Building an e-health system for health awareness campaigns in poor areas

Submitted in fulfilment
of the requirements of the degree of

Master of Science

of Rhodes University

Chikumbutso David Gremu

Grahamstown, South Africa
March 2015

Abstract

Appropriate e-services as well as revenue generation capabilities are key to the deployment and the sustainability for ICT installations in poor areas, particularly common in developing country. The area of e-Health is a promising area for e-services that are both important to the population in those areas and potentially of direct interest to National Health Organizations, which already spend money for Health campaigns there.

This thesis focuses on the design, implementation, and full functional testing of HealthAware, an application that allows health organization to set up targeted awareness campaigns for poor areas. Requirements for such application are very specific, starting from the fact that the preparation of the campaign and its execution/consumption happen in two different environments from a technological and social point of view. Part of the research work done for this thesis was to make the above requirements explicit and then use them in the design. This phase of the research was facilitated by the fact that the thesis' work was executed within the context of the Siyakhula Living Lab (SLL; www.siyakhulaLL.org), which has accumulated multi-year experience of ICT deployment in such areas.

As a result of the found requirements, HealthAware comprises two components, which are web-based, Java applications that run in a peer-to-peer fashion. The first component, the Dashboard, is used to create, manage, and publish information for conducting awareness campaigns or surveys. The second component, HealthMessenger, facilitates users' access to the campaigns or surveys that were created using the Dashboard. The HealthMessenger was designed to be hosted on TeleWeaver while the Dashboard is hosted independently of TeleWeaver and simply communicates with the HealthMessenger through webservices.

TeleWeaver is an application integration platform developed within the SLL to host software applications for poor areas. Using a core service of TeleWeaver, the profile service, where all the users' defining elements are contained, campaigns and surveys can be easily and effectively targeted, for example to match specific demographics or geographic locations.

Revenue generation is attained via the logging of the interactions of the target users in the communities with the applications in TeleWeaver, from which billing data is generated according to the specific contractual agreements with the National Health Organization.

From a general point of view, HealthAware contributes to the concrete realizations of a bidirectional access channel between Health Organizations and users in poor communities, which not only allows the communication of appropriate content in both directions, but get 'monetized' and in so doing becomes a revenue generator.

Acknowledgements

I would like to thank my supervisors Prof. Alfredo Terzoli and Dr. Mosiuoa Tsietsi for their untiring efforts to assist me during the whole period I was doing research. They were available every time I needed them. Without their guidance, I would not have achieved the results I obtained.

I would also like to thank my family for their support. Their patience, understanding, and frequent phone calls to encourage me to work hard were some of the reasons I always looked forward to working on my research every day.

I would also like to thank Sisonke Mawonga, a Masters student in the Department of African Languages at Rhodes University for her assistance with translations to make the system that was developed as part of the research accessible in English and isiXhosa.

Last but not least, I would like to acknowledge the financial support of Telkom SA, Tellabs, Genband, Easttel, Bright Ideas 39, THRIP and NRF SA (TP13070820716) through the Telkom Centre of Excellence in the Department of Computer Science at Rhodes University.

Author Publications

The research work discussed in this thesis resulted in the following publications:

1. Chikumbutso Gremu, Alfredo Terzoli and Mosiuoa Tsietsi. A User-Oriented, Bi-Directional Information Channel for e-Health Interventions in Marginalised Rural Areas. *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 1 - 4 September 2013, Spier Wine Estate, Stellenbosch, Western Cape, South Africa .

2. Chikumbutso Gremu, Alfredo Terzoli and Mosiuoa Tsietsi. A Consumer Health Informatics Application for e-Health Interventions in Marginalised Rural Areas. *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 31 August - 3 September 2014, The Boardwalk, Port Elizabeth, Eastern Cape, South Africa.

Contents

1 Introduction to the Research **1**

1.1 Introduction 1

1.2 Background 1

1.3 Siyakhula Living Lab 2

1.4 Research Problem 3

1.5 Research Objectives 3

1.6 Opportunities 4

1.7 Scope and Limitations 4

1.8 Research Approach 5

1.9 Challenges 6

1.10 Thesis Structure 6

1.11 Summary 7

2 Situation Analysis and Related Work **8**

2.1 Introduction 8

2.2 The South African Public Health System 8

2.2.1 The Primary Level 9

2.2.2 The Secondary Level 10
2.2.3 Use of e-Health to Deliver Public Health Services in South Africa . 10
2.3 The Use of the Internet in South Africa 11
2.4 Health Challenges in South Africa 12
2.4.1 Major Health Challenges 12
2.4.2 Efforts by the Government to Address the Challenges 12
2.4.3 Other Possible Solutions to the Challenges 13
2.5 Initiatives to Provide Health Information to the Public in South Africa . . 13
2.5.1 Project Masiluleke 14
2.5.2 LoveLife 15
2.5.3 MomConnect 16
2.5.4 Analysis of the Projects 17
2.6 Living Labs 17
2.7 Siyakhula Living Lab Project 17
2.7.1 Dwesa 18
2.7.2 ICT Infrastructure and Training in Dwesa 19
2.7.3 TeleWeaver 19
2.7.4 TeleWeaver Business Model 19
2.7.5 TeleWeaver Applications 21
2.7.5.1 Profile Application 21
2.7.5.2 Billing Application 21
2.7.5.3 Message Dissemination Application 21
2.8 Financing the Development of ICT in Poor Areas 21
2.9 Summary 22

3 Background Literature 24

3.1 Introduction 24

3.2 Information Communication to Consumers 24

3.2.1 Communication Systems 24

3.2.2 Communication Channels 25

3.2.3 Communication Types 25

3.2.3.1 Asynchronous Communication 25

3.2.3.2 Synchronous Communication 26

3.2.4 Communication Tools 26

3.2.4.1 Briefings 26

3.2.4.2 Mailing 26

3.2.4.3 Exhibits 26

3.2.4.4 Fact Sheets 27

3.2.4.5 Newsletters 27

3.2.5 Communication Campaigns 27

3.3 Health Communication 28

3.3.1 Stages in Health Communication 29

3.3.1.1 Reaching the Audience 30

3.3.1.2 Gaining the Target Audience's Attention 30

3.3.1.3 Understanding the Message 30

3.3.2 Health Communication Through the Internet 30

3.3.2.1 Advantages 31

3.3.2.2 Disadvantage 31

3.3.3 Health Communication Strategies 31
3.3.3.1 Mass Communication 32
3.3.3.2 Targeted Communication 32
3.3.3.3 Tailored Communication 33
3.4 Consumer Health Informatics 35
3.5 Social Marketing 36
3.5.1 Social Marketing Principles 37
3.5.2 What is Involved in Social Marketing 38
3.6 Information and Data Gathering 38
3.6.1 Advantages of Surveys 39
3.6.2 Disadvantages of Surveys 39
3.7 Summary 39

4 Review of Health Websites 41

4.1 Introduction 41
4.2 Common Features of Consumer Health Websites 41
4.2.1 Publicity 42
4.2.2 Events 42
4.2.3 News 42
4.2.4 Disease and Conditions 43
4.2.5 Drugs and Medications 44
4.2.6 Ask An Expert 44
4.2.7 Multimedia 44

4.2.8 Facilities 44
4.2.9 Services 45
4.3 Observations 46
4.4 Summary 46

5 Requirements and Design 47

5.1 Introduction 47
5.2 Requirements 47
5.2.1 System Architecture 48
5.2.2 Software Architecture 50
5.2.3 Language Switching 51
5.2.4 Support Functions 52
5.2.5 Targeting 52
5.3 Functional Specifications of Component 1 52
5.3.1 Campaigns Function 54
5.3.1.1 Creating a Campaign 54
5.3.1.2 Campaign States 54
5.3.1.3 Updating a Campaign 54
5.3.1.4 Deleting a Campaign 55
5.3.1.5 Stopping and Restarting a Campaign 55
5.3.1.6 Adding or Updating Campaign Messages 56
5.3.1.7 Targeting a Campaign 56
5.3.1.8 Archiving a Campaign 56

5.3.1.9 Publishing a Campaign 57
5.3.2 Surveys Function 57
5.3.3 Events Function 58
5.3.4 News Function 58
5.3.5 Topics Function 58
5.3.6 Creating and Managing Facilities and Services 59
5.3.7 Users Function 59
5.3.7.1 Creating a User 61
5.3.7.2 Managing Users 61
5.3.7.3 User Authentication 61
5.3.7.4 States of User Accounts 62
5.3.7.5 User Authorisation 64
5.3.8 Search Function 64
5.3.9 Bulk Functions 64
5.4 Functional Specifications of Component 2 64
5.4.1 HealthMessenger Requirements 66
5.4.2 Campaigns Function 66
5.4.2.1 Pull Mode 67
5.4.2.2 Push Mode 68
5.4.3 Surveys Function 69
5.4.4 Events Function 69
5.4.5 News Function 69
5.4.6 Facilities Function 70

5.4.7 Topics Function 70
5.5 Logging of Auditing and Billing Data 70
5.6 Data Flow in HealthAware 72
5.7 Summary 73

6 Implementation **74**

6.1 Introduction 74
6.2 Tools Used 74
6.2.1 Host Server Environment 75
6.2.2 Development Tools 75
6.2.2.1 Creating Java Web Application Projects 75
6.2.2.2 Project Library Dependency Management 76
6.2.2.3 Wiring Dependencies 77
6.3 Implementation of HealthAware Functions 78
6.3.1 Presentation Layer 78
6.3.1.1 Creation of Forms and Form Elements 79
6.3.1.2 Creation of Tables and a Search Function 79
6.3.1.3 Creating of Menu Icons 80
6.3.1.4 Data/Input Validation 80
6.3.1.5 Client and Server Communication 81
6.3.2 Business Layer 82
6.3.2.1 The Service Layer 82
6.3.2.2 The App Services/Work Flows 82

6.3.2.3 The Domain Model 82
6.3.3 Data Access Layer 83
6.4 Implementation of the Dashboard Functions 84
6.4.1 Implementation of the Campaigns Function 85
6.4.1.1 Business Layer 86
6.4.1.2 Data Access Layer 88
6.4.1.3 Publishing a Campaign 88
6.4.2 Implementation of the Surveys Function 89
6.4.2.1 Business Layer 90
6.4.2.2 Data access Layer 91
6.4.3 Implementation of the Events Function 91
6.4.3.1 Business Layer 91
6.4.3.2 Data access Layer 93
6.4.4 Implementation of the News Function 93
6.4.4.1 Business Layer 93
6.4.4.2 Data access Layer 94
6.4.5 Implementation of the Topics Function 94
6.4.5.1 Business Layer 95
6.4.5.2 Data Access Layer 96
6.4.6 Implementation of the Facilities and Services Functions 96
6.4.6.1 Business Layer 96
6.4.6.2 Data Access Layer 97
6.4.7 Implementation of the Users Function 98

6.4.7.1 Business Layer 98
6.4.7.2 Data Access Layer 100
6.4.7.3 User Authentication 101
6.4.7.4 User Authorisation 102
6.4.8 Report Generation 107
6.5 Implementation of the HealthMessenger Functions 107
6.5.1 Campaigns Function 109
6.5.1.1 Creating a Campaign 109
6.5.1.2 Logging of Auditing Data 109
6.5.1.3 Logging of Billing Data 109
6.5.2 Surveys Function 109
6.5.2.1 Completing a Survey 110
6.5.2.2 Logging of Auditing Data 110
6.5.3 Events Function 110
6.5.3.1 Creating an Event 110
6.5.3.2 Registering and Deregistering from an Event 110
6.5.4 News Function 111
6.5.5 Facilities Function 111
6.5.6 Targeting 111
6.5.6.1 Profile Application 111
6.5.6.2 Selection of Targeted Health Information 112
6.5.6.3 Accessing Health Information in a Pull Mode 113
6.5.6.4 Accessing Health Information in a Push Mode 113

6.5.7 Language Switching . . . 114
6.5.8 Reporting . . . 115
6.6 Implementation of the Store and Forward Function . . . 116
6.7 Summary . . . 117

7 Testing **118**

7.1 Introduction . . . 118
7.2 Software Testing Approaches . . . 118
7.2.1 Black Box Testing . . . 118
7.2.2 White Box Testing . . . 119
7.2.3 Specific Testing Methods . . . 119
7.2.3.1 Unit Testing . . . 119
7.2.3.2 Integration Testing . . . 119
7.2.3.3 System Testing . . . 119
7.3 Test Environment and Approach . . . 120
7.3.1 Test Environment . . . 120
7.3.2 Test Approach . . . 120
7.4 Testing the Campaigns Function . . . 120
7.4.1 Creating a Campaign . . . 122
7.4.2 Publishing a Campaign . . . 122
7.4.3 Stopping and Restarting a Campaign . . . 123
7.4.4 Archiving a Campaign . . . 125
7.4.5 Accessing Campaigns . . . 125

7.4.5.1 Accessing Campaigns in a Push Mode 127
7.4.5.2 Accessing Campaigns in a Pull Mode 129
7.4.6 Generating Campaign Hits Report 134
7.5 Testing the Surveys Function 134
7.5.1 Creating a Survey 135
7.5.2 Completing a Survey 139
7.6 Testing the Events Function 140
7.6.1 Creating an Event 140
7.6.2 Registering for an Event 141
7.7 Testing the News Function 141
7.7.1 Creating a News Article 141
7.7.2 Accessing News Through the HealthMessenger 142
7.8 Testing the Topics Function 143
7.8.1 Creating a Topic 143
7.8.2 Accessing Topics Through the HealthMessenger 146
7.9 Testing the Facilities Function 147
7.9.1 Creating a Facility 147
7.9.2 Creating a Service 148
7.9.3 Adding and Removing Services From a Facility 148
7.9.4 Accessing Facilities and Services Using the HealthMessenger 149
7.10 Testing the User Management Function 151
7.10.1 Creating a User 151
7.10.2 Activating and Deactivating a User 151

7.10.3 Testing Locking and Unlocking a User Account 152

7.10.4 Logging in and Logging out a User 154

7.11 Testing Language Switching 154

7.12 Testing Store and Forward 155

7.13 Discussion 157

7.14 Summary 158

8 Summary and Conclusion **159**

8.1 Introduction 159

8.2 Summary 159

8.2.1 Situation Analysis and Related Work 159

8.2.2 Review of Related Literature 160

8.2.3 Review of Consumer Health Websites 160

8.2.4 Design and Implementation of HealthAware 161

8.2.5 Testing of HealthAware 161

8.3 Meeting the Research Objectives 161

8.3.1 e-Health System to Facilitate Targeted Awareness Campaigns . . . 161

8.3.2 A Function to Generate Revenue to Support the Development and Upkeep of ICT Infrastructure 162

8.4 Fulfilling Solution Requirements 162

8.4.1 Hosted on TeleWeaver 163

8.4.2 Accessible in Multiple Languages 163

8.4.3 Usable in Low Speed and Intermittent Internet Connectivity Environments 163

8.4.4 Logging of User Interactions for Billing . . . 163
8.5 Contributions . . . 164
8.6 Future Work . . . 164
8.7 Summary and Conclusion . . . 165

References **166**

A Spring **175**

A.1 Spring Functional Areas . . . 175
A.2 Dependency Injection . . . 176

B Message Tailoring **178**

B.1 Personalisation . . . 178
B.1.1 Identification . . . 178
B.1.2 Raising Expectation . . . 178
B.1.3 Contextualisation . . . 179
B.2 Feedback . . . 179
B.2.1 Descriptive Feedback . . . 179
B.2.2 Comparative Feedback . . . 179
B.2.3 Evaluative Feedback . . . 180
B.3 Content Matching . . . 180

C Configuration Details for pom.xml **181**

List of Tables

7.1 Test schedule for campaigns 130

List of Figures

2.1 HIV/AIDS Messages sent in the unused PCM space. Source: [67]. 15

2.2 MomConnect mobile messaging service [68]. 16

2.3 Action space for Living Labs along the technology adoption cycle [28]. . . . 18

2.4 TeleWeaver Business Model [79]. 20

3.1 Stages in Health Communication [64]. 29

3.2 Overview of the process of message tailoring [61]. 34

4.1 MedlinePlus web interface for Health Topics 43

4.2 Healthfinder.gov website 45

5.2 System Architecture 48

5.1 Overall use case diagram for HealthAware 49

5.3 The logical view of a layered software architecture [57]. 51

5.4 Dashboard use case 53

5.5 State diagram of a campaign 55

5.6 State diagram for stopping and restarting a campaign 56

5.7 State diagram for publishing a campaign 57

5.8 State diagram for News 59

5.9 Use case diagram for user management 60

5.10 State diagram for user account state transitions 62

5.11 Mock screen listing users 63

5.12 Mock screen showing a list of campaigns 65

5.13 Use Case of the HealthMessenger 66

5.14 Activity diagram of the pull mode 67

5.15 Activity diagram of the push mode 68

5.16 Use case for registering for an event 70

5.17 Sequence diagram for pull strategy 71

5.18 Sequence diagram for push strategy 72

6.1 The Layered Architecture used to implement the functions of HealthAware [30]. 79

6.2 Project Structure for the Dashboard 85

6.3 Classes used to implement the business and data access layers of the Campaigns function 86

6.4 Domain Model classes for the *Campaigns* function 87

6.5 Classes used to implement the business and data access layers of the *Surveys* function 90

6.6 Domain Model classes used to implement the *Surveys* function 91

6.7 Classes used to implement the business and data access layers of the *Events* function 92

6.8 Domain Model classes for the *Events* function 92

6.9 Classes used to implement the business and data access layers of the *News* function 93

6.10 Domain Model classes used to implement the *News* function 94

6.11 Classes used to implement the business and data access layers of the *Topics* function 95

6.12 Domain Model classes for the *Topics* function 96

6.13 Classes used to implement the business and data access layers of the *Facilities* function 97

6.14 Domain Model classes for the *Facilities* function 98

6.15 Classes used to implement the business and data access layers of the *Users* function 99

6.16 Domain Model classes for the *Users* function 100

6.17 Project files for the implementation of the HealthMessenger 108

7.1 List of campaigns (screenshot taken on 08-12-2014) 121

7.2 New campaign 122

7.3 Published campaign 123

7.4 A campaign in running state 124

7.5 Stopping a campaign 124

7.6 Archived campaign 125

7.7 List of campaign targets as extracted from the database 126

7.8 List of campaign targets showing targeted locations 126

7.9 List of users and their demographics as extracted from the database 126

7.10 List of user addresses 127

7.11 Welcome page for an authenticated user 127

7.12 A notification for Kaka Mandimba 128

7.13 Notification for Nuno Gomez 129

7.14 No campaigns accessed by Anga Ndimaso (a.ndimaso). 131

7.15 A campaign accessed by Kaka Madimba (k.madimba) 131

7.16 Campaigns accessed by Red Spencer (red.spencer) 131

7.17 No campaigns accessed Osman Phiri (o.phiri). 131

7.18 Campaigns accessed by Nuno Gomez (n.gomez). 132

7.19 Campaigns accessed by Benson Msakambewa (b.msakambewa). 132

7.20 Campaigns accessed by Pepe Kale (p.kale). 133

7.21 Campaign message after a user click on a campaign link 133

7.22 Number of hits 134

7.23 Campaign hits report 135

7.24 Creating a Survey 136

7.25 A form that enables a user to select question type 137

7.26 A form for adding questions that can have one answer only 137

7.27 A form for adding questions that can have multiple answers 138

7.28 List of survey questions 138

7.29 List of surveys 139

7.30 Screenshot of a page to complete a survey 139

7.31 Form to create an event 140

7.32 Events a user can register 141

7.33 A form to create a news article 142

7.34 List of news items 142

7.35 News details 143

7.36 Creating a topic 144

7.37 Creating topic details 145

7.38 List of health topics 145

7.39 A list of health topics 146

7.40 Detailed view of a topic 146

7.41 Creating a facility 147

7.42 Creating a service 148

7.43 Adding and removing services 149

7.44 List of health facilities 150

7.45 A user searching for facilities by district (Amathole) 150

7.46 Displaying services offered by a facility 151

7.47 New user created 152

7.48 User deactivated 153

7.49 Locked user account 153

7.50 User logged in 154

7.51 Switching language to isiXhosa 155

7.52 HealthMessenger accessible in isiXhosa 155

7.53 Communication when one broker is offline 156

7.54 Communication when both brokers are online 157

A.1 Spring framework's basic functional areas [89]. 176

Glossary of Terms

AOP	Aspect-Oriented Programming
API	Application Programming Interface
ART	Antiretoviral Therapy
CCS	Cascading Style Sheets
CHI	Consumer Health Informatics
CORS	Cross-Origin Resource Sharing
DAN	Digital Access Node
DI	Dependency Injection
HIV/AIDS	Human Immunodeficiency Virus/Acquired Immune Deficiency Syndromme
HTML	HyperText Markup Language
ICT	Information and Communications Technology
ICTD	Information and Communications Technology for Development
IDE	Integrated Development Environment
IoC	Inversion of Control
IP	Internet Protocol
JAR	Java Archive
JAX-RS	Java API for RESTful Web Services
JDBC	Java Database Connectivity

JMS	Java Message Service
JSON	JavaScript Object Notation
MSM	Males who have Sex with Males
NDoH	National Department of Health
ORM	Object-Relation Mapping
OSGi	Open Systems Gateway Initiative
OWASP	Open Web Application Security Project
OXM	Object/XML Mapping
PCM	Please Call Me
PIN	Personal Identification Number
POJO	Plain Old Java Object
POM	Project Object Model
RDI	Research Development and Innovation
REST	Representational State Transfer
SLL	Siyakhula Living Lab
SMS	Short Message Service
SQL	Structured Query Language
STD	Sexually Transmitted Diseases
TB	Tuberculosis
URL	Uniform Resource Identifier
USAASA	Universal Service and Access Agency of South Africa
VM	Virtual Machine
XML	Extensible Markup Language

Chapter 1

Introduction to the Research

1.1 Introduction

This chapter introduces the thesis by setting the research discussed into context. The chapter is divided into several sections. Section 1.2 provides a background to the research. Section 1.3 introduces the Siyakhula Living Lab (SLL). Section 1.4 states the problem the research aims to address. Section 1.5 discusses the research objectives. Section 1.6 describes the opportunities to implement the work done as part of the research in poor areas. Section 1.7 discusses what was included and excluded from the research. Section 1.8 discusses the approach that was taken to conduct the research. Section 1.9 discusses the challenges encountered in the course of the research and the measures taken to ensure that the research objectives are still met. Section 1.10 provides an outline of the thesis. Section 1.11 provides a summary of what has been discussed in the chapter.

1.2 Background

Traditionally, physicians or medical practitioners were the custodians of medical and health related knowledge [51]. Patients and non-patients depended on them as the primary and possibly only source of health information. This changed with the advent of the Internet, which facilitates consumers' access to a broad range of e-Health applications such as online health informatics websites, interactive electronic health records, health decision support systems, tailored health programmes, health care system portals, mobile

health communication programmes, and telemedicine applications. The producers of health information use these applications to disseminate health information to a large number of consumers, at comparatively low cost and at a time and place that is convenient to different individuals [34]. In this thesis, a health information consumer or a consumer should be interpreted as an individual who seeks information on how to stay healthy, prevent diseases, treat specific conditions, and manage various health conditions and chronic diseases etc [51]. A consumer can be a patient, caregiver, friend or family member of a patient,or any member of the general public who is concerned about health.

The ubiquity of online health information facilitated by the Internet has led to changes in the consumer-health provider relationship [47]. Consumers are now more knowledgeable about health, making a consumer-centered approach, where consumers actively participate in making decisions on their health possible. However, this is more likely to increase health disparities and disadvantage those having limited access the Internet [12]. This disparity is called the digital divide and is defined as the gap between persons who have and those who do not have or have restricted access to technology [45]. In the context of access to health information, the disparity is mainly caused by the lack of competencies to effectively use technology and the means to access information.

The digital divide mainly affects those who are of low economic status, the majority being those living in poor areas. In South Africa, there are several efforts to bridge the digital divide by providing the people in poor areas access to computers with Internet connectivity. An example of such efforts is the SLL, which is discussed in the next section.

1.3 Siyakhula Living Lab

The SLL is multi-stakeholder operation that consists of the Telkom Centers of Excellence at Rhodes University and the University of Fort Hare, sponsors from provincial governments and the industry, and marginalized communities that facilitate user-driven innovations in the Information and Communications Technology for Development (ICTD) domain [73]. The operation was launched in 2005 in Dwesa, a rural area in the vicinity of Dwesa-Cwebe Nature Reserve in Mbashe municipality in the Eastern Cape province of South Africa, where a community lab was opened. Other labs have been opened in Grahamstown and the town of Alice in the same province. The partners of the project supported the installation of Information and Communications Technology (ICT) infrastructure and computers with Internet connectivity at all the labs. The setup at the labs,

which is called a Digital Access Node (DAN), has terminals that are realised as thin clients that run on Edubuntu or Ubuntu Operating System of the Linux family. To ensure that the computers benefit the surrounding communities, the members are trained free of charge by researchers in the field ICTD from Rhodes University, the University of Fort Hare, and other research institutions. Software applications for the communities are developed by the researchers as part of their research, and by Reed House Systems, a software development company that emerged as part of the efforts to build software applications for poor areas. The company designed and developed a service integration platform called TeleWeaver, which is installed at DANs to host these applications.

1.4 Research Problem

The Internet has brought many benefits to the field of health. The ubiquity of online health information has empowered consumers to actively participate in making decisions on their health. However, there still exists other challenges, which need to be addressed. The first challenge is that most of the health information that is found online is developed for the general audience and cannot address the specific needs of individuals or groups. This therefore requires consumers of health information to sift through irrelevant information before arriving at the relevant information. The information is also mostly in English and other foreign languages with no option to access it in other languages such as the South African local languages. This disadvantages consumers who do not understand the languages. The other challenge is that the ICT infrastructure that is installed in poor areas such as Dwesa to address the digital divide is not sustainable without the support of external funding partners. This results in the efforts to bridge the digital divide not fruitful because once the equipment breaks down, the target users do not have access to technology.

1.5 Research Objectives

The research has two objectives. The first objective is to develop an e-health system to facilitate health awareness campaigns carried out by health service organisations such as the South African Department of Health that target people in poor areas. The awareness campaigns should be targeted to ensure that the target audience access only the health information that is relevant them. The system should also be used to collect data from

the target audience that can be used to determine the services they need. The second objective is to extend the functions of the system in order to generate revenue to support the upkeep of ICT infrastructure in the target areas. The following should be fulfilled by the system in order to meet the research objectives:

- target users should access awareness campaign messages through TeleWeaver instances in their areas;
- it should be accessible in multiple languages to accommodate different users;
- it should be usable with minimal challenges when the Internet connectivity is intermittent or not available at all; and
- user interactions with the system should be logged for the purpose of generating bills for the organisations using the system.

1.6 Opportunities

The use of software applications to provide health information to people living in poor areas is a challenge because the ICT infrastructure in the areas is either underdeveloped or nonexistent and the ICT literacy among the people is low. These challenges have been addressed in some areas in Dwesa, Grahamstown, and Alice through the SLLs. The research targets these areas to take advantage of the opportunities they present. The use of TeleWeaver to host the e-health system has the following advantages:

- awareness campaign messages can be created and disseminated to a particular community in a language that is understood by the majority;
- the user profile information in TeleWeaver, as discussed in Sub-section 2.7.5.1, can be used to select awareness campaigns that are relevant to a particular user; and
- it can be accessible to the target users all the time through DANs in their areas.

1.7 Scope and Limitations

The research has the following scope and limitations:

- it involves the development of a system to enable health service organisations to carry out awareness campaigns in poor areas and it excludes the assessment of their impact;
- target consumers those with access to a SLL and have attended at least one training session to use of computers and the Internet;
- it excludes the development of a billing application and a message dissemination application to push messages to consumers - these applications will be developed by Reed House Systems; and
- it excludes the processes to obtain rights to process personal information as required by the South African Protection of Personal Information Act of 2013 [70].

1.8 Research Approach

The study commenced with a review of literature to learn the communication strategies that are used to ensure that consumers access only the health information that is relevant to them. The second stage was to review websites developed to disseminate health information to consumers, identify their common functions and select appropriate ones that were included in the system that was developed. A search was also done for open source solutions similar to the system to be developed. The search did not yield any results, therefore the system was developed from scratch. System development used an evolutionary prototyping approach, where the initial implementation of an application evolves towards the final version [9]. The prototypes of the system were demonstrated to Raphael Centre, a non governmental organisation that provides care and support to people with AIDS, supports and develops community action, and fights stigma and discrimination the people living with HIV suffer [77], officials from the Eastern Cape Department of Health, delegates at the Southern Africa Telecommunication Networks and Applications conference of 2014 and delegates at the Eastern Cape ICT summit of 2014. The feedback obtained from the audience was used to modify the functions of the system. The following chapters will provide the details of the approach taken.

1.9 Challenges

Several challenges were met during the development of the system. At the onset, the version of TeleWeaver that was available was based on Open Systems Gateway Initiative (OSGi) and development work targeted that environment. A review of that version recommended moving from OSGi to WildFly 8.2.0.Final and development work changed to target the new environment. Wildfly is based on JBOSS[1], which is an enterprise application platform that was developed to build, deploy, and host Java applications and services. The new version of TeleWeaver was not ready when the system was completed, therefore, the scope of work was extended to include user authentication in order to test targeting of awareness campaigns. The initial design of the system assumed that user authentication would be done by the profile application in TeleWeaver. The other challenge is the legal requirement to get clearance to use and process personal information as required by South African Protection of Personal Information Act of 2013. The unavailability of TeleWeaver and the clearance prevented testing of the system in real-life.

1.10 Thesis Structure

The thesis is organised into several chapters. Chapter 1 introduces the thesis by providing a background to the research. The rest of the chapters are organised as follows:

Chapter 2 provides a brief discussion of the South African public health system, use of the Internet South Africa, initiatives to provide the South African general public with health information, the Living Lab methodology and financing of ICT infrastructure in poor areas. The discussion of the South African public health system includes the challenges it faces and the use of electronic health (e-Health) systems by the National Department of Health to improve the delivery of public health services to overcome the challenges. The chapter concludes with a discussion of how different countries are financing the development of ICT in poor areas.

Chapter 3 reviews related literature. It includes a discussion of the strategies that are used in health communication to ensure that consumers access health messages that are relevant to them and Consumer Health Informatics (CHI), which is the development of applications to enable consumers access health information.

[1] http://www.jbossweb.jboss.org

Chapter 4 discusses a review of consumer health websites that was done to identify their common functions and select appropriate ones included in the system that developed.

Chapter 5 discusses the requirements and design of the system. The functions of the system were derived from the investigation of consumer health websites in **Chapter 4** and the need to collect data from the target audience.

Chapter 6 discusses the implementation of the system. The discussion includes the tools used and how they helped to achieve the implementation of specific functions.

Chapter 7 discusses the tests that were done to verify that the system meets the requirements discussed in **Chapter 5**.

Chapter 8 summarises what is discussed in this thesis and reviews the research objectives to assesses if they were met.

1.11 Summary

This chapter introduces the thesis by setting the research being discussed into context. The introduction includes a discussion of the background information, problem statement, objectives, opportunities, scope and limitations, and the approach that was used to conduct the research. The next chapter presents a brief situation analysis of the South African public health system, use of the Internet in South Africa and initiatives to provide the South African general public with health information.

Chapter 2

Situation Analysis and Related Work

2.1 Introduction

The previous chapter introduced the thesis by setting the research discussed into context. This chapter analyses and discusses the South African public health system, how people use the Internet in South Africa, initiatives to provide the South African general public with health information, how Siyakhula Living Labs (SLLs) are bridging the digital divide between those with and without Internet access in South Africa and how countries are financing the development of Information and Communications Technology (ICT) in poor and marginalised areas.

2.2 The South African Public Health System

Public health services in South Africa are delivered across three levels of government, namely national, provincial, and local [27]. The national government focuses on legislation, policy, norms, standards, and ensuring equity. The provincial government focuses on planning, budgeting, and delivery of health services at the coalface. The local government is responsible for the delivery of municipal health services. Public health services are provided free of charge to children under six years and pregnant or breastfeeding mothers [56]. The services are accessed at two levels, which are primary and secondary levels.

2.2.1 The Primary Level

This is the first point of entry for people seeking public health services. Facilities at this level are clinics and community health centres. Patients who are treated at this level are those who are able to walk on their own with no need to be confined to a bed. Those who require more specialised care are referred to the next (secondary) level of care by clinical staff. The services offered at this level include preventive, promotional, curative, and rehabilitative services such as mother and child care, immunisation, family planning, treatment of sexually transmitted infections, minor trauma, and care for those with chronic illnesses (e.g. diabetes and hypertension). The services are run by nurses and doctors regularly visit the facilities.

The primary level faces many challenges. According to Visser *et al.* [86], only 47% of the facilities at this level are visited by doctors. As reported by Cullinan [23], based on the National Primary Health Care Facilities Survey of 2003, only a quarter of the clinics surveyed had piped water; about 10% did not meet the sanitary requirements, were not connected to the national electricity grid and did not have telecommunications equipment such as phones or faxes; only 40% had trained primary healthcare nurses; and only half of them were reported to have functional clinic or community committees meaning that most communities are not involved in health issues. The facilities have a high patient to health professional ratio because they fail to attract and retain experienced professional nurses [23]. This is worsened by the requirement that it should be the point of entry for people seeking public health services, which leads to congestion. The congestion forces clinicians to refer most cases that can be handled at that level to the next level of care, creating further congestion at that level.

The challenges at the primary level mainly affect people living in poor areas and of a low economic status. This is because private health services are expensive and not affordable to them or the primary healthcare facilities are the only facilities available in their areas. The challenges such as congestion at the facilities can be alleviated, to some extent, if preventative efforts are intensified in order to decrease the frequency at which people seek medical or other assistance. Equipping the people with information for informed decision-making is one of the measures.

2.2.2 The Secondary Level

Facilities at the secondary level are hospitals. They are primarily for those who need in-patient care. They are categorised as either Acute or Specialised. Acute hospitals include Central, Tertiary, Regional, and District hospitals, while specialised hospitals include Chronic, Orthopaedic, Psychiatric, Rehabilitation, and Tuberculosis (TB) hospitals. Below is a brief description of the services that are offered by acute hospitals.

- **District Hospitals**: they are the first level of referral. General practitioners provide basic diagnostic and therapeutic services, such as X-rays and basic laboratory tests.
- **Regional Hospitals**: they provide care that requires the intervention of specialists and general practitioners.
- **Tertiary Hospitals**: they provide specialist and sub-specialist care. They receive patients from and provide sub-specialist support to Regional Hospitals.
- **Central Hospitals**: they are highly specialised and provide an environment for multi-specialty clinical services, innovation, and research.

2.2.3 Use of e-Health to Deliver Public Health Services in South Africa

The South African National Department of Health (NDoH) has been implementing electronic health (e-Health) programmes for a number of years in order to improve the delivery of public healthcare services. e-Health is the delivery of health services and information through the Internet and related technologies [32]. In the years 1999/2000, the Department together with the National Health Information System Committee of South Africa initiated a national Telemedicine Programme [27]. Telemedicine is the use of ICT to deliver healthcare services to individuals who are located at a distance from a healthcare provider [72]. The programme was expected to overcome the challenges of the rural health settings and offer better services to citizens who are marginalised because of their location. According to the 2012 - 2016 National e-Health strategy [27], the programme failed to achieve its objectives and stalled after hardware was installed and various services were handed over to provinces for maintenance and expansion.

In 2012, the Department launched its 2012-2016 e-Health strategy [27]. The mission of the strategy is to establish e-Health as an integral part of the transformation and improvement of healthcare services in South Africa. One mandate of the strategy is to endeavour to reach out to communities, including vulnerable groups, with health services appropriate to their needs. The strategy defined Consumer Health Informatics (CHI) but no further reference was made to it afterwards. Its focus is the development of e-Health tools to enable physicians to provide better services to patients. These tools include electronic medical records, healthcare information systems, surveillance systems, business intelligence for health, electronic content management systems, decision support systems, and knowledge management systems. Progress has been made in developing some of the proposed tools, which include the District Health Information System, which records routine information on facility-based services, community-based campaigns, infrastructure, and human resources, the National Electronic TB Register that monitors cohort groups of TB patients, and the Western Cape Primary Healthcare Information System and Patient Master Index [27].

2.3 The Use of the Internet in South Africa

The number of Internet users worldwide, including sub-Saharan Africa, has been increasing over the years. As reported by Akue-Kpakpo [2], between the years 2006 and 2010, the average growth in number of Internet users in sub-Saharan Africa was 35%. Similarly, the number of Internet users in South Africa has been increasing over the years, the majority of new users being from the low income category [26]. As reported by De Lanerolle [26], the ratio of Internet users in South Africa in 2008 was one person in seven. In 2012, it was one person in three and the total number of users was 12.3 million. It was projected that if the growth rate is maintained, the ratio will be one person in two in 2014 and two people in three in 2016. The least number of Internet users in 2012 were those living below the poverty line (19%) followed by those of low income (23%). According to South Africa: The Good News [76], a news website that highlights positive developments in South Africa, 41% of South Africans were using the Internet in 2013. The remaining 49% were facing challenges of barriers of means of access (devices and networks), cost of access, knowledge of the existence of the Internet and language barriers. Those with Internet access mainly used it to get information (Internet search), to socialise, for study, for work or business, and to look for jobs.

The rate at which the number of Internet users is growing presents an opportunity to use

the Internet as a channel to disseminate health information to a larger population in the low income bracket including those living in poor areas. Although the target reach can be relatively low as compared to using other channels, with the projected growth rate, it will be possible to reach out to a larger population in future.

2.4 Health Challenges in South Africa

This section discusses some of the health challenges that South Africa is facing and the efforts to alleviate the challenges.

2.4.1 Major Health Challenges

South Africa faces a number of health challenges. Some of the major challenges are communicable diseases (especially HIV/AIDS), noncommunicable diseases, maternal, neonatal and child deaths, and deaths from injury and violence [80]. South Africa has the highest prevalence of HIV/AIDS in the world [1]. It accounts for 18% of the global burden of HIV infection despite having only 0.7% of the world's population [83]. In 2011, 5.58 million South Africans were living with HIV, 43% of deaths were due to HIV/AIDS and 65% of the patients suffering from tuberculosis (TB) were infected with HIV [55]. South Africa is one of only a dozen countries that has seen an increase in maternal mortality [55]. In 2007, overall maternity mortality was 650 in every 100 000 as opposed to 150 in every 100 000 in 1998 [10]. In 2005, infant mortality ranged from 18 per 1 000 live births among white people to 74 per 1 000 live births among black people [10]. The other major health challenge is the inequalities in health status and access to health services across the provinces. Only 14% of citizens are able to access services provided by the private healthcare sector, even though it benefits from up to 60% of the national health expenditure [80].

2.4.2 Efforts by the Government to Address the Challenges

The South African government has made progress in addressing some of the challenges the country is facing. According to the Gap Report of 2014 by the UNAIDS [83], the country has the largest antiretoviral therapy (ART) programme in the world. The number of AIDS-related deaths decreased by 51% between 2009 and 2013. The largest number

of new people added to the list of those receiving ART in the world during the same period was in South Africa (33%). However, despite these achievements, the challenges of HIV/AIDS are still persistent. According to the report, about 58% of the people living with HIV in South Africa are still not accessing ART. In 2013, South Africa accounted for 25% of the people living with HIV in sub-Saharan Africa and had the highest proportion of new HIV infections by country (16%). However, in absolute numbers, the number of new infections was 98,000 fewer than in 2010.

The government has been increasing its funding in response to the interlinked HIV and TB epidemics [55]. In the 2009/2010 budget, the government spent R4.5 billion towards the epidemics. The expenditure was increased to R8.4 billion in the 2010/2011 budget. The increase was done to expand the ART, scale up prevention of mother-to-child transmission programmes, promote HIV and TB treatment integration, and increase investments in HIV prevention.

2.4.3 Other Possible Solutions to the Challenges

Challenges such as the HIV/AIDS pandemic can be alleviated, to some extent, if the population is provided with information on how they can contract HIV and take care of themselves in case they have contracted it or are suffering from AIDS. Mayonsi *et al.* [55] posits that a big obstacle to HIV prevention and treatment is poor knowledge or denial about HIV status and associated risks. If information on the HIV/AIDS pandemic is made available to most people, it is possible that they can be encouraged to test for HIV. In the case they test negative, they can know how to take care of themselves and avoid contracting it. In the case they test positive, they will be knowledgeable about what to do to ensure that they live longer and do not infect others.

2.5 Initiatives to Provide Health Information to the Public in South Africa

The previous section discussed some of the health challenges that South Africa is facing and the steps taken by the government of South Africa to alleviate the challenges. This Section discusses efforts to provide the citizens with health information. The aims are to learn how others did or are doing it and to show that the work reported in this thesis

was not done in isolation but as part of larger work to provide health information to the people of South Africa.

2.5.1 Project Masiluleke

Project Masiluleke was a signature project of the PopTech Accelerator in collaboration with iTeach, Praekelt Foundation, Frog Design, MTN South Africa, Nokia Siemens Networks, and the Geographic Society [67]. The project was using mobile technology to sensitise people about HIV/AIDS and TB in South Africa. The aim of the project was "to raise widespread public awareness about how to access help; move people to take action, resulting in their getting tested for HIV and TB; get those who test positive into treatment and help them adhere to effective individualised treatment plans that will extend their lives and reduce the human, community, and economic losses associated with what would otherwise be certain and untimely death" [67, p. 2]. The project took advantage of the ubiquity of mobile phones in South Africa to reach as many South Africans as possible with HIV/AIDS and TB messages. It emphasised reaching out to those who are difficult to reach including men, youth, and those living in the rural areas with limited access to healthcare information and services.

The project proposed mobile solutions that interact directly with end-users through Short Message Service (SMS) messages. One solution was using the unused space of the "Please Call Me" (PCM) text messages, a special, free form of SMS text widely used in South Africa, to send HIV/AIDS and TB messages. The text messages informed the users of where they can obtain accurate healthcare information and referral to regional healthcare centres capable of providing voluntary HIV testing and counselling, TB screening and antiretoviral therapies, and anti-TB medication. The messages were developed in local languages to ensure that many users understood them. The campaign using this solution started on 1 October, 2008. Early results showed that 1 million PCM messages were sent every day and the number of calls to the AIDS helpline increased by 300% [67]. Figure 2.1 is a screen shot of a sample of the messages that were sent as part of the sensitisation efforts of the project.

The success of the solution was measured by the number of SMSs that were sent everyday and the number of calls to the AIDS helpline. This method of measuring success has a number of flaws. It is possible that some of recipients received messages that were in a language they did not understand or the messages were irrelevant to them. For example, the message in Figure 2.1 is only relevant to an HIV positive person who is mistreated

Figure 2.1: HIV/AIDS Messages sent in the unused PCM space. Source: [67].

by family and friends. If another person in a different situation receives the message, it is more likely that he or she would ignore it, unless he or she knows someone who is mistreated by family and friends who would benefit from such assistance. Another reason that might make the recipients ignore the messages is that PCM text messages are sent with the aim of requesting the recipient to call the sender, which might be all that most recipients are only interested in. This is supported by Davis [25], who argues that the challenge of communicating generic messages is that the recipients disregard particulars of the message that do not apply to them and focus only on the information that is personally applicable.

2.5.2 LoveLife

LoveLife is a health intervention programme that is aimed at preventing the transmission of HIV among the youth in South Africa [52]. The programme implements a variety of community and outreach support projects for young people. The main aim of the programme is to disseminate information about healthy sexuality, positive lifestyle, arts and culture, and family health. It has a number of projects among which is Love4Life,

a health sexuality project that targets the youth and provides them with relevant and accurate information about their sexual and reproductive health. The target youth are those between the ages of 12 and 19. The project targets schools and LoveLife community hubs (such as LoveLife Y-Centres, community-based organisations and youth friendly clinics) and is implemented throughout South Africa. Most of the projects by LoveLife are targeted and use age to segment the target population. This is the main difference from Project Masiluleke in addition to not focusing on the use of technology only.

2.5.3 MomConnect

MomConnect is a messaging service that provides pregnant women with vital antenatal and postnatal information for free [5]. The service was launched by the Minister of Health, Dr. Aaron Motsoaledi on 21st August, 2014. Pregnant women are registered with the service to receive stage-based, personalised SMS messages (A pregnant woman receives messages depending on her stage of pregnancy). After delivery, mothers still receive SMSs for up to a year. The messages they receive advise them on how to take care of their babies by recommending exclusive breastfeeding, encouraging mothers to go for immunisation and follow family planning methods, and reminding them of check-up periods at a health facility. The messages are delivered in the language of their choice. Available languages are Afrikaans, English, Zulu, Sotho, Xhosa, and Setswana (see Figure 2.2). The women are registered with the NDoH Pregnancy Registry by dialing *134*550# on their mobile phones or by a health worker at a government antenatal clinic. In the case where a woman has registered herself, she still needs to go to an antenatal clinic to complete the registration and be examined by a nurse. The women can also send free SMSs to the NDoH, either to compliment or complain about services provided at a particular public healthcare facility in South Africa.

Figure 2.2: MomConnect mobile messaging service [68].

2.5.4 Analysis of the Projects

The Project Masiluleke has a positive and a negative side. The positive side is that many people were reached with health messages free of charge. The negative side, as discussed, is that the impact of a message depends on whether it is of interest to the recipient and in a language he or she can understand. This is because the target population is not well-defined, which renders some messages not useful to the recipients. LoveLife and MomConnect projects improve on the shortfalls of the project Masiluleke by setting a well-defined target population. This ensures that the recipients receive only messages that are relevant to them.

2.6 Living Labs

According to the Directorate-General for the Information Society and Media of the European Commission [28], a Living Lab is a user-driven, open innovation ecosystem that is conceived as a partnership of businesses, citizens, and government to enable users to take an active role in research, development, and the innovation process. Budweg *et al.* [16, p. 595], defines a Living Lab as an 'environment of user-driven, open innovation resulting in the development and implementation of concrete innovation projects through a process of experimentation and evaluation'. Common in the definitions is that Living Labs are user-driven, open innovations involving several players (businesses, citizens, and government). Figure 2.3 is a graphical representation of the technology adoption cycle of Living Labs. The Living Labs are driven by two main ideas, which are, involving users as co-creators on equal grounds with the rest of the participants and experimentation in real-world settings [3]. They bridge the gap between technology ideation and development on the one hand and market entry and fulfillment on the other [28]. They provide a demand-driven 'concurrent innovation' approach by iteratively engaging all stakeholders in all phases of development with the user as the driver of the innovation process. It is believed that the user-driven innovation method can significantly improve the efficiency of the innovation process and increase the update of research and development results.

2.7 Siyakhula Living Lab Project

As introduced in Section 1.3, the Siyakhula Living Lab (SLL) is multi-stakeholder operation that consists of the Telkom Centers of Excellence at Rhodes University and the

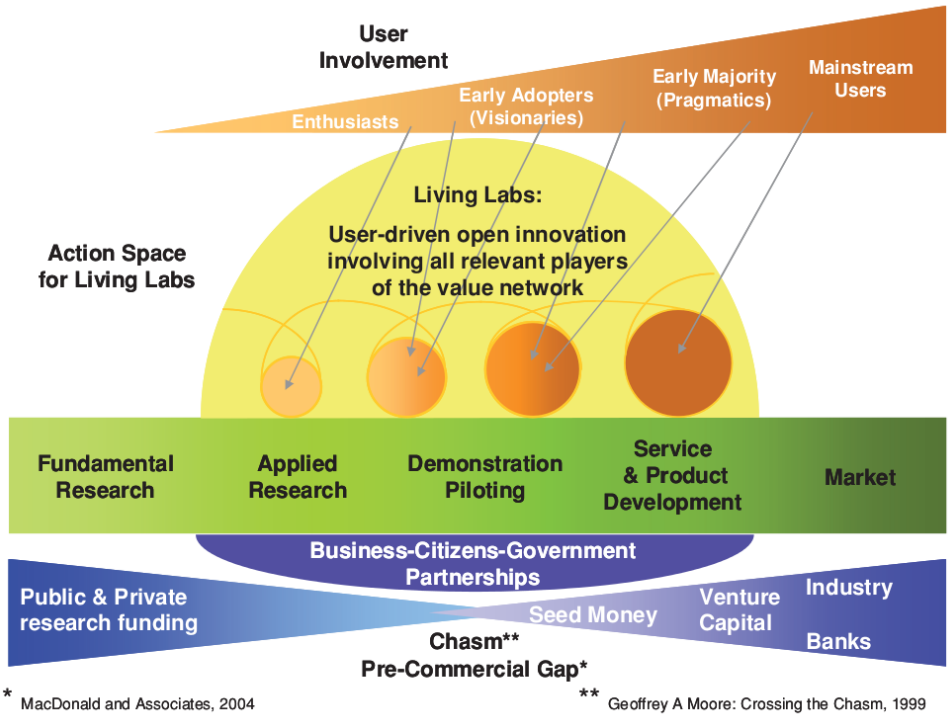

Figure 2.3: Action space for Living Labs along the technology adoption cycle [28].

University of Fort Hare, sponsors from provincial governments and the industry, and marginalized communities that facilitate user-driven innovations in the Information and Communications Technology for Development (ICTD) domain. As reported by Dugmore [29], its organised along the lines of the emerging Research Development and Innovation (RDI) Living Lab methodology, which is based on the principle of co-creation of solutions with empowered users [34].

2.7.1 Dwesa

As discussed in Section 1.3, Dwesa was the first deep rural site the SLL installed a community lab. It is representative of rural areas in South Africa and other African countries [24]. Just like most rural areas in developing countries, it faces challenges of lack of electricity, under-developed ICT infrastructure and poor roads. Most of the households in the area are yet to be connected to the national electricity grid. In most villages, the most likely buildings to be connected to electricity are schools and health facilities. The road that connects Dwesa to the nearest town (Willowville) is surfaced with gravel. It is hardly passable during the rainy season due to flooding of the Mbashe and Nqabara rivers. Most basic services are not available in the area except for basic education and

primary level healthcare services. To access other services, people travel to Willowville, which is at a distance of over 40 km.

2.7.2 ICT Infrastructure and Training in Dwesa

The SLLs are setup at schools to enable community members to access computers with Internet connection. The schools were selected because they have electricity, are fairly secure and able to host ICT infrastructure, and are centres of knowledge generation and dissemination. As of the year 2012, a total of 17 communities in Dwesa were interconnected wirelessly through the labs [29]. Students from Rhodes University, the University of Fort Hare, and other research institutions conducting research in the field of ICT for Development (ICTD) travel frequently to Dwesa, where among other activities they train community members to use computers and access the Internet and engage them in the development of software applications for use in poor areas. According to Dugmore [29], the active user base of the SLL in 2012 was approximately 200 community members and 4500 learners.

2.7.3 TeleWeaver

As discussed in Section 1.3, TeleWeaver is a software integration platform that was developed by Reed House Systems. The platform was customised to fit environments where hardware and software resources are limited [69]. The idea to develop it followed the need to address the challenge researchers were facing of hosting software developed for poor areas as individual and disconnected services [36]. The aim of TeleWeaver is to provide a common or single environment to host the services so that they can interact and interoperate (share application data and functions), which can help reduce the development cycle of applications.

2.7.4 TeleWeaver Business Model

TeleWeaver fits into a business model developed to generate revenue to support ICT infrastructure in poor areas. Figure 2.4 is a graphical representation of the model, which works as follows:

- an entity (for example a municipality) sets up a Digital Access Node (DAN) in a marginalised area;
- the entity procures a license for TeleWeaver from Reed House Systems, which installs it with startup applications at the DAN;
- the entity opens up TeleWeaver for use by other organisations (both public and private) that have an interest to provide services to communities in the area - the organisations can provide the services through applications available in TeleWeaver or can request Reed House Systems to develop new applications based on their specifications; and
- the organisations are billed for using the applications and the revenue generated is used to support the development and upkeep of ICT infrastructure at the DAN.

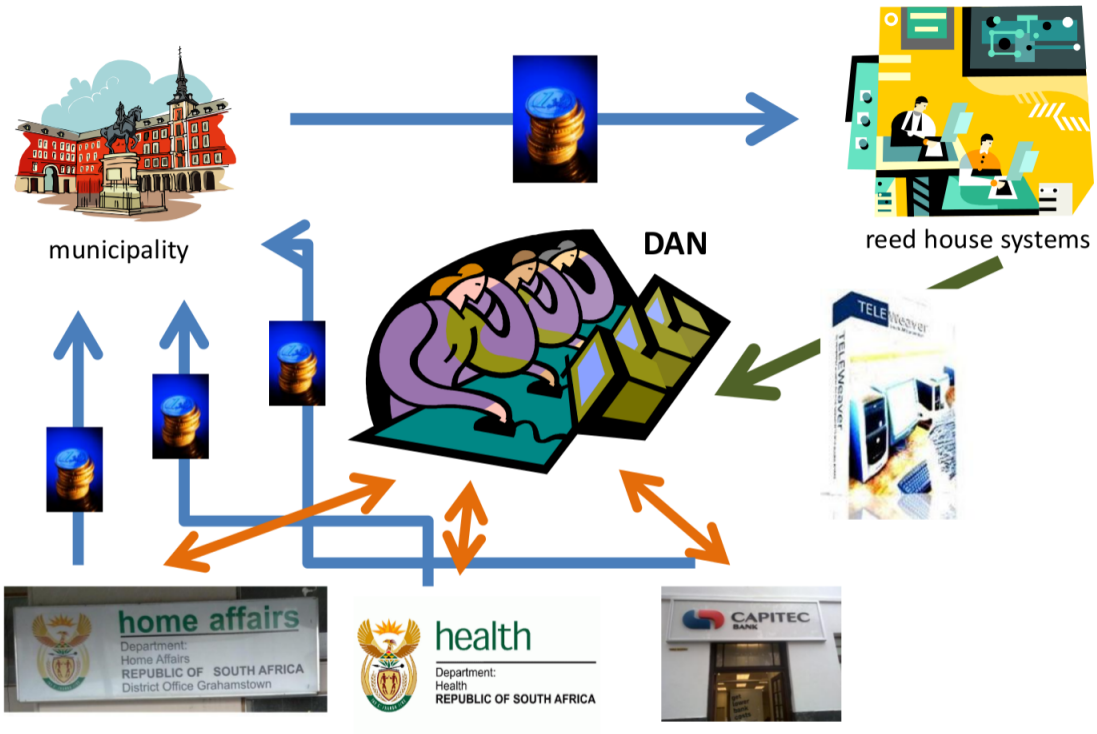

Figure 2.4: TeleWeaver Business Model [79].

The license for TeleWeaver includes pre-agreed contracts with organisations with an interest to provide services to people living in poor areas. This ensures that revenue generation starts once the users start accessing and using services in TeleWeaver.

2.7.5 TeleWeaver Applications

A new instance of TeleWeaver is installed together with applications that provide common functions required by most applications. Some of the applications are discussed in the following subsections.

2.7.5.1 Profile Application

The profile application manages users' profile information and their authentication to use hosted applications. It implements a single-sign-on to enable users to access all the applications in TeleWeaver at single login. A profile of a user includes as much of a user's information as possible that can be used for other purposes such as targeted advertising, message targeting (discussed in Sub-section 3.3.3.2) and message tailoring (discussed in Sub-section 3.3.3.3).

2.7.5.2 Billing Application

The billing application is for billing organisations using applications in TeleWeaver to provide services to people in poor areas. The billing information is generated depending on the type of service. For example, billing information is generated when a user views a an awareness campaign message in the e-health system that was developed as part of the research discussed in this thesis.

2.7.5.3 Message Dissemination Application

The message dissemination application is designed to push messages to users to attract their attention to access a specific service. It handles pushing of messages for all the applications in TeleWeaver that require the function. The messages are delivered as pop-up message boxes. Users have the option to ignore the messages, or to accept them in order to access more information.

2.8 Financing the Development of ICT in Poor Areas

There are significant challenges to finance the delivery of efficient ICT infrastructure in poor areas in both rich and poor countries [84]. To alleviate the challenges, there

were calls to create an international fund to support relevant ICT interventions. These calls did not yield enough funds and this prompted countries and organisations to find alternative means of financing their own interventions. A number of countries including Nigeria, Kenya, South Africa, and Uganda created their own Universal Service Funds to drive the development of new ICT interventions. In 2007, Nigeria enacted a law that established the National Information Technology Development Fund that sought to compel banks, insurance companies, and ICT companies to dedicate 1% of their profits to the fund.[1] The Kenya Information Communications Act of 2009 [2], under Section 84J, created a Universal Service Fund to support widespread access to ICT for capacity building and innovation. South Africa was one of the first countries in Africa to create such a fund. As discussed by Attwood *et al.* [6], the Universal Service and Access Agency of South Africa (USAASA), which is under the Department of Communications, created a Universal Service and Access Fund aimed at extending IT services to the most in need. By 2000, USAASA had helped to establish 65 telecentres containing between one and four computers, telephone lines and Internet access devices, in mainly rural, disadvantaged areas. By 2001, 32% of the telecentres surveyed were not operating and only 8% offered access to the Internet. One reasons for their failure was the lack of financial sustainability plans for the ICT infrastructure.

The failure of telecentres established by USAASA calls for alternative means to finance ICT investments in poor areas to ensure that they are sustainable. The model developed within the SLL to finance the sustainability of ICT infrastructure in poor areas is both innovative and unique and may help to solve some of the problems.

2.9 Summary

This chapter discussed the South African public health system, how people use the Internet in South Africa, work related to that done as part of the research, how the SLLs are bridging the digital divide between those with and without Internet access in South Africa and how countries are financing the upkeep of ICT infrastructure in poor areas. The use of the Internet in South Africa indicates an increasing number of users, the majority of new users being from the low income category. This makes it a potential platform to deliver health interventions to the public, including in poor areas. The failure of projects aimed

[1] http://www.researchictafrica.net/countries/nigeria/National_Information_Technoloyg_Development_Agency_Act_2007.pdf

[2] http://www.ca.go.ke/images/downloads/sector_legislation/Kenya%20Information%20Communications%20Act.pdf

at bridging the digital divide indicates the need to find alternative means to support the development and upkeep of ICT infrastructure in poor areas. There is an opportunity within the SLL to test a model that is designed to generate revenue to support ICT infrastructure in poor areas. The next chapter discusses the background literature related to the research.

Chapter 3

Background Literature

3.1 Introduction

The previous chapter analysed and discussed the South African public health system, how people use the Internet in South Africa, initiatives to provide the general public with health information and how countries are financing the upkeep of ICT infrastructure in poor areas. This chapter presents the background literature to the research. It includes a presentation of health communication, Consumer Health Informatics (CHI), Social Marketing, and information and data gathering methods.

3.2 Information Communication to Consumers

This section discusses how information is communicated from producers to consumers. The aim is to provide a picture of how it is done and the factors that determine its success or failure. At the end of the section, a campaign is discussed as an example of communication.

3.2.1 Communication Systems

Communication systems are formal or informal structures that are used by organisations to support their communication needs [21]. A good communication system ensures

that the target audience has timely access to relevant information for informed decision-making. According to Kreuter and McClure [48], the fundamental factors of a communication system are the message source, content, media delivery channel, receiver of the message, and destination of the message. It is therefore important to select and use credible sources of information, appropriate media delivery channels, and well-researched communication strategies to ensure that communication is successful.

3.2.2 Communication Channels

A communication channel is a "virtual pipe" along which messages travel from producers to consumers [21]. According to Kreuter and McClure [48], for communication to happen, at the basic level, the target audience must have access to a communication channel. Every channel has its own unique attributes such as sensory appeal, level of interactivity, and the expected number of people that can be reached. The appropriateness, credibility, accessibility, and feasibility (e.g cost) of a channel should therefore be considered to ensure that communication is effective [21, 48, 62].

3.2.3 Communication Types

There are two types of communication which are asynchronous and synchronous communication.

3.2.3.1 Asynchronous Communication

Asynchronous communication is a form of communication that does not happen in real time [78]. There is a time lag between when a message is sent and when a recipient acts upon it. Some examples of this type of communication method include email, voicemail, and some forms of video communications. This type of communication is flexible because there is less pressure on a recipient to act on a message. It is appropriate in situations where urgent action on a message is not required by the recipient. The disadvantage is that it can render messages outdated if they are not attended to in time.

3.2.3.2 Synchronous Communication

Synchronous communication is a form of communication that happens in real time [78]. It requires simultaneous participation of both the sender and recipient of a message. Some examples of this type of communication include telephone conversations and online chat sessions. The advantage of this type of communication is that a message is acted upon immediately by the recipient. It is more applicable in situations where immediate feedback is required. Its disadvantage is that it requires the sender and the recipient of a message to be present for communication to happen. If the communication is mediated by technology, the technology should be reliable enough to support real-time interaction.

3.2.4 Communication Tools

There are several tools that can be used to disseminate health messages to the target audience. According to Oak Ridge Institute for Science and education [62], an institute under the U.S. Department of Energy that among other activities specialises in audience and multimedia-based approaches to health communication, some of the communication tools include briefings, mailing, exhibits, fact sheets, and newsletters.

3.2.4.1 Briefings

Briefings are commonly used to notify the audience of any developments. Some of the developments include an introduction of an organisation, explaining its role and work process, and presentations of results of studies.

3.2.4.2 Mailing

Mailing involves sending information by e-mail to key contacts and concerned and involved members. This tool disseminates information quickly and easily in writing. It is useful in situations where the information communicated is easy to understand.

3.2.4.3 Exhibits

Exhibits are visual displays such as maps, charts, diagrams, or photographs that help to illustrate the messages that are communicated. If they are well presented, they can

make technical information easily understandable to lay persons. Their disadvantage is that they only support one-way communication and do not provide an opportunity for feedback if they are used as the only communication tool.

3.2.4.4 Fact Sheets

A fact sheet is a brief report summarising messages or information that is communicated. It can be used to introduce an organisation, explain particular health risks, guide people in precautionary health actions, announce new findings, etc. The limitation is that it is used for one-way communication only. In order to make the technical information easily understandable to the target audience, it requires careful design and presentation.

3.2.4.5 Newsletters

A newsletter is a publication that is circulated to subscribers at regular intervals [58]. It can be used to inform the target audience of activities, findings, health precautions, and other information concerning health. The advantage of this tool is that recipients can keep a newsletter and refer to it any time. Its disadvantage is that it is mostly used for one way communication and does not allow interaction.

3.2.5 Communication Campaigns

According to Snyder [75, p. S32], a communication campaign is 'an organised communication activity, directed at a particular population for a particular period of time, to achieve a particular goal'. The goal of campaigns is to promote behavioural change in the target audience. Some of the behaviours that campaigns have tried to promote include the use of seat belts, dietary change, physical exercise, smoking cessation, substance use prevention, family planning, use of health services, and testing and screening for diseases. Some of the tools that are used to conduct campaigns include posters, handouts, public announcements, and discussion groups. Snyder [75] recommends the following to ensure that campaigns are successful:

- campaign messages should be delivered multiple times in a given period of time to increase the chances of users acting on them - the greater the frequency of a campaign (number of times a person is reached by a campaign), the greater the likelihood that campaign messages will be remembered more accurately;

- message communication should use multiple channels (such as television, radio, pamphlets, outreach workers, clinic staff, and special events) at the same time to increase exposure - the choice of message delivery channels should ensure that a large percentage of the target population is reached; and
- campaigns should emphasise information that is new to the target audience and essential for their behaviour to be changed - for example, if nearly all people are aware of the need for fibre intake but are unsure of how to get it, a campaign should focus on telling them which food products are high in fibre rather than on the need to take it.

Presentation of campaign messages determines whether the target recipients accept or reject them. As discussed by Snyder [75], messages should be designed to capture attention in order to increase the chances of the target recipients acting on them. They should be simple to ensure that the target recipients understand them.

3.3 Health Communication

The previous section discussed communication systems, channels, tools, and how campaigns should be conducted to ensure that they achieve the goal of changing behaviours of the target population. This section continues with the discussion of communication by looking at health communication. Health communication is the use of communication strategies to inform, influence, and motivate individual, institutional, and community decisions that enhance health [59, 66]. Some of its aims are to communicate steps to be taken when consumers are faced with a particular health situation and to provide comparative information on alternative health actions [44]. An effective health communication can lead to a better-informed population that is aware of diseases and conditions, and common health risks; understands how to prevent or mitigate health risks; and is able to identify avenues to obtain help or where and how to access health services. For example, those who understand the importance of eating a nutritious diet, exercising regularly, quitting smoking, limiting alcohol consumption, stopping drug abuse, and practicing safe sex, take necessary steps to ensure that they stay healthy.

Health information can be communicated through different media channels such as radio, television, newspapers, brochures, posters, billboards, the Internet, and face-to-face communication between healthcare providers and consumers. The National Cancer Institute

of America [59] posits that well-researched strategies that shape messages and determine communication channels to deliver health messages to the intended audiences are key to successful health communication and can help achieve the following:

- increase the intended audience's knowledge and awareness of a health issue, problem, or solution;
- influence perceptions, beliefs, and attitudes that may change social norms;
- increase demand or support for health services; and
- cause a sustained change in which an individual adopts and maintains a new health behaviour.

3.3.1 Stages in Health Communication

According to the Open University [64], health communication must pass through several stages in order to achieve the desired changes in behaviour. Figure figure 3.1 shows the stages, some of which are discussed in the following sections.

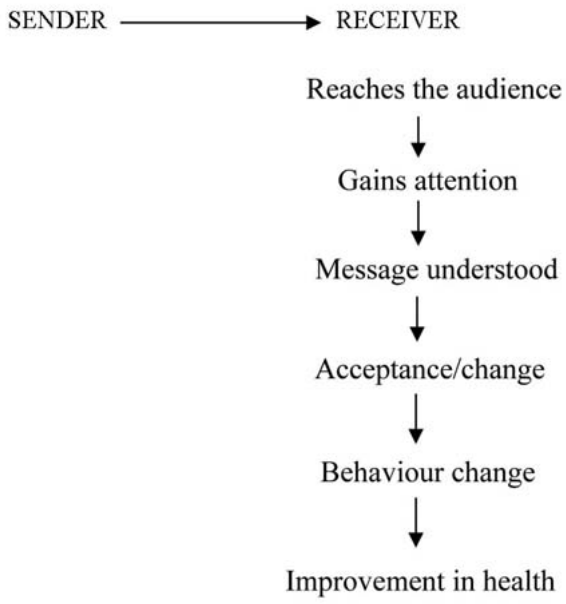

Figure 3.1: Stages in Health Communication [64].

3.3.1.1 Reaching the Audience

For health communication to be effective, it must reach the intended audience. The communication should avoid 'preaching to the converted'. This means that messages to influence a particular behaviour should be not be targeted at those practicing the intended behaviour. For example, a poster encouraging pregnant women to go for antenatal services might have less impact if it is posted at antenatal clinics because it is targeting those who are already motivated and practicing the intended behaviour. It is therefore important to focus on channels that will reach out to those who are not practicing the intended behaviour.

3.3.1.2 Gaining the Target Audience's Attention

Attracting the attention of the target audience is key to successful communication. Only when their attention is attracted, will they pay attention and engage with a communication message.

3.3.1.3 Understanding the Message

When people receive messages, they try to understand and interpret them. There is a chance that different people might interpret the same message differently, which might be different from the meaning intended by the sender. It is therefore necessary to keep the messages simple so that users can easily understand them. If the intended audience are lay users, technical terms should be avoided or should be explained in simple language. Communicating too much information should be avoided because it can be overwhelming, leading to information overload.

3.3.2 Health Communication Through the Internet

Books, encyclopaedias, and brochures were among the first forms of available mass media to share health information [45]. Other media that followed include radio, television, newspapers, magazines, and the Internet. The popularity of the Internet among people of all ages and different economic statuses makes it a good platform to deliver public health interventions. Internet interventions are systematic treatment/prevention programmes that are developed to address one or more determinants of health and are delivered to

users through the Internet [11]. These types of interventions have been growing in popularity as observed from the number of studies that used the Internet platform. Some of the studies include the prevention of unplanned pregnancies and spread of sexually transmitted diseases and Human Immunodeficiency Virus (HIV) in rural adolescents at two public high schools in rural Appalachia [71]; and HIV prevention targeting males who have sex with males (MSM) [14]. Most of the Internet intervention studies concluded that the Internet is efficacious as a platform for conducting public health interventions [11, 14].

3.3.2.1 Advantages

Some of the advantages of Internet-based health interventions are:

- they provide a cost-effective way of reaching a large number of people - the ubiquity of the Internet enables users to access health information at a time and place of their convenience;
- users have access to regularly updated and upgraded health content as compared to paper-based interventions; and
- the interventions can be structured to provide users with tailored health information to enhance relevance.

3.3.2.2 Disadvantage

The disadvantage is that the digital divide hinders those who do not have access to the Internet or those with no knowledge of how to use the Internet from accessing health information.

3.3.3 Health Communication Strategies

Health messages developed to influence behaviour change can be directed at individuals, organisations, communities, and a society as a whole. Different audiences call for different strategies for the messages to have an impact. Some strategies work by directing messages at the target audience, while others at people who may have an influence on the target audience [75]. Direct communication assumes that the target audience has the

capacity to change their own behaviours. If it is unlikely that direct communication will produce the desired effects, targeting those who have influence on the target audience is the most appropriate. For example, it might be easier to target those responsible for the procurement and preparation of food to influence the diet of consumers.

As discussed by Hawkins *et al.* [39], health communication strategies are grouped into three distinct categories which are discussed in the following subsections.

3.3.3.1 Mass Communication

This is one of the earliest strategies to be used in health communication. During the first half of the 20th century, it was called health publicity [25]. Its aim was to disseminate general health information and threat of diseases to the general public. In this strategy, relatively large, heterogeneous audiences receive identical messages [39]. It is based on a 'one-size-fits-all', sometimes called 'all things to all people' approach [25, 37]. It is more appropriate for creating awareness and reaching out to a large audience [18]. Some of the media channels that are used with this strategy are newspapers, magazines, radio, brochures, and television. The strategy is based on the belief that quality of life of the public would improve if people understand health issues [25]. Its advantage is that a large audience can be reached and informed using the same messages. This makes it cost effective as compared to other strategies because less time is spent segmenting the audience and developing communication materials for the segments. Its disadvantage is that it fails to consider specific characteristics of individuals. It does not consider the finer details that vary from person to person and that uniquely affect an individual's health-related decisions and behaviours [50]. This approach is not effective in producing widespread behaviour changes because messages can be difficult to understand and remember and individuals may find them not directly applicable to their situation [39, 25].

3.3.3.2 Targeted Communication

Targeted interventions involve the development and use of a single intervention approach for a defined population subgroup [48, 51]. The subgroups are derived through a practice called audience segmentation. Audience segmentation is the process of dividing a large and heterogeneous population into smaller, more homogeneous subgroups [48]. Some of the factors that are used to segment a population include their demographics, behaviour, psychosocial, geography, culture, and risk factor characteristics. The practice divides

the audience into more defined, homogeneous subgroups that are similar internally but different from each other [39, 60]. The assumption is that sufficient homogeneity exists among the members of the subgroup to warrant using a single intervention approach [25]. A subgroup can be composed of the target audience or those who have an influence on the target audience.

In this strategy, a single set of messages is developed and used for all the members of a subgroup. The practice of designing messages that resonate with a particular group or subgroup is called message targeting [60]. It is based on the belief that when an audience has more similarities, the members will react similarly to the same messages. The contents of the messages and their presentation are based on an the understanding of the needs and concerns of the target segment [48]. This type of communication is advantageous because it reduces the effort necessary to identify information relevant to an individual within the targeted subgroup [25]. Its disadvantage is that it cannot address variations between individuals on factors that are not used to segment a population.

3.3.3.3 Tailored Communication

Message tailoring is the practice of designing health messages that are relevant to an individual based on his or her unique characteristics [25, 60]. It uses an individual's needs, interests, and concerns to develop custom messages and materials that provide direct feedback to address his or her needs. It starts by assessing an individual based on a variety of characteristics that are relevant to the behaviour under study [60, 61] (see Figure figure 3.2). Based on the assessment, appropriate messages are retrieved, customised to fit the situation, and determine delivery channels [4, 39]. It is believed that messages that are customised to an individual are viewed as more 'personally relevant', are more likely to be read, remembered, and cognitively processed, and have a better chance of stimulating behaviour change [61]. Common behaviours that have been studied using message tailoring include smoking cessation, diet [38], mammography screening [60], weight loss [49], and self-management of type 2 diabetes [88].

Message tailoring mimics counselling sessions where questions are asked and feedback is given based on responses. It has been simplified by the advancement of computing and Internet communication technologies. When using computer-based technologies, information used to tailor messages is solicited from an individual or queried from existing records. Computer algorithms are then used with the information to retrieve appropriate messages from a message library to address a particular behaviour in an individual [51, 60]. The

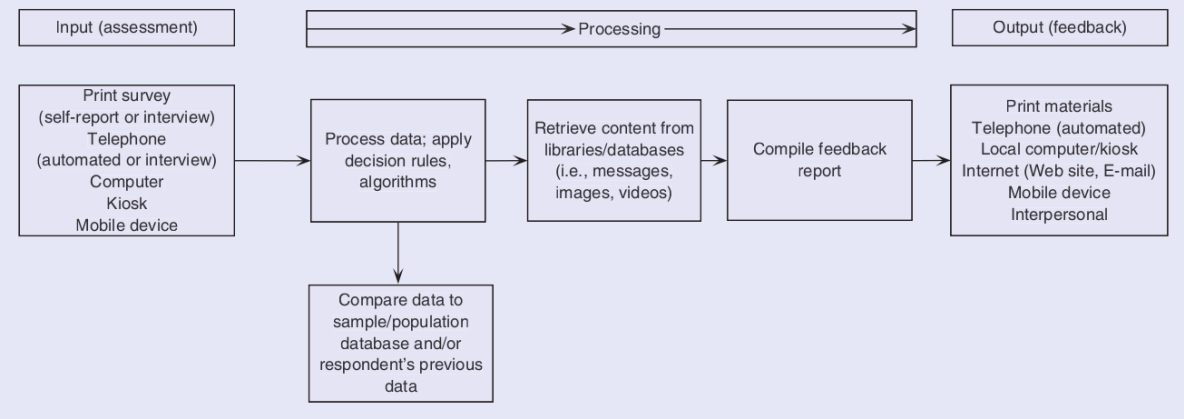

Figure 3.2: Overview of the process of message tailoring [61].

library can contain hundreds or even thousands of messages. Message tailoring should be used only when there is both considerable variation across the intended audience on important factors known to influence the outcome of interest and when the outcome of interest is more complex [50]. This is because producing content tailored to an individual is more expensive compared to a defined group or for everyone [37].

Most reviews have concluded that tailored messages are more effective at impacting health behaviour changes as compared to non-tailored or targeted messages [39, 60]. Some of the reasons are that it enhances the relevance of the information presented to a user and ultimately produces greater desired outcomes as compared to both targeted and undifferentiated mass communication [39]; reduces the burden of sifting through potentially non-relevant information imposed by generic materials [25]; and addresses the specific informational and behavioural needs of an individual [25]. However, in the study by Haerens *et al.* [38], most adolescents did not perceive tailored intervention messages as interesting, new, personally relevant, or correct. The reason for this could be the choice of the sample population. In the study, participants were recruited in the classrooms and the motivation to participate could have been low (as shown in the responses where most of the participants reported that they had not read the tailored feedback) compared to studies that recruited patients, as in the study by Weymann *et al.* [88] where individuals were motivated to see changes in their health. Additional information on message tailoring are discussed in Appendix B.

3.4 Consumer Health Informatics

The previous section discussed health communication. The discussion included the stages communication goes through, the communication that happens through the Internet, and the strategies that are used to communicate messages to achieve a specific objective. This section discusses Consumer Health Informatics (CHI), which is the field the research belongs to. The aim is to present how CHI impacts on the knowledge and understanding of health issues and well-being of non health professionals.

As discussed in the introduction to this thesis, historically, medical practitioners (or physicians) championed health information communication and education to consumers [41, 51]. Information flow was usually unidirectional, from practitioners to consumers, and was communicated on a "need to know" basis. This has changed with the advent of the CHI, which is the study, development, and implementation of computer applications that facilitate consumer access to health information [4, 45, 51]. The aim of CHI is to provide health information to consumers to promote self-care, health behaviours, peer information exchange, and social support, which ultimately foster better public health outcomes [40, 46]. It analyses health information needs of consumers, studies and implements methods to enable them to access and understand health information, and models and integrates consumers' preferences into medical information systems or applications [4, 54]. It is believed that consumers can be empowered to participate in decision-making on issues concerning their health, if consumer health applications are developed based on their needs [31].

The advent of the Internet has simplified the task of building health applications that are accessible to consumers. It has changed the mechanics and economics of mass delivery of health information to consumers, leading to an increase in the number of consumers and the modalities in which they seek health information [45]. Consumers with Internet access now have continuous, on-demand access to health information, which was traditionally only accessible to medical practitioners and other health professionals. Consumer knowledge of health issues has increased, leading a new physician-consumer relationship that emphasises participation of both parties of the relationship [40]. In this relationship, consumers take an active role in their care by asking questions and indicating their preferences, and physicians provide their expert opinions. Decision-making is shared between a consumer and a physician. Physicians are now able to interact with consumers (patients and their immediate caregivers and healthy persons) and consumers are also able interact among themselves through the Internet. Some of the interactions that are possible through the Internet include the following:

- physicians diagnose patients from geographically remote locations using telemedicine;
- physicians provide advice to patients and caregivers to promote their self-care through clinical emails; and
- patients with similar conditions form online relationships and communities where they share stories, give each other advice, and provide each other the needed support. (**Note**: Their interactions are not moderated by physicians which encourages the members to freely contribute to the forums. Where inaccurate information is provided, other members provide corrections.)

In order for consumers to be able to use the information that is made available through computer applications and the Internet, they require significant knowledge and skills to use electronic media [45]. The other requirement is that they should have access to technology. Keselman *et al.* [45] posits that it is more likely that those in need of health information usually have no access to technology such as the Internet. For example, in poor areas most people are of low economic status and are unable to afford the costs of accessing the Internet and yet their sources of health information are limited due to their location. Another challenge is health literacy, which is the capacity of individuals to obtain, process, and understand basic health information and services for them to make informed decisions on their health [45]. In the case where consumers have access to unlimited health information, if they are health illiterate, it cannot be useful to them. These challenges can be addressed if consumers are provided access to technology equipment and are trained in how to use it and the information is developed in simple language.

3.5 Social Marketing

The previous section discussed how CHI has helped to improve consumer knowledge on health issues, and how it has shaped the relationship between health professionals and consumers. This section discusses how Social Marketing can help to achieve desired behaviour changes in the target audience.

Social Marketing is the application of commercial marketing technologies to the analysis, planning, execution, and evaluation of programmes designed to influence voluntary behaviour change in the target audience in order to improve their personal welfare and that of their communities [20]. It uses principles used in selling goods such as shoes, food items to convince people to change their behaviours. As posited by Grier and Bryant

[35], it facilitates acceptance, rejection, modification, abandonment, or maintenance of particular behaviours. Its defining features include exchange theory, audience segmentation, competition (behaviour options that compete with public health recommendations and services), market mix, consumer orientation, and continuous monitoring.

3.5.1 Social Marketing Principles

There are four basic principles of Social Marketing that were adopted from commercial marketing, known as the four Ps: product, price, place, and promotion.

- **Product**: This represents what is marketed. In social marketing, it refers to the behaviour you are trying to change [22]. For people to change their behaviours, they need to see the benefits associated with the desired behaviour change [35]. For an intervention to be successful, the product must provide a solution to the problems that consumers are facing, and it should offer them benefits they truly value. The benefits that appeal most to the target audience should be identified and incorporated in the design of the product.

- **Price**: This refers to the cost or sacrifice required to change behaviours [17, 35]. It encompasses intangible costs, such as diminished pleasure, embarrassment, and loss of time. A good social marketing plan should reduce the costs of behaviour change.

- **Place**: According to the California STD/HIV Prevention Training Center [17], place refers to where and how the target population can be reached. For example, distributing free or inexpensive condoms at schools, bars, or restrooms. Social marketing efforts should make it easier to change behaviour by making sure the necessary support is not only available but accessible. The less people need to go out of their way to make a change, the more likely they will change.

- **Promotion**: This is the way to notify the target population about intervention messages and it includes advertising [17]. For example, incorporating messages about a recommended behaviour change in all existing programmes in order to reinforce the messages at multiple levels. This needs determining of the communication channels and activities that will best reach the audience to promote the benefits of the desired behaviour [63].

3.5.2 What is Involved in Social Marketing

According to Community Tool Box [22], a service of the work group for community health and development at the University of Kansas, social marketing involves the following steps:

- identifying a behaviour to change;
- identifying the audience, which involves the use of audience segmentation to improve the impact of the intervention;
- identifying barriers to change;
- reducing the barriers to change, which involves planning ways to make it easier for the target audience to change behaviour;
- pretesting the idea on a small number of people and then modifying the plan according to the results; and
- publicising both the benefits of change and the efforts to make the change easier.

3.6 Information and Data Gathering

In order to determine the needs of the target audience, it is necessary to gather data from them. There are several methods that can be used to gather data such as surveys, structured and unstructured interviews, brainstorming, direct observation, and focus group discussions [19]. Each method has its own strengths and limitations with some more participatory than others. This thesis discusses the use of surveys as a data collection method. A survey uses a list of questions to gather information from individuals. It can be used to assess needs, gather baseline data for an outcome evaluation and forming initiative goals, and for process evaluation [19, 42]. Surveys can be conducted on a telephone, using paper-based or web-based questionnaires, or face-to-face interviews. The reason for discussing this method only is that it was found to be appropriate for collecting data using the system that was developed, which was based on the advantages discussed below.

3.6.1 Advantages of Surveys

Some of the advantages of surveys are:

- they can be used to reach a large audience;
- they provide a written record of responses that can be referred to later;
- they can be quick and anonymous;
- the use of specific questions helps to obtain a clear data set of results; and
- they simplify gathering of data and analysis of results.

3.6.2 Disadvantages of Surveys

Some of the disadvantages of surveys are as follows:

- they are less flexible in terms of responses (responses are limited to the questions and answers in the survey questionnaire);
- the prerequisite for (paper and web-based) surveys is that the respondents should be able to read and write and this limits the target population;
- analysis of responses might require knowledge of statistical methods;
- they tend to have a low response rate; and
- the questions can be misunderstood by the respondents.

3.7 Summary

This chapter reviewed literature related to the research. The review included CHI, health communication through the Internet, and communication strategies that are used to disseminate health information. While mass communication has the impact of reaching a large audience, targeted and tailored communication are superior because they ensure that messages disseminated to the target audience make an impact. The development of e-Health applications that are accessible to consumers through the Internet has led

to a shift in the consumer-health professional relationship. Consumers who have access to on-line health information are empowered to participate in decision-making on their health, which was not the case in the past. The next chapter reviews web applications that were developed to provide consumers with access to health information.

Chapter 4

Review of Health Websites

4.1 Introduction

The previous chapter reviewed background literature to the research. The review included the strategies that are used to ensure that health messages have an impact on the target audience. This chapter discusses a review of consumer health websites that was done to identify their common features, and select appropriate ones for the system that was developed as part of this research. The review excluded websites that are developed to manage personal health records (e.g. Microsoft Health Vault[1]) and specific health conditions over a period of time (e.g. applications to manage conditions such as diabetes and smoking cessation). From this point up to the end of the thesis, the system will be called HealthAware.

4.2 Common Features of Consumer Health Websites

There is a large number of health websites that are available both nationally and internationally. Some of them are developed to introduce the owning organisations, while others are developed to provide consumers with health information. This review includes both local and international websites. The local websites are those for the South Africa NDoH and six provincial departments of heath, Health-e,[2], and Health24.[3] The websites for

[1] https://www.healthvault.com
[2] http://www.health-e.org.za/
[3] http://www.health24.com/

the remaining three provincial departments were either inaccessible or do not exist. The international websites are those for the WebMD,[4] HealthFinder.gov,[5] FamilyDoctor.org,[6] National HIV and STD Testing Resources,[7] ConsumerHealthChoices,[8] Centers for Disease Control and Prevention,[9], and MedlinePlus.[10] The common functions that were found on the websites are discussed in the following subsections.

4.2.1 Publicity

The publicity function is used for awareness or promotion activities. Examples of promotions are hand washing, living a health lifestyle, and women's health. Though the websites investigated categorised publicity information, most of it was static (disseminated as pdf files). The function was considered useful for a consumer health application and was selected to be included in HealthAware. The function was renamed to *Campaigns*.

4.2.2 Events

The *Events* function enables consumers to access information on upcoming health events. The function was considered useful for a consumer health application and was selected and included in HealthAware.

4.2.3 News

The *News* function enables users to access latest news. Most websites that were investigated provide this function. Health-e website is exclusively for health news. The function was considered useful for a consumer health application and was selected to be included in HealthAware.

[4] http://www.webmd.com/
[5] http://healthfinder.gov/
[6] http://familydoctor.org/
[7] http://hivtest.cdc.gov/
[8] http://consumerhealthchoices.org/
[9] http://www.cdc.gov/
[10] http://www.nlm.nih.gov/medlineplus/

4.2.4 Disease and Conditions

The *Disease* and *Conditions* enables users to obtain information on different diseases and conditions. It is a common function in most websites. In some websites it is called *Health Topics* (see Figure 4.1 for MedlinePlus website showing a list of Health Topics). The function was considered useful for a consumer health application and was selected to be included in HealthAware. The function was renamed *Topics*.

Figure 4.1: MedlinePlus web interface for Health Topics

4.2.5 Drugs and Medications

The *Drugs* and *Medications* function enables consumers to access information on drugs and medications for different diseases and conditions. The function was considered useful for a consumer health application in order to promote self care. However, it was not considered necessary for HealthAware because it can encourage consumers to perform self-diagnosis and prescription, which might be risky if not properly done. The disease and conditions function was considered a good substitute to provide users with information on the steps to take for a particular disease or condition.

4.2.6 Ask An Expert

The *Ask an expert* function enables consumers to ask a health expert questions that concern their health. The questions and their responses are accessible to other consumers. The function was considered useful in order to save consumers' time to travel and see a health expert, who is most likely not available in poor areas. However, getting an expert specifically for the people in target areas can be a challenge. The function was therefore not selected for inclusion in HealthAware.

4.2.7 Multimedia

This function enables users to play multimedia files like online videos. A combination of audio and video can facilitate consumers' understanding of health topics. They can also be useful to those who cannot read if the audio is in a language they can understand. The function, however, is difficult to implement for HealthAware because of low Internet speed and intermittent connectivity in poor areas. The architecture of HealthAware, which is discussed in the next chapter provides more information on the reasons this function is a challenge.

4.2.8 Facilities

The *Facilities* function enables consumers to access information about health facilities (e.g. clinics, hospitals, and mobile services such as mobile clinics). The function was considered useful for a consumer health application and was selected to be included in HealthAware.

4.2.9 Services

The *Services* function enables consumers to obtain detailed information of health services offered by different facilities. Examples of services are antenatal care, contraception, termination of pregnancy, physiotherapy, and speech therapy. Most websites that were investigated provide this function. On the NDoH website, services are categorised into those for children, mothers, businesses, and those living with a disease or a condition. The services are either provided as a list or as links and there is no search function. Other websites such as HealthFinder.gov allow users to search for specific services. For example, a user can search for a dentist or a doctor (see Figure 4.2). The function was considered useful for a consumer health application and was selected to be included in HealthAware.

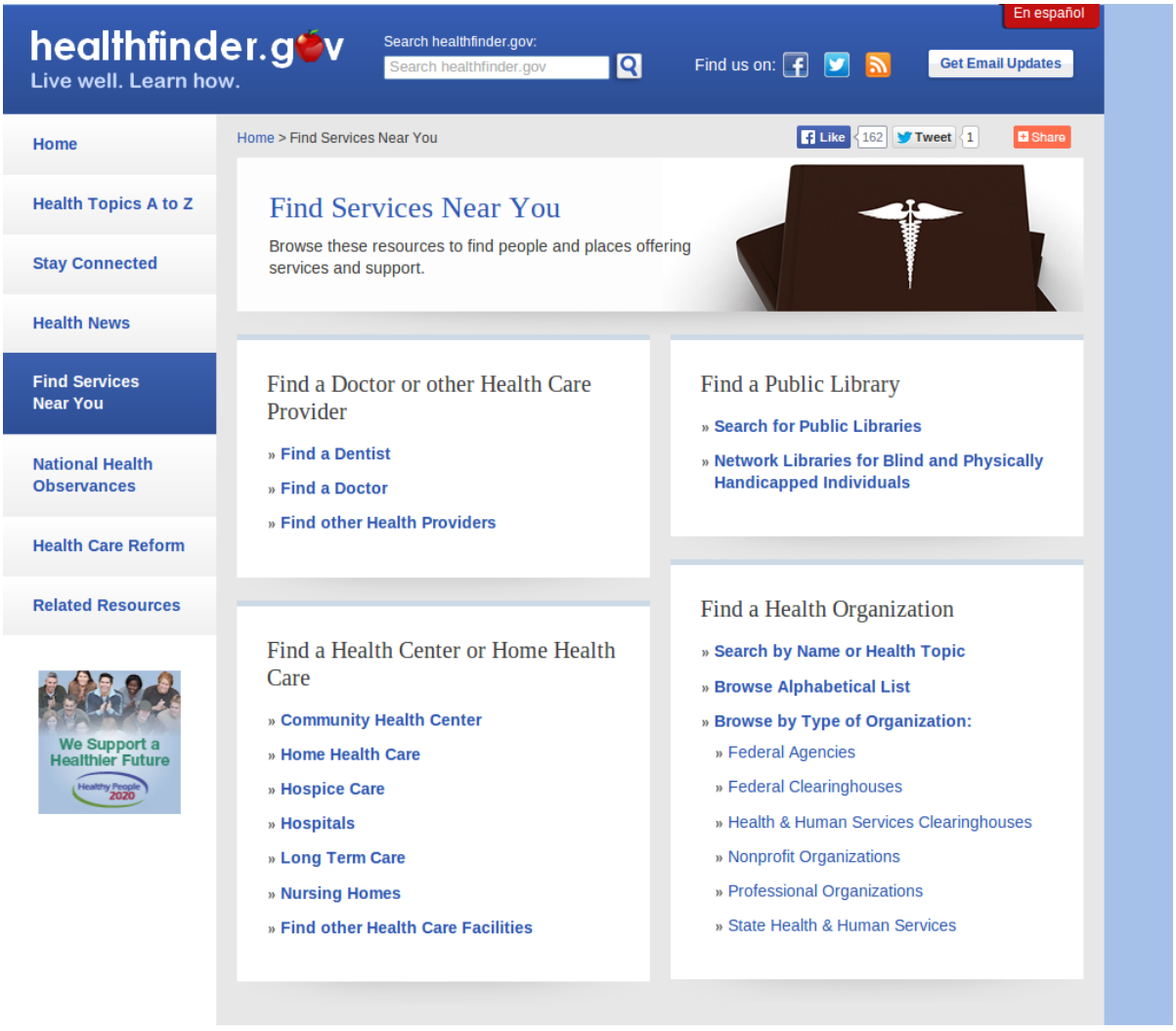

Figure 4.2: Healthfinder.gov website

4.3 Observations

Several observations were made during the review of the websites. The websites for the NDoH and the provincial departments were developed mainly to introduce the organisations even though some of their content can be used by consumers for informed decision-making. Health24 and Health-e websites were developed specifically to disseminate health information to consumers. Health24 provides information on a wide range of health topics, while Health-e is dedicated to health news. The content on all these websites is developed for general audience and is exclusively in English. The NDoH groups the information on health services into that for children, mothers, businesses, and those living with a disease or condition.

The international websites that were reviewed are dedicated to providing health information to consumers. The websites target consumers in the countries they were developed. For example, a search for a dentist on the HealthFinder.gov website yields information of dentists in the United States of America, which cannot be useful to South Africans with no intention to travel to the United States of America for medical attention. The HealthFinder.gov, Centers for Disease Control and Prevention, National HIV and STD Testing Resources, and FamilyDoctor.org websites enable consumers to access health information either in English or Spanish indicating the importance of providing content in multiple languages to accommodate consumers who understand different languages.

4.4 Summary

This chapter reviewed health websites and analysed their common functions in order to select appropriate ones for HealthAware. Out of the 9 functions that were analysed, 6 were selected to be included in HealthAware. The functions selected are *Campaigns*, for implementing sensitisation campaigns; *News and Events*, for the dissemination of latest health news and upcoming events; *Disease and Conditions*, to disseminate information on various diseases and conditions; *Facilities*, to enable consumers to search for different health facilities and link to the services they offer; and *Services*, to enable consumers to find information on different health services and link to the facilities that offer them. The functions that were excluded were considered either challenging to implement for the target environment or not useful to achieve the research objectives. The next chapter discusses the requirements and design of HealthAware.

Chapter 5

Requirements and Design

5.1 Introduction

The previous chapter reviewed websites that provide consumers with health information. The functions that were common in the websites were evaluated, and those that were considered appropriate for HealthAware were selected. The functions that were selected were *Campaigns, Events, News, Topics, Facilities*, and *Services*. This chapter discusses the requirements and design of HealthAware.

5.2 Requirements

This section discusses the general requirements of HealthAware in order to provide an overall picture of how it is meant to work. Specific requirements are discussed in sections 5.3 and 5.4.

As discussed in Section 1.5, the first objective to use HealthAware to facilitate health awareness campaigns carried out by health service organisations that target people in poor areas. Target users should access the awareness campaigns' information/messages through DANs in their areas. The campaigns should be targeted to ensure that the target audience access or receive only what is relevant to them. HealthAware should also include a function to collect data and visualise the results of the campaigns. To enable health organisations to collect data, the *Surveys* function that enables the organisations

to conduct surveys should be included. The function is appropriate for collecting data based on the advantages discussed in Section 3.6.

The second objective is to use HealthAware to generate revenue to support the development and upkeep of ICT infrastructure in poor areas. The revenue should be generated via the logging of the interactions of the target users with the HealthAware, from which billing data is generated according to the specific contractual agreements with health organisations.

Figure 5.1 is a graphical representation of the functions of HealthAware. From this point up to the end of the thesis, all references to 'awareness campaign' should be interpreted as campaign, news, or event and 'health information' should be interpreted as information for a campaign, survey, news, event, service, or facility.

5.2.1 System Architecture

HealthAware should comprise two independent components as shown in Figure 5.2. Component 1 should be hosted independent of TeleWeaver and should be used by health service organisations to create, manage, and publish health information to be accessible through component 2. Component 2 should be hosted on TeleWeaver and should be used by TeleWeaver users to access the health information created using Component 1.

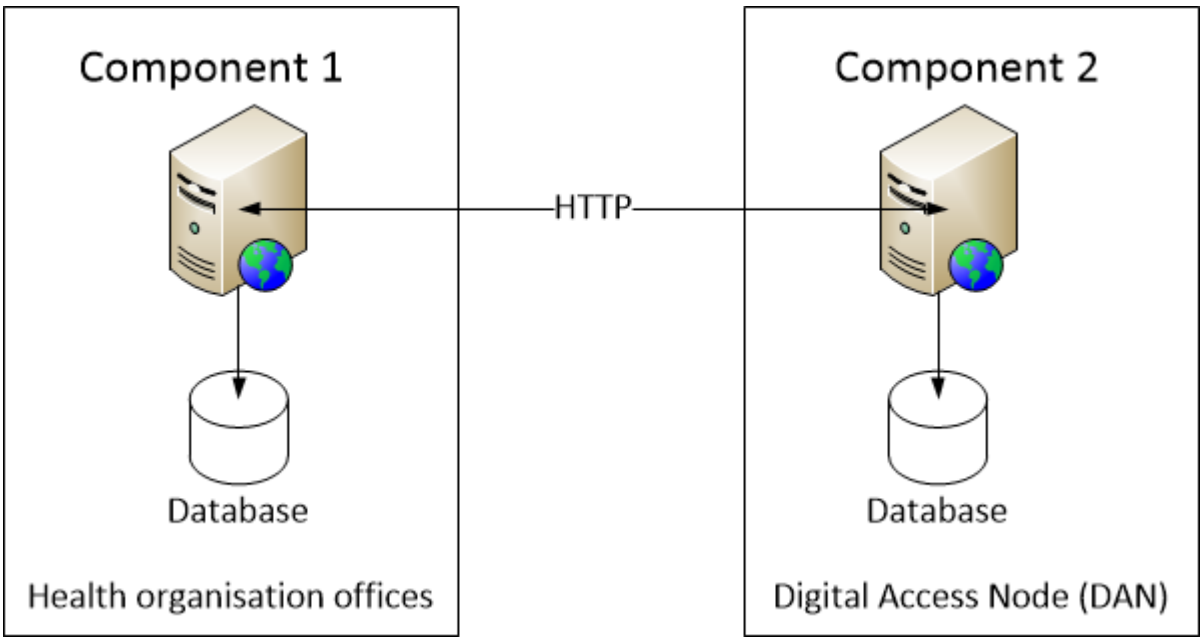

Figure 5.2: System Architecture

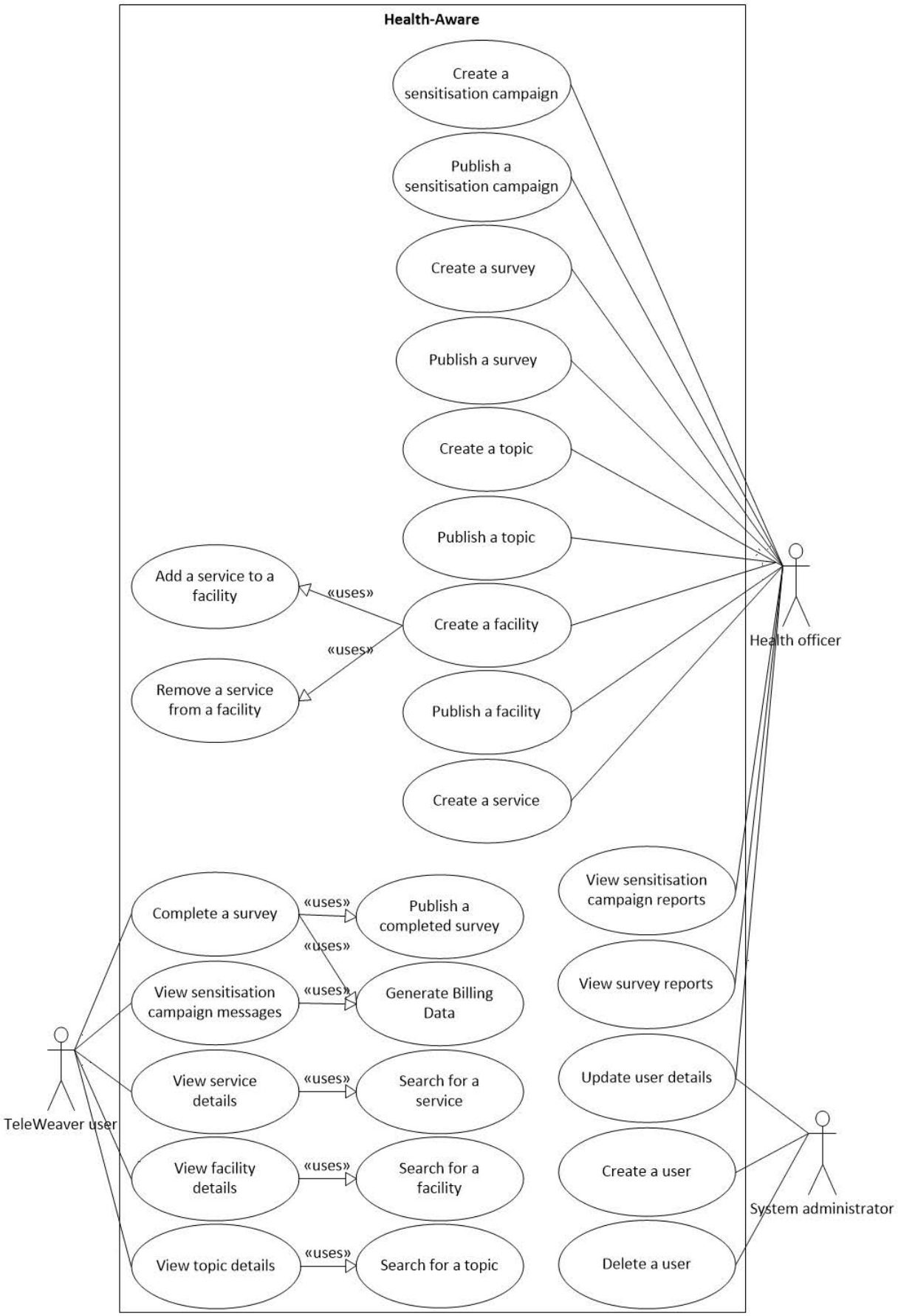

Figure 5.1: Overall use case diagram for HealthAware

The components should be web applications, each having an underlying database. Web applications are ideal because they are used independent of the operating system and could be easily deployed in the cloud in future, when the speed and stability of the Internet connectivity in the targeted areas improves. Health information created in one component should not be accessible to users of the other component, unless published. Publication of health information from a component should retrieve a copy of the information from the database of the component and post it to the other component where a record of the information should be created in the database. One major drawback of this approach is that it requires both components to be online when the information is published. To enable users to publish even when one of the components is offline, a store-and-forward technology should be used.

The advantage of using the proposed architecture is that it enables both components to be usable at all times despite the challenges of Internet connectivity in poor areas. In such areas, the Internet connectivity is, at best, of low speed and unstable. If both components are to be hosted on TeleWeaver, users from health service organisations cannot use Component 1 outside the DAN if TeleWeaver is not accessible online. It also requires having Component 1 in all instances of TeleWeaver and health service organisations to use a different instance of Component 1 to target a particular population. If both components are hosted on the cloud, users in poor areas cannot use Component 2 most of the time because of the challenges of Internet connectivity in their areas. The proposed architecture ensures that only one instance of Component 1 is available to service all health service organisations that uses the HealthAware and is accessible at all times through the Internet and that Component 2 is accessible at all times through the DANs.

5.2.2 Software Architecture

A software architecture organises or structures a system to represent a collection of components that accomplish a specific function or a set of functions [57]. According to Meier *et al.* [57], a good software architecture divides an application into distinct features with as little overlap in functionality as possible (separation of concerns); ensures that each component or module is responsible for only one specific feature or function; ensures that a component or object does not know about the internal details of other components or objects; ensures that a specific function is implemented in only one component; and only designs what is necessary.

A layered software architecture should be used to develop the functions of the compo-

nents of HealthAware. The architecture meets the characteristics of a good architecture described above by grouping related functions of an application into distinct layers that are stacked vertically on top of each other [57]. This is done by splitting the implementation of a function into presentation, business, and data access layers. The presentation layer contains user-oriented functionality for managing user interaction with an application. The business layer implements the core functionality of an application and encapsulates relevant business logic. The data access layer provides access to data hosted within the boundaries of an application and that exposed by other networked systems, perhaps accessed through web services. Figure 5.3 is a graphical representation of the layered architecture.

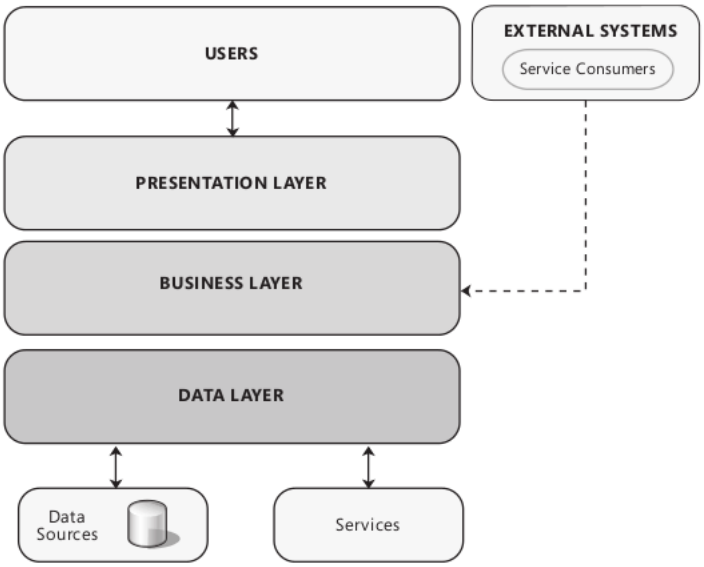

Figure 5.3: The logical view of a layered software architecture [57].

5.2.3 Language Switching

The components should be accessible in multiple languages. A user should access the system in his or her language of preference by default. The language should be specified when creating or updating a user profile. Switching of languages should be possible from the web interface. It should update the web contents to the selected language, but not the preferred language of the user. When creating health information using Component 1, the language the information is developed should be specified so that it is matched with the preferred language of a TeleWeaver user.

5.2.4 Support Functions

Component 1 should include support functions that should be accessible to a system administrator only. These should include *Users, Locations, Organisations, Service types Occupations,* and *TeleWeavers*. The *Users* function should be used to create and manage users. The *Locations* function should be used to create and manage geographical location information that should be used when setting targets. The *Organisations* function should be used to create and manage organisations that use HealthAware. The *Service types* function should be used to create and manage categories of services that are provided by different facilities. The *Occupations* function should be used to create and manage occupation information that should be used when setting targets. The *TeleWeavers* function should be used to record addresses of TeleWeaver instances that should be used when publishing health information to a specified TeleWeaver instance. Any number of support functions can be included depending on the need.

5.2.5 Targeting

Awareness campaigns and surveys should be targeted so that they are appropriately matched with the target users. Targeting should use a combination of demographic characteristics and geographical locations. To determine if an awareness campaign or a survey is relevant to user, the target information should be compared with demographic and geographical locations of a user.

5.3 Functional Specifications of Component 1

This section discusses the functional specifications of Component 1. The component is called the Dashboard and should be hosted independently of TeleWeaver. The main functions of the component should include *Campaigns, Surveys, Events, News, Topics, Facilities*, and *Services*. The support functions should include *Users, Locations, Organisations, Service types, Occupations*, and *TeleWeavers*. The Dashboard should enable users from health service organisations to create, manage (update, delete, stop, restart, or archive), and publish health information, view reports of awareness campaigns and surveys, and a system administrator to manage users. Figure 5.4 is a graphical representation of the functions of the Dashboard. In this section, all references to a 'user' should be interpreted as a 'Dashboard user' unless specified.

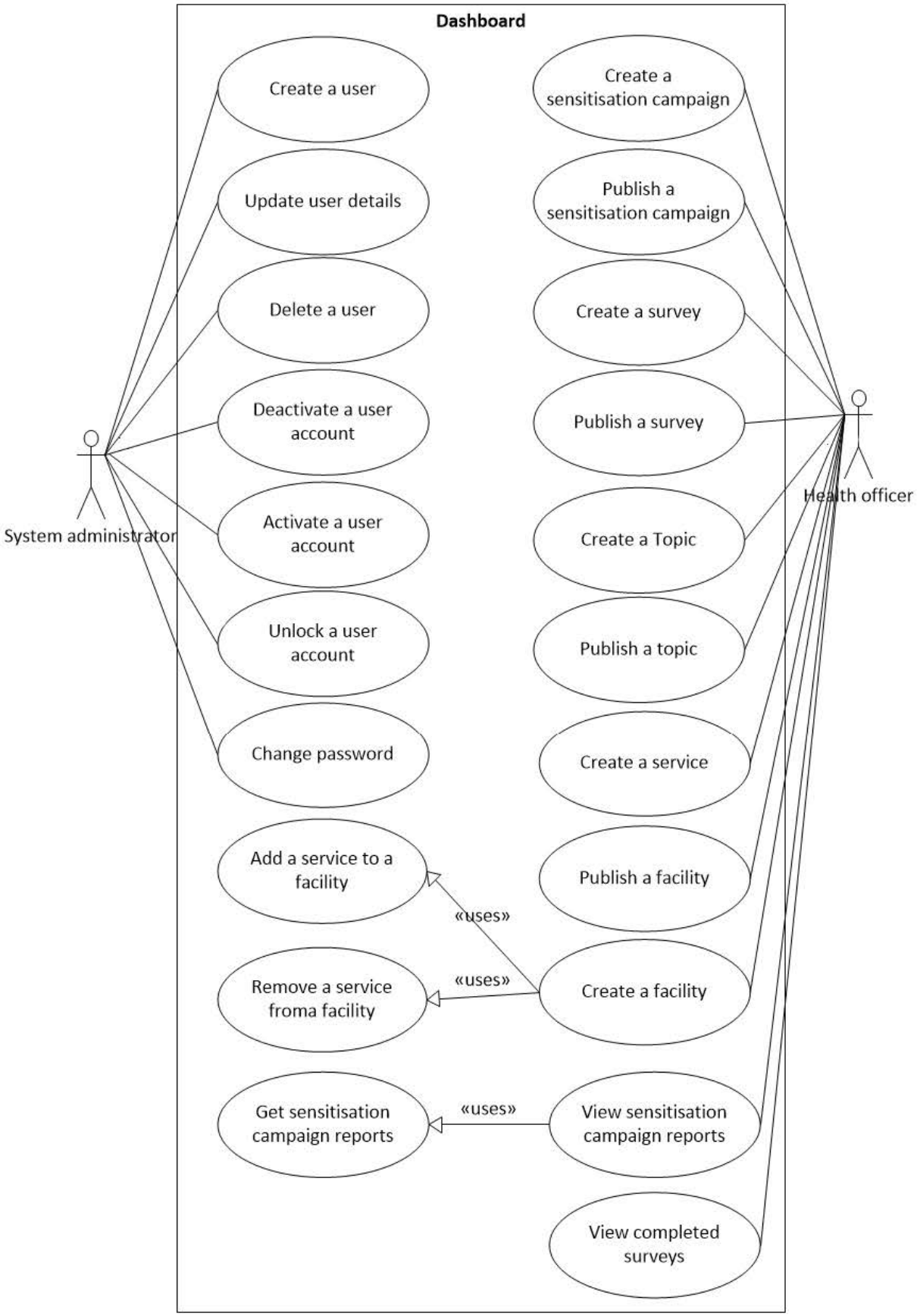

Figure 5.4: Dashboard use case

5.3.1 Campaigns Function

The *Campaigns* function should be used to create, update, delete, stop, restart, archive, or publish a campaign.

5.3.1.1 Creating a Campaign

The details that should be recorded when creating a campaign include a `title, start date and time, end date and time, slogan, aim`, and `description`. All these, except the `description`, should be provided to create a campaign. Other details include `messages` and `target` (demographic characteristics and geographical locations for targeting a campaign). When creating a campaign, a check should be made to ensure that the `start date and time` are greater than the current date and time and less than than the `end date and time`.

5.3.1.2 Campaign States

A campaign can be in *Active*, *Published*, *Running*, *Stopped/Inactive*, *Finished*, or *Archived* state. A new campaign should, by default, be in *Active* state. A user should be able to edit/delete/publish a campaign, add/edit/delete its messages, and add/update its target information when it is in *Active* state. If a campaign has been published and the current date and time are less than the `start date and time`, it should be in *Published* state. If it has been published and the current date and time are greater than the `start date and time` and less than the `end date and time` of a campaign, it should be in a *Running* state. If it has been published and the current date and time are greater than the `end date and time` of a campaign, it should be in a *Finished* state. The state of a campaign should determine the actions a user can perform on it. Figure 5.6 and 5.7 are graphical representations of the states a campaign can pass through.

5.3.1.3 Updating a Campaign

A user should be able to update a campaign only when it is in *Active*, *Published*, or *Stopped* stated. When updating a campaign, a check should be made to ensure that mandatory fields are completed and that the `end date and time` are greater than the `start date and time`. All the details of a campaign should be updateable. User confirmation is required for a campaign to be updated because of the critical nature of the operation.

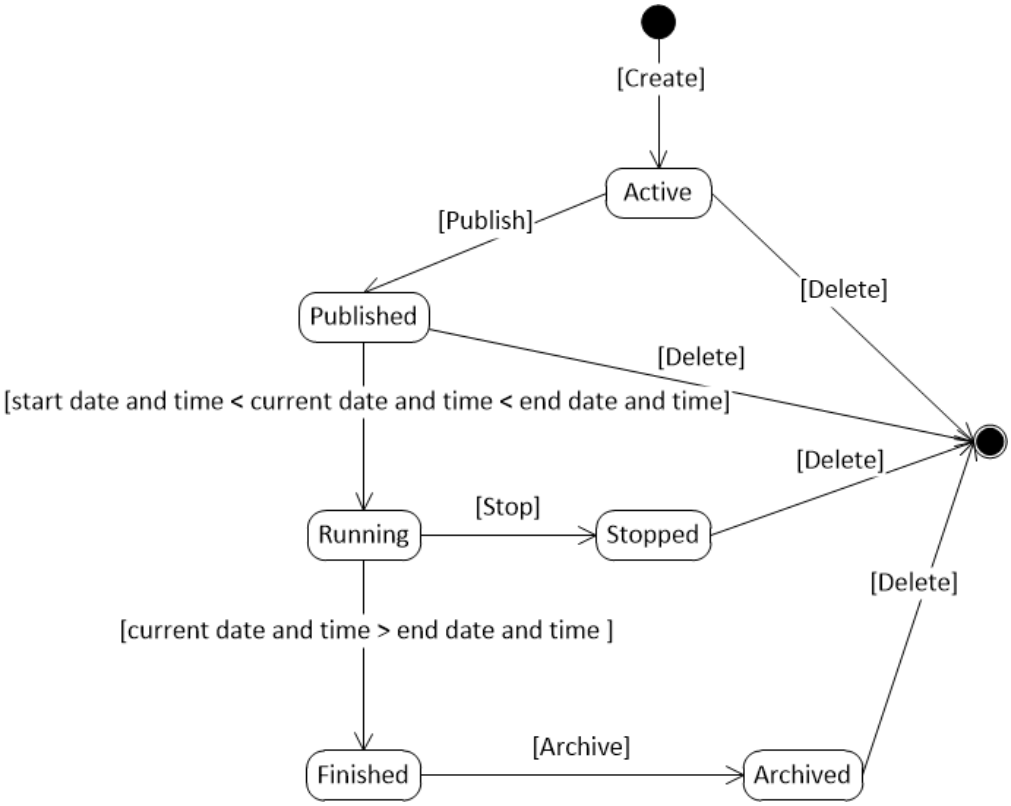

Figure 5.5: State diagram of a campaign

5.3.1.4 Deleting a Campaign

A user should be able to delete a campaign only when it is in *Active, Published, Stopped*, or *Archived* state. User confirmation is required to delete a campaign because of the critical nature of the operation.

5.3.1.5 Stopping and Restarting a Campaign

A user should be able to stop a campaign only when it is in *Running* state (see Figure 5.7). When it has been stopped, its state should change to *Stopped*. When it has been restarted, its state should change to *Running* if the current date and time are greater than the `start date and time` and less that the `end date and time`, or *Finished* if the current date and time are greater than the `end date and time` (see Figure 5.6). User confirmation is required to stop or restart a campaign because of the critical nature of the operation. If a campaign has been stopped, a user should only be able update (edit campaign details, add or edit target details, and add or edit messages), restart, or delete it.

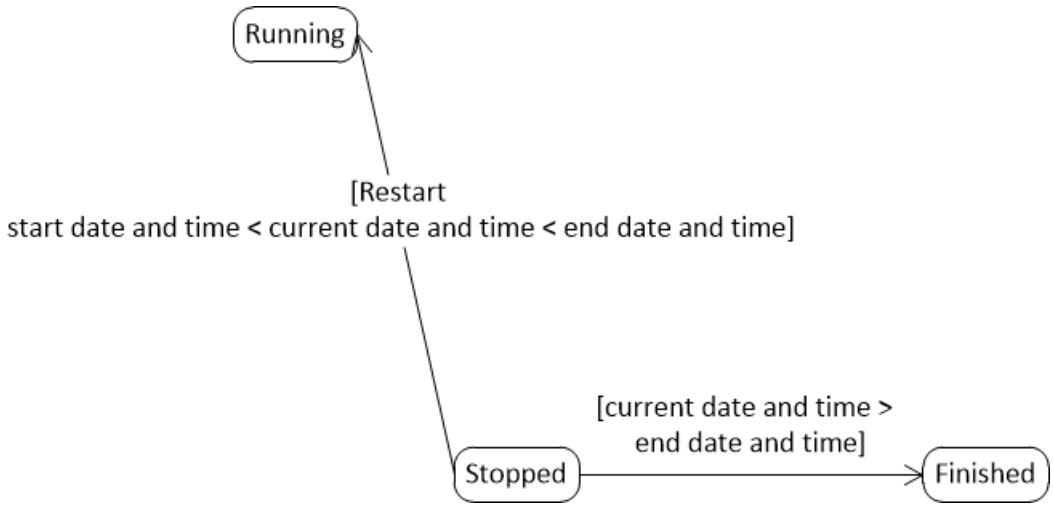

Figure 5.6: State diagram for stopping and restarting a campaign

5.3.1.6 Adding or Updating Campaign Messages

A user should be able to add or update campaign messages only it is in *Active*, *Published*, or *Stopped* state. A campaign message should have a `title` and `description`. A campaign can have any number of messages. A user should be able to add, update or delete campaign messages.

5.3.1.7 Targeting a Campaign

Targeting a campaign involves setting demographic characteristics and geographical locations to be used to determine it is appropriate a TeleWeaver user. A user should be able to target a campaign only when it is in *Active*, *Stopped*, or *Published* state. Any number of demographic characteristics can be set.

5.3.1.8 Archiving a Campaign

A user should be able to archive a campaign only when it is in a *Finished* state (see Figure 5.7).When a campaign is archived, its state should change to *Archived*. User confirmation is required to archive a campaign because of the critical nature of the operation. A user should only view campaign information, delete it, or generate campaign report when it has been archived.

5.3.1.9 Publishing a Campaign

Publication of a campaign should be possible only when its state is *Active* or *Published*. When the state is *Active*, publication should create a new record of the campaign information in Component 2. Subsequent publications should update the information. When a campaign has been published, its state should change to *Published*. If published, and the current date and time are greater than the `start date and time` and less than the `end date and time`, its state should change to *Running*. If published, and the current date and time are greater than the `end date and time` its state should change to *Finished* (see Figure 5.7). User confirmation is required to publish a campaign because of the critical nature of the operation.

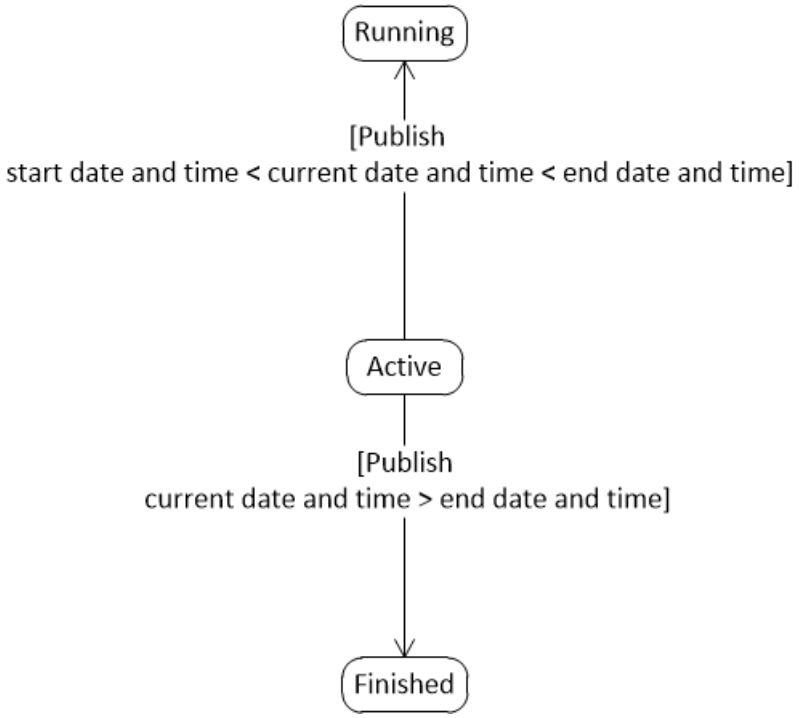

Figure 5.7: State diagram for publishing a campaign

5.3.2 Surveys Function

The *Surveys* function should be used to create, update, delete, stop, restart, archive, and publish a survey. The details that should be recorded when creating a survey include the `title`, `description`, `start date and time` and `end date and time`. All these are mandatory and should be provided to create a survey. Other details include `questions` and `target`. A question should have multiple choice answer options that allow a user to

respond by selecting one or multiple answers. A survey can have any number of questions and a question can have any number of answer options. Targeting and publication of a survey should work similarly to targeting and publication of a campaign. A survey should go through the same states as a campaign and each state should determine the actions a user can perform.

5.3.3 Events Function

The *Events* function should be used to create, update, delete, stop, restart, archive, and publish an event. The details that should be recorded when creating an event include the `name`, `start date and time`, `end date and time`, and `description`. All these are mandatory and should be provided to create an event. Targeting and publication of an event should work similarly to targeting and publication of a campaign. An event should go through the same states as a campaign and each state should determine the actions a user can perform.

5.3.4 News Function

The *News* function should be used to create, update, delete, archive, or publish a news article. The details that should be recorded when creating a news article include the `title`, `author`, and `content`. All these are mandatory and should be provided to create a news article. Targeting and publication of a news article should work similarly to targeting and publication of a campaign. A news article should be in *Active* state when created and in *Archived* state when archived (see Figure 5.8). If it has been archived, a user should only be able to view its information or delete it.

5.3.5 Topics Function

The *Topics* function should be used to create, update, delete, archive, or publish health topics. The details that should be recorded when creating a topic include the `title`, `description`, and `topic details/sections`. The `title` and `description` are mandatory and should be provided to create a topic. A topic can have any number of topic sections. A `topic section` should include a `title` and `body`. Topics should not be targeted.

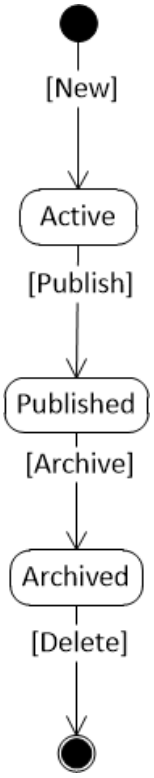

Figure 5.8: State diagram for News

5.3.6 Creating and Managing Facilities and Services

The *Facilities* function should be used to create and manage health facilities. The details that should be recorded when creating a facility include the `name`, `type`, and `location`. The `location` information should include `province`, `district`, `municipality`, and `village`. All the details are required and should be provided to create a facility. A user should be able to add and remove health services from a facility. When a service is added to a facility, it indicates that the facility provides the service. If it is no longer provided, it can be removed. The *Services* function should be used to create and manage health services that are provided by health facilities. The details that should be recorded when creating a service include the `name`, `description`, and `type` of service and all these are required to create a service.

5.3.7 Users Function

The *Users* function should be used by the system administrator to create and manage users. Management of users include updating, deactivating, activating, unlocking, and deleting a user account. Figure 5.9 is a graphical representation of the functions.

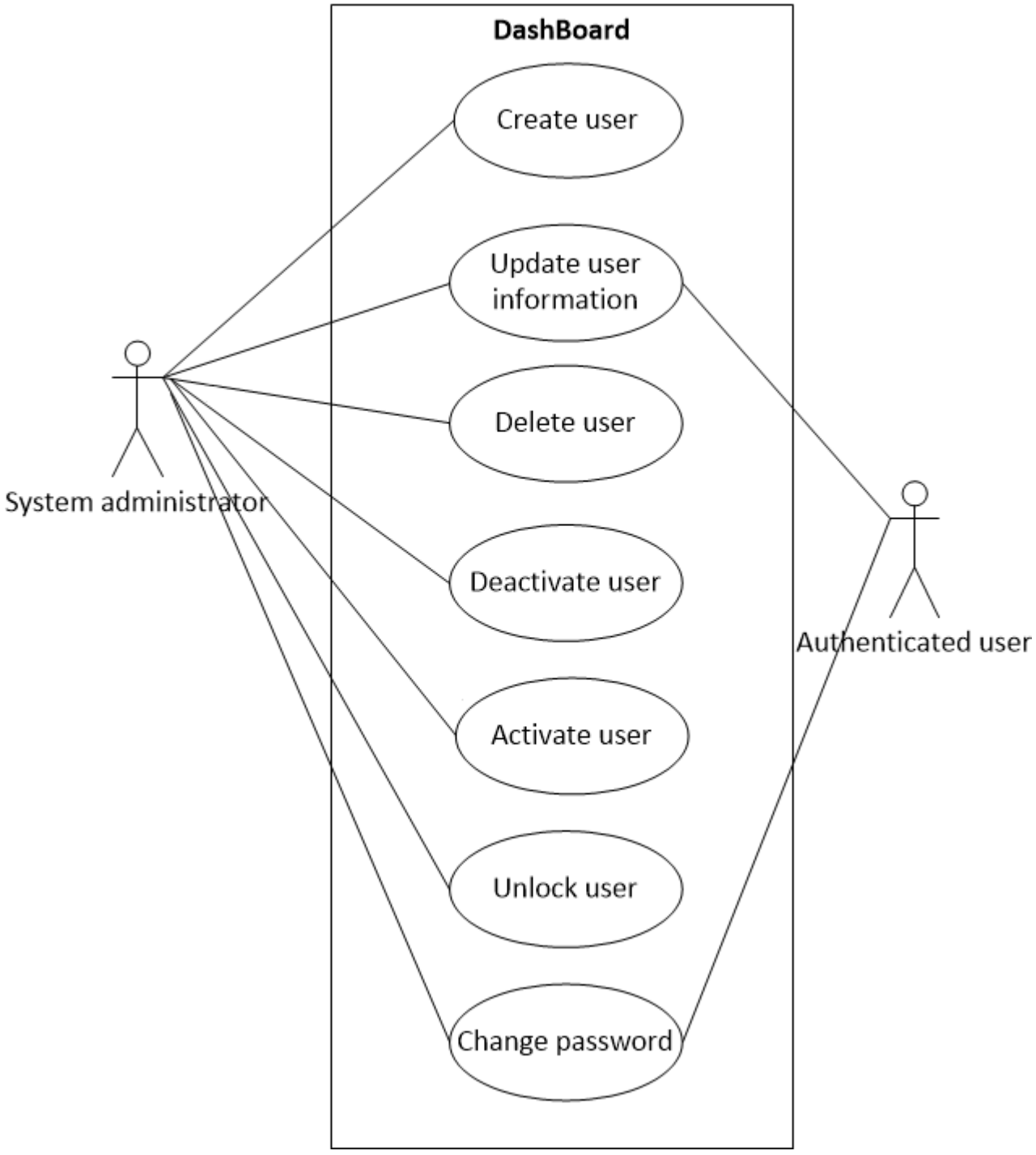

Figure 5.9: Use case diagram for user management

5.3.7.1 Creating a User

The details that should be recorded when creating a user include the `user name`, `first name`, `last name`, `password`, `organisation`, and `role`. All these are mandatory and should be provided to create a user. A new user account should be in a *Logged out* state.

5.3.7.2 Managing Users

A system administrator should be able to update information of any user account and deactivate, activate, unlock, and delete any user account except his or her account. A user account that has been deactivated or locked should not be used. Any use of the account should result in an error and a user should be informed of the reason the account cannot be used and the steps to take to ensure that it is usable.

5.3.7.3 User Authentication

User authentication in an application is the process of verifying the identity of a user or who he or she claims to be [7, 15]. As discussed by Barreiro [7], there are three basic factors that can be used to authenticate a user, which are:

- **Something only a user knows:** a password, which is usually used in combination with a user name, is an example. It is the simplest and the most inexpensive way of authenticating users. Some of its advantages are that users can share a password or if it is weak others can guess or crack it, leading to unauthorised access.
- **Something unique a user has:** the use of an ATM card to withdraw money, or a card to gain access to a building are some examples. In the case of ATM cards, what a user has in possession is used in combination with a Personal Identification Number (PIN) to increase security.
- **Something a user is:** the use of biometrics (fingerprints, voice, or iris scan) is an example. It is more expensive but it ensures that a registered user is present for authentication to succeed.

A user name and password combination should be used to authenticate Dashboard users because of its simplicity and low cost. Measures should be taken to ensure that the

password is strong. If a user enters a wrong user name and password combination, he or she should be informed. One of the measures that can be taken to guard against password guessing is limiting the number of failed login attempts to five. On the fifth attempt, if a user inputs a wrong password, the user account should lock.

5.3.7.4 States of User Accounts

When a user is successfully authenticated, his or her account should change to *Logged in* state. If he or she logs out, it should change to *Logged out* state. If it is locked, it should change to *Locked* state. If it is unlocked, it should change to *Logged out* state. If it is deactivated, it should change to *Inactive* state. If it is activated, it should change to *Logged out* state. A system administrator should be able to delete a user account only when it is in *Inactive*, *Logged out* or *Locked* state. Figure 5.10 is a graphical representation of the states a user account should go through. Figure 5.11 is a mock screen showing a list of user accounts in different states and the actions a system administrator is permitted to perform on a user account in a particular state.

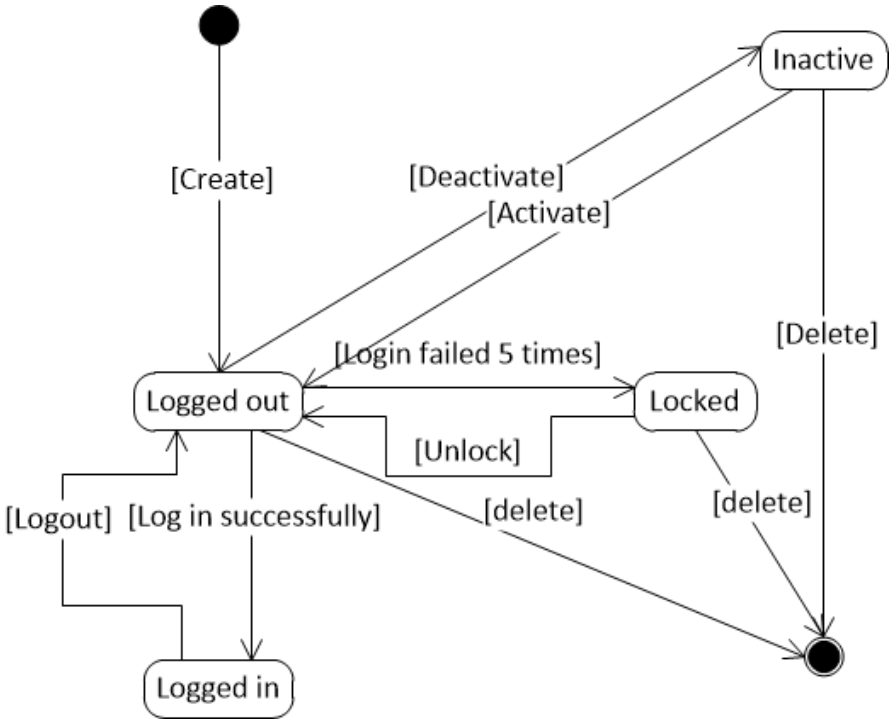

Figure 5.10: State diagram for user account state transitions

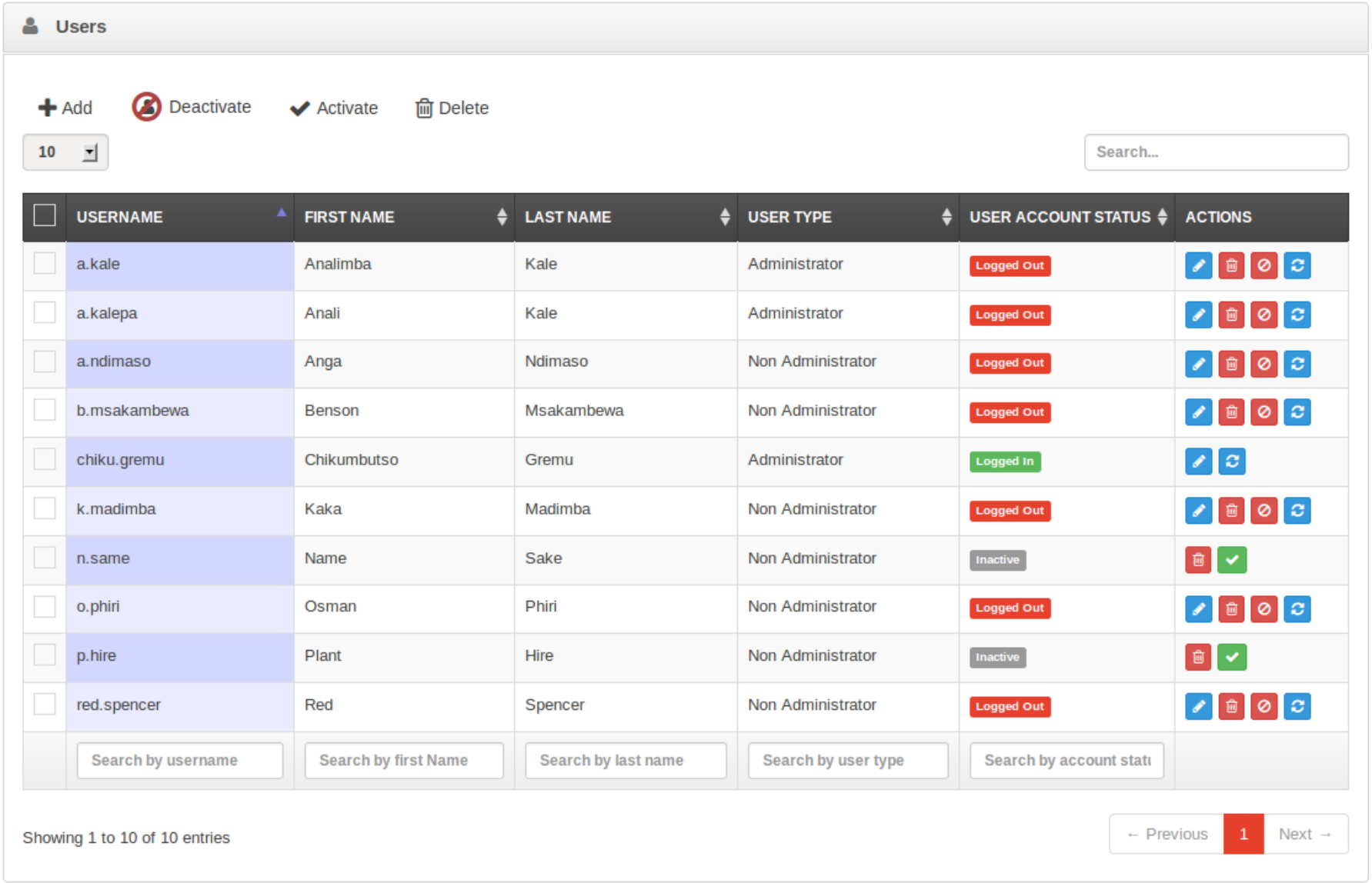

Figure 5.11: Mock screen listing users

5.3.7.5 User Authorisation

User authorisation should be implemented to ensure that a user performs only the functions he or she is allowed. Two user roles, which are *administrator* and *authenticated*, should be used to determine the functions a user can perform. A user with an *administrator* role is a super user (system administrator) and should perform all the functions of the Dashboard. A user with an *authenticated* role should create and manage health information only. This is a standard practice in application development where super users are separated from ordinary users by the functions they perform.

5.3.8 Search Function

The *Search* function should enable a user to search for specific information. A user should search for information using several criteria. For example, a campaign can be searched by name, start date, end date, and status as in Figure 5.12.

5.3.9 Bulk Functions

The *Bulk* function should enable a user to perform several functions at the click of a button. For example, a user can delete more than one campaign, or publish more than one survey at the click of a button.

5.4 Functional Specifications of Component 2

This section discusses the functional specifications of Component 2. The component is called the HealthMessenger and should be hosted on TeleWeaver. In this section, all references to 'user' should be interpreted as 'TeleWeaver user' unless specified. The functions of the component are *Campaigns, Surveys*, *Events*, *News, Topics*, and *Facilities. The Campaigns* function should enable a user to access campaign messages. The *Surveys* function should enable a user to complete surveys. The *Events* function should enable a user to register or deregister from an event. The *News* function should enable a user to access health news. The *Topics* function should enable a user to access health topics. The *Facilities* function should enable a user to access information of healthcare facilities and the services they provide.

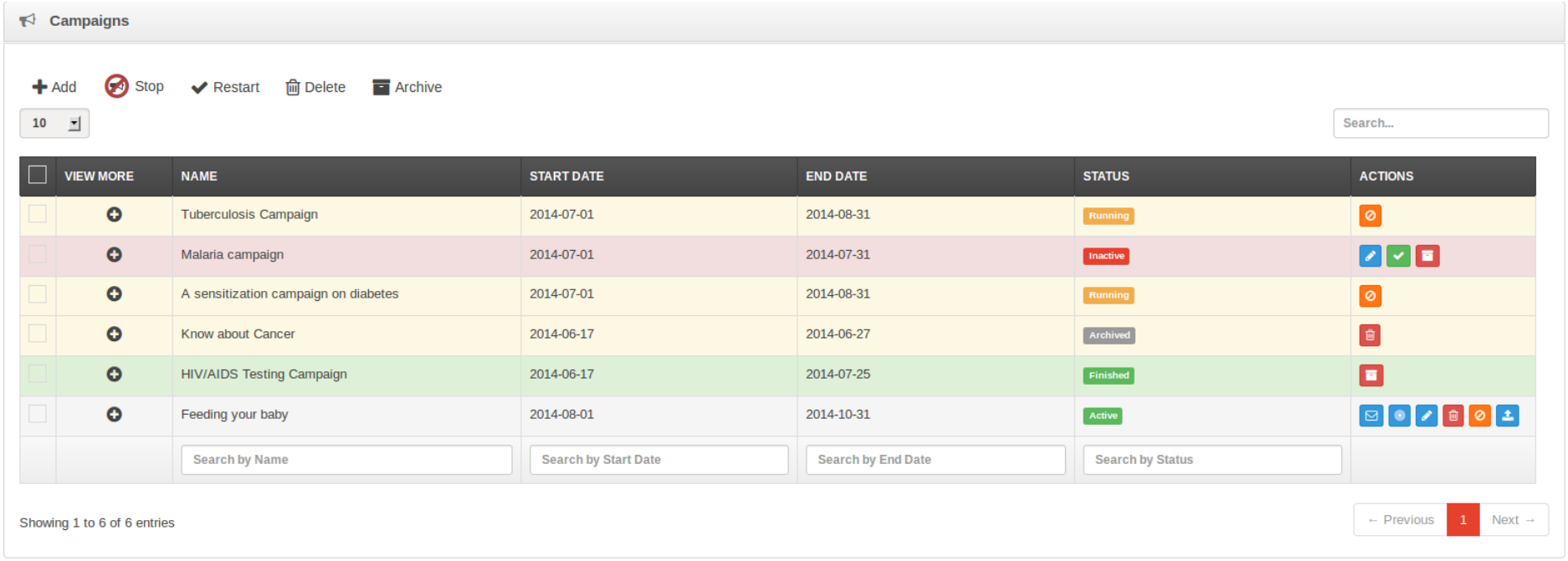

Figure 5.12: Mock screen showing a list of campaigns

5.4.1 HealthMessenger Requirements

The HealthMessenger should enable a user to access health information that is created and published using the Dashboard. The information should be accessible in either a pull or a push mode. In a pull mode, a user accesses information on demand, while in a push mode, information is retrieved and delivered to a user [33]. A user should access awareness campaigns and surveys in either a pull or a push mode and health topics, facilities, and services in a pull mode. Figure 5.13 is a graphical representation of the functions in the HealthMessenger.

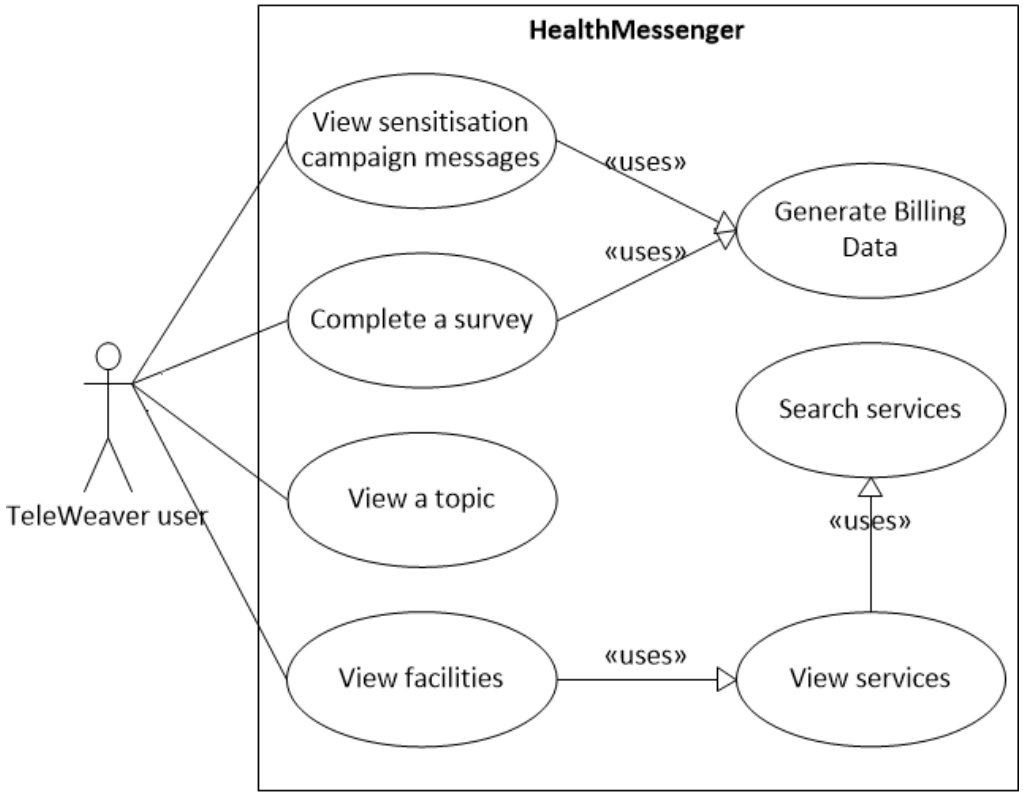

Figure 5.13: Use Case of the HealthMessenger

5.4.2 Campaigns Function

As discussed above, a user should be able to access campaigns in either a pull or a push mode. In both modes, a targeted strategy should be used to select campaigns that a user is a target. This should be done by comparing demographic characteristics and geographical locations that are set when creating a campaign and demographic and geographical locations of a user. The detailed steps of how the two strategies should work are discussed in the following subsections.

5.4.2.1 Pull Mode

The pull mode, illustrated in Figure 5.14 below works as follows:

- an authenticated user clicks on a menu to access campaigns;
- profile information of the user and campaigns in a *Running* state are retrieved;
- target information of each campaign is compared with demographic information of the user that is obtained from his or her profile information in order to determine if they match. (**Note**: A match happens if the target information of a campaign is a subset or a complete match of the user's demographic information); and
- all campaigns that do not match are discarded and matched campaigns are presented as a list and a user can access campaign messages by clicking on its link.

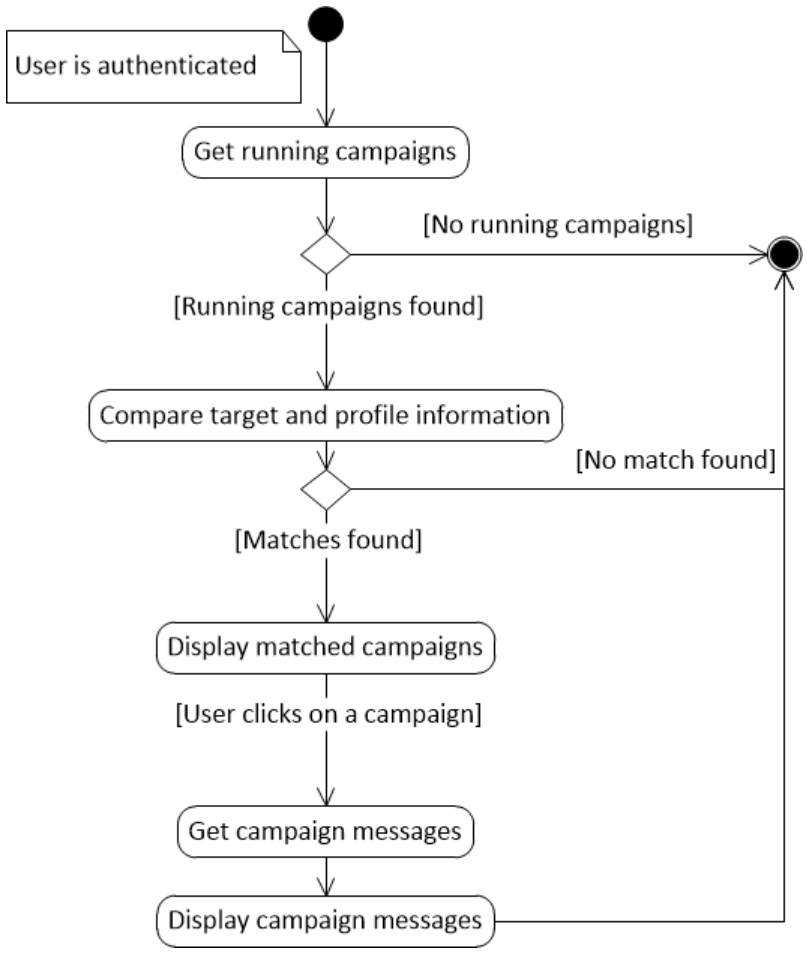

Figure 5.14: Activity diagram of the pull mode

5.4.2.2 Push Mode

A targeted strategy should be used to select campaigns that a user is a target as discussed in the pull mode above. If several campaigns are selected, their priorities should be compared in order to select one of them. The priority can be based on the gravity of the case being advanced by the campaign. For example, if there is an outbreak of a disease, which is costing lives of people, a campaign to raise awareness of the disease can have a high priority. The `title` of the selected campaign and the name of the user should be used to create a custom message that should be used to attract his or her attention to engage with campaign messages. This follows a recommendation by the Open University [64] that attracting the attention of the target audience is key to successful communication. The push mode is illustrated in Figure 5.15.

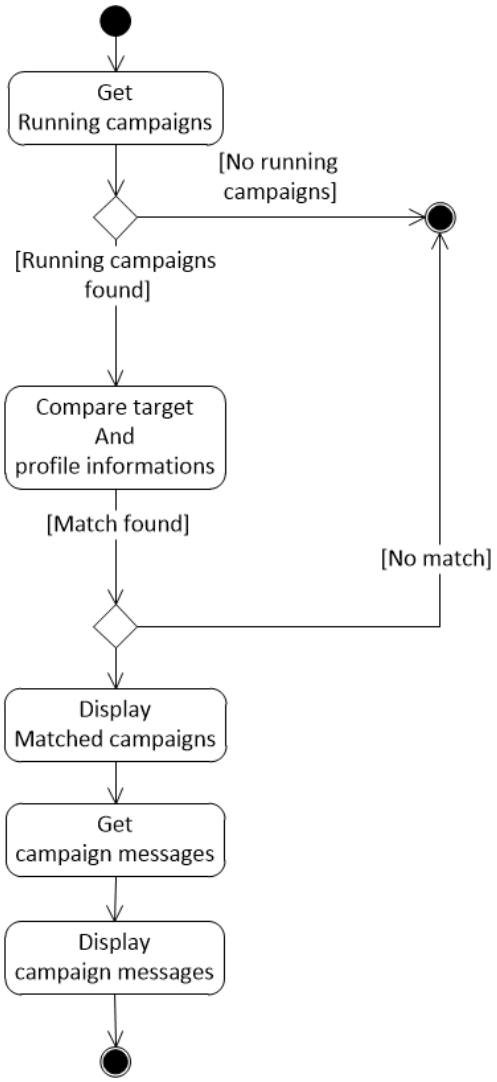

Figure 5.15: Activity diagram of the push mode

As discussed in Sub-section 2.7.5.3, a user has an option to click on the custom message in order to access related messages, or close it. If a user choses to access related messages, a URL should open to enable the user to view campaign messages. A message that has been accepted by a user should not be delivered to him or her again in the push mode. However, a user can access campaign messages any number of times in a pull mode.

5.4.3 Surveys Function

The *Surveys* function should enable a user to respond to survey questions. A user should access surveys in either a pull or push mode. The modes should work similarly to how they work for a campaign. If a user clicks of a custom message in a push mode, a URL should open to enable him or her to respond to survey questions. If there are many questions, they should be split into several pages. A user should be able to save survey responses on each page. Survey responses should be saved to the HealthMessenger database and a copy should be posted to the Dashboard for production of survey completion report.

5.4.4 Events Function

The *Events* function should enable a user to register and deregister from an event. A user should access events in either a pull or push mode. The modes should work similarly to the *Campaign* function. If a user clicks of a custom message in a push mode, a URL should open to enable him or her to register for an event. Registration information should be saved in the HealthMessenger database and a copy should be posted to the Dashboard for generating event registration report. Figure 5.16 is a graphical representation of the *Events* function.

5.4.5 News Function

The *News* function should enable a user to read news articles. A user should access news in either a pull or push mode. The modes should work similarly to the *Campaign* function. If a user clicks of a custom message in a push mode, a URL should open to enable him or her to read a news article.

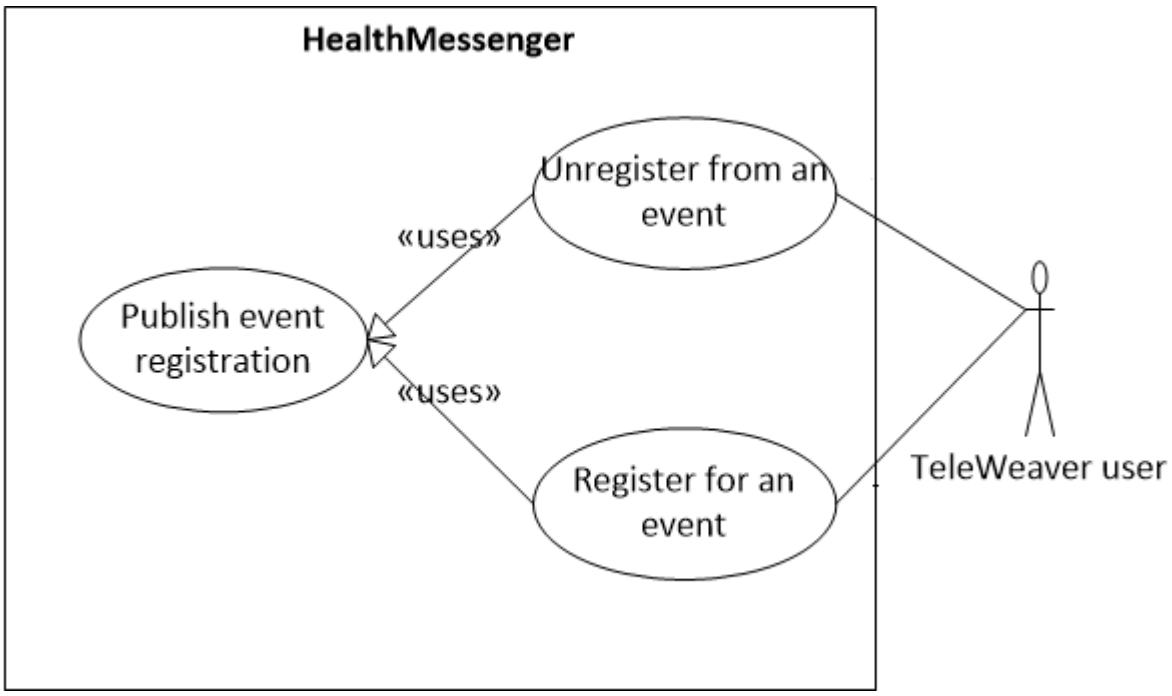

Figure 5.16: Use case for registering for an event

5.4.6 Facilities Function

The *Facilities* function should enable a user to access information of healthcare facilities and the services they offer in a pull mode. A user should be able to access any information on facilities and services. A link should be provided on a facility to enable a user to access information of the services it offers. A search function should be provided to enable a user to search for facilities and services.

5.4.7 Topics Function

The *Topics* function should enable a user to access information of health topics. A user should be able to access information on any topic in a pull mode. A search function should be provided to enable users to search for a specific topic.

5.5 Logging of Auditing and Billing Data

The second objective of the research is to extend the functions of HealthAware to be a source of revenue to support the development and upkeep of ICT infrastructure in poor

areas. The organisations using HealthAware to run campaigns or surveys, send events information, or disseminate news should be billed using a charging framework similar to pay-per-click framework used by Google, Yahoo, and Bing to charge advertisers. In pay-per-click, advertisers pay each time a potential customer clicks on an advert [13, 65]. The advertisers are allowed to propose a maximum amount they are willing to pay for a click on an advert. Health service organisations should be allowed to pay a standard amount or to propose to pay an amount that is higher than the standard amount just like in the pay-per-click. They should be billed every time a user clicks to view a campaign message, complete a survey, read a news article, or register for an event. Figures 5.17 and 5.18 are graphical representations of logging of auditing and billing data.

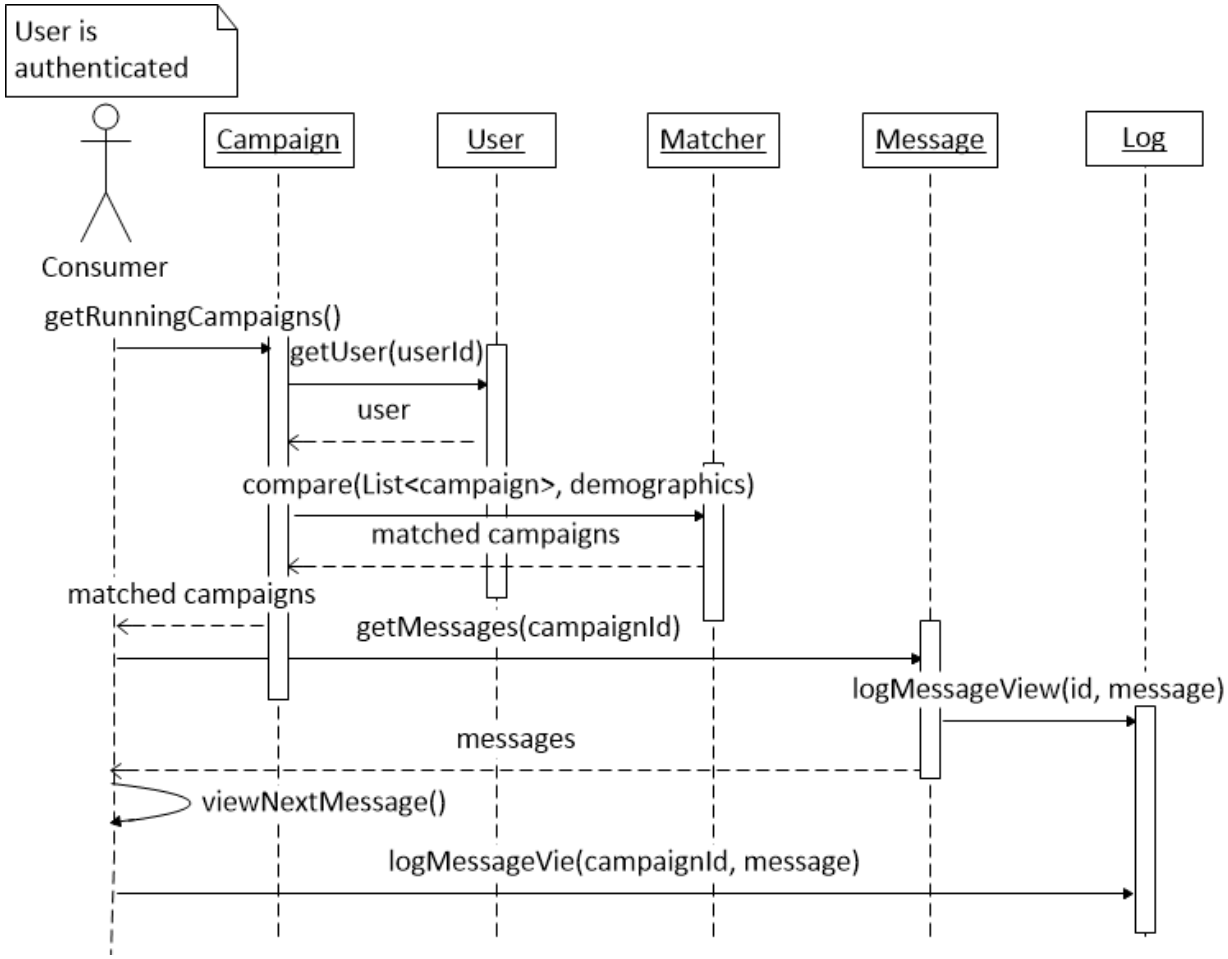

Figure 5.17: Sequence diagram for pull strategy

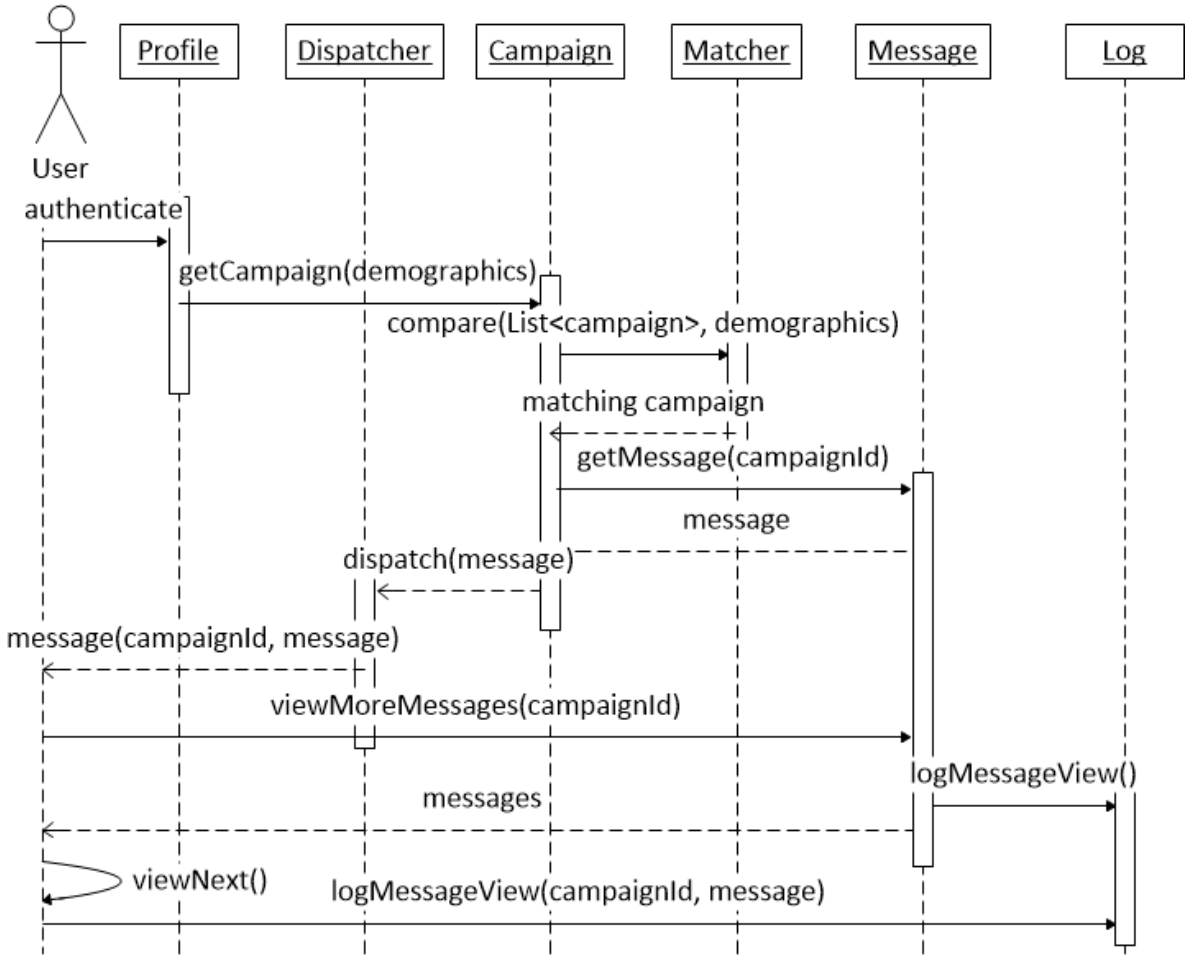

Figure 5.18: Sequence diagram for push strategy

5.6 Data Flow in HealthAware

Data flow in HealthAware should be bi-directional. The campaign function is used to demonstrate the flow of data, which is as follows:

- a user from a health service organisation, herein referred to as a health officer, uses the Dashboard to create a campaign and set parameters that should be used to determine who should access or be reached with specific campaign messages;
- the officer publishes the campaign so that it is accessible to TeleWeaver users through HealthMessenger;
- when a TeleWeaver user has been authenticated, a targeted strategy in HealthMessenger is used to select a campaign he or she matches as a target;

- the title of the campaign and the TeleWeaver user's name are used to create a custom message that is sent to him or her as a pop-up message box in order to attract his or her attention to engage with campaign messages;
- when the TeleWeaver user clicks on the message, a page with a campaign message is displayed (**Note**: In the case where there are many messages, a user views one message at a time); and
- clicking on the message generates data that is posted to the Dashboard for generation of campaign reports.

Note: When a user accesses campaigns through a menu, matching campaigns are displayed as a list. The user accesses a campaign's messages by clicking on its link. Clicking on a link generates data that is sent to the Dashboard for generating campaign reports.

5.7 Summary

This chapter discussed the requirements and design of HealthAware. The HealthAware comprises two components called the Dashboard and the HealthMessenger. The Dashboard enables health service organisations to create, target, and publish health information to be accessible to TeleWeaver users through the HealthMessenger in either a pull or push mode. HealthAware includes a function to generate data for billing purposes. The function is based on the pay-per-click charging framework used by Google, Bing, and Yahoo to charge advertisers. The next chapter discusses the implementation of HealthAware.

Chapter 6

Implementation

6.1 Introduction

The previous chapter discussed the requirements and design of HealthAware. This chapter discusses its implementation. Section 6.2 discusses the general tools used. Section 6.3 discusses the implementation of the functions of HealthAware from a general perspective. Section 6.4 discusses the implementation of the functions of the Dashboard. Section 6.5 discusses the implementation of the functions of the HealthMessenger. The implementation of the functions of the Dashboard is discussed in detail. The implementation of the functions of the HealthMessenger is limited to features that are different because it mirrors the implementation of the functions of the Dashboard.

6.2 Tools Used

This Section discusses the general tools used to develop the components of HealthAware. Specific tools used to implement specific features are discussed when the feature implementation is described. The tools used to implement the functions the HealthMessenger were constrained by TeleWeaver. As discussed in Section 5.4, the HealthMessenger will be hosted on TeleWeaver, which is based on JBOSS. Java and associated tools were used to develop the HealthMessenger to ensure that it deploys on TeleWeaver. The same tools were used to implement the functions the Dashboard for simplicity. The advantage of using homogeneous tools is that the same skill sets were required to implement the functions of both components, and will be required to maintain them later.

6.2.1 Host Server Environment

As specified in subsections 5.2.1 and 5.2.2, the components were implemented as separate Java web applications, each having a client/server architecture. Wildfly version 8.2.0.Final was used to host the components during the development stage. The use of this environment ensures that the HealthMessenger deploys on TeleWeaver with few or no changes at all.

6.2.2 Development Tools

Some of the tools used to implement HealthAware include Maven, Spring Framework, and Spring Tool Suite. Maven is a software project management tool based on the concept of a project object model (POM).[1] Version 2.2.1 of Maven was used to create the project and manage project library dependencies. Spring Framework is an open source framework created to simplify the development of enterprise Java software [89]. Four functional areas of the framework, namely Data Access/Integration, Web, Aspect-Oriented Programming (AOP), and the Core Container were used to simplify implementation of specific functions of HealthAware. The Data Access/Integration function was used to simplify the implementation of object-relation mapping (ORM). The Web function was used to integrate the Spring into web applications. AOP was used to simplify the management of transactions. The Core Container function was used to simplify the management of object references using Dependency injection (DI). Details of the Spring Framework, which include all of its functional areas, are found in Appendix A. Spring Tool Suite is a plugin for Eclipse that is customised for developing Spring applications.[2] Eclipse is an open source integrated development environment (IDE) developed in Java that is used to edit and debug program source code.[3] The plugin was used to edit and debug the Java source code, XML, and HTML scripts.

6.2.2.1 Creating Java Web Application Projects

A Maven template called maven-archetype-webapp, which is used to create Java web application projects, was used to create the projects to implement the two components. The projects were created on the command line, converted to eclipse projects, and then imported as Maven projects by Spring Tool Suite.

[1] http://maven.apache.org/
[2] http://spring.io/tools/sts
[3] https://www.eclipse.org/

6.2.2.2 Project Library Dependency Management

Maven was used to manage the projects' Java Archive (JAR) library dependencies. Configuration details to enable Maven to download the required JAR files were added to the pom.xml file that is created for every Maven project. Listing 6.1 is an example of configuration details to enable Maven to download dependencies for Jersey to use Spring beans as Java API for RESTful Web Services (JAX-RS) components. Jersey is an open source framework for developing Web Services in Java according to the Representational State Transfer (REST) architectural pattern.[4] It provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation. The JAX-RS API provides support for developing RESTful Web Services using Java. All the configuration details made in the pom.xml are available in Appendix C.1.

```
<dependency>
    <groupId>com.sun.jersey.contribs</groupId>
    <artifactId>jersey-spring</artifactId>
    <version>1.12</version>
    <scope>compile</scope>
    <exclusions>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Listing 6.1: pom.xml configuration details

The meaning of the XML elements used in Listing 6.1 are:

[4] https://jersey.java.net/

- **groupId**: This is the unique group ID of the project.
- **artifactId**: This refers to the name of the project.
- **version**: This refers to the version of the project
- **scope**: this refers to the classpath of the task at hand and is used to limit the transitivity of a dependency. Five different scopes are available:
  - **compile** - this is the default scope and indicates that the dependency should be available in all classpaths. Furthermore, those dependencies are propagated to dependent projects.
  - **provided** - this is similar to **compile**, but indicates that the JDK or a container is to provide the dependency at runtime.
  - **runtime** - this scope indicates that the dependency is not required for compilation, but only during execution of the program.
  - **test** - this scope indicates that the dependency is not required for normal use of the application and is only available for the test compilation and execution phases.
  - **system** - this is similar to **provided** except that the JAR file which contains the dependency has to be provided.
- **exclusions**: this explicitly tells Maven not to include the specified project that is a dependency of this dependency (in other words, it is a transitive dependency).

6.2.2.3 Wiring Dependencies

Spring Framework version 3.2.8.RELEASE was used to implement the components. Spring configuration details were added to the pom.xml file of the Spring Tool Suite to enable Maven to download the required Spring dependencies. Four XML files, namely *reedhousystems-data-context.xml, reedhousystems-component-scan-context.xml, reedhousystems-root-context.xml*, and *reedhousystems-servlet-context.xml* were used to configure (or wire) Spring dependencies and to indicate the packages to scan for annotated classes that need to be registered automatically as beans in the Spring container.

The *reedhousystems-data-context.xml* configures hibernate session factory, data source, transaction manager, and hibernate properties. The hibernate session factory (*org.spring-*

framework.orm.hibernate4.LocalSessionFactoryBean) is used to obtain a physical connection to the database. The data source (*org.apache.commons.dbcp.BasicDataSource*) provides database connection pooling. The transaction manager (*org.springframework.orm.hibernate4.HibernateTransactionManager*) manages hibernate sessions. The hibernate properties are used by the hibernate session factory to obtain a connection to the database.

The *reedhousystems-component-scan-context.xml* file lists the packages to scan for classes that should be automatically registered as beans in the Spring container. The *reedhousystems-root-context.xml* file imports the configuration details in the *reedhousystems-data-context.xml* and the *component-scan-context.xml* and combines them in one configuration file. The *reedhousystems-servlet-context.xml*, which is located in the WEB-INF, imports the configuration details in the *reedhousystems-data-context.xml* and the *reedhousystems-component-scan-context.xml,* by importing the *reedhousystems-root-context.xml.* The WEB-INF is a directory that is created automatically when a Java web application project is created and contains metadata about the application. It is not part of the public document tree of the application, therefore, no file contained in the directory can be served directly to a client by the container. Other dependencies were wired using annotations in Java code and will be discussed in the following sections.

6.3 Implementation of HealthAware Functions

This section discusses the implementation of the functions of HealthAware from a general perspective. The layered architecture introduced in Sub-section 5.2.2 was used to implement the functions. The architecture splits the implementation of a function into presentation, business, and data access layers. Interface-based programming style, where a concrete class implements an interface, was used to implement the business and data access layers. The advantage of using this style is that it enables changing of the implementation details on the server without affecting how a client communicates with it [53]. The layered architecture used is shown in Figure 6.1.

6.3.1 Presentation Layer

HTML, CSS, and JavaScript tools were used to implement the presentation layer. Some of the tools used and the functions they were used to implement are discussed in the following subsections.

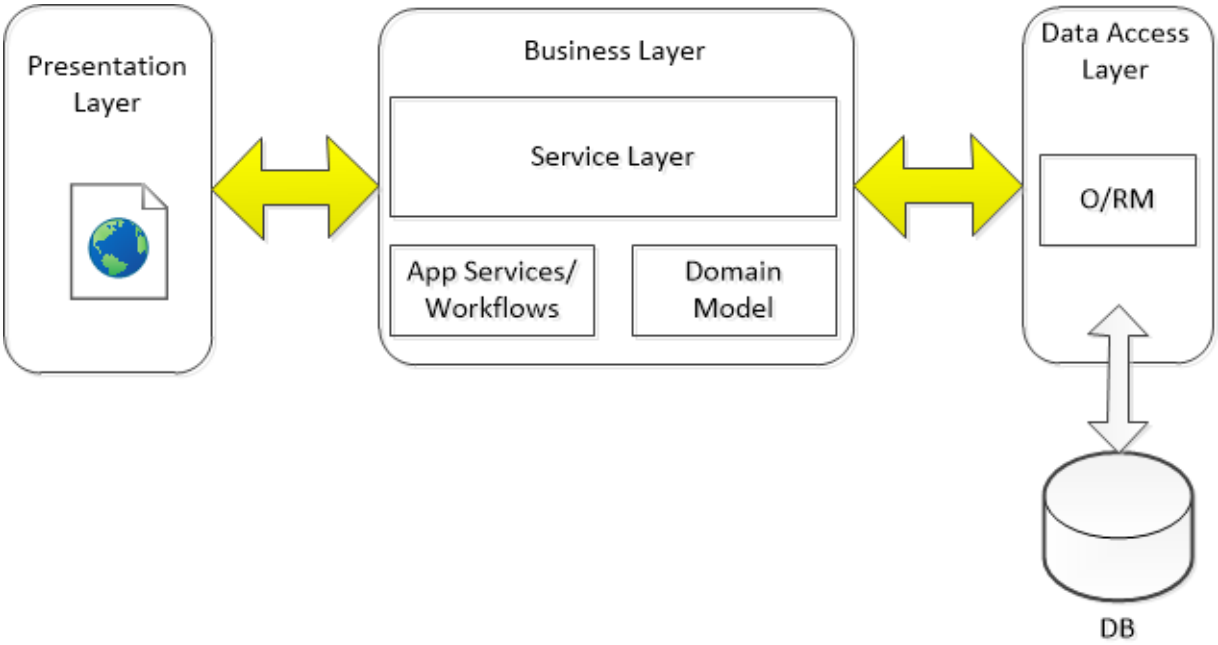

Figure 6.1: The Layered Architecture used to implement the functions of HealthAware [30].

6.3.1.1 Creation of Forms and Form Elements

Version 3.2.0 of Bootstrap was used to create forms and form elements, menus, and other presentation layer entities. Bootstrap is an open source framework designed for building user interface components.[5] The framework simplifies creation of presentation layer components by combining HTML, CSS, and JavaScript.

6.3.1.2 Creation of Tables and a Search Function

Version 1.9.4 of DataTables was used to develop interactive HTML tables and to implement a search function in the presentation layer. DataTables is a plugin for jQuery JavaScript library, built to simplify the development of interactive HTML tables.[6] Settings to hide columns, sort data in a table using one or more columns, and display table data in multiple pages if records exceed a certain number (paging) were configured on the plugin and applied to tables. Two types of search were implemented using the plugin: a global search that searches all the columns and a column search that searches only one column at a time in a table.

[5] http://getbootstrap.com/
[6] http://www.datatables.net/

6.3.1.3 Creating of Menu Icons

Version 4.0.3 of Font Awesome was used to create icons that represented the functions to add, update, delete, stop, restart, and archive campaigns, surveys, events etc. Font Awesome is a CSS library that provides scalable vector icons that are customisable - their colour, size, drop shadow etc is customisable.[7]

6.3.1.4 Data/Input Validation

Data validation is done to control what users provide to an application through input form. According to Open Web Application Security Project (OWASP) [81], an organisation that focuses on improving security of software, there are three strategies that can be used to validate data:

- **Accept known good**: This strategy is also known as whitelisting or positive validation. It works by accepting only the data that belongs to a set of tightly known good values and rejecting anything else.
- **Reject known bad**: This strategy is also known as negative or blacklist validation. It works by rejecting data that is known to be bad. The strategy has a flaw because the set of possible bad data is not easily manageable.
- **Sanitise**: This strategy works by changing user input into acceptable format as opposed to accepting or rejecting it. Sanitise can use either the whitelist or the blacklist strategy. Sanitise with whitelist works by removing, encoding, or replacing any data not part of an approved list. Sanitise with blacklist works by eliminating or translating data in an effort to make it safe.

A jQuery validation plugin[8] called Validate was used to implement input validation using whitelisting strategy. The plugin comes with a useful set of validation methods and is customisable. It provides an Application Programming Interface (API) that allows users to write their own validation methods. Acceptable data is specified by writing rules to be applied to input data. Some of the validations implemented using the plugin include ensuring a user provides data for all the required fields and restricting the length of input

[7] http://fortawesome.github.io/Font-Awesome/
[8] http://jqueryvalidation.org/

data. Data validation using the plugin only works if JavaScript is enabled in a browser. A second input validation check was implemented on the server side of the components to complement the check done at the presentation layer and this is discussed later.

6.3.1.5 Client and Server Communication

All HTTP requests from the client (web browser) to the server are done asynchronously. The jQuery.ajax() function of the jQuery API[9] was used to implement the requests. The function is configured using a set of key/value pairs, all of which are optional. The data that is transmitted between the client and the server is in JavaScript Object Notation (JSON). JSON is a lightweight data-interchange format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages.[10] It is simple for humans to read and write and for machines to parse and generate. Some of the configuration details made to the jQuery.ajax() function include the following:

- **contentType:** this is used to indicate the type of content that is sent along with a request. The key/value pair was set to **contentType: 'application/json'** to indicate that data in a JSON format will be sent with the requests.
- **data:** this is used to indicate the data that is sent with a request. The data is converted to JSON using *JSON.stringify* method of JavaScript.
- **dataType:** this is used to indicate the type of data expected from the server. The key/value pair was set to **dataType: 'application/json'** to indicate that data from the server to the client is in JSON format.
- **error:** this is used to indicate the function to be called if a request fails.
- **header:** this is used to send header information with a request as key/value pairs using XMLHttpRequest.
- **success:** this is used to indicate the function to be called if a request succeeds.
- **type:** this is used to indicate the type of request (e.g. GET, POST, DELETE etc.).
- **url:** this is used to indicate the URL a request is sent to.

[9] http://api.jquery.com/jquery.ajax/
[10] http://www.json.org/

6.3.2 Business Layer

The business layer was split into Service Layer, App Services/Work Flows, and Domain Model as shown in Figure 6.1. In this context, the Domain Model should be interpreted as a conceptual framework consisting of domain object classes and their attributes and associations.

6.3.2.1 The Service Layer

The Service Layer exposes a function as a RESTful web service. Jersey RESTful Web Services framework was used to implement this layer. Configuration details to enable Maven to download Jersey and jersey-json dependencies, were added to the pom.xml file. The jersey-json is a module that provides support for JSON to Jersey. The classes of the Service Layer were annotated with `@Component` Spring annotation. The `@Component` flags a bean so that the component-scanning mechanism can pick and register it with the container [89]. A JavaBean is a specially constructed Java class that is coded according to the JavaBeans API specifications.[11] Each class of the service layer has a dependency on a class in the app services, which is injected automatically using DI.

6.3.2.2 The App Services/Work Flows

The App Services (see Figure 6.1) contains classes that implement the application logic or work flow. The classes were annotated with `@Service` and `@Transactional` and they implement an interface. The `@Service` annotation is a specialisation of the `@Component` annotation and it enables the component-scanning mechanism to discover and register a class with the container. The `@Transactional` annotation indicates that all public methods within a class are transactional. Each class in the App Services has a dependency on a class in the data access layer, which is injected automatically using DI. The methods of the classes are annotated with HTTP methods such as @POST, @PUT, @DELETE, and @GET on order to map client requests to appropriate methods.

6.3.2.3 The Domain Model

The Domain Model contains persistent classes. The classes were annotated with Java persistence annotations, which are `@Entity`, to indicate that a class is an entity and `@Table`, to

[11] http://www.tutorialspoint.com/jsp/jsp_java_beans.htm

map a class to a table in the database. The classes extend the *BaseEntity* class. The fields of the *BaseEntity* are `id` and `uuid`. The `id` has a long data type and was annotated with @Id and @GeneratedValue(strategy=GenerationType.AUTO) Java persistence annotations to indicate that it is the primary key and that its value should be generated automatically. The `uuid` has a string data type and was annotated with @Column(length=36) to indicate that it is a persistent field and its length is limited to 36 characters. The uuid is generated using the randomUUID() method of the *java.util.UUID* class and is used as a key to retrieve a specific record from the database.

Some of the Domain Model classes used include *Target* and *Organisation*. An object of the *Target* class contains information that is used to target campaigns, surveys, events, and news. Some of the fields of the class include `language, gender, occupation, maritalStatus, sexOrientation`, and `location`. The `language, gender, occupation, maritalStatus, sexOrientation` fields were annotated with @Column. The location field has a *Location* class data type, which is a Domain Model class and was annotated with @ManyToOne Java persistence annotation. The annotation indicates that there can be many targets with one location (ie. there is a many-to-one relationship between *Target* and *Location* objects). The fields of *Location* class are `province`, `district`, `municipality`, and `village` and are annotated with @Column and @NotNull. The @NotNull indicates that the fields cannot be null. An object of the *Organisation* class contains information of a health organisation. The fields of the *Organisation* class are `name`, `description`, `addressLine1`, `addressLine2`, `addressLine3`, `phoneNumber`, and `emailAddress`.

6.3.3 Data Access Layer

The data access layer provides access to persistent data. Data persistence in an application is the act of storing data so that it exists even when the application is closed. An example is storing data in a relational database using structured query language (SQL). Version 4.0.1.Final of Hibernate, an open source ORM service implementation, was used to implement data persistence. ORM "is the automated (and transparent) persistence of objects in a Java application to the tables in a relational database, using metadata that describes the mapping between the objects and the database" [8, p. 25]. It mediates an application's interaction with a relational database, leaving the developer to concentrate on the business problem at hand. Configuration details to enable Maven to download required hibernate dependencies were added to the pom.xml file. The dependencies were used for development purposes only and were excluded from the packaged applications

because Wildfly provides the hibernate JAR files. The dependencies were excluded by specifying a value of provided for the scope element, as discussed in Sub-section 6.2.2.2. This is one of the advantages of using Maven because it frees a programmer from extra work of managing dependencies to avoid conflicts.

The classes in the data access layer extend the *BaseDao* class and implement an interface. The classes were annotated with @Repository, which is a specialisation of the @Component annotation. The @Repository annotation works in the same way as the @Component annotation, in addition to making unchecked exceptions [89]. The *BaseDao* contains the `sessionFactory` field, which has a *org.hibernate.SessionFactory* data type that is used to access the database. The field was annotated with @Autowired Spring annotation so that it is injected automatically with a database session.

The components use MySQL as the underlying database to store persistent data. Wildfly was extended with mysql-connector-java-5.1.25-bin.jar file to enable it to support MySQL. Configuration details to enable Maven to download MySQL dependencies were added to pom.xml file. The dependencies were used for development purposes only and were excluded from packaged applications, in the same way as the Hibernate files.

6.4 Implementation of the Dashboard Functions

This section discusses the implementation of the Dashboard functions. The functions were implemented using the layered architecture as introduced in Section 6.3. The presentation layer for all the functions was implemented as discussed in Sub-section 6.3.1, therefore only the business and data access layers are discussed. Several packages were created to organise project files as shown in Figure 6.2. Some of the packages include the *com.reedhousesystems.services.core.dashboard.api*, herein referred to as the API package, which contains interface declarations; the *com.reedhousesystems.services.core.dashboard.resource*, herein referred to as the resource package, which contains classes that implement the JAX-RS interface; the *com.reedhousesystems.services.core.dashboard.dao*, herein referred to as the DAO package, which contains classes that manage database access; the *com.reedhousesystems.services.core.dashboard.service*, herein referred to as the service package, which contains the business logic; and the *com.reedhousesystems.services.core.dashboard.model*, here in referred to as the model package, which contains domain models or persistent classes.

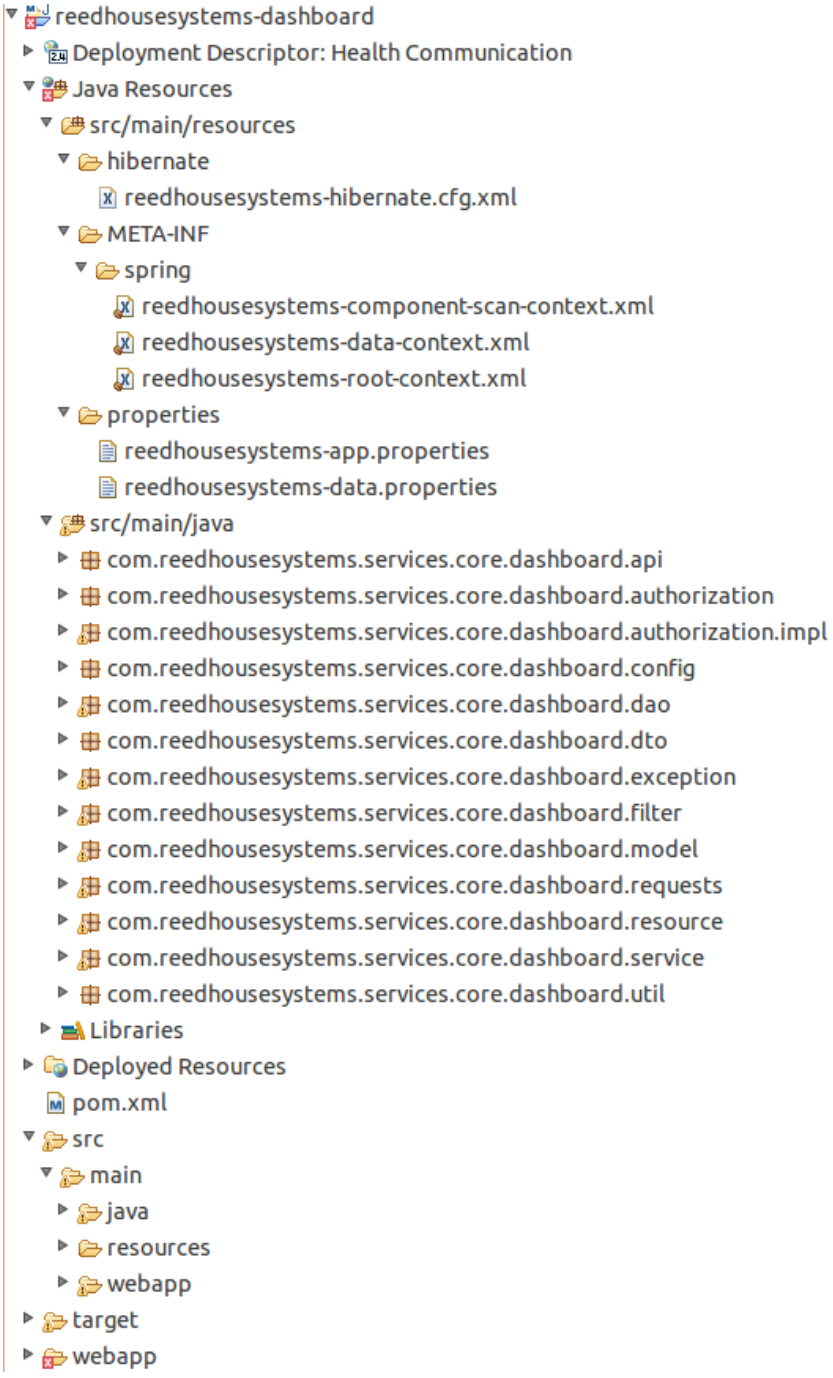

Figure 6.2: Project Structure for the Dashboard

6.4.1 Implementation of the Campaigns Function

The *CampaignResource*, *CampaignServiceImpl*, *Campaign*, and *Message* classes were used to implement the business layer. The *CampaignDaoImpl* class was used to implement the

data access layer. Figures 6.3 and 6.4 are high-level diagrams showing the classes and how they relate to each other.

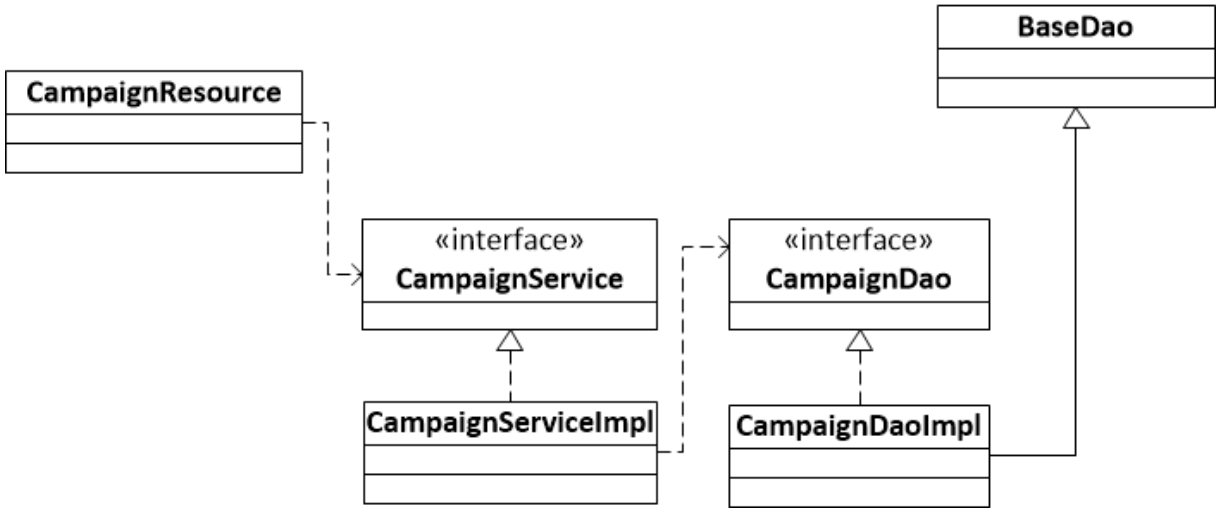

Figure 6.3: Classes used to implement the business and data access layers of the Campaigns function

6.4.1.1 Business Layer

The *CampaignResource* class, located in the resource package, was used to implement the Service Layer to expose the *Campaigns* function as a RESTful web service. It has a `campaignService` field of *CampaignService* interface data type, which is annotated with `@Autowired` so that it is automatically injected with its dependency using DI. The class has methods to create, update, delete, and retrieve campaign data, and set campaign targets and messages.

The *CampaignServiceImpl* class, located in the service package, was used to implement the App Services. It implements the *CampaignService* interface. It has a `campaignDao` field of the *CampaignDao* interface data type, which is annotated with the `@Resource` of the *javax.annotation.Resource* data type so that it is automatically injected with its dependency using DI. The class has methods that correspond to those of the *CampaignResource* class.

The *Campaign* and *Message* classes, located in the model package, were used to implement the Domain Model. The fields of the *Campaign* class are `title, aim, description, slogan, creationDate, startDate, endDate, status, target, user, organisation`, and `messages`. The `title, aim, description, slogan, creationDate, star-`

`tDate, endDate`, and `status` are annotated with @Column and @NotNull. The `target` field has a *Target* class data type, the `organisation` field has a *Organisation* class data type and `user` field has a *User* class data type. The fields are annotated with @JsonBackReference and @ManyToOne. The @JsonBackReference indicates that the annotated field is part of a two-way linkage between fields and that its role is 'child' (or 'back') link.[12] The @ManyToOne indicates that many campaigns can have the same target, can be created by the same user, or can belong to the same organisation, but a campaign can have one and only one target, can be created by one and only one user, and can belong to one and only organisation. The *Organisation* and *User* classes have a campaign field of the *Campaign* class data type that is annotated with @OneToMany and @JsonManagedReference. The @OneToMany annotation is a reverse of the @ManyToOne. The @JsonManagedReference indicates that the annotated field is part of a two-way linkage between fields and that its role is 'parent' (or 'forward') link.[13] The *messages* field has a *java.util.List<Message>* data type and is annotated with @OneToMany and @JsonManagedReference. The fields of the *Message* class are `messageTitle`, `message`, and `campaign`. The `messageTitle` and `message` fields are annotated with @Column and @NotNull. The `campaign` field has a *Campaign* class data type and is annotated with @JsonBackReference and @ManyToOne. Figure 6.4 shows the classes that implement the Domain Model and how they relate to each other.

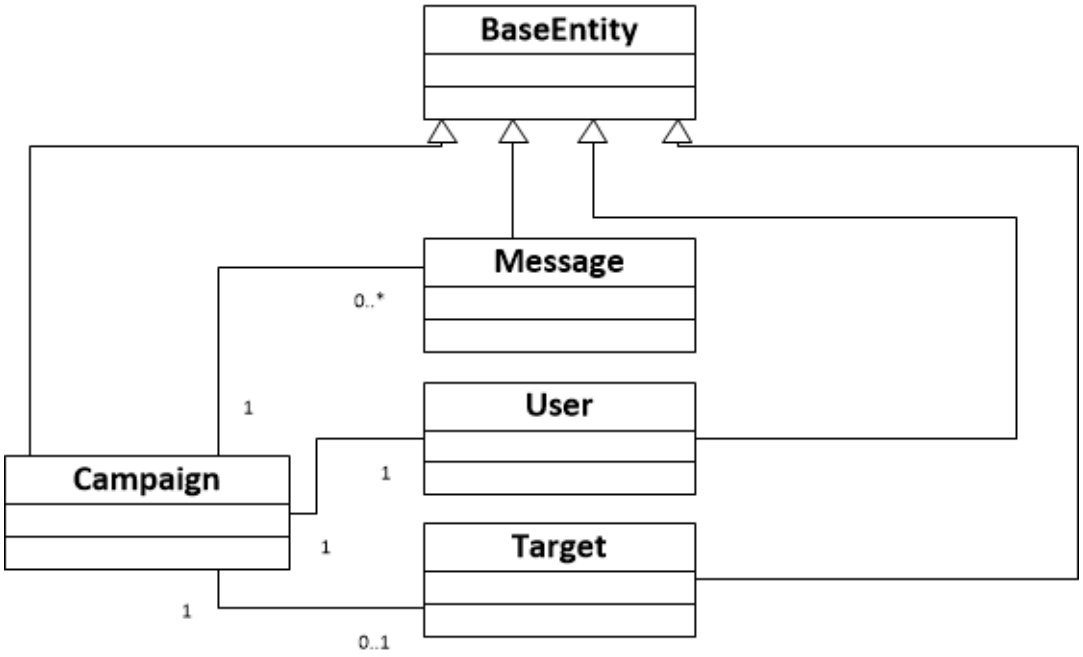

Figure 6.4: Domain Model classes for the *Campaigns* function

[12] http://jackson.codehaus.org/1.6.0/javadoc/org/codehaus/jackson/annotate/JsonBackReference.html
[13] http://jackson.codehaus.org/1.6.0/javadoc/org/codehaus/jackson/annotate/JsonManagedReference.html

6.4.1.2 Data Access Layer

The *CampaignDaoImpl* class, located in the DAO package, was used to implement the data access layer. It extends the *BaseDao* class and implements the *CampaignDao* interface. The class has methods to create, delete, and retrieve campaign information.

6.4.1.3 Publishing a Campaign

According to the requirements specification, publication a campaign retrieves a copy of the information from the Dashboard database and posts it to the HealthMessenger where a record is created in the database. Publishing from one domain (e.g. www.example1.com) to another domain (e.g. www.example2.com) is subject to same-origin restrictions to network requests enforced by user agents (web browsers) [85]. These restrictions are done to prevent a client-side web application running on one domain from executing (POST, GET, DELETE etc.) requests at another domain. The aim is to limit unsafe HTTP requests that can be automatically launched towards destinations on a domain that is different from where the request is coming from. To allow cross-origin network requests, the World Wide Web Consortium (W3C)[14] developed a specification named Cross-Origin Resource Sharing (CORS) to enable an API to make cross-origin requests to resources. The specification was used to implement data interchange between the Dashboard and the HealthMessenger. The implementation used a Jersey response filter and works as follows:

1. when making a cross-domain request (publishing a campaign), a request called a preflight request is made in order to discover whether the cross-origin resource (HealthMessenger) is prepared to accept the request - this is done automatically by the user agent using the OPTIONS HTTP method);

2. the preflight request executes a method in the HealthMessenger, which is annotated with @OPTIONS (see lines 13 - 17 of Listing 6.2) and sends back an HTTP 200 OK status code as a response;

3. the Jersey response filter adds the "Access-Control-Allow-Origin" header to the response, which has a value of the URL of the Dashboard to indicate that it accepts cross-domain requests from that URL (**Note**: The response is cached on the Dashboard user agent for 24 hours and any request made within the 24 hour period is not preceded by a preflight request); and

[14] http://www.w3.org/TR/cors/

4. the preflight request is followed by the actual request to post campaign data to the HealthMessenger.

```
1  @Path("/campaign")
2  @Component
3  @Produces({MediaType.APPLICATION_JSON})
4  public class CampaignResource {
5
6          private CampaignService campaignService;
7
8          @Autowired
9          protected void setCampaignService(CampaignService campaignService) {
10                 this.campaignService = campaignService;
11         }
12
13         @OPTIONS
14         @CorsPreflight
15         public Response optionRequest() {
16                 return Response.ok().build();
17         }
18
19         @POST
20         @Cors
21         @Consumes({MediaType.APPLICATION_JSON})
22         public Response saveOrUpdateCampaign(CampaignDTO campaign) {
23                 campaignService.saveOrUpdateCampaign(campaign);
24                 return Response.noContent().build();
25         }
26
27         @RolesAllowed({"administrator", "authenticated"})
28         @GET
29         public Response getRunningCampaigns(@Context SecurityContext sc) {
30         ExternalUser userMakingRequest = (ExternalUser)sc.getUserPrincipal();
31                 List<CampaignRequest> campaignList = campaignService.getRunningCampaigns(
                       userMakingRequest);
32                 return Response.ok().entity(campaignList).build();
33         }
34 }
```

Listing 6.2: Extract of the source code for CampaignResource class

6.4.2 Implementation of the Surveys Function

The *SurveyResource*, *SurveyServiceImpl*, *Survey*, *Question*, and *AnswerOptions* classes were used to implement the business layer. The *SurveyDaoImpl* class was used to implement the data access layer. Figures 6.5 and 6.6 are high-level diagrams showing the classes and how they relate to each other.

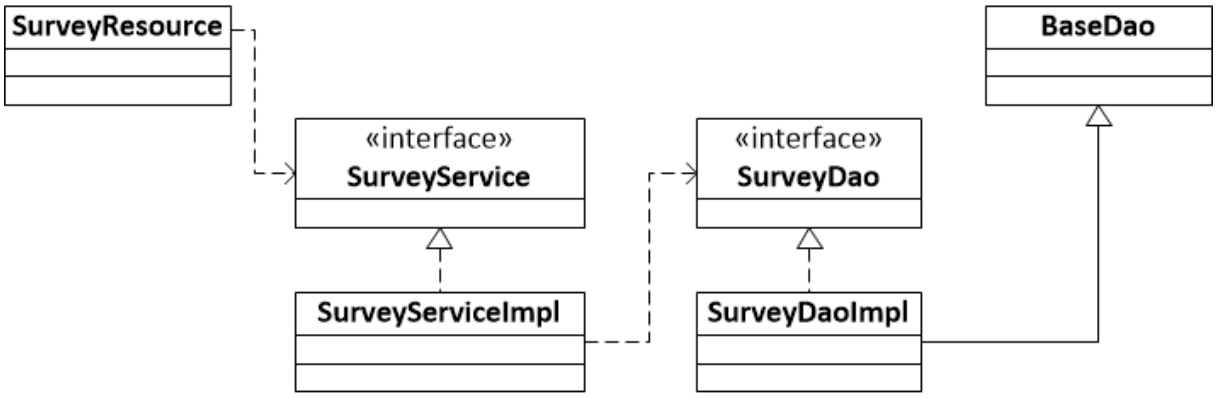

Figure 6.5: Classes used to implement the business and data access layers of the *Surveys* function

6.4.2.1 Business Layer

The *SurveyResource* class, located in the resource package, was used to implement the Service Layer to expose the *Surveys* function as a RESTful web service. It has a `surveyService` field of the *SurveyService* interface data type, which is annotated with @Autowired. The class has methods to add, update, delete, and retrieve a survey, add questions and set survey target.

The *SurveyServiceImpl* class, located in the service package, was used to implement the App Services. It implements the *SurveyService* interface. It has a `surveyDao` field of the *SurveyDao* interface data type, which is annotated with the @Resource. The class has methods that correspond to those of the *SurveyResource* class.

The *Survey*, *Question*, and *AnswerOptions* classes, located in the model package, implement the Domain Model. The fields of the *Survey* class are `title`, `description`, `status`, `creationDate`, `startDate`, `endDate`, `target`, `organisation`, `questions`, and `user`. The `title`, `description`, `status`, `creationDate`, `startDate`, and `endDate` fields are annotated with @Column and @NotNull. The `questions` field has a *java.util.List<Question>* data type. It is annotated with @JsonBackReference and @OneToMany. The fields of the *Question* class are `question`, `questionType`, `survey`, and `answerOptions`. The `survey` field has a *Survey* class data type and is annotated with @JsonBackReference and @ManyToOne. The `answerOptions` has a the *AnswerOptions* class data type and is annotated with @JsonManagedReference and @OneToMany. The fields of the *AnswerOptions* class are `question` and `answerOption`. The `question` has a *Question* class data type and is annotated with @JsonBackReference and @ManyToOne. Figure 6.6 is a high-level diagram of the domain classes used to implement the *Surveys* function.

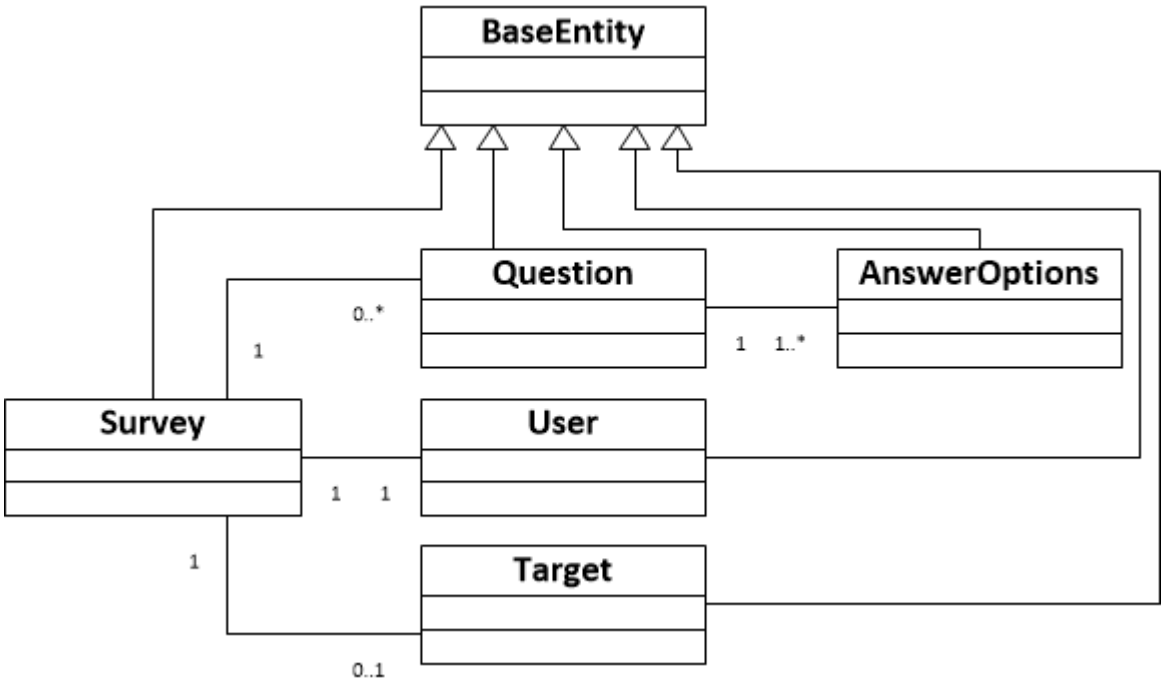

Figure 6.6: Domain Model classes used to implement the *Surveys* function

6.4.2.2 Data access Layer

The *SurveyDaoImpl* class, located in the DAO package, was used to implement the data access layer. It extends the *BaseDao* class and implements the *SurveyDao* interface. It has methods to add, delete, and retrieve surveys.

6.4.3 Implementation of the Events Function

The *EventResource*, *EventServiceImpl*, and *Event* classes were used to implement the business layer. The *EventsDaoImpl* class was used to implement the data access layer. Figures 6.7 and 6.8 are high-level diagrams showing the classes and how they relate to each other.

6.4.3.1 Business Layer

The *EventResource* class, located in the resource package, was used to implement the Service Layer to expose the Events function as a RESTful web service. It has the `eventService` field of the *EventService* interface data type, which is annotated with

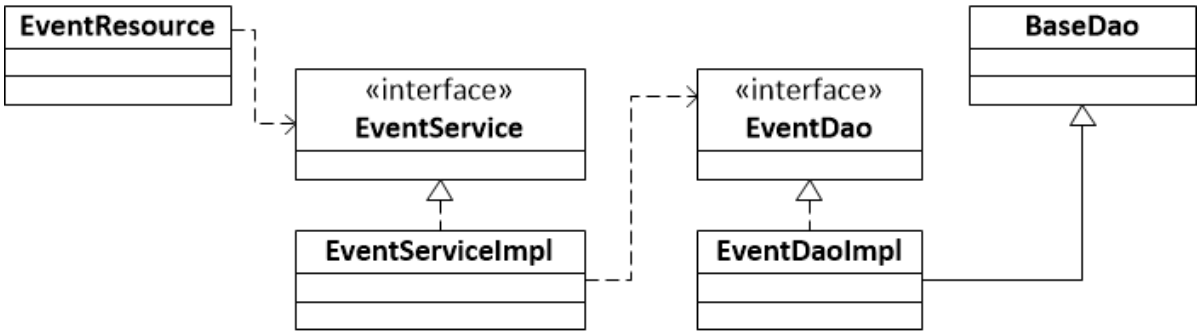

Figure 6.7: Classes used to implement the business and data access layers of the *Events* function

@Autowired. The class has methods to add, update, delete, and retrieve an event, and to set event targets.

The *EventServiceImpl* class, located in the service package, was used to implement the App Services. It implements the *EventService* interface. It has an `eventDao` field of the *EventDao* interface data type, which is annotated with the @Resource. The class has methods that correspond to those of the *EventResource* class.

The *Event* class, located in the model package, was used to implement the Domain Model. The fields of the class are `name`, `description`, `status`, `creationDate`, `startDate`, `endDate`, `target`, `organisation`, and `user`. The `name`, `description`, `status`, `creationDate`, `startDate`, and `endDate` fields are annotated with @Column and @NotNull. Figure 6.8 is a high-level diagram showing the domain model classes used to implement the *Events* function and how they relate to each other.

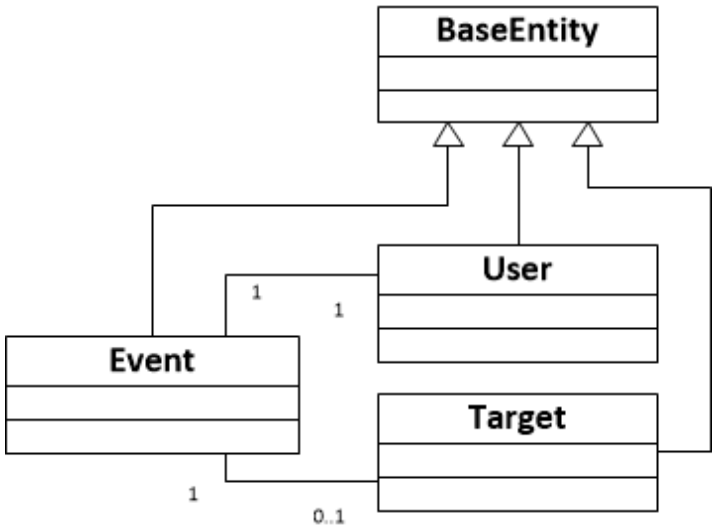

Figure 6.8: Domain Model classes for the *Events* function

6.4.3.2 Data access Layer

The *EventDaoImpl* class, located in the DAO package, was used to implement the data access layer. It extends the *BaseDao* class and implements the *EventDao* interface. It has methods to add, delete, and retrieve events.

6.4.4 Implementation of the News Function

The *NewsResource*, *NewsServiceImpl*, and *News* classes were used to implement the business layer. The *NewsDaoImpl* class was used to implement the data access layer. Figures 6.9 and 6.10 are high-level diagrams showing the classes and how they relate to each other.

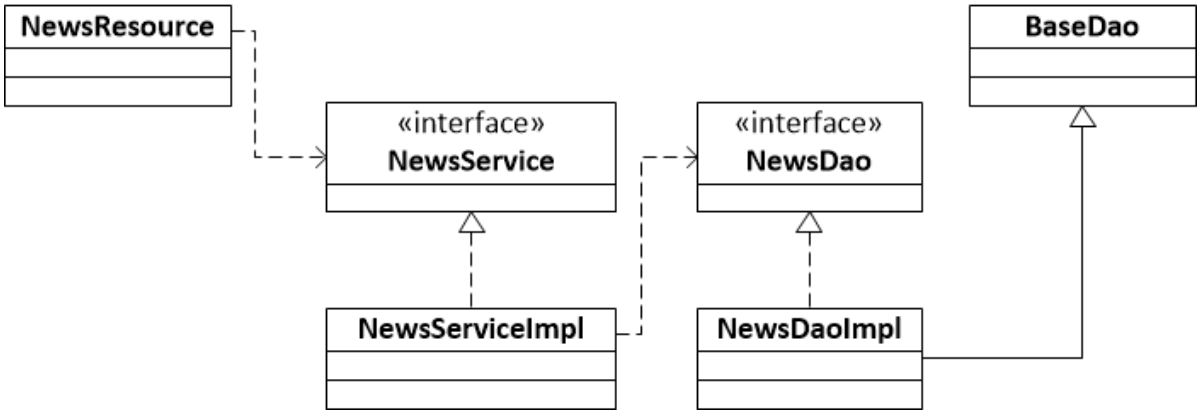

Figure 6.9: Classes used to implement the business and data access layers of the *News* function

6.4.4.1 Business Layer

The *NewsResource* class, located in the resource package, was used to implement the Service Layer to expose the *News* function as a RESTful web service. It has a `newsService` field of the *NewsService* interface data type, which is annotated with `@Autowired`. The class has methods to add, update, delete, and retrieve news, and to set news target.

The *NewsServiceImpl* class, located in the service package, was used to implement the App Services. It implements the *NewsService* interface. It has a `newsDao` field of the *NewsDao* interface data type, which is annotated with `@Resource`. The class has methods that correspond to those of the *NewsResource* class.

The *News* class, located in the model package, was used to implement the Domain Model. The fields of the class are `title, content, author, status, creationDate, target, organisation`, and `user`. The `title, content, author, status`, and `creationDate` fields are annotated with `@Column` and `@NotNull`. Figure 6.10 is a high-level diagram showing the relationship between the domain model classes used to implement the *News* function and how they relate to each other.

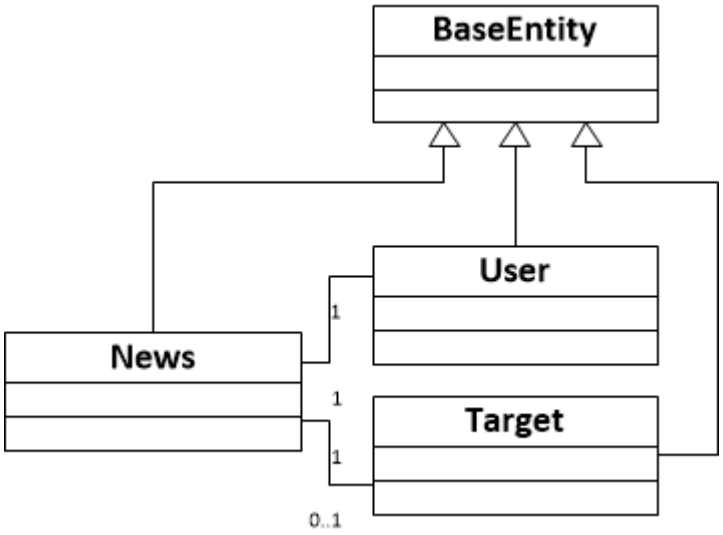

Figure 6.10: Domain Model classes used to implement the *News* function

6.4.4.2 Data access Layer

The *NewsDaoImpl* class, located in the DAO package, was used to implement the data access layer. It extends the *BaseDao* class and implements the *NewsDao* interface. It has methods to add, delete, and retrieve news.

6.4.5 Implementation of the Topics Function

The *TopicResource*, *TopicServiceImpl*, *Topic*, and *TopicDetails* classes were used to implement the business layer of the *Topics* function. The *TopicDaoImpl* class was used to implement the data access layer. Figures 6.11 and 6.12 are high-level diagrams showing the classes and how they relate to each other.

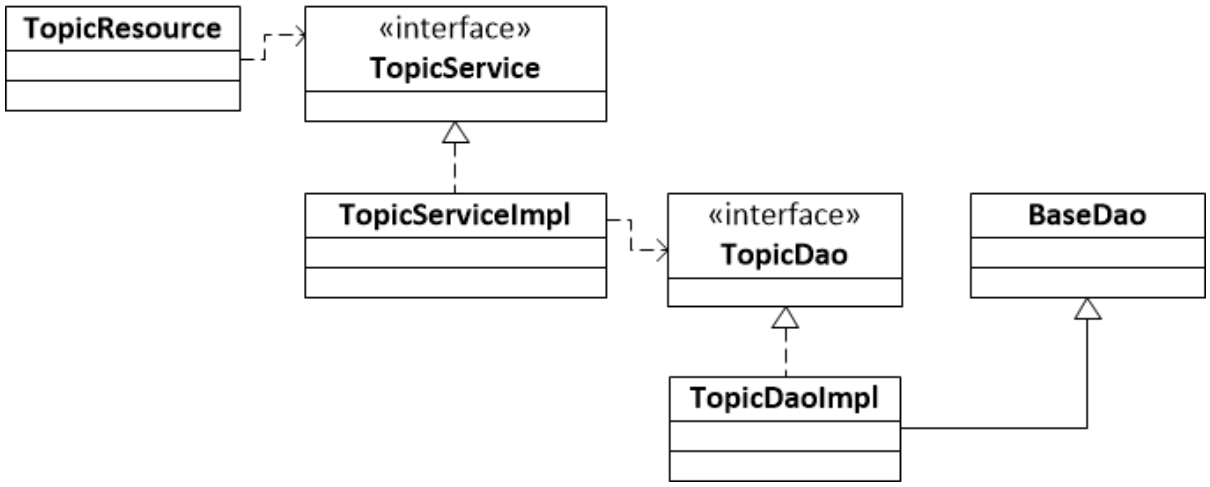

Figure 6.11: Classes used to implement the business and data access layers of the *Topics* function

6.4.5.1 Business Layer

The *TopicResource* class, located in the resource package, was used to implement the Service Layer to expose the *Topics* function as a RESTful web service. It has a `topicService` field of the *TopicService* interface data type, which is annotated with `@Autowired`. The class has methods to add, update, delete, and retrieve a topic.

The *TopicServiceImpl* class, located in the service package, was used to implement the App Services. It implements the *TopicService* interface. It has a `topicDao` field of the *TopicDao* interface data type, which is annotated with the `@Resource`. The class has methods that correspond to those of the *TopicResource* class.

The *Topic and TopicDetails* classes, located in the model package, were used to implement the Domain Model. The fields of the *Topic* class are `title, creationDate, language, topicDetails`, and `user`. The `title, creationDate`, and `language` fields are annotated with `@Column` and `@NotNull`. The `topicDetails` field has a *java.util.List<TopicDetails>* data type. The fields of the *TopicDetails* class are `sectionName`, `sectionDescription`, and `topic`. The `topic` field has a *Topic* class data type and is annotated with `@JsonBackReference` and `@ManyToOne`. Figure 6.12 is a high-level diagram showing the domain model classes used to implement the *Topics* function and how they relate to each other.

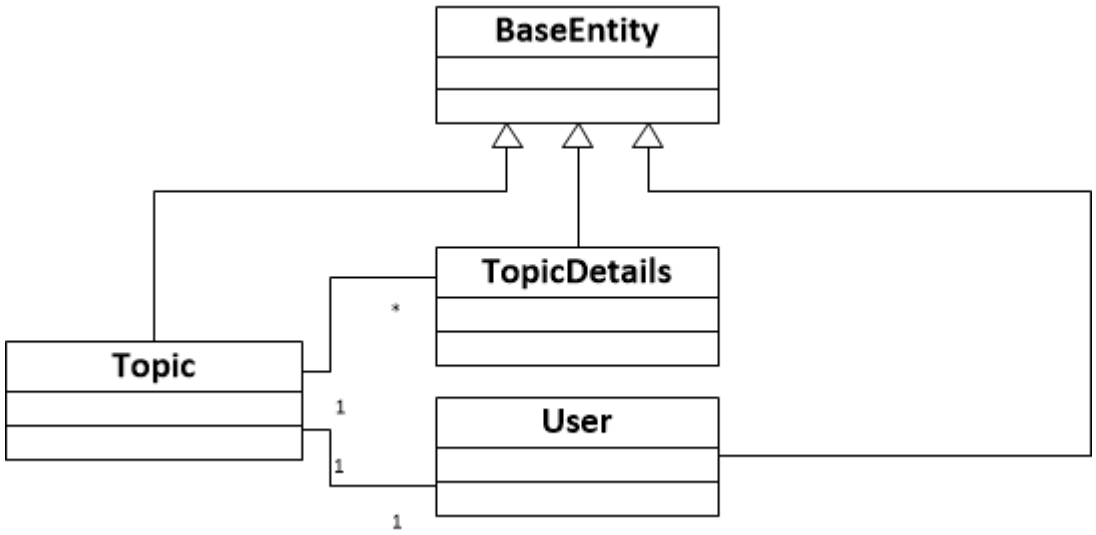

Figure 6.12: Domain Model classes for the *Topics* function

6.4.5.2 Data Access Layer

The *TopicDaoImpl* class, located in the DAO package, was used to implement the data access layer. It extends the *BaseDao* class and implements the *TopicDao* interface. It has methods to add, delete, and retrieve topics.

6.4.6 Implementation of the Facilities and Services Functions

The *Facilities* and *Services* functions, collectively referred to as the *Facilities* function, were implemented together. The *FacilityResource*, *FacilityServiceImpl*, *Facility*, and *Services* classes were used to implement the business layer. The *FacilityDaoImpl* class was used to implement the data access layer. Figures 6.13 and 6.14 are high-level diagrams showing the classes and how they relate to each other.

6.4.6.1 Business Layer

The *FacilityResource* class, located in the resource package, was used to implement the Service Layer to expose the *Facilities* function as a RESTful web service. It has a `facilityService` field of the *FacilityService* interface data type, which is annotated with `@Autowired`. The class has methods to add, update, delete, and retrieve facilities and services, and to add and remove services from a facility.

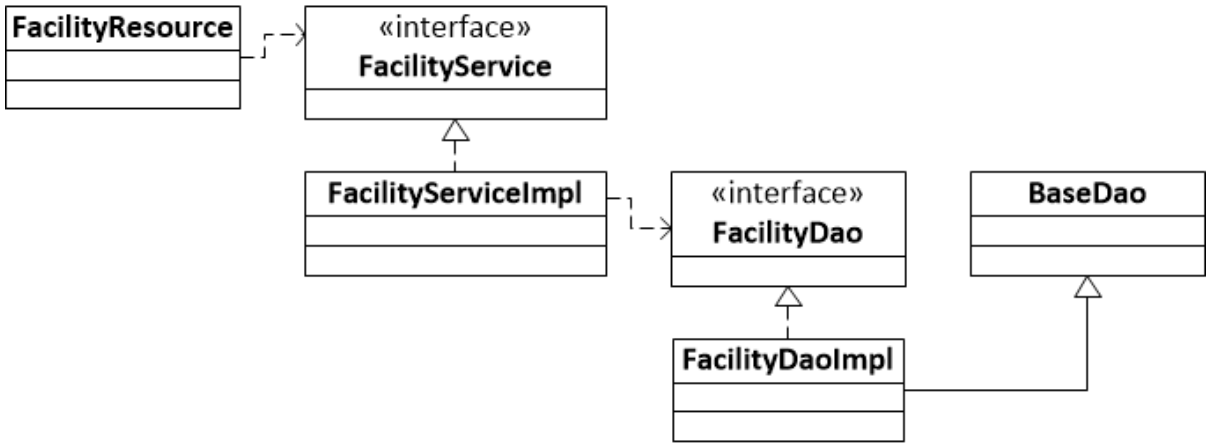

Figure 6.13: Classes used to implement the business and data access layers of the *Facilities* function

The *FacilityServiceImpl* class, located in the service package, was used to implement the App Services. It implements the *FacilityService* interface. It has a `facilityDao` field of the *FacilityDao* interface data type, which is annotated with the @Resource. The class has methods that correspond to those of the *FacilityResource* class.

The *Facility* and *Services* classes, located in the model class, were used to implement the Domain Model. The fields of the *Facility* class are `name, type, creationDate, location, services`, and `user`. The `name, type`, and `creationDate` fields are annotated with @Column and @NotNull. The `location` field has a *Location* class data type, which is a domain model class and is annotated with @ManyToOne. The `services` field has a *java.util.List<Services>* data type and is annotated with @ManyToMany. The @ManyToMany annotation indicates that a service can be offered by different facilities and a facility can offer many services. The fields of the *Services* class are `name, description`, and `type`. The fields are annotated with @Column and @NotNull. Figure 6.14 is a high-level diagram showing the domain model classes used to implement the *Facilities* function and how they relate to each other.

6.4.6.2 Data Access Layer

The *FacilityDaoImpl* class, located in the DAO package, was used to implement the data access layer. It extends the *BaseDao* and implements the *FacilityDao* interface. It has methods to add, delete, and retrieve facilities.

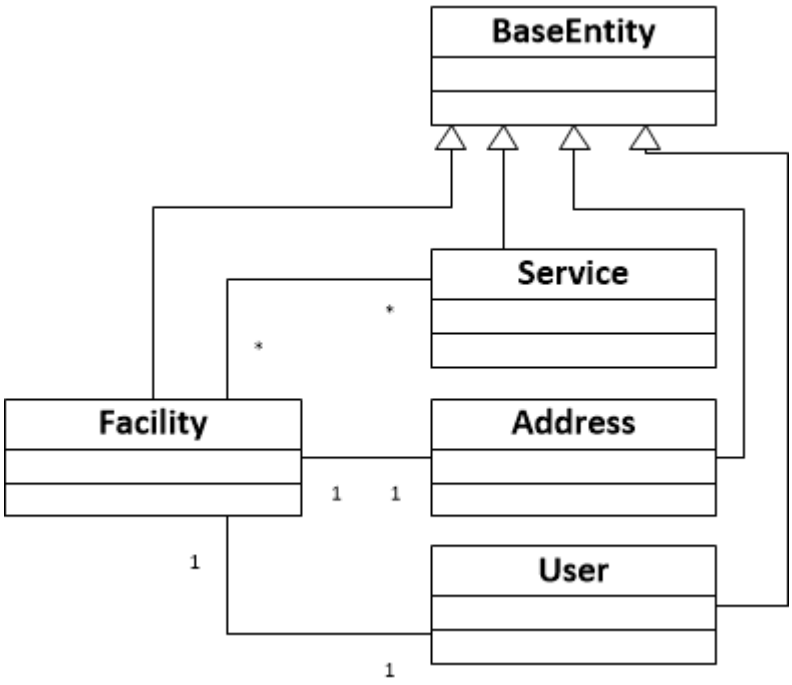

Figure 6.14: Domain Model classes for the *Facilities* function

6.4.7 Implementation of the Users Function

The *UserResource*, *UserServiceImpl*, *VerificationTokenServiceImpl*, *VerificationToken*, *SessionToken*, and *User* classes were used to implement the business layer. The *UserDaoImpl* and *VerificationTokenDaoImpl* classes were used to implement the data access layer. Figures 6.15 and 6.16 are high-level diagrams of the classes and how they relate to each other.

6.4.7.1 Business Layer

The *UserResource* class, located in the resource package, was used to implement the Service Layer to expose the *Users* function as a RESTful web service. It has a `userService` field of the *UserService* interface data type and a `verificationTokenService` field of the *VerificationTokenService* interface data type, which are annotated with `@Autowired`. The *UserResource* class has methods to create, update, delete, retrieve, login, and logout a user.

The *UserServiceImpl* and *VerificationTokenServiceImpl* classes, located in the service package, were used to implement the App Services. They implement the *UserService* interface and *VerificationTokenService* interface respectively. The *UserServiceImpl* has

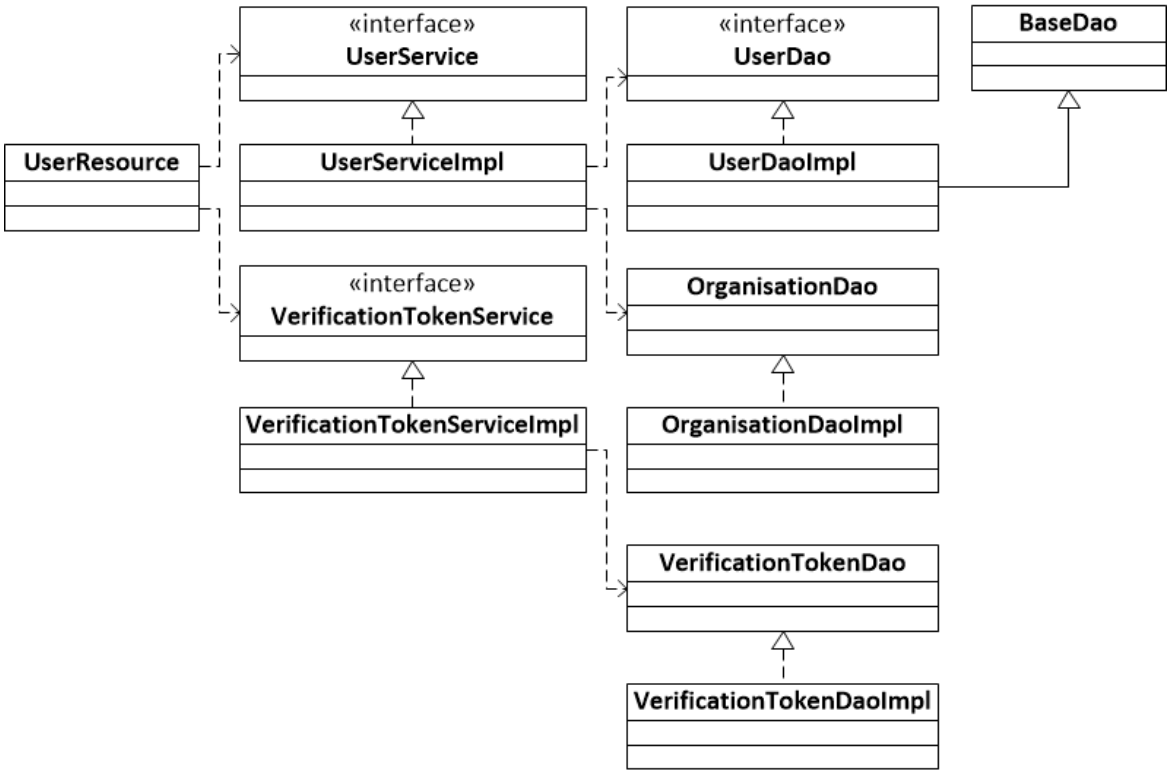

Figure 6.15: Classes used to implement the business and data access layers of the *Users* function

`userDao` and `organisationDao` fields of the *UserDao* and *OrganisationDao* interface data types respectively, which are annotated with `@Resource`. The methods of the *UserServiceImpl* correspond to those of the *UserResource* class. The *VerificationTokenServiceImpl* class is used to manage tokens obtained when a user is successfully authenticated.

The *User* and *VerificationToken* classes, located in the model package, were used to implement the Domain Model. The fields of the *User* class are `firstName, lastName, password, loginAttemps, salt, contact, organisation, role, verificationTokens`, and `sessions`. The `firstName, lastName, userName, password, loginAttempts`, and `salt` fields are annotated with `@Column` and `@NotNull`. The `password` field is annotated with `@Length(min=8)` to set the minimum length of a password to 8 characters. This is part of the data input validation check, which is done on the server side to complement the check at the presentation layer level. The `contact` field has a *Contact* class data type, which is a domain model class. It is annotated with `@Embedded` so that the fields of the *Contact* class are embedded within the *User* class. The fields of the *Contact* class are `emailAddress` and `phoneNumber`. The `organisation` field has a *Organisation* class data type. It is annotated with with `@JsonBackReference` and `@ManyToOne`. The `role` field has a *Role* data type, which is an *Enum* and is used to store the role of

an authenticated user. It is annotated with @Enumerated(EnumType.string) to indicate that it has a *javax.persistence.Enumerated* data type. The `salt` field is used to store a text called a salt that is used to hash a password. The `verificationToken` field has a *VerificationToken* class data type. The fields of the *VerificationToken* class are `token`, `expiryDate`, and `verified`. The `token` field is annotated with @Column(length=36) to indicate that it is a persistent field limited to 36 characters. The `sessions` field has a *java.util.SortedSet<SessionToken>* data type and is annotated with @OneToMany. The fields of the *SessionToken* class are `token`, `timeCreated, lastUpdate`, and `user`. The `token`, `timeCreated`, and `lastUpdate` are annotated with @Column. The `user` field has a *User* class data type and is annotated with @ManyToOne. Figure 6.14 is a high-level diagram showing the domain model classes used to implement the *Users* function and how they relate to each other.

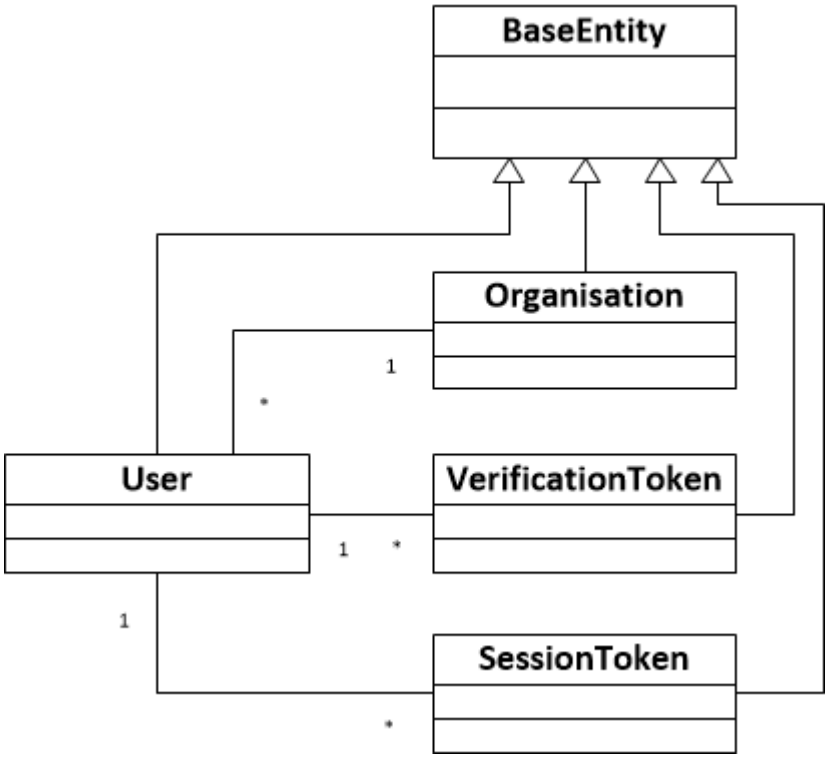

Figure 6.16: Domain Model classes for the *Users* function

6.4.7.2 Data Access Layer

The *UserDaoImpl* and *VerificationTokenDaoImpl* classes, located in the DAO package, were used to implement the data access layer. They extend the *BaseDao* class and implement the *UserDao* interface and the *VerificationTokenDao* interface respectively. The

UserDaoImpl class has methods to add, delete, and retrieve users. The *VerificationTokenDaoImpl* class is used to retrieve a token that is generated when a user is successfully authenticated.

6.4.7.3 User Authentication

User authentication in the Dashboard uses a username and password combination. The minimum length of a password is 8. Token-based authentication was used to implement the authentication. According to the World Wide Web Consortium (W3C) [90], token-based authentication works as follows:

- a user provides a username and password;
- a server verifies the credentials by comparing them with those stored on the server;
- if the credentials match any of those stored on the server, a token is generated and sent to the user;
- in the subsequent requests for resources, the user provides the token instead of a username and password; and
- the token expires after a specific period of time but it is renewable.

Token-based authentication was implemented as follows:

- a user enters a username and password;
- a check is made to ensure that the username and password are provided (A blank username or password is not acceptable and results in an error);
- the credentials are sent to the server for verification;
- the username and password pair is compared with username and password pairs in the database;
- a match with one of the pairs results in a successful authentication otherwise it is unsuccessful;

- if it is unsuccessful, the user is informed, otherwise a token is generated and sent to the user. (**Note**: Other information sent with the token are a unique user ID and the first name and last name. The token and the unique user ID are used in the subsequent requests for resources as discussed in Section 6.4.7.4. The first name and last name are used to display the name of the current logged in user. These are stored using store.js,[15] a JavaScript library for storing data locally, which is browser independent and does not use cookies).

6.4.7.4 User Authorisation

Authorisation is the process of verifying that a user has the rights to perform an action or access resources in an application [15]. An example is determining whether a user has the rights to create other users or access other users' information. Implementation of user authorisation starts with verification of the token that is sent with a user's request. When a user sends an HTTP request to a server, information used to authorise a user to execute a method is sent along with the request in the HTTP headers. The headers, which are "Authorisation", "x-healthcomm-date", and "nonce" (see Lines 12 - 16 of Code Listing 6.3), are sent with every request that requires authorisation. The "Authorisation" header (see Line 13 of Code Listing 6.3) is derived from a combination of a unique user ID obtained after a user has been successfully authenticated and a result of Sha256 encoding followed by Base64 encoding of the session token obtained after a user has been successfully authenticated, a relative URL and HTTP method of the request, an ISO date, and a nonce value (see Line 6 of Code Listing 6.3). The ISO date and nonce values are derived as shown in lines 3 and 4 of Code Listing 6.3. CryptoJS,[16] a cryptographic algorithm implemented in JavaScript, is used to implement the Sha256 and Base64 hashing.

```
1  //A function to to perform a HTTP POST request
2  var postAuth = function (url, data, success, error) {
3    var time = get_iso_date()
4    var nonce = makeRandomString()
5    var string_to_hash = $.cookie('token') + ':' + url + ',POST,' + time + "," + nonce;
6    var authorisation = $.cookie('userId') + ':' + hash(string_to_hash)
7    $.ajax({
8      url: url,
9      type: "POST",
10     contentType: "application/json",
11     data: JSON.stringify(data),
12     headers: {
13       'Authorisation' : authorization,
14       'x-healthcomm-date' : time,
```

[15] https://github.com/marcuswestin/store.js/
[16] http://code.google.com/p/crypto-js/

```
15        'nonce' : nonce
16      },
17      dataType: "json",
18      success : success,
19      error : error
20    });
21  }
22
23  //Creating a nonce
24  function makeRandomString() {
25    return Math.random().toString(36).substring(2, 15) +
26      Math.random().toString(36).substring(2, 15);
27  }
28
29  //A function to derive an ISO date
30  /**
31  * Return the current time as an ISO 8061 Date
32  * @return {string} 2012-06-30T12:00:00+01:00
33  */
34
35  var get_iso_date = function () {
36    var d = new Date();
37    function pad(n) {
38     return n<10 ? '0'+n : n
39    }
40    return d.getUTCFullYear() + '-'
41      + pad(d.getUTCMonth()+1) + '-'
42      + pad(d.getUTCDate()) + 'T'
43      + pad(d.getUTCHours()) + ':'
44      + pad(d.getUTCMinutes()) + ':'
45      + pad(d.getUTCSeconds()) + 'Z'
46  }
47
48  /**
49  * SHA256, then base64 encode a string
50  * @param {string}
51  * @return {string}
52  */
53  var hash = function (string) {
54    var hash = CryptoJS.SHA256(string)
55    return hash.toString(CryptoJS.enc.Base64)
56  }
```

Listing 6.3: Code listing for a POST transaction

On the server, a SecurityContextFilter class that implements *com.sun.jersey.spi.container.ResourceFilter* and the *com.sun.jersey.spi.container.ContainerRequestFilter* interfaces handles HTTP requests. An in-bound HTTP request of *com.sun.jersey.spi.container.ContainerRequest* data type is used to read header information, the relative path and the HTTP method of the request (see lines 20-23 of Code Listing 6.4). The user ID, which is sent in the "Authorisation" header (Line 13 of Code Listing 6.3) is used to retrieve the information of the user making the request. The token generated when a user is successfully authen-

ticated is retrieved from the user information. A Sha256 hash followed by a Base64 hash of the token, URL, HTTP method, ISO date, and nonce values retrieved in lines 20-23 of Code Listing 6.4 are generated in a similar manner to line 5 of Code Listing 6.3. A encodeBase64() method of the *org.apache.commons.codec.binary.Base64* class is used to calculate the Base64 hash and a sha256() method of the *class org.apache.commons.codec.digest.DigestUtils* class is used to calculate the Sha256 hash. The hash obtained is compared with the hash generated in Line 5 of Code Listing 6.3, which is sent in the 'Authorisation' header. A match of the two hashes indicates that the token is correct and belongs to the user making the request. The token is updated as per bullet 5 of W3C specification in Section 6.4.7.3 and the user is allowed to proceed to the second phase of user authorisation, which is discussed next.

```
1  ...
2  import com.sun.jersey.spi.container.ContainerRequest;
3  import com.sun.jersey.spi.container.ContainerRequestFilter;
4  import com.sun.jersey.spi.container.ContainerResponseFilter;
5  import com.sun.jersey.spi.container.ResourceFilter;
6  ..
7
8  public class SecurityContextFilter implements ResourceFilter, ContainerRequestFilter {
9
10     private static final Logger LOG = LoggerFactory.getLogger(SecurityContextFilter.class
           );
11     protected static final String HEADER_AUTHORIZATION = "Authorization";
12     protected static final String HEADER_DATE = "x-healthcomm-date";
13     protected static final String HEADER_NONCE = "nonce";
14
15     @Autowired
16     @Qualifier("requestSigningAuthorizationService")
17     private AuthorizationService authorizationService;
18
19     public ContainerRequest filter(ContainerRequest request) {
20         String authToken = request.getHeaderValue(HEADER_AUTHORIZATION);
21         String requestDateString = request.getHeaderValue(HEADER_DATE);
22         String nonce = request.getHeaderValue(HEADER_NONCE);
23         AuthorizationRequestContext context = new AuthorizationRequestContext(request.
               getPath(), request.getMethod(), requestDateString, nonce, authToken);
24         ExternalUser externalUser = authorizationService.authorize(context);
25         request.setSecurityContext(new SecurityContextImpl(externalUser));
26         return request;
27     }
28
29     public ContainerRequestFilter getRequestFilter() {
30         return this;
31     }
32
33     public ContainerResponseFilter getResponseFilter() {
34         return null;
35     }
36
37 }
```

Listing 6.4: Code listing for Security Context Filter

The second phase of user authorisation is made at the method level to ensure that only authorised users execute methods. Annotations imported from *javax.annotation.security*[17] package are used to annotate the methods. The package contains security common annotations used to annotate class methods to indicate who is allowed to execute the methods. The @RolesAllowed annotation is used to list the roles that are permitted to execute specified method(s) in an application.[18] The @DenyAll annotation specifies that no security roles are allowed to execute the specified method(s).[19] The @PermitAll annotation specifies that all security roles are allowed to execute the specified method(s).[20] Line 27 of Listing 6.2 is an example of a method annotated with @RolesAllowed, listing 'administrator' and 'authenticated' as the roles allowed to execute the *getRunningCampaigns()* method. All methods that require authorisation are annotated with the *javax.annotation.security* annotations to ensure that only allowed roles execute them. If a user is not of the authorised role, a message is sent to inform him or her.

```
1  @Path("/campaign")
2  @Component
3  @Produces({MediaType.APPLICATION_JSON})
4  public class CampaignResource {
5
6          private CampaignService campaignService;
7
8          @Autowired
9          protected void setCampaignService(CampaignService campaignService) {
10                 this.campaignService = campaignService;
11         }
12
13         @OPTIONS
14         @CorsPreflight
15         public Response optionRequest() {
16                 return Response.ok().build();
17         }
18
19         @POST
20         @Cors
21         @Consumes({MediaType.APPLICATION_JSON})
22         public Response saveOrUpdateCampaign(CampaignDTO campaign) {
23                 campaignService.saveOrUpdateCampaign(campaign);
24                 return Response.noContent().build();
25         }
26
```

[17] http://docs.oracle.com/javaee/6/api/javax/annotation/security/package-summary.html
[18] http://docs.oracle.com/javaee/6/api/javax/annotation/security/RolesAllowed.html
[19] http://docs.oracle.com/javaee/6/api/javax/annotation/security/DenyAll.html
[20] http://docs.oracle.com/javaee/6/api/javax/annotation/security/PermitAll.html

```
27          @RolesAllowed({"administrator", "authenticated"})
28          @GET
29          public Response getRunningCampaigns(@Context SecurityContext sc) {
30          ExternalUser userMakingRequest = (ExternalUser)sc.getUserPrincipal();
31                  List<CampaignRequest> campaignList = campaignService.getRunningCampaigns(
                        userMakingRequest);
32                  return Response.ok().entity(campaignList).build();
33          }
34  }
```

Listing 6.5: Code listing showing security annotation

To determine if a user has a the role listed in the @RolesAllowed annotation, the *isUserInRole(java.lang.String role)* method of the *SecurityContextImpl* class that implements the *javax.ws.rs.core.SecurityContext* interface (see Code Listing 6.6) is used. The *javax.ws.rs.core.SecurityContext* is an injectable interface that provides access to security related information.[21] The interface has the *getAuthenticationScheme()*, *getUserPrincipal()*, *isSecure()*, and *isUserInRole(java.lang.String role)* methods, which are implemented by the *SecurityContextImpl*. The *getAuthenticationScheme()* method returns a string value of the authentication scheme used to protect a resource. The *getUserPrincipal()* method returns a *java.security.Principal* object containing the name of the current authenticated user. The isSecure() method returns a boolean indicating whether a request was made using a secure channel, such as HTTPS. The *isUserInRole(java.lang.String role)* method returns a boolean indicating whether the authenticated user is included in the specified logical 'role'. If the *isUserInRole(java.lang.String role)* method returns false, a message is sent to inform the user.

```
1   ...
2   import java.security.Principal;
3   import javax.ws.rs.core.SecurityContext;
4   ...
5
6   /**
7    * Implementation of {@link javax.ws.rs.core.SecurityContext}
8    *
9    */
10
11  public class SecurityContextImpl implements SecurityContext {
12      private final ExternalUser user;
13
14      public SecurityContextImpl(ExternalUser user) {
15          this.user = user;
16      }
17
18      public Principal getUserPrincipal() {
19          return user;
```

[21] http://docs.oracle.com/javaee/6/api/javax/ws/rs/core/SecurityContext.html

```
20      }
21
22      public boolean isUserInRole(String role) {
23          if(user == null) {
24              throw new InvalidAuthorizationHeaderException();
25          }
26          return user.getRole().equalsIgnoreCase(role);
27      }
28
29      public boolean isSecure() {
30          return false;
31      }
32
33      public String getAuthenticationScheme() {
34          return SecurityContext.BASIC_AUTH;
35      }
36  }
```

Listing 6.6: Implementation of SecurityContext interface

In addition to annotating methods with *javax.annotation.security* annotations, the functions that can be performed by different roles are accessible using separate login pages. A system administrator has a separate login page to perform system settings. If a user with the authenticated role tries to log in to the page, an error message is generated informing him or her that the login page is for system administrators only and that he or she should login in to an appropriate page.

6.4.8 Report Generation

The reports that can be generated in the HealthAware include a graphical view of the number of people reached by a campaign, who have registered for an event and who have read news article. Other reports include a graphical view of the number of hits on a campaign, event, or news article, responses to a survey and the demographic information of the users who have been reached by a sensitisation campaign, or completed a survey. A hit is a click on a campaign, event, or news link. Generation of reporting data is discussed together with the implementation of the HealthMessenger.

6.5 Implementation of the HealthMessenger Functions

The previous section discussed the implementation of the Dashboard functions. This section discusses the implementation of the HealthMessenger functions. As discussed

in Sub-section 5.4.1, the HealthMessenger should enable users to access the health information that is created and published using the Dashboard in either a pull or push mode. The functions were implemented using the layered architecture and mirrored the implementation of the Dashboard functions. A project to implement the HealthMessenger functions was created similarly to that used to implement the Dashboard functions. Several packages were created to organise project files as shown in Figure 6.17.

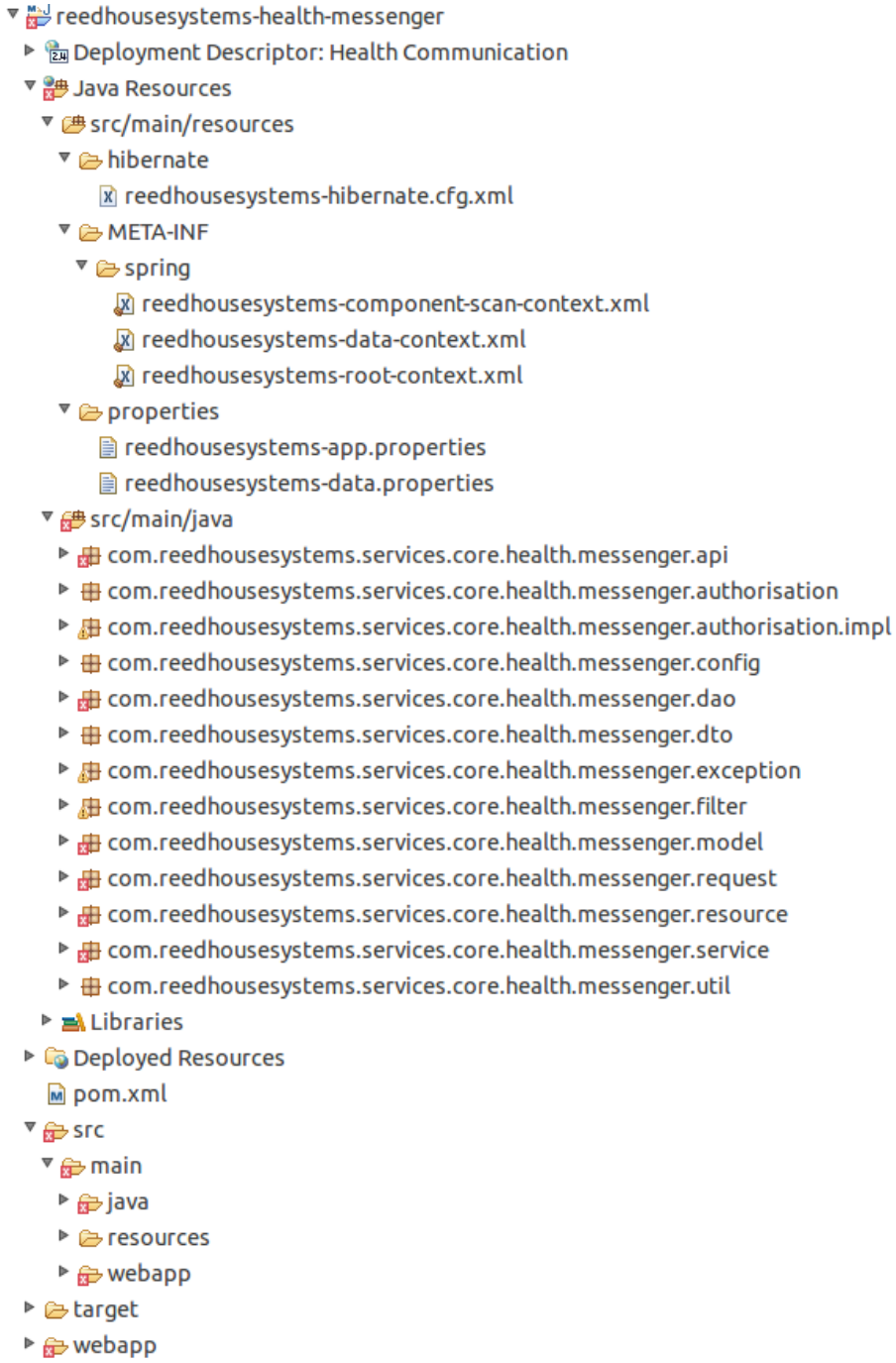

Figure 6.17: Project files for the implementation of the HealthMessenger

6.5.1 Campaigns Function

The *Campaigns* function was implemented using the classes similar to those used to implement the *Campaigns* function of the Dashboard. The difference in their implementation is that a HealthMessenger user cannot add, update, or delete a campaign.

6.5.1.1 Creating a Campaign

A campaign is created in the HealthMessenger when a user publishes it in the Dashboard. It is updated when a Dashboard user updates and republishes it.

6.5.1.2 Logging of Auditing Data

Every user access to a campaign is logged to enable the organisation running the campaign to generate campaign reach reports. The *Campaigns* class of the Dashboard was modified to include the `users` field, which has a *java.util.List<User>* data type. The field replaced the `user` field of the *User* class data type. The field was annotated with `@ManyToMany`. Only one instance of user access to a campaign is logged. When a user accesses a campaign, an object containing the user's details is created and added a campaign's `users` list.

6.5.1.3 Logging of Billing Data

According to the requirements specification, every view of a campaign message should be logged regardless of the number of views. When a user views a campaign message, an object containing the details of the campaign, message, and user is created but is not persisted to the database. When the billing application becomes available, they will be persisted to the database of the application.

6.5.2 Surveys Function

The *Surveys* function of the HealthMessenger was implemented using the classes similar to those used to implement the same function in the Dashboard. As is the case with the *Campaigns* function, a HealthMessenger user cannot add, update, or delete a survey. He or she can only complete a survey. A survey is created when a Dashboard user publishes it. It is updated when a Dashboard user updates and republishes it.

6.5.2.1 Completing a Survey

When a user completes a survey, a record of survey answers is created in the HealthMessenger database and a copy of the completed survey is posted to and created in the Dashboard database. Recording of survey answers was implemented using the *Answer* class. The fields of the class are response, which has an *Integer* data type and `answerOption`, which has a *AnswerOption* data type. The response field is annotated with @JsonBackReference and @ManyToOne. When a user selects an answer option, the related answerOption is incremented. No record of user details is made but only their answers. This was done to ensure anonymity of user responses. Completion of a survey triggers logging of billing data. Logging of the data was implemented similarly to the campaign messages and includes details of the survey and user.

6.5.2.2 Logging of Auditing Data

Logging of auditing data of a Survey was implemented similarly to logging of auditing data a campaign by including a `users` field of *java.util.List<User>* data type in the *Survey* class.

6.5.3 Events Function

The *Events* function was implemented using the classes similar to those used to implement the same function in the Dashboard. A HealthMessenger user can only view or register for an event.

6.5.3.1 Creating an Event

An event is created when a Dashboard user publishes it. It is updated when a Dashboard user updates and republishes it.

6.5.3.2 Registering and Deregistering from an Event

User registration to an event was implemented similarly to logging of user access to a campaign. User deregistration deletes the record that is created when a user registers for an event. Registering for an event triggers logging of billing data just like in a survey.

6.5.4 News Function

The *News* function was implemented using similar classes to those used to implement the same function in the Dashboard. A HealthMessenger user can only read a news article. A news article is created in the HealthMessenger when a Dashboard user publishes it. It is updated when a Dashboard user updates and republishes it. Logging of user access to a news article was implemented similarly to logging of user access to a campaign. When a user reads a news article, it triggers logging of billing data just like in a survey.

6.5.5 Facilities Function

The *Facilities* function of the HealthMessenger was implemented using classes similar to those used to implement the same function in the Dashboard. A facility is created in the HealthMessenger when a Dashboard user publishes it. It is updated when a Dashboard user updates and republishes it. The Facility function in the HealthMessenger includes the function to get services provided by a particular facility and also to search for a particular service independent of a facility.

6.5.6 Targeting

Targeting in the HealthMessenger involves providing a user access to only the awareness campaigns and surveys that are relevant to him or her. The relevance is determined by comparing the target information of an awareness campaign or survey and the demographic information and geographical location of the user. Targeting depends on a user having been authenticated. In the absence of the TeleWeaver, a profile application was developed to create, manage, and authenticate users. (**Note**: User creation, management, and authentication were originally not the functions of the HealthMessenger. The assumption was that user creation, management, and authentication would be done in TeleWeaver. It was included in in the project scope to implement and test targeting)

6.5.6.1 Profile Application

The profile application was implemented as a separate web application. Token-based authentication was used to implement user authentication just like in the Dashboard. The

user authentication was modified to ensure that all users complete their profile information before accessing any awareness campaigns or surveys and works as follows:

- a user enters a username and password on the web interface;
- a check is made to ensure that the username and password fields are filled - a blank username or password is not acceptable and results in an error;
- on the server, the username and password pair is compared with username and password pairs in the database - a match with one of the pairs results in a successful authentication otherwise it is unsuccessful;
- if user authentication is unsuccessful, the user is informed otherwise a token is generated and sent to the user (**Note**: Other information sent with the token are a unique user ID, the first name and last name, and an indication of whether the user has completed his or her profile. The token and the unique user ID are used in the subsequent requests for resources as discussed in Sub-section 6.4.7.4. The first name and last name are used to personalise health messages presented to a user using personalisation tactic of message tailoring strategy discussed in Appendix B.1 and to indicate the logged in user);
- if a user's profile is not completed, a profile update page is displayed - it is a requirement that an authenticated user completes his or her profile before accessing any other information; and
- if the user profile is completed, a welcome message is presented to a user with a link to access the HealthMessenger.

6.5.6.2 Selection of Targeted Health Information

The *Campaigns* function is used to describe how targeted strategy was implemented. Selection of targeted information for the other functions was implemented similarly. As described in Sub-section 5.4.2.1, in order to get campaigns that match a user as a target, his or her demographic information and geographical location is compared with the target information of campaigns in a *Running* state. The request to get campaigns in a *Running* state is done through the *getRunningCampaigns(@Context SecurityContext sc)* method (lines 27 - 33 of Code listing 6.2) of the *CampaignResource* class. The `@Context` annotation in the argument of the method (Line 29 of Code Listing 6.2) has a *javax.ws.rs.core.Context*

data type and is used to inject the *SecurityContext*. The *SecurityContext* is used to obtain profile information of a user making a request as shown in Line 30 of Code Listing 6.2. The profile information is sent with a request to retrieve campaigns in a *Running* state. Demographic information and geographical location of a user is retrieved from the profile information and then compared with the target information of each of the campaigns in a *Running* state to determine if they match. All matching campaigns are retained, while those that do not match are discarded.

6.5.6.3 Accessing Health Information in a Pull Mode

User access to health information in a pull mode is done through a link provided when he or she is successfully authenticated. When a user clicks on the link, a menu to access campaigns, surveys, events, news, topics, and facilities and services is displayed. When a user clicks on one of the menus, appropriate information is retrieved for the user as discussed in Sub-section 6.5.6. The information is presented as a list.

A user accesses campaign messages by clicking on a campaign link on the list. When there is more than one message, a user views one at a time. He or she moves between the messages by clicking on the next and previous buttons. A user completes a survey by clicking on a survey link on the list of surveys. When clicked, a form to enable a user to complete the survey is presented. A user can register or deregister from an event by clicking on the button next to it. A user can read a news article by clicking on its link. A user is allowed to search for a specific facility or service. Several criteria, which include name, type, and location can be used to conduct the search. A user can search for a specific service by name. He or she can access information of the services offered by a facility by clicking on a link to services next to a facility.

6.5.6.4 Accessing Health Information in a Push Mode

When a campaign, survey, event, or news is published, a record containing the id, name/title, type (which can be a campaign, survey, event, or news), priority, start date and time, end date and time, and target is created in the Log table of the HealthMessenger database. When a user is successfully authenticated, appropriate campaigns, surveys, events, and news are selected from the Log table as discussed in Sub-section 6.5.6. The one with a high priority is selected and used create a message that is displayed to the user as a desktop notification to attract his or her attention to access health information. A desktop

notification is used because it does not prevent a user from performing other functions when it is displayed as is the case with modal dialog boxes. The use of personal information to create a message followed the recommendation by Snyder [75] that messages should be designed to capture a user's attention in order to increase the chances him or her acting on it. When a user clicks on the notification, a web page is opened with the information about the selected campaign, survey, event, or news.

The desktop notification was implemented using a JavaScript notification plugin called PNotify. PNotify is a plugin enables creation of non modal desktop notifications.[22] To use the plugin, CSS, and JavaScript files of the plugin were added to the project. The parameters of the plugin, which include the title and text (message to the user) were set. When a user clicks on the notification, it executes a function that opens a page to enable the user to access health information that was selected for him or her. When the notification is closed, it disappears without any action.

6.5.7 Language Switching

Switching of languages was implemented using a jQuery plugin called jquery-lang-js[23]. The plugin enables switching of languages without reloading a page or sending a request to the server. The plugin was used to translate static web contents. To use the plugin, a library file of the plugin was added to the project. To persist language selection across pages automatically, a jQuery cookie plugin called jquery-cookie[24] was added to the project. After adding the required libraries, a language pack, which provides translation from English (the default language) to other languages was created. Only translations from English to isiXhosa were done but as many languages as possible can be used to deliver the content. The plugin provides two ways to switch languages. A language pack to translate page contents can be loaded dynamically or upfront. In dynamic loading, the path of the language pack is defined and the plugin handles its on-demand loading. The language packs are saved as JSON files with .json extension. In up-front loading, the language packs are loaded with the page and used straight away and they are saved as JavaScript files with .js extension. Upfront loading was used to implement the language switching.

To implement access to HealthAware in English and isiXhosa, a language pack to translate text from from English to isiXhosa was created using JSON key/value pairs. The

[22] https://github.com/sciactive/pnotify
[23] https://github.com/coolbloke1324/jquery-lang-js
[24] https://github.com/carhartl/jquery-cookie

translations were done with the help of Sisonke Mawonga, a student from the Department of African Languages at Rhodes University. The words to be translated were enclosed in HTML elements with a 'lang' attribute. For example Home indicates that 'Home' will be translated to isiXhosa if a language pack for isiXhosa is loaded. When a user selects a language, a JavaScript method called **change()** is called and passed a two-letter language code of the language to switch to. For example, to switch to isiXhosa, the method is passed the code 'xh'. The switcher was implemented using the code below:

```
1 <li>
2 <a href="#lang-en" onclick="window.lang.change('en');return false;">Switch to English</a>
3 </li>
4 <li>
5 <a href="#lang-en" onclick="window.lang.change('xh');return false;">Switch to Xhosa</a>
6 </li>
```

Listing 6.7: Code listing for the switcher

To update the dynamic web contents to the selected language, a request is sent to the server to retrieve content in the selected language. Language switching works as follows:

- when a user is successfully authenticated, his or her preferred language is obtained;
- the content, which is developed in the preferred language is retrieved;
- a language cookie is used to persist the language across all web pages automatically; and
- when a user switches languages, the page contents are updated to reflect the selected language and the language cookie is set to the selected language.

6.5.8 Reporting

The *Report* class was used to persist data for generating sensitisation campaigns reach and hits report. The class is a Domain Model and its fields are `reportId, title`, `type`, `hits`, `reach`, `organisation`, `demographics`. The `reportId, title`, `type`, `hits`, and `reach` are annotated with `@Column` and `@NotNull`. The `reportId` contains the `uuid` of the campaign, event, or news. The `title` field contains the title/name of a campaign, event, or news. The `type` field can be 'campaign', 'event', or 'news' and is used to determine report type. The `hits` field contains the number of hits on a campaign, event,

or news and is updated at every click. The reach field contains the number of people who have been reached by a campaign, who have registered for an event, or who have read a news article. The `organisation` field has the *Organisation* class data type and is used to select sensitisation campaigns for a specific organisation. The `demographics` field has a *Demographics* class data type and contains demographics of TeleWeaver users that have been reached with sensitisation campaigns. When reporting data is generated, it is persisted to the HealthMessenger database and a copy is posted to and persisted to the Dashboard database.

6.6 Implementation of the Store and Forward Function

The store and forward function was implemented using Apache ActiveMQ. Apache ActiveMQ is an open source message broker that implements the Java Message Service (JMS) specification and is used for remote communication between systems [74]. ActiveMQ was integrated in Wildfly using activemq-rar-5.10.0.rar. A transport connector that provides client-to-broker communication was used to configure ActiveMQ as an embedded broker using Virtual Machine (VM) protocol. In this case, the clients are the HealthMessenger and the Dashboard. The VM protocol enables client applications to communicate directly with the broker within the container (Wildfly). When a broker is configured using VM protocol, it is starts when the container is started.

Another instance of Wildfly was configured similarly in a separate environment in order to implement communication between brokers (network of brokers) so that they can send messages to each other. A network of brokers creates a cluster composed of multiple ActiveMQ instances that are interconnected. Part of the configuration details on one of the brokers are shown in program listing 6.8. The broker was running on a machine with an Internet Protocol (IP) address of 146.231.123.109 and was listening for incoming messages on port 61616. The broker was configured to communicate with the other broker running on a machine with an IP address of 146.231.121.143 and listening for incoming messages on port 61616 using a static protocol/connector. The prerequisite of using the static connector is that the addresses of all the brokers used should be known. The static protocol was used for testing purposes only.

The ideal situation is to use the fanout connector. The fanout connector is used to network brokers so that a broker can send a message to multiple brokers within a network. Fanout can be configured using either a static connector or a multicast connector. The multicast

connector is a network technique used to send messages from one source to a group of receivers over an IP network. The uniform resource identifier (URI) syntax for the multicast connector is multicast://ipadaddress:port?key=value. The URL syntax for fanout connector is fanout:(fanoutURI)?key=value. To configure fanout using static connector, the syntax is fanout:(static:(tcp://host1:61616,tcp://host2:61616,tcp://host3:61616)) and to use fanout with the multicast connector the syntax is fanout:(multicast://default).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans
3    xmlns="http://www.springframework.org/schema/beans"
4    xmlns:amq="http://activemq.apache.org/schema/core"
5    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.
       springframework.org/schema/beans/spring-beans-3.0.xsd
7  http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-
     core.xsd">
8
9    <broker xmlns="http://activemq.apache.org/schema/core" useJmx="true" useShutdownHook="
       false">
10
11     <transportConnectors>
12       <transportConnector name="vm" uri="vm://146.231.123.109:61616"/>
13     </transportConnectors>
14
15     <networkConnectors>
16       <networkConnector name="messenger" uri="static:(tcp://146.231.121.143:61616)"/>
17     </networkConnectors>
18
19   </broker>
20 </beans>
```

Listing 6.8: Broker configuration

6.7 Summary

This chapter discussed the implementation of HealthAware. The layered architecture that splits the implementation of a function into presentation, business, and data access layers was used to implement individual functions. The layered architecture helps to minimise duplication of function implementation and simplifies the program code. The next chapter discusses the tests that were done to determine if HealthAware meets the requirements

Chapter 7

Testing

7.1 Introduction

The previous chapter discussed the implementation of HealthAware. This chapter discusses the tests done to verify that it meets the requirements discussed in Chapter 5.

7.2 Software Testing Approaches

There are two main software testing approaches called black box and white box testing.

7.2.1 Black Box Testing

Black box testing tests units of code that are complete. It is concerned with verifying input against expected output and not the internal logic of the code [9, 43]. Examples are user acceptance and system testing. It is also called functional testing because it uses knowledge of a function of the program and not how it works. It commences with the development of a test specification or schedule, where test data and expected outcomes are written down before the actual test. The test schedule is the used to carry out the actual test, where expected outcomes and the actual outcomes are compared.

7.2.2 White Box Testing

White box testing uses knowledge of how a program works as the basis for devising test data [9]. It is also called structural testing because it uses the structure of a program. It analyses the program code and internal logic looking for any discrepancies. Every statement in the program is executed during the testing, including any exception handling carried out by the program.

7.2.3 Specific Testing Methods

Specific testing methods include unit, integration, and system testing.

7.2.3.1 Unit Testing

In unit testing, discrete components of the final product, called units, are tested independently through the use of 'test harnesses', which simulate the context in which the unit will be integrated with other units [43]. A test harness provides a number of known inputs and measures the outputs of the unit under test, which are then compared with the expected values to determine any discrepancies.

7.2.3.2 Integration Testing

In integration testing, smaller units are integrated into larger units, which are further integrated into the overall system. This type of integration testing is called incremental testing. Another integration testing method starts by testing each component individually and then bringing them together and test a complete system.

7.2.3.3 System Testing

In system testing, all the software components are combined and tested together. It is in this test that the full complexity of the product is present. It ensures that the product responds correctly to all possible input conditions and that all exceptions are handled properly.

7.3 Test Environment and Approach

7.3.1 Test Environment

The tests were done in an experimental environment. Separate environments for the Dashboard and the HealthMessenger were prepared using Ubuntu 12.04 as the operating system and Wildfly-8.2.0.Final as the host environment. Wildfly was extended with mysql-connector-java-5.1.25-bin.jar file to support MySQL database and activemq-rar-5.10.0.rar for message queuing. The Dashboard was accessible through http://146.231.123.109:8080-/Dashboard and the HealthMessenger through http://146.2-31.121.143:8080/heath-messenger.

7.3.2 Test Approach

White box testing was done during code development. JUnit version 4.8.2 was used to conduct the tests. What remained after HealthAware was completed was to test the overall system (System testing), which is discussed in this chapter. The discussion is done on a function by function basis and includes creating and publishing health information using the Dashboard, accessing the information using the HealthMessenger, and generating reports. The tests included checking, validating, and verifying input data and accessing health information in both pull and push modes. Only the test results of the *Campaigns* function are discussed in detail. The test of the other functions exclude those similar to the *Campaigns* function.

7.4 Testing the Campaigns Function

The features of the *Campaigns* function that require testing are creating, publishing, stopping, restarting, and archiving campaigns, accessing campaigns in push and pull modes, and generating campaign reach reports. Figure 7.1 is a screenshot of campaigns that were created for testing purposes. In the figure, the icons in the **Actions** column indicate the actions a user can perform on a campaign in a particular state. The icon labeled 1 is for adding and updating campaign messages, 2 is for setting or updating targets, 3 is for editing campaign details, 4 is for deleting a campaign, 5 is for publishing a campaign, 6 is for restarting a stopped campaign, 7 is for archiving a campaign, 8 is for viewing campaign report and 9 is for stopping a campaign.

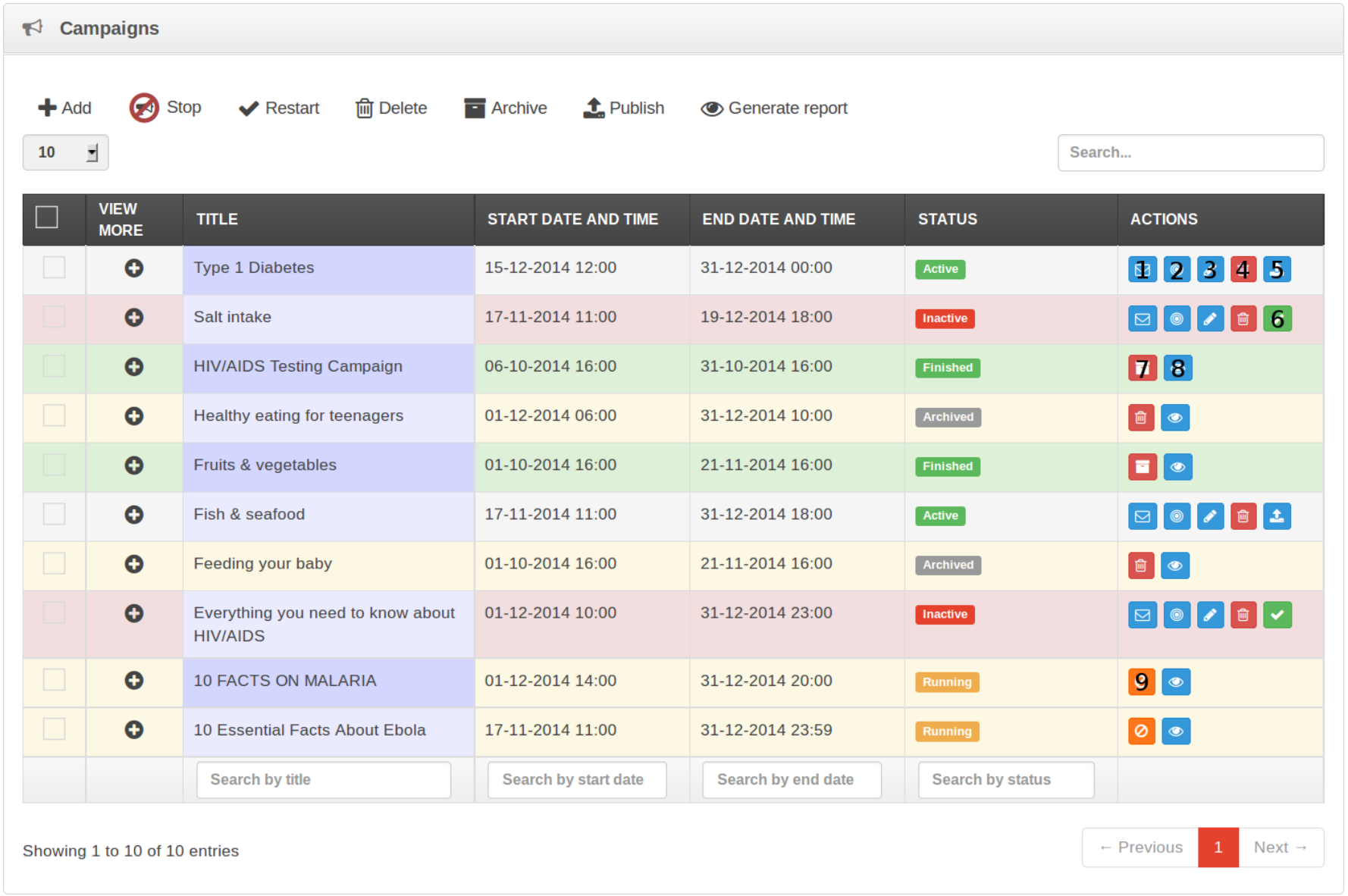

Figure 7.1: List of campaigns (screenshot taken on 08-12-2014)

7.4.1 Creating a Campaign

The feature that require testing when creating a campaign is that a new campaign should be in *Active* state. A campaign in *Active* state can be updated (edit its details, add/edit/delete its messages, and add/edit/delete its target information), deleted, and published. Figure 7.2 is a screenshot that was taken when a campaign about the Ebola virus was created. As shown in the screenshot, the campaign is in *Active* state. The state of the campaign together with the icons in the **Actions** column confirm that the function passes the test.

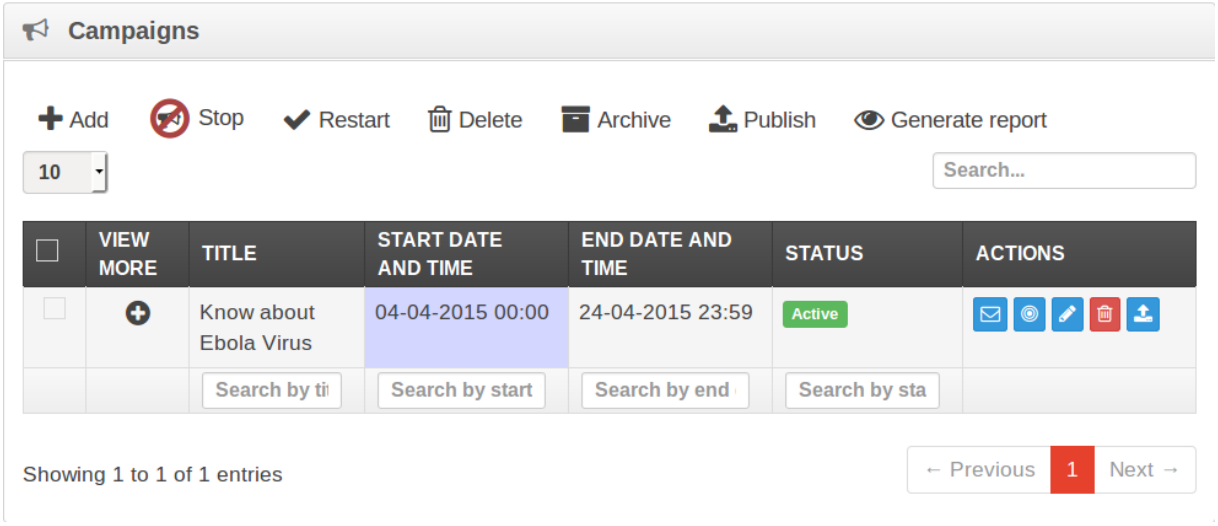

Figure 7.2: New campaign

7.4.2 Publishing a Campaign

A campaign can be published by clicking icon 5 in Figure 7.1. The feature that require testing when a campaign is published is that its state changes toshould be *Published*. A campaign can be published when it is in *Active* or *Published* state. Publishing of a campaign in the *Published* state is allowed to enable a user to update and republish it. A campaign that has been published can be updated and deleted. Figure 7.3 is a screenshot taken when the Campaign about Ebola virus was published. The status of the campaign and the actions that can be performed on it as shown in the **Actions** colums confirm that the function passes the test.

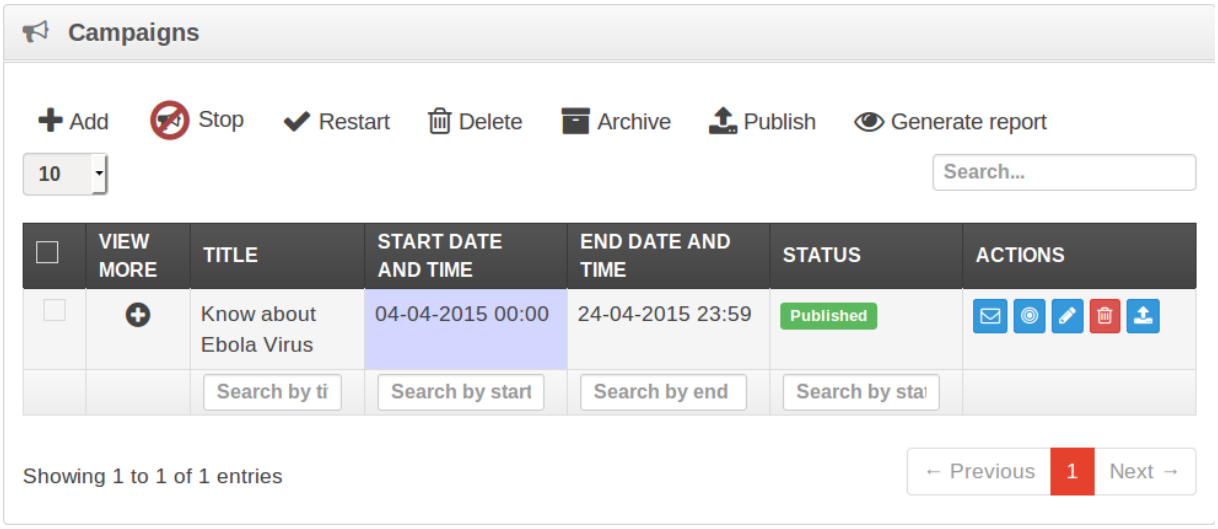

Figure 7.3: Published campaign

7.4.3 Stopping and Restarting a Campaign

A campaign can be stopped by clicking on icon 9 and can be restarted by clicking on icon 6. The feature that requires testing when stopping a campaign is that its state should change to *Inactive*. A campaign can be stopped when it is in *Running* state. A campaign should be in a *Running* state if the current date and time is greater than the start date and time and less than the end date and time. A campaign in *Inactive* state can be updated (edit its details, add or edit target details and add or edit its messages), restarted, or deleted. Figure 7.4 is a screenshot of the campaign about Ebola virus in a *Running* state. The date and time of the test was 30-03-2015 15:00 and the start date and time of the campaign was changed so that the state of the campaign changed to *Running*.

Figure 7.5 is a screenshot taken when the campaign about Ebola virus was stopped. The status of the campaign together with the actions that can be performed on it as shown by the icons in the **Actions** column of Figure 7.5 confirm that the function passes the test.

When the campaign about the Ebola virus is restarted, the screenshot is as shown in Figure 7.4 if the current date and time is greater than the start date and time and less than the end date and time. If the start date and time is greater than the current date and time then the screenshot is as shown in Figure 7.3. The function to restart the campaign therefore passes the test.

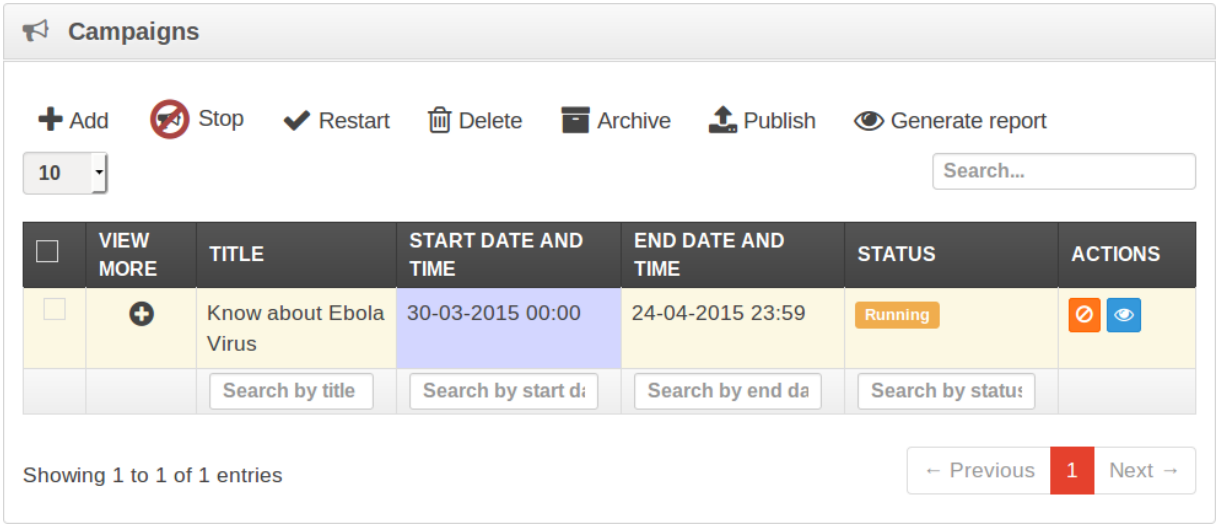

Figure 7.4: A campaign in running state

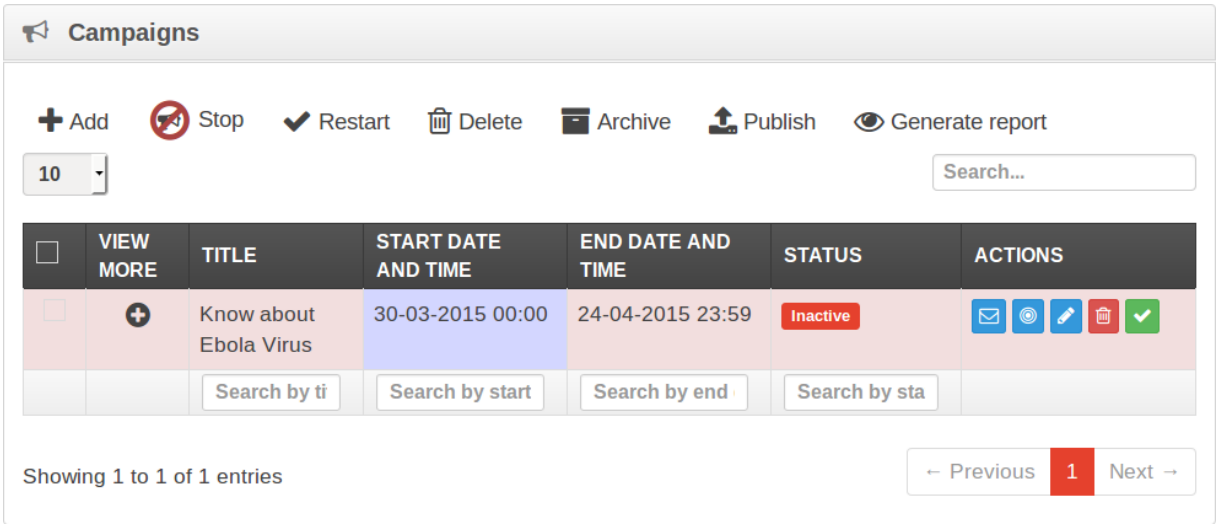

Figure 7.5: Stopping a campaign

7.4.4 Archiving a Campaign

A campaign can be archived by clicking icon 7. The feature that requires testing when archiving a campaign is that its state should change to *Archived*. A campaign can be archived if it is finished. A campaign is finished if the end date and time is less than the current date and time. A user can view campaign details, delete it, or generate a reach report when it is in *Archived* state. Figure 7.6 is a screenshot taken when the campaign about Ebola virus is archived. The status of the campaign and the actions that can be performed when archived confirms that it passes the test.

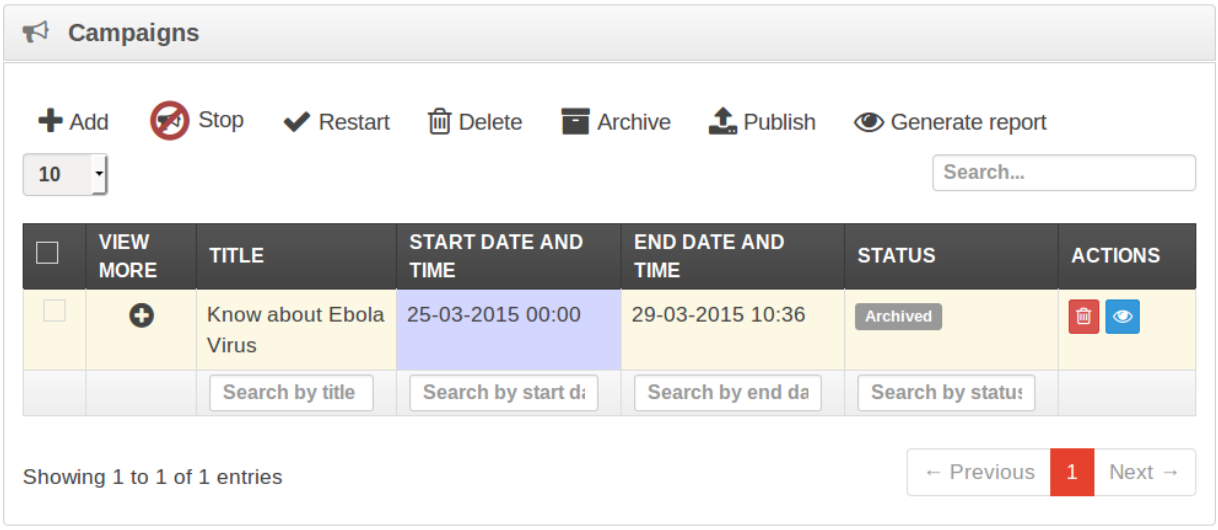

Figure 7.6: Archived campaign

7.4.5 Accessing Campaigns

The features that require testing are that a user should access campaigns in either a pull or a push mode; and should access only those he or she matches as a target. To conduct the tests, a number of campaigns targeting different users were created using the Dashboard and published to be accessible through the HealthMessenger. Figures 7.7 and 7.8 are extracts from the database showing the campaigns details. The details are split into two figures for clarity.

```
mysql> SELECT c.title, DATE_FORMAT(t.dob_from, '%d-%m-%Y') as dob_from, DATE_FORMAT(t.dob_to, '%d-%m-%Y') as dob_to, t.gender, t.language, t.marit
al_status as m_status, t.occupation, t.sex_orientation FROM CAMPAIGN_TBL c, TARGET_TBL t WHERE c.target_id = t.id;
+----------------------------------------+------------+------------+--------+----------+----------+----------------------+----------------+
| title                                  | dob_from   | dob_to     | gender | language | m_status | occupation           | sex_orientation |
+----------------------------------------+------------+------------+--------+----------+----------+----------------------+----------------+
| Know about Cancer                      | 01-02-1982 | 31-12-2009 | NULL   | English  | NULL     | NULL                 | NULL           |
| 10 Essential Facts About Ebola         | NULL       | NULL       | NULL   | English  | NULL     | NULL                 | NULL           |
| 10 FACTS ON MALARIA                    | 01-01-1983 | 31-12-2005 | NULL   | English  | NULL     | NULL                 | NULL           |
| Type 1 Diabetes Facts                  | NULL       | NULL       | NULL   | English  | NULL     | NULL                 | NULL           |
| Reducing saturated fats                | NULL       | NULL       | NULL   | English  | NULL     | NULL                 | NULL           |
| Healthy eating for teenagers           | 01-01-1992 | 31-12-2001 | NULL   | English  | NULL     | NULL                 | NULL           |
| Fruits & vegetables                    | 01-06-1994 | 31-10-2009 | NULL   | English  | NULL     | NULL                 | NULL           |
| Everything you need to know about HIV/AIDS | 01-10-1989 | 01-10-2008 | NULL   | English  | NULL     | NULL                 | NULL           |
| Salt intake                            | 01-01-1990 | 31-12-2000 | Female | English  | Single   | Commercial Sex Workers | Bisexual       |
| Fish & seafood                         | 01-01-1995 | 31-12-2010 | Male   | English  | Single   | Miners               | Gay            |
| HIV/AIDS Testing Campaign              | NULL       | NULL       | Female | English  | NULL     | Commercial Sex Workers | NULL           |
| Feeding your baby                      | 01-01-1990 | 31-12-1999 | Female | English  | NULL     | NULL                 | NULL           |
+----------------------------------------+------------+------------+--------+----------+----------+----------------------+----------------+
12 rows in set (0.01 sec)
```

Figure 7.7: List of campaign targets as extracted from the database

```
mysql> SELECT c.title, l.province, l.district, l.municipality, l.village FROM CAMPAIGN_TBL c, TARGE
T_TBL t, LOCATION_TBL l WHERE c.target_id = t.id and t.location_id = l.id;
+----------------------------------------+--------------+----------+--------------+----------+
| title                                  | province     | district | municipality | village  |
+----------------------------------------+--------------+----------+--------------+----------+
| Know about Cancer                      | Eastern Cape | Cacadu   | Makana       | Joza     |
| Reducing saturated fats                | Eastern Cape | Cacadu   | Makana       | Joza     |
| 10 Essential Facts About Ebola         | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| Type 1 Diabetes Facts                  | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| Healthy eating for teenagers           | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| Fruits & vegetables                    | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| Everything you need to know about HIV/AIDS | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| Fish & seafood                         | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| Feeding your baby                      | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| 10 FACTS ON MALARIA                    | Eastern Cape | Ncedo    | Mbashe       | Dwesa    |
| Salt intake                            | Eastern Cape | Ncedo    | Mbashe       | Dwesa    |
| HIV/AIDS Testing Campaign              | Eastern Cape | Ncedo    | Mbashe       | Dwesa    |
+----------------------------------------+--------------+----------+--------------+----------+
12 rows in set (0.00 sec)
```

Figure 7.8: List of campaign targets showing targeted locations

A number of users with different demographic characteristics were created using the profile application described in Sub-section 2.7.5.1. Figures 7.9 and 7.10 are extracts from the database showing the list of users.

```
mysql> SELECT user_name, DATE_FORMAT(date_of_birth, '%d-%m-%Y') as date_of_birth, gender, language, marital_stat
us, occupation, sexOrientation FROM USER_TBL;
+-------------+---------------+--------+----------+----------------+-----------------------+----------------+
| user_name   | date_of_birth | gender | language | marital_status | occupation            | sexOrientation |
+-------------+---------------+--------+----------+----------------+-----------------------+----------------+
| c.gremu     | NULL          | NULL   | NULL     | NULL           | NULL                  | NULL           |
| k.madimba   | 26-06-1983    | Male   | English  | Engaged        | Commercial Sex Workers | Gay            |
| o.phiri     | 16-05-1982    | Male   | isiXhosa | Single         | Miners                | Straight       |
| b.msakambewa | 28-01-1985   | Male   | English  | Single         | House Maids           | Gay            |
| red.spencer | 18-09-1990    | Female | English  | Single         | Commercial Sex Workers | Bisexual       |
| a.ndimaso   | 13-02-1975    | Female | isiXhosa | married        | House Maids           | Straight       |
| p.kale      | 29-09-1999    | Male   | English  | Single         | Miners                | Gay            |
| n.gomez     | 01-01-1995    | Female | English  | Single         | Commercial Sex Workers | Lesbian        |
| s.campbell  | NULL          | NULL   | NULL     | NULL           | NULL                  | NULL           |
+-------------+---------------+--------+----------+----------------+-----------------------+----------------+
9 rows in set (0.00 sec)
```

Figure 7.9: List of users and their demographics as extracted from the database

```
mysql> select u.user_name, l.province, l.district, l.municipality, l.
village from USER_TBL u LEFT JOIN LOCATION_TBL l on u.location_id = l
.id;
+--------------+--------------+----------+--------------+----------+
| user_name    | province     | district | municipality | village  |
+--------------+--------------+----------+--------------+----------+
| c.gremu      | NULL         | NULL     | NULL         | NULL     |
| k.madimba    | Eastern Cape | Ncedo    | Mbashe       | Dwesa    |
| o.phiri      | Eastern Cape | Cacadu   | Makana       | Joza     |
| b.msakambewa | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| red.spencer  | Eastern Cape | Ncedo    | Mbashe       | Dwesa    |
| a.ndimaso    | Eastern Cape | Ncedo    | Mbashe       | Dwesa    |
| p.kale       | Eastern Cape | Amathole | Buffalo City | Qoboqobo |
| n.gomez      | Eastern Cape | Cacadu   | Makana       | Joza     |
| s.campbell   | NULL         | NULL     | NULL         | NULL     |
+--------------+--------------+----------+--------------+----------+
9 rows in set (0.00 sec)
```

Figure 7.10: List of user addresses

The user accounts were used to access the campaigns. When a user is successfully authenticated and his or her profile information is complete, a welcome page is presented to him or her with a link to proceed to access health messages using the HealthMessenger as shown in Figure 7.11.

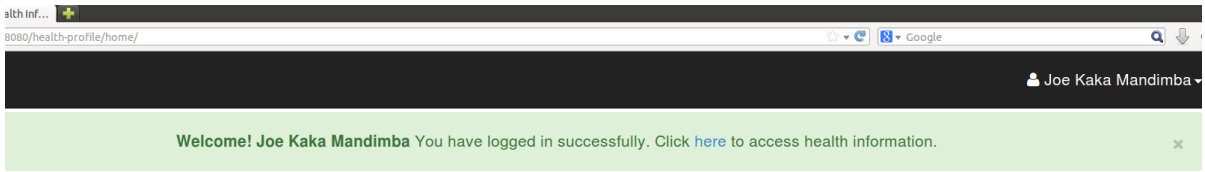

Figure 7.11: Welcome page for an authenticated user

7.4.5.1 Accessing Campaigns in a Push Mode

When a user is successfully authenticated and there are campaigns, surveys, events, or news he or she matches as their target, one of them is selected. The title/name of the selected campaign, survey, event, or news is used with the name of the user to create a custom message to attract his or her attention to act on the message. The message is displayed as a sticky desktop notification. A user can ignore the notification, close it, or click it to access health information selected for him or her. Figure 7.12 is a screenshot for

a sticky desktop notification created after Joe Kaka Mandimba was successfully authenticated. Figure 7.13 is a screenshot for a sticky desktop notification created after Nuno Gomez was successfully authenticated. When the notification in Figure 7.12 is clicked, the details for a campaign entitled '10 Facts on Malaria' are displayed and a user can access campaign messages when a link provided is clicked. When the notification in Figure 7.13 is clicked, the details for a campaign entitled 'Reducing saturated fats' are displayed and a user can access campaign messages when a link provided is clicked.

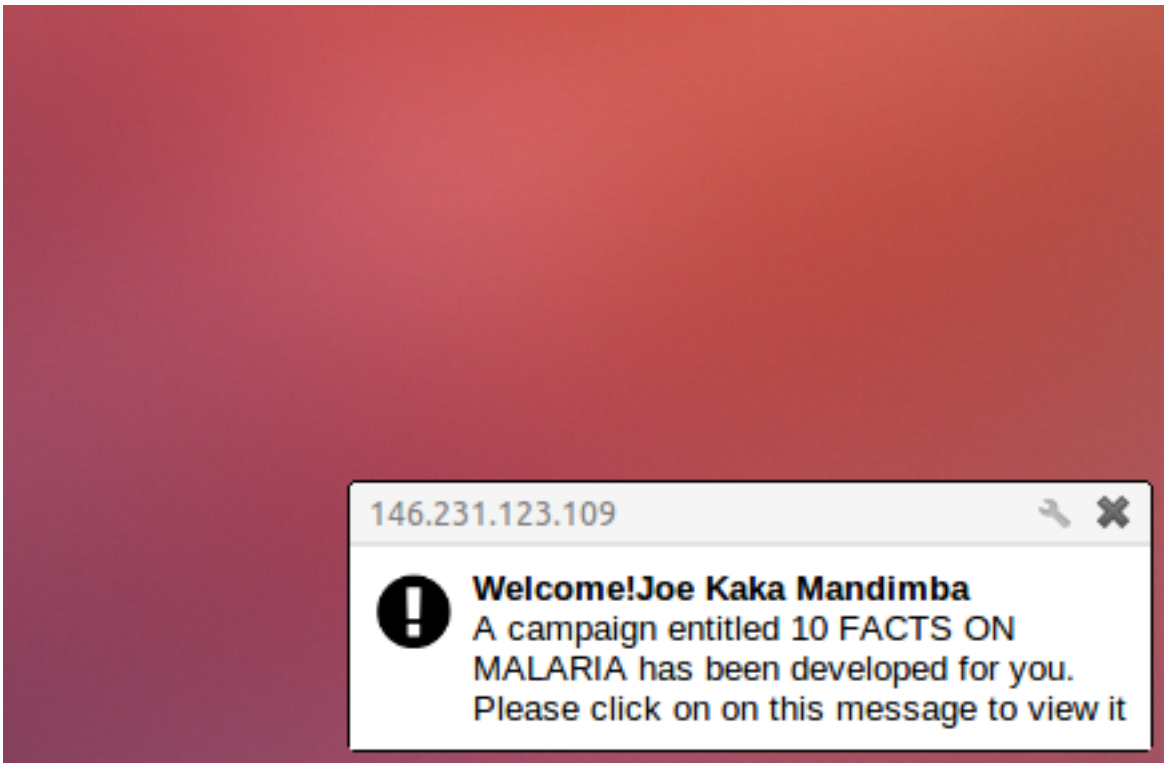

Figure 7.12: A notification for Kaka Mandimba

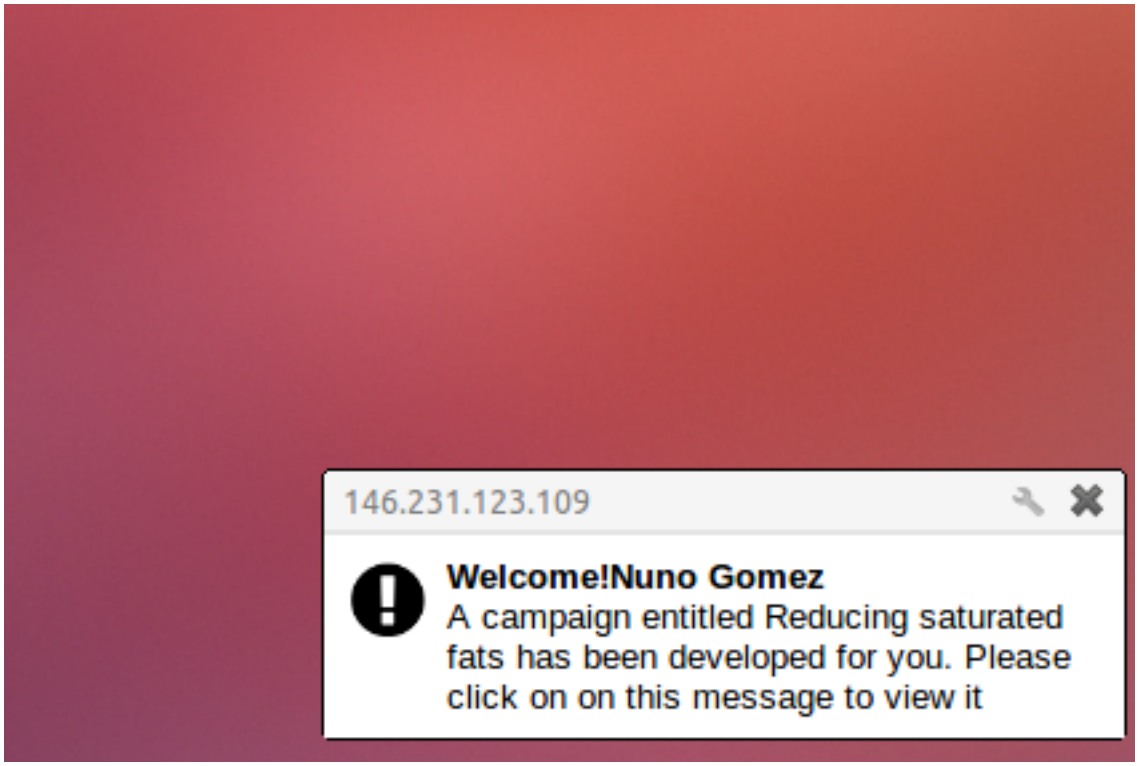

Figure 7.13: Notification for Nuno Gomez

7.4.5.2 Accessing Campaigns in a Pull Mode

A test schedule was prepared to test access to campaigns in the pull mode and is shown in Table 7.1. The expected outcome was obtained by comparing the target information of each campaign in Figure 7.7 with the demographic information of each user in Figure 7.9. Figures 7.14 to 7.20 are screenshots of the results obtained when each user account shown in Figure 7.9 was used to access campaigns. The expected and the actual results match showing that the test was successful. The results also match that obtained when testing the push mode.

USER NAME	MATCHING CAMPAIGNS	REASON
a.ndimaso	None	There are no campaigns in isiXhosa
k.madimba	10 facts on Malaria	Eastern Cape, Ncedo, Mbashe, Dwesa, and English match the demographics of k.madimba. His date of birth (26-06-1983) is in the range of (1-1-1983 - 21-12-2005)
red.spencer	10 Facts on Malaria	Eastern Cape, Ncedo, Mbashe, Dwesa, and English match the demographics of red.spencer. Her date of birth (18-09-1990) is within the range (1-1-1983 - 21-12-2005)
	Salt intake	Eastern Cape, Ncedo, Mbashe, Dwesa, Single, Commercial sex worker, bisexual, and English match the demographics of red.spencer. Her date of birth of k.madimba (18-09-1990) is within the range of (1-1-1990 - 31-12-2000)
	HIV/Aids testing campaign	Eastern Cape, Ncedo, Mbashe, Dwesa, Female, Commercial sex worker, and English match the demographics of red.spencer. Her date of birth (18-09-1990) is within the specified range of (1-1-1983 - 21-12-2005)
o.phiri	None	There are no campaigns in isiXhosa
n.gomez	Know about Cancer	Eastern Cape, Cacadu, Makana, Joza, and English match the demographics of n.gomez. The date of birth (1-1-1995) is between the specified range of (1-2-1982 - 31-12-2009)
	Reducing saturated fats	Eastern Cape, Cacadu, Makana, Joza, and English match the demographics of n.gomez.
b.msakambewa	Type 1 Diabetes facts	Eastern Cape, Amathole, Buffalo City, Qoboqobo, and English match the demographics of b.msakambewa.
	10 essential facts about Ebola	Eastern Cape, Amathole, Buffalo City, Qoboqobo, and English match the demographics of b.msakambewa.
p.kale	Type 1 Diabetes facts	Eastern Cape, Amathole, Buffalo City, Qoboqobo, and English match the demographics of p.kale.
	10 essential facts about Ebola	Same reason as for "Type 1 Diabetes facts".
	Fruits & vegetables	Same reason as for "Type 1 Diabetes facts" and the date of birth (29-9-1999) is between the specified range of (1-6-1994 - 31-10-2009).
	Everything you need to know about HIV/AIDS	Same reason as for "Type 1 Diabetes facts" and the date of birth (29-9-1999) is between the specified range of (1-10-1989 - 1-10-2008).
	Healthy eating for teenagers	Same reason as for "Type 1 Diabetes facts".
	Fish and seafood	Same reason as for "Type 1 Diabetes facts" and the date of birth (29-9-1999) is between the specified range of (1-1-1995 - 31-12-2010).

Table 7.1: Test schedule for campaigns

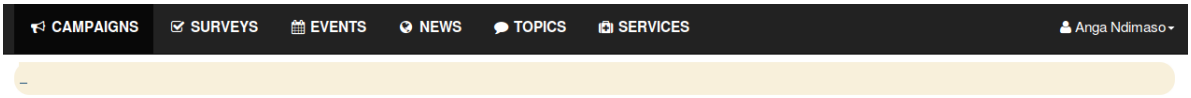

Figure 7.14: No campaigns accessed by Anga Ndimaso (a.ndimaso).

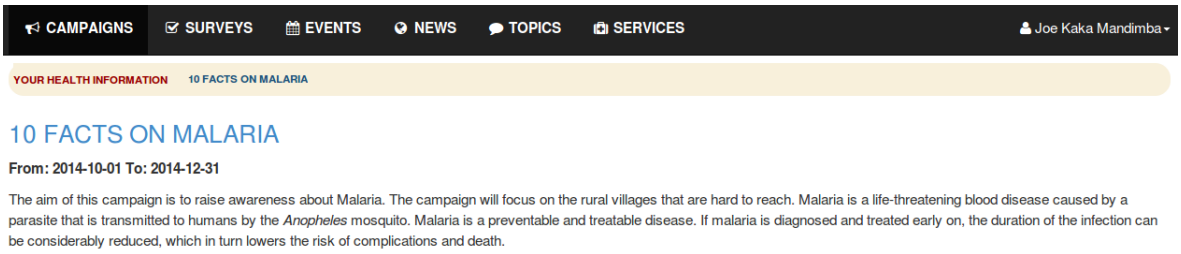

Figure 7.15: A campaign accessed by Kaka Madimba (k.madimba)

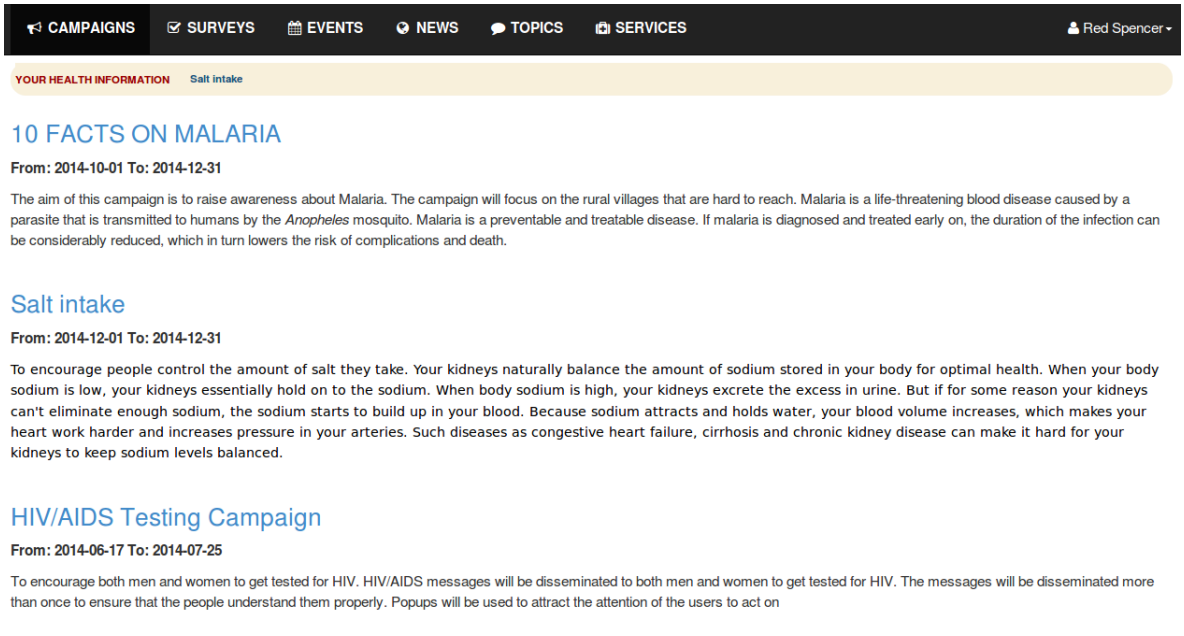

Figure 7.16: Campaigns accessed by Red Spencer (red.spencer)

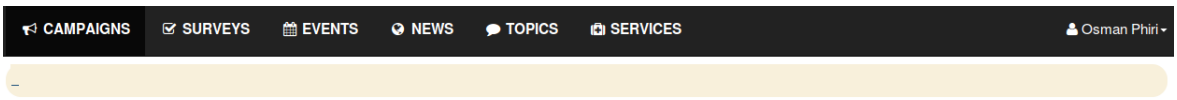

Figure 7.17: No campaigns accessed Osman Phiri (o.phiri).

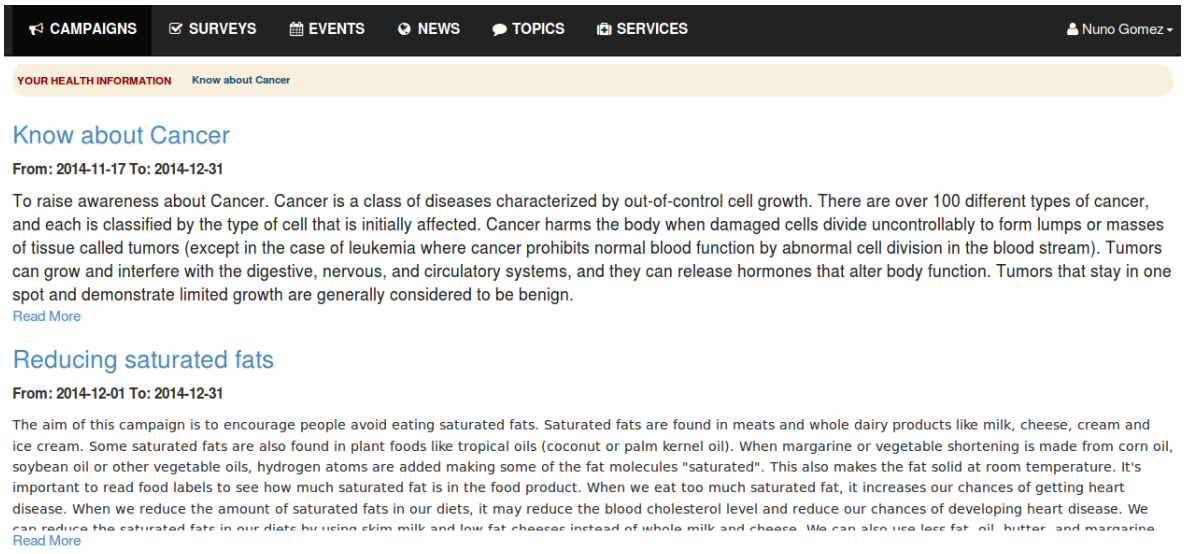

Figure 7.18: Campaigns accessed by Nuno Gomez (n.gomez).

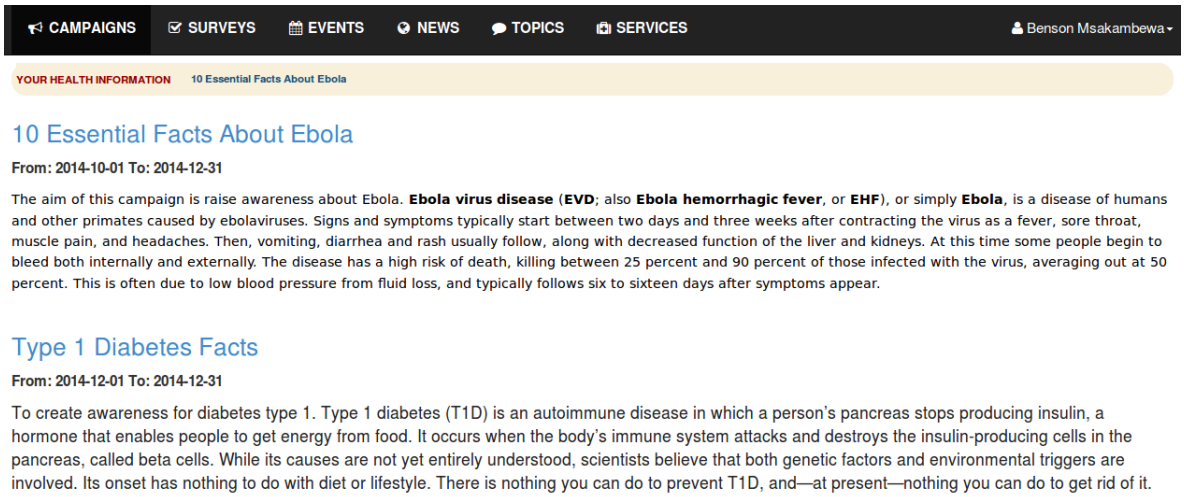

Figure 7.19: Campaigns accessed by Benson Msakambewa (b.msakambewa).

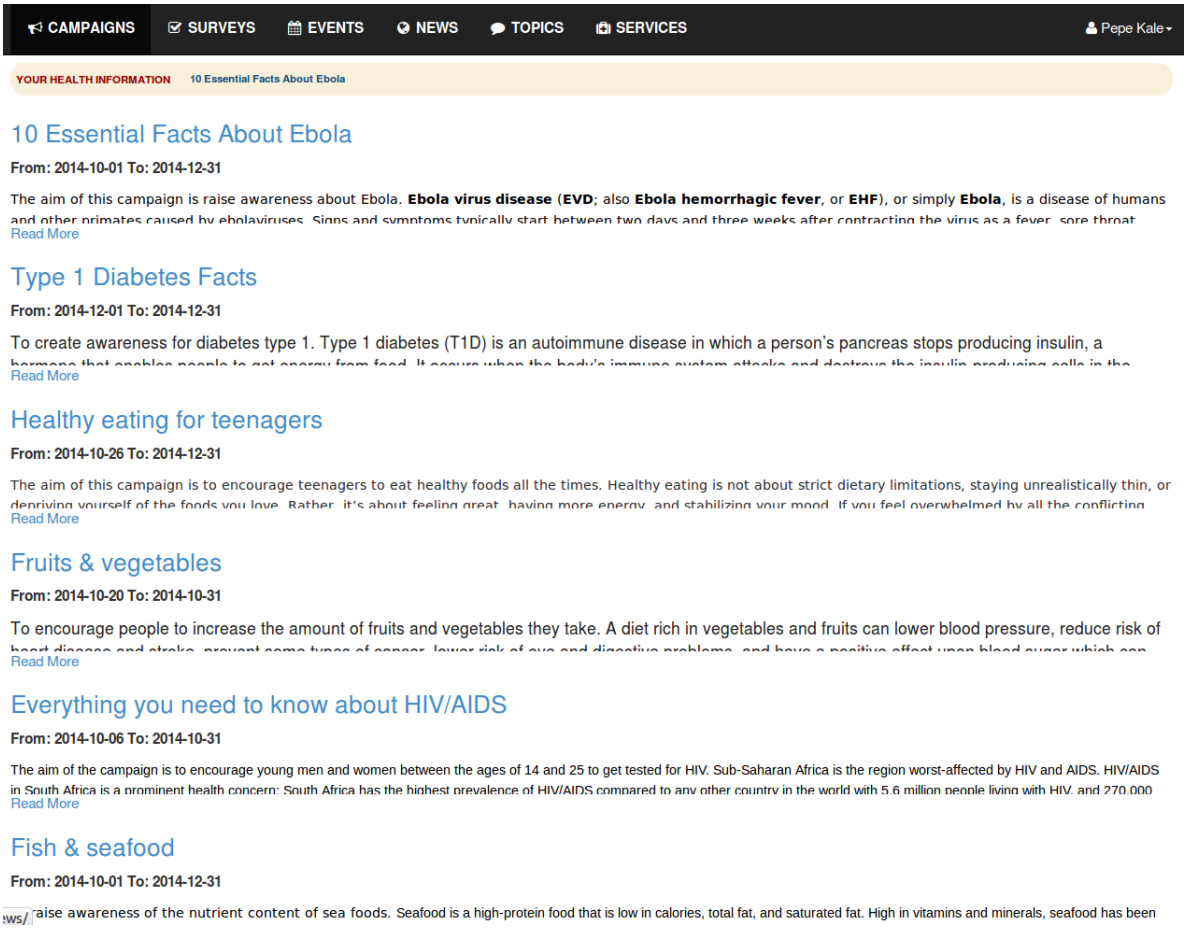

Figure 7.20: Campaigns accessed by Pepe Kale (p.kale).

When a user clicks on a title of a campaign, messages are displayed one at a time. A user views other messages by clicking on next and previous buttons (see Figure 7.21).

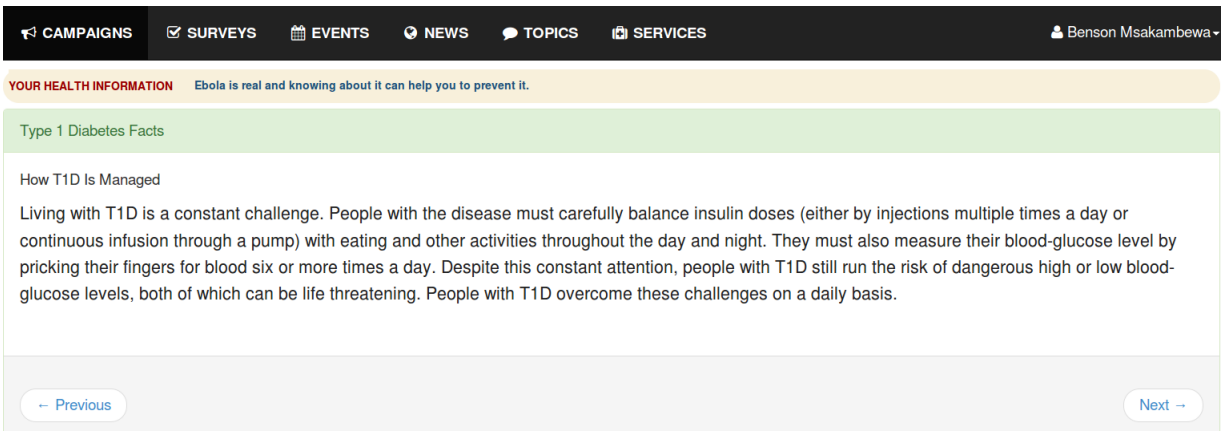

Figure 7.21: Campaign message after a user click on a campaign link

7.4.6 Generating Campaign Hits Report

A hits report for all campaigns of an organisation is generated by clicking the icon labelled generate reports (Figure 7.1). A hit report for a specific campaign is generated by clicking on icon 8 (Figure 7.1). Figure 7.22 is a screenshot of an extract from the database of campaign hits for a specific organisation.

title	number of hits
Type 1 Diabetes	15
10 Essential Facts About Ebola	19
10 FACTS ON MALARIA	0
Healthy eating for teenagers	0
Fruits & vegetables	42
Everything you need to know about HIV/AIDS	0
Salt intake	0
Fish & seafood	32
HIV/AIDS Testing Campaign	0
Feeding your baby	0
bdsfbdsfbsd	0
asdf dsafasd	0

12 rows in set (0.00 sec)

Figure 7.22: Number of hits

Figure 7.23 is the hits report, which is presented using a bar chart. When a cursor hovers over a bar, a popup provides more details, which include the name of a campaign and number of hits.

7.5 Testing the Surveys Function

The features of the *Surveys* that require testing are creating, updating, deleting, stopping, restarting, archiving, publishing, and completing surveys; and generating completed surveys report. In this section, only creating and completing a survey are discussed because the other features are similar to the *Campaigns* function.

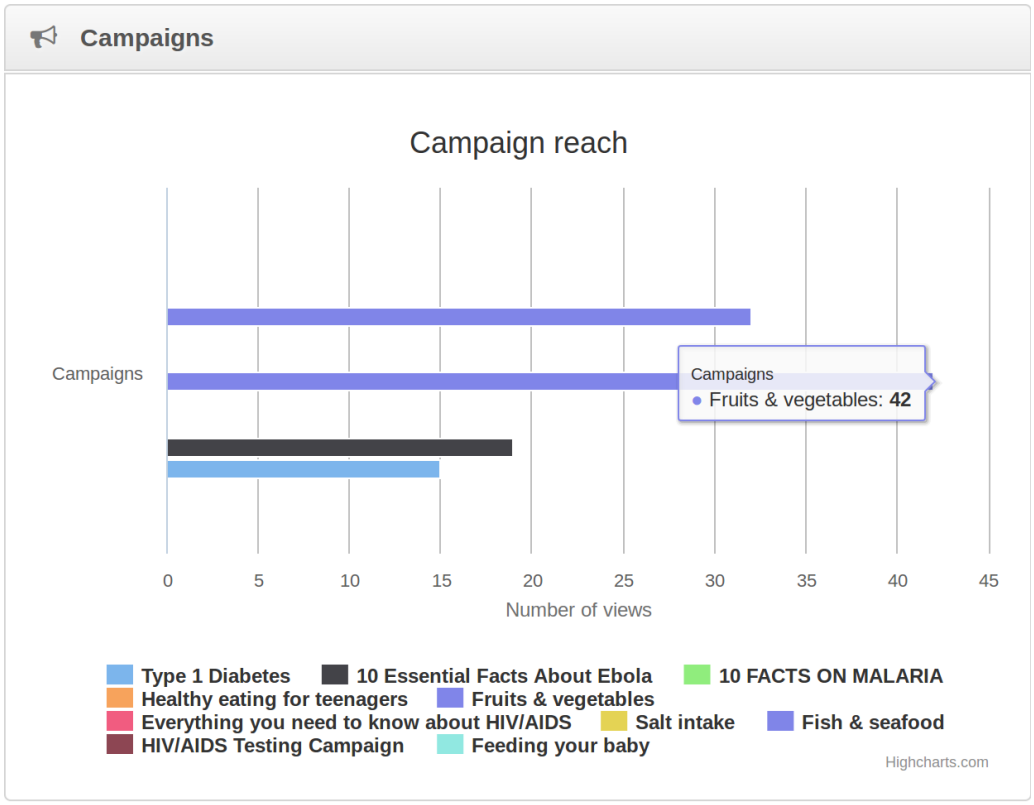

Figure 7.23: Campaign hits report

7.5.1 Creating a Survey

The features that require testing when creating a survey include recording the `title`, `description`, `start date and time`, `end date and time`, `questions`, and `target`. Figure 7.24 is a screenshot for recording the `title`, `description`, `start date and time`, and `end date and time`. Figures 7.25, 7.26 and 7.27 are screenshots for creating questions. Question types are indicated by selecting Multiple Choice (One answer) option to create questions that require one answer only, or Multiple Choice (Multiple answers) to create questions that can have more than one answer. The answer options for Multiple Choice (One answer) questions are presented as radio buttons, while those for Multiple Choice (Multiple answers) questions are presented as check boxes.

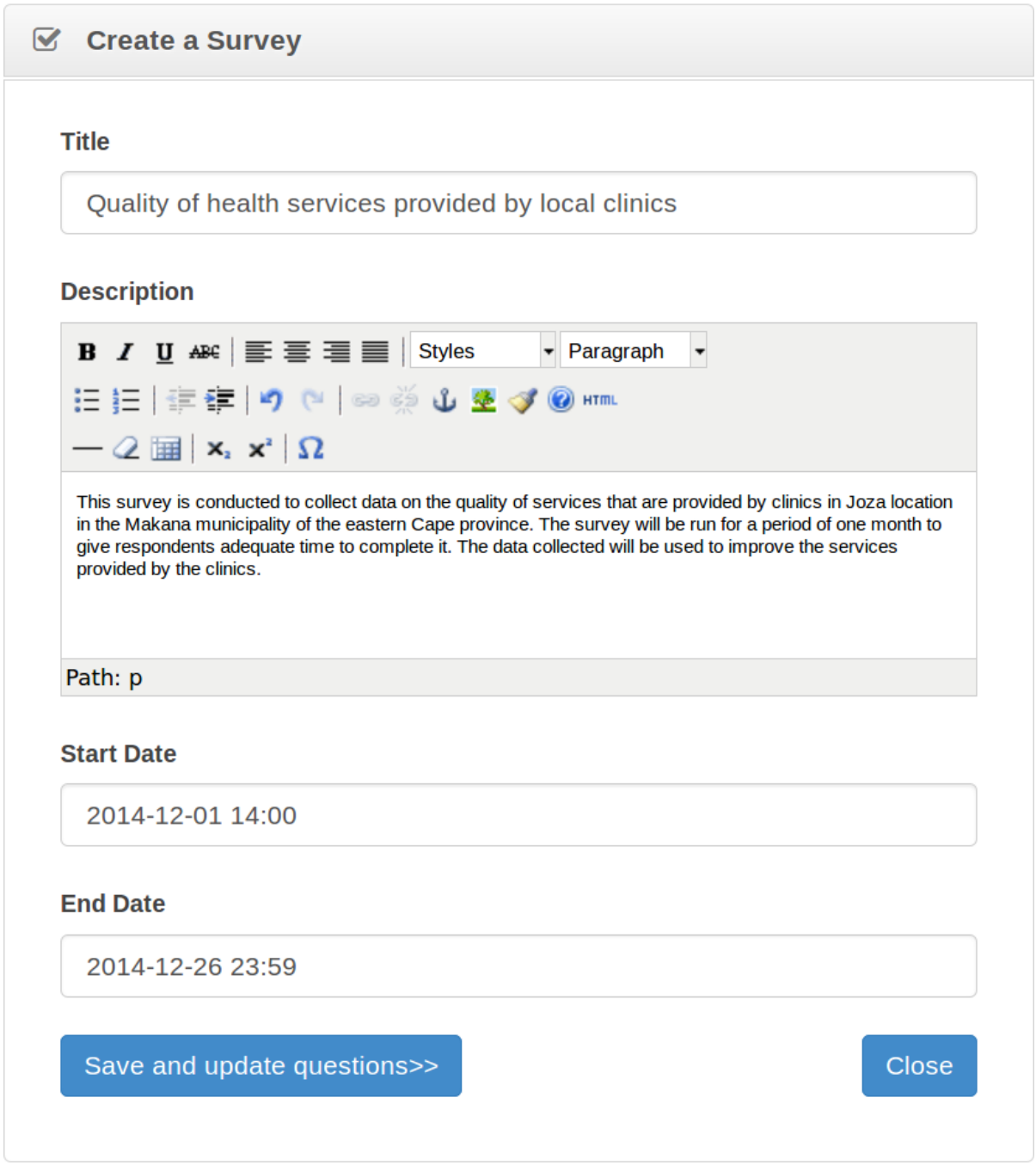

Figure 7.24: Creating a Survey

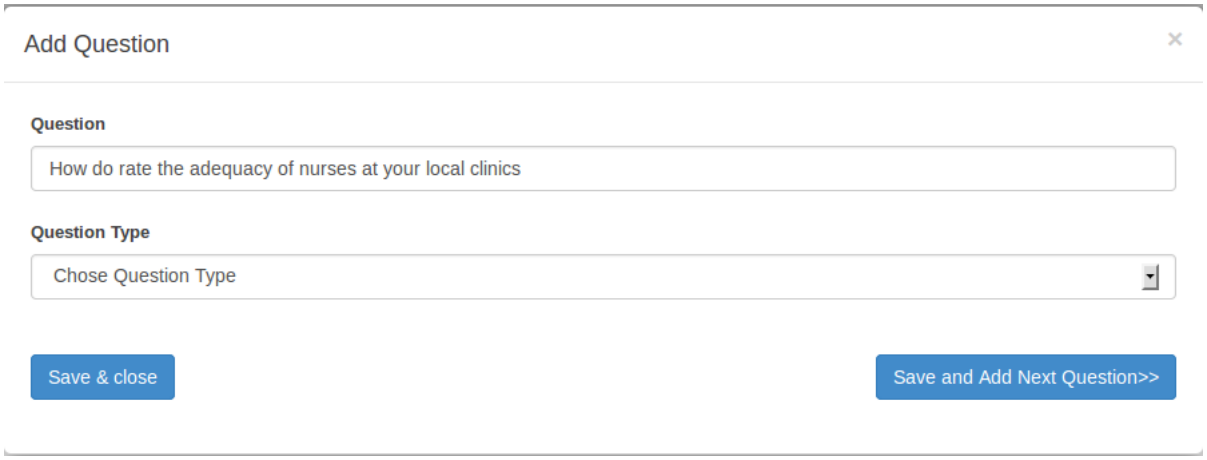

Figure 7.25: A form that enables a user to select question type

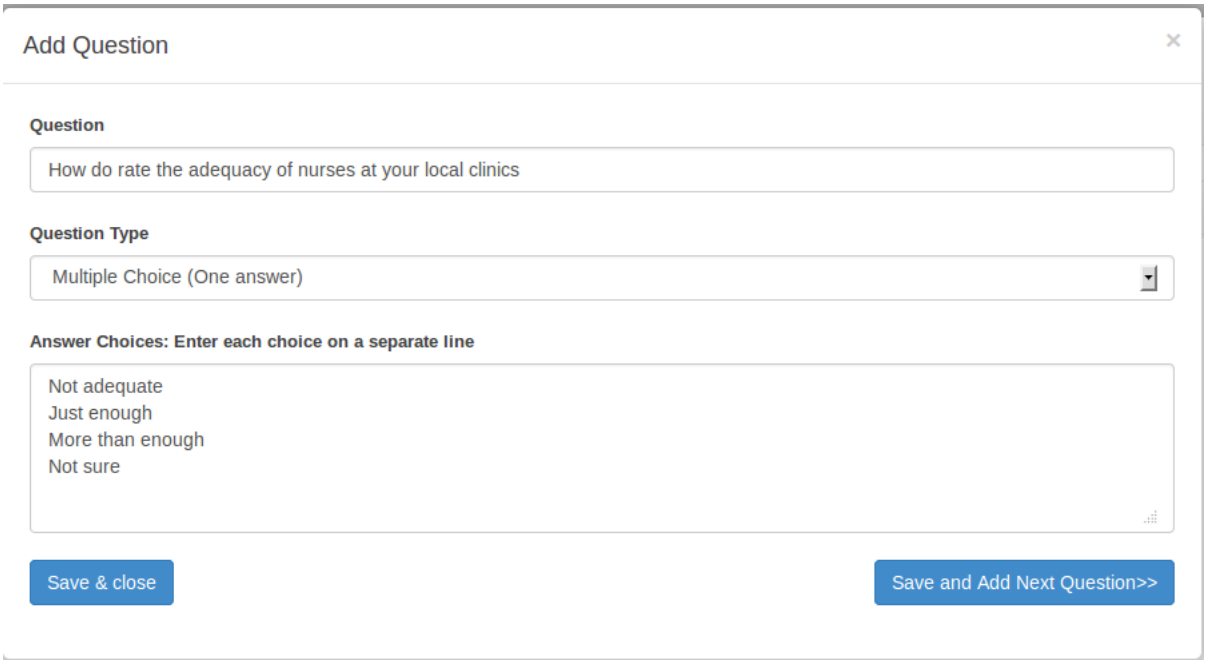

Figure 7.26: A form for adding questions that can have one answer only

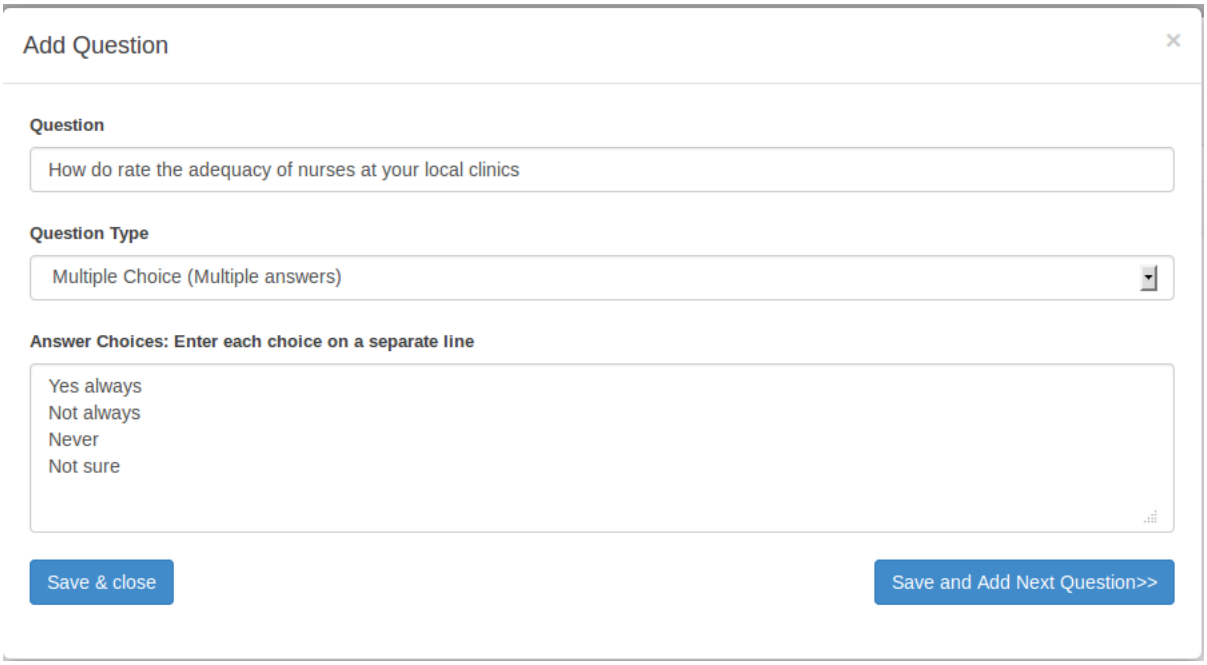

Figure 7.27: A form for adding questions that can have multiple answers

When a user creates survey questions, they can be edited and deleted. Figure 7.28 shows the questions created using the screenshots in Figures 7.25, 7.26 and 7.27.

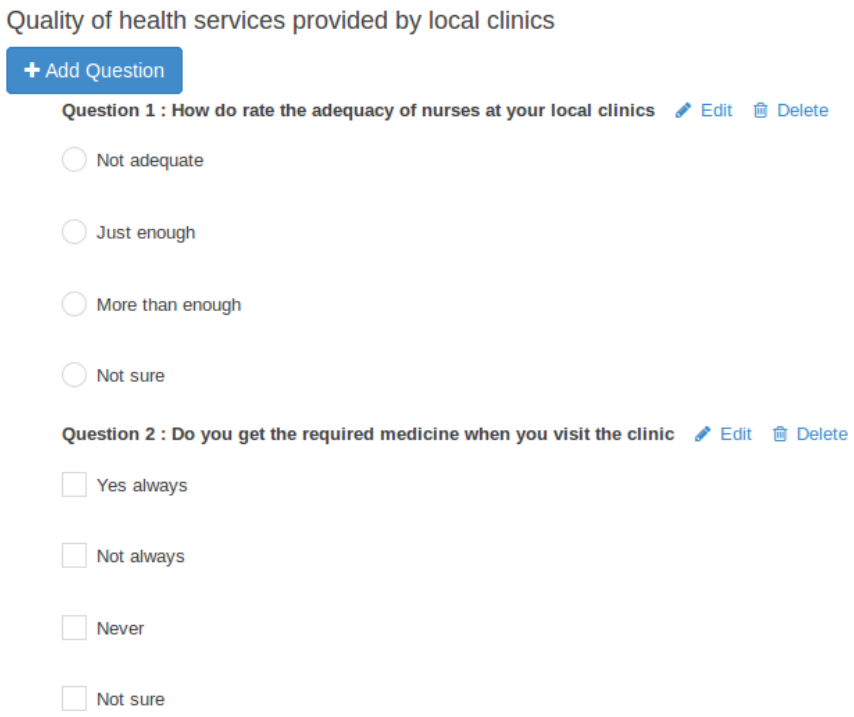

Figure 7.28: List of survey questions

7.5.2 Completing a Survey

The first step to complete a survey is to click the Surveys menu item in the HealthMessenger or a sticky notification displayed when successfully authenticated. Surveys that match a user as a target are presented as a list, when a user clicks on the menu. When a sticky notification is clicked, a page with survey details is presented. Figure 7.29 is a screenshot of a list of surveys accessed by a user named Joe Kaka Mandimba.

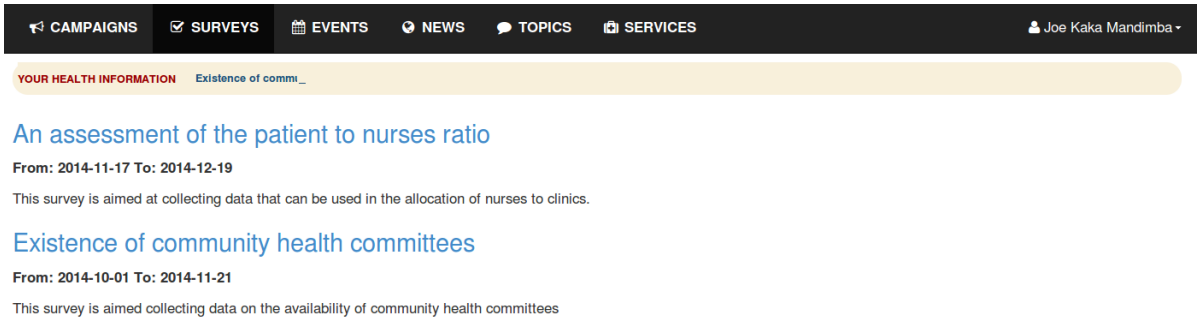

Figure 7.29: List of surveys

When the user clicks of the survey entitled "An assessment of the patient to nurses ratio", the page shown in 7.30 is presented.

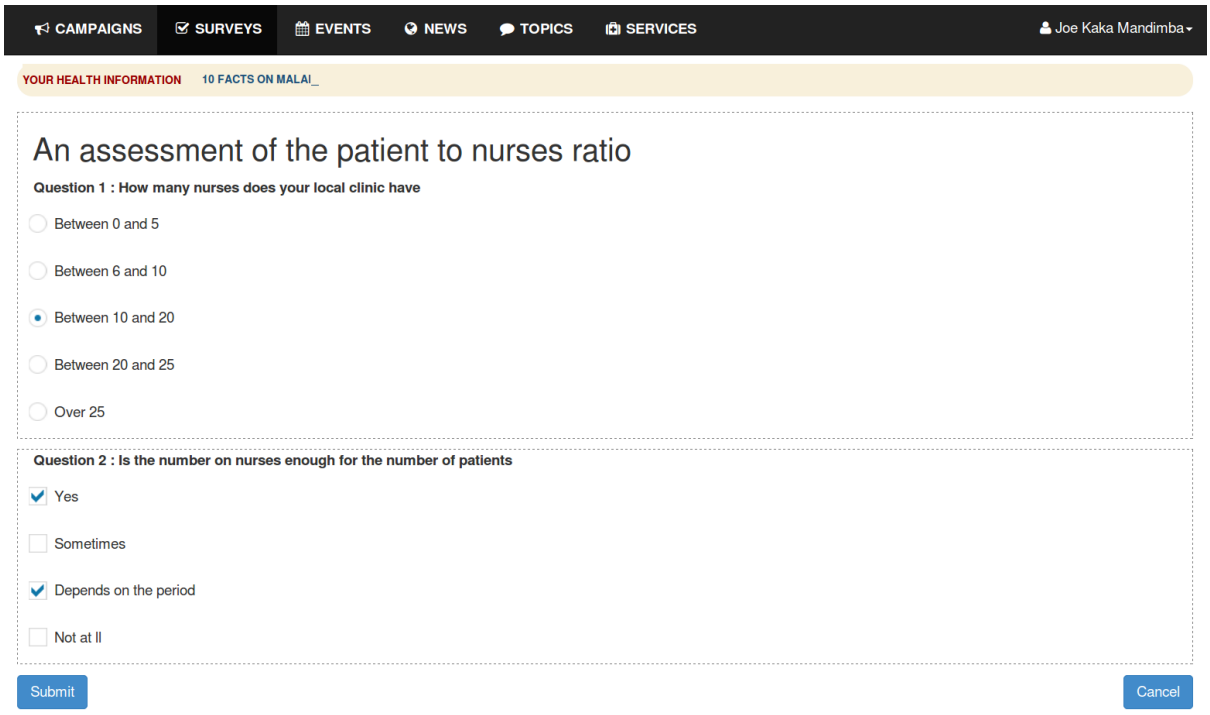

Figure 7.30: Screenshot of a page to complete a survey

The user completes the survey by selecting answer options and clicking the Submit button. After clicking on the button, the responses are published and accessible through the Dashboard.

7.6 Testing the Events Function

The features of the *Events* function that require testing are creating, updating, deleting, stopping, restarting, archiving, and publishing events; registering and deregistering from an event; and viewing completed events. In this section, only creating and registering for an event are discussed because the other features are similar to the *Campaigns* function.

7.6.1 Creating an Event

The feature that requires testing is recording the `name`, `start date and time`, `end date and time`, and `description` of an event. Figure 7.31 is a screenshot of a form to create an event.

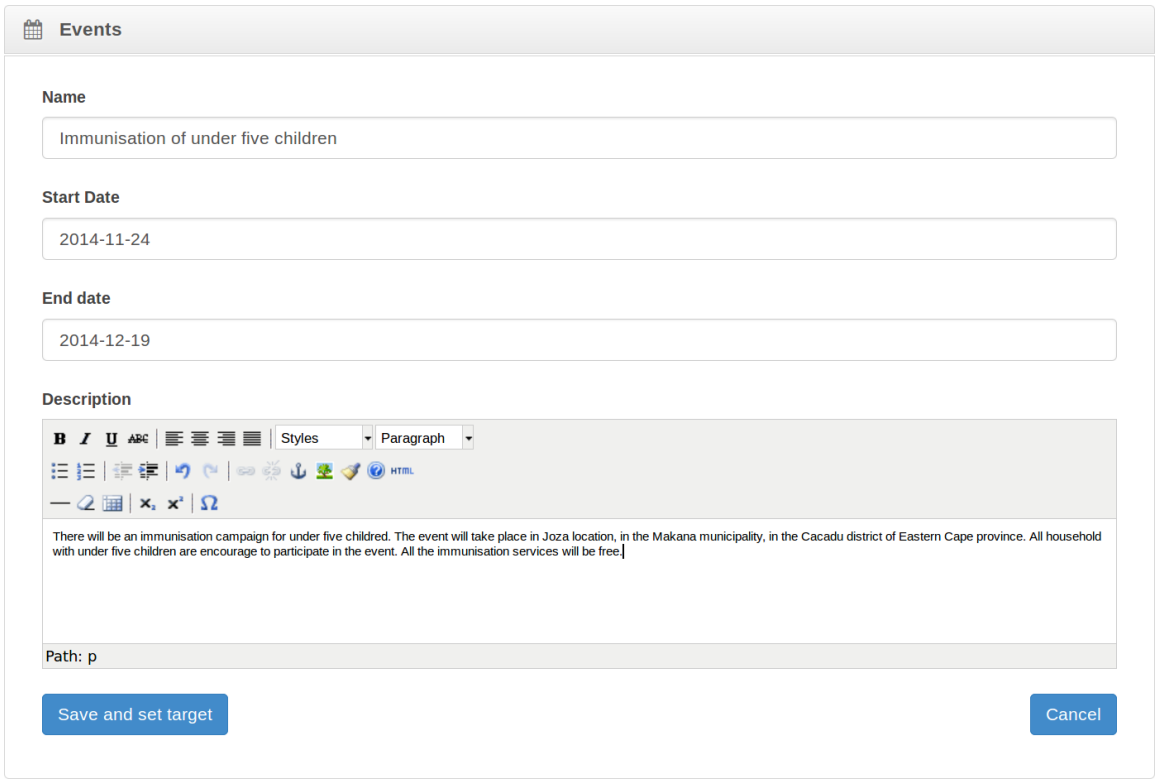

Figure 7.31: Form to create an event

7.6.2 Registering for an Event

A user registers for an event by clicking on the "Register for an event" button and deregisters by clicking on the "Deregister from an event" button. Figure 7.32 is a screenshot of events Joe Kaka Mandimba can register or deregister from.

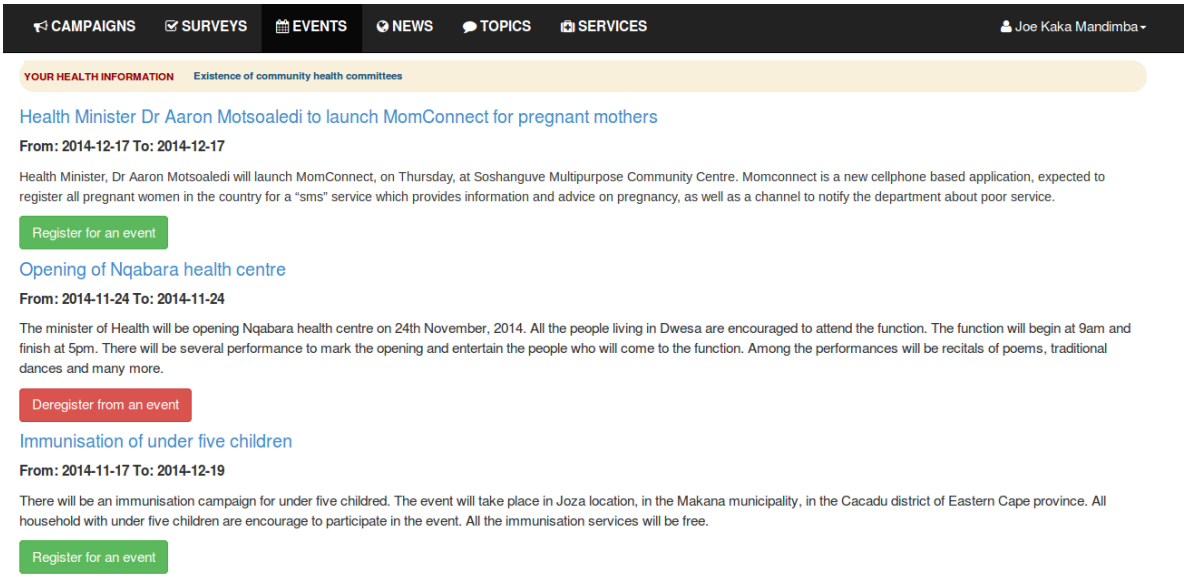

Figure 7.32: Events a user can register

7.7 Testing the News Function

The features of the *News* function that require testing are creating, updating, deleting, archiving, and publishing news; reading news; and generating news access reports. In this section, only creating and reading news are discussed because the other features are similar to the *Campaigns* function.

7.7.1 Creating a News Article

The feature that require testing is recording the `title`, `author`, and `content`. Figure 7.33 is a screenshot of a form to create a news article.

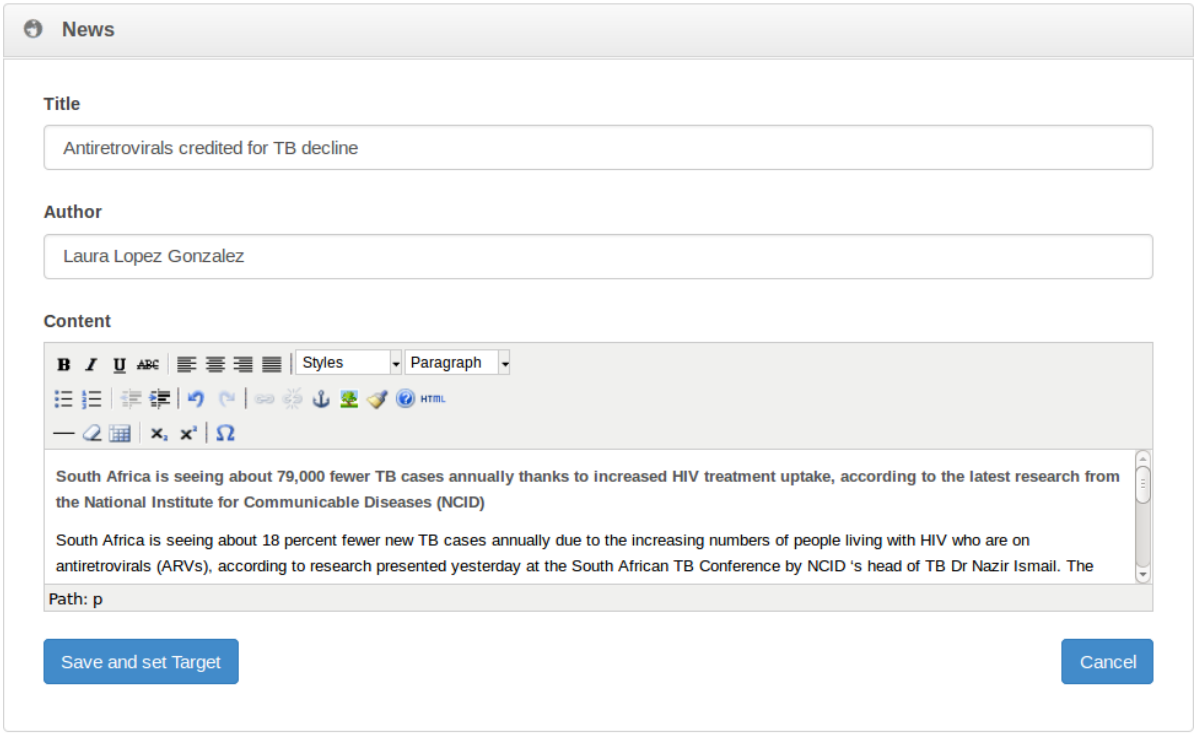

Figure 7.33: A form to create a news article

7.7.2 Accessing News Through the HealthMessenger

Figure 7.34 is a screenshot of a list of news articles accessed by Joe Kaka Mandimba after clicking on the News menu.

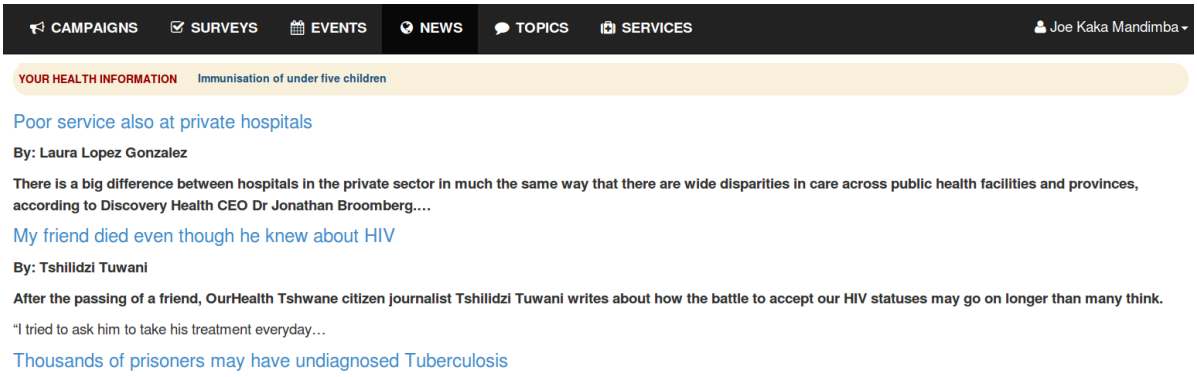

Figure 7.34: List of news items

When the user clicks on the title of a news article, the whole article is displayed as shown in Figure 7.35.

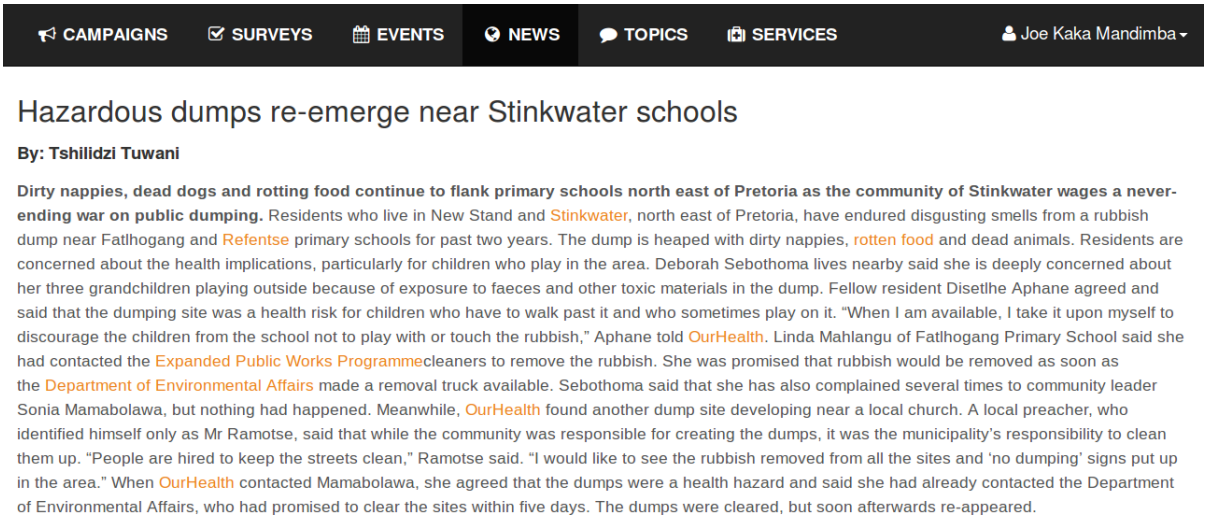

Figure 7.35: News details

7.8 Testing the Topics Function

The features of the *Topics* function that require testing are creating, updating, deleting, and publishing topics; and accessing topic information. In this section, only creating and accessing topics are discussed because the other features are similar to the *Campaigns* function.

7.8.1 Creating a Topic

The feature that require testing is recording the `title`, `description`, and `topic details/sections`. Figure 7.36 is a screenshot for recording the `title` and `description`.

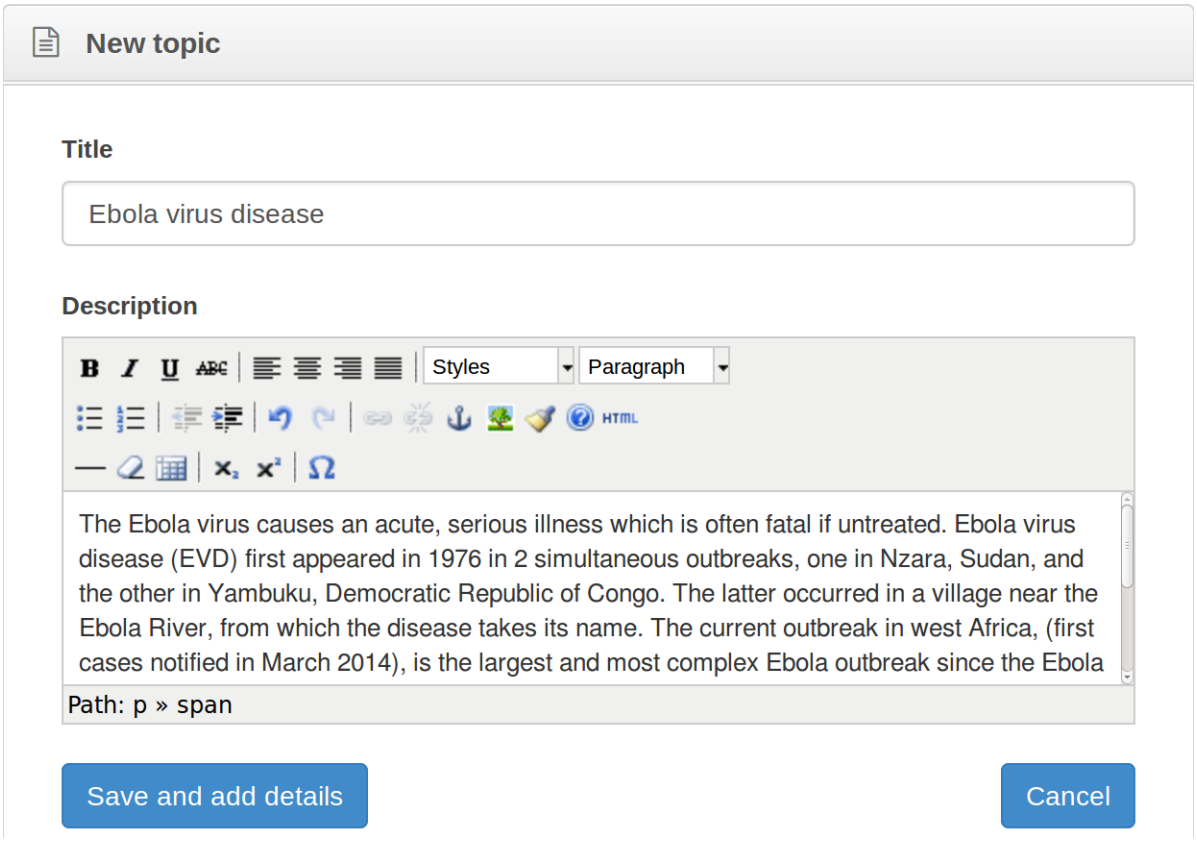

Figure 7.36: Creating a topic

Figure 7.37 is a screenshot of a form to create topic sections or topic details. A `topic section` includes a `title` and `body` or `description` and can be updated or deleted independently of other sections.

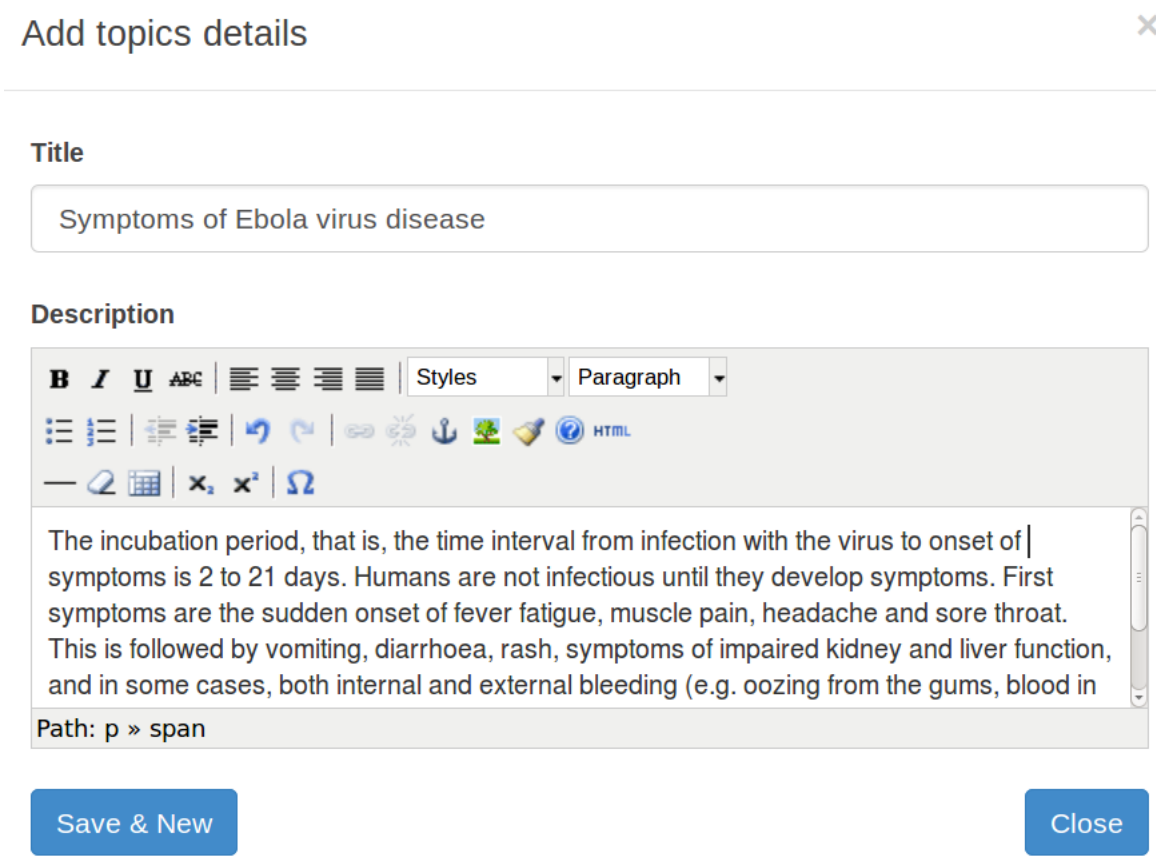

Figure 7.37: Creating topic details

Figure 7.38 is a screenshot showing the details of a topic entitled 'Ebola virus disease'. The details can be edited or deleted independently of each other as per the specifications.

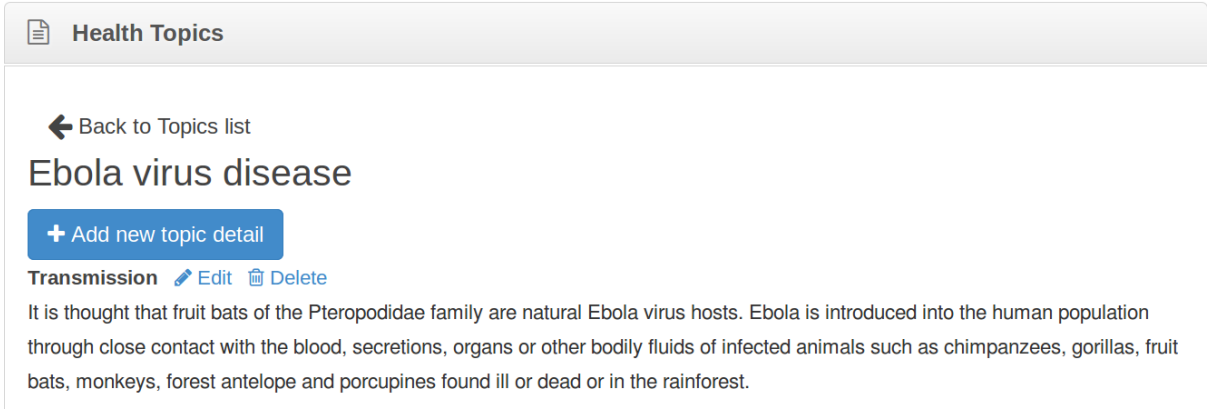

Figure 7.38: List of health topics

7.8.2 Accessing Topics Through the HealthMessenger

Figure 7.39 is a screenshot showing a list of topics published through the Dashboard and accessible through the HealthMessenger (only the title and description are shown).

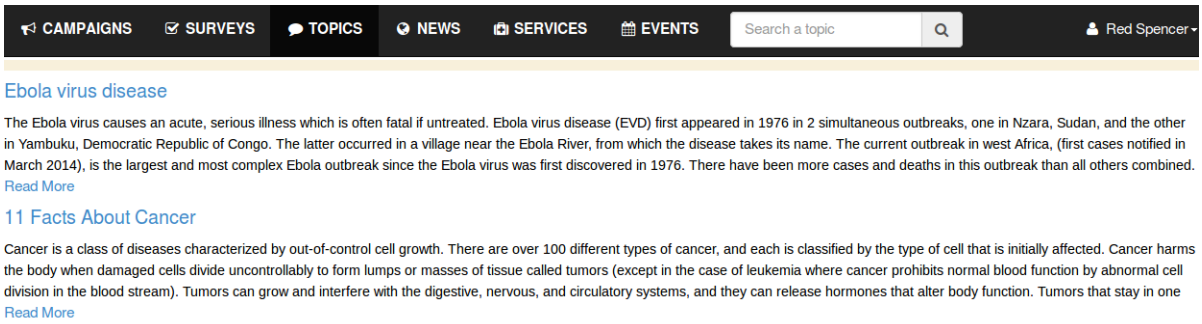

Figure 7.39: A list of health topics

If a topic description is large, only a part of it is shown, with an option to expand its view by clicking on the *Read More* link. To view all the details, a user clicks on a topic link, which opens a new page to view detailed information. Figure 7.40 is a screenshot obtained when the **Ebola virus disease** topic link is clicked.

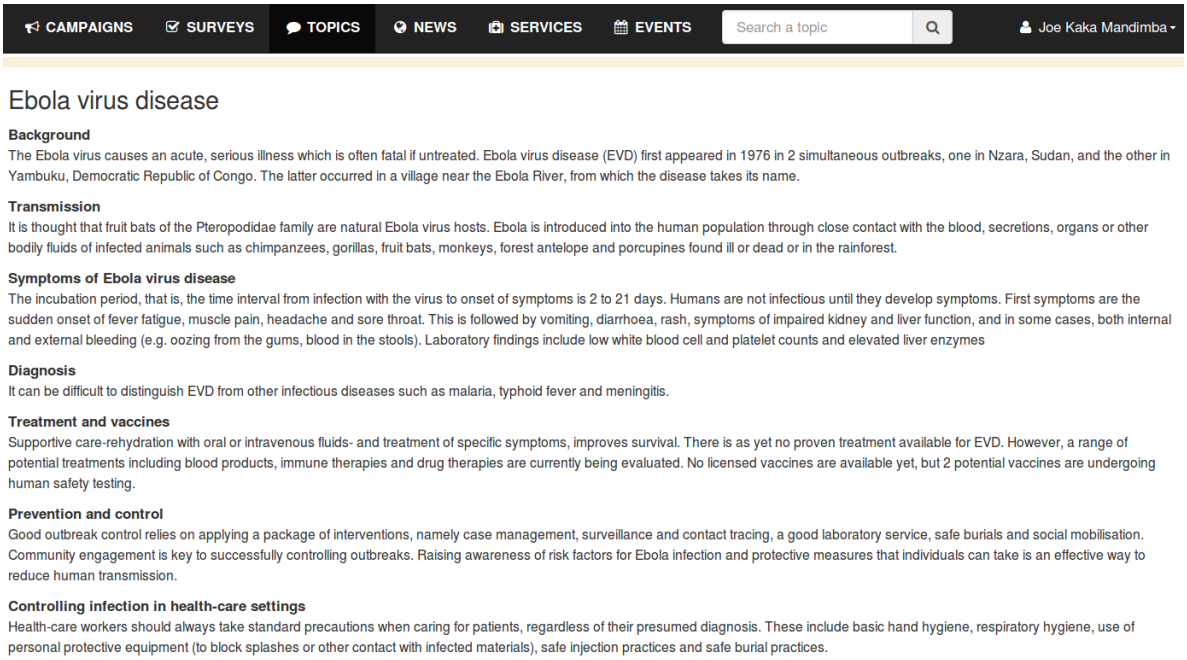

Figure 7.40: Detailed view of a topic

7.9 Testing the Facilities Function

The features of the *Facilities* function that require testing are creating, updating, deleting, and publishing facilities; and accessing information of facilities. Creating facilities includes creating, adding, and removing services from a facility. Accessing facilities includes searching for a facility or a service and getting services offered by a facility. In this section, only creating and accessing facilities are discussed because the other features are similar to the *Campaigns* function.

7.9.1 Creating a Facility

The feature that requires testing is recording the `name`, `type`, and `location` of a facility. The location information includes `province`, `district`, `municipality`, and `village`. Figure 7.41 is a screenshot of a form to create a facility.

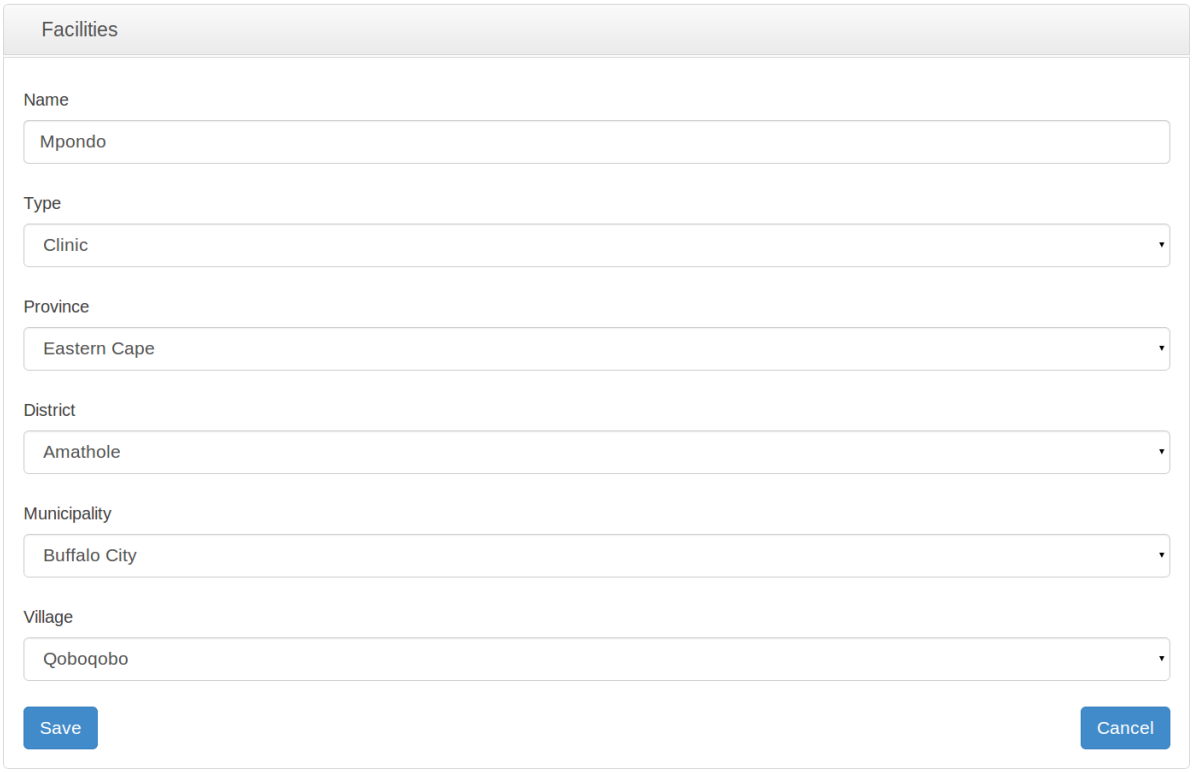

Figure 7.41: Creating a facility

7.9.2 Creating a Service

The feature that requires testing when creating a service is recording the `name`, `description`, and `type`. Figure 7.42 is a screenshot of a form to create a service.

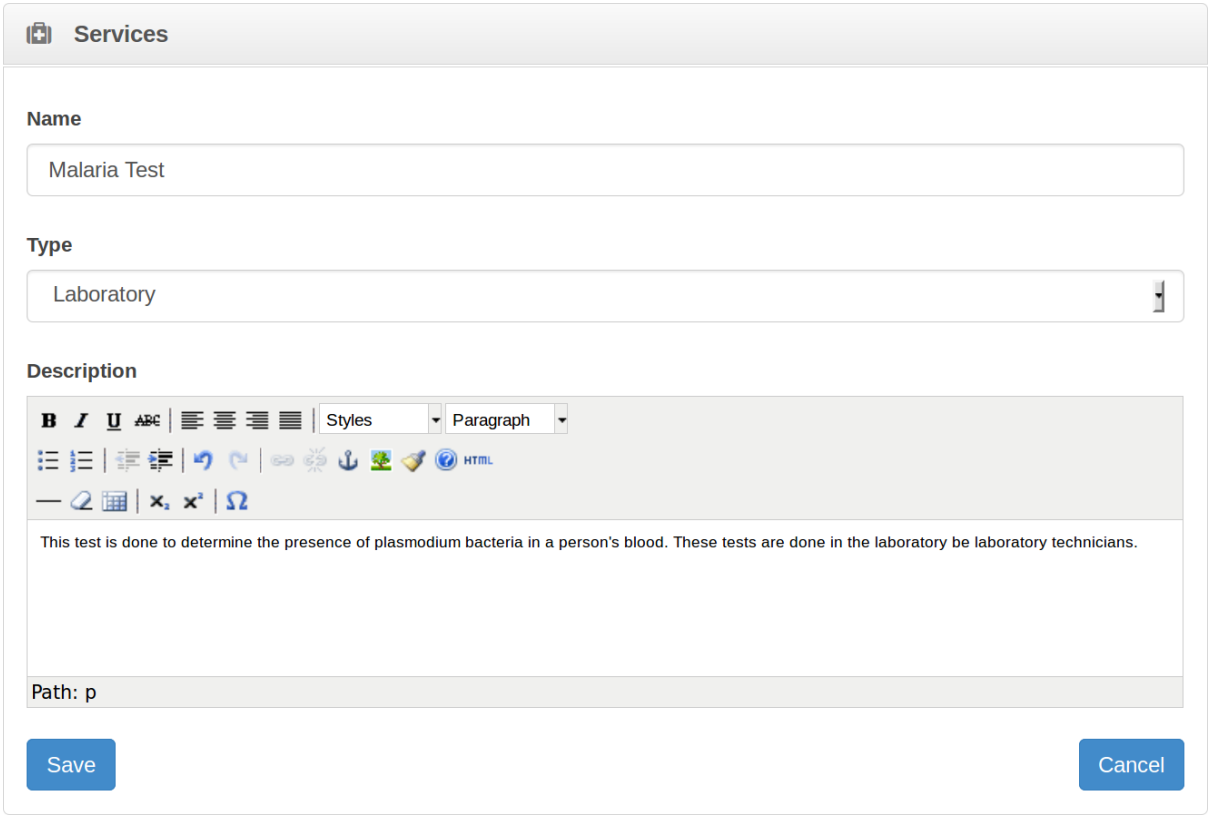

Figure 7.42: Creating a service

7.9.3 Adding and Removing Services From a Facility

Figure 7.43 is a screenshot for adding and removing services from a facility.

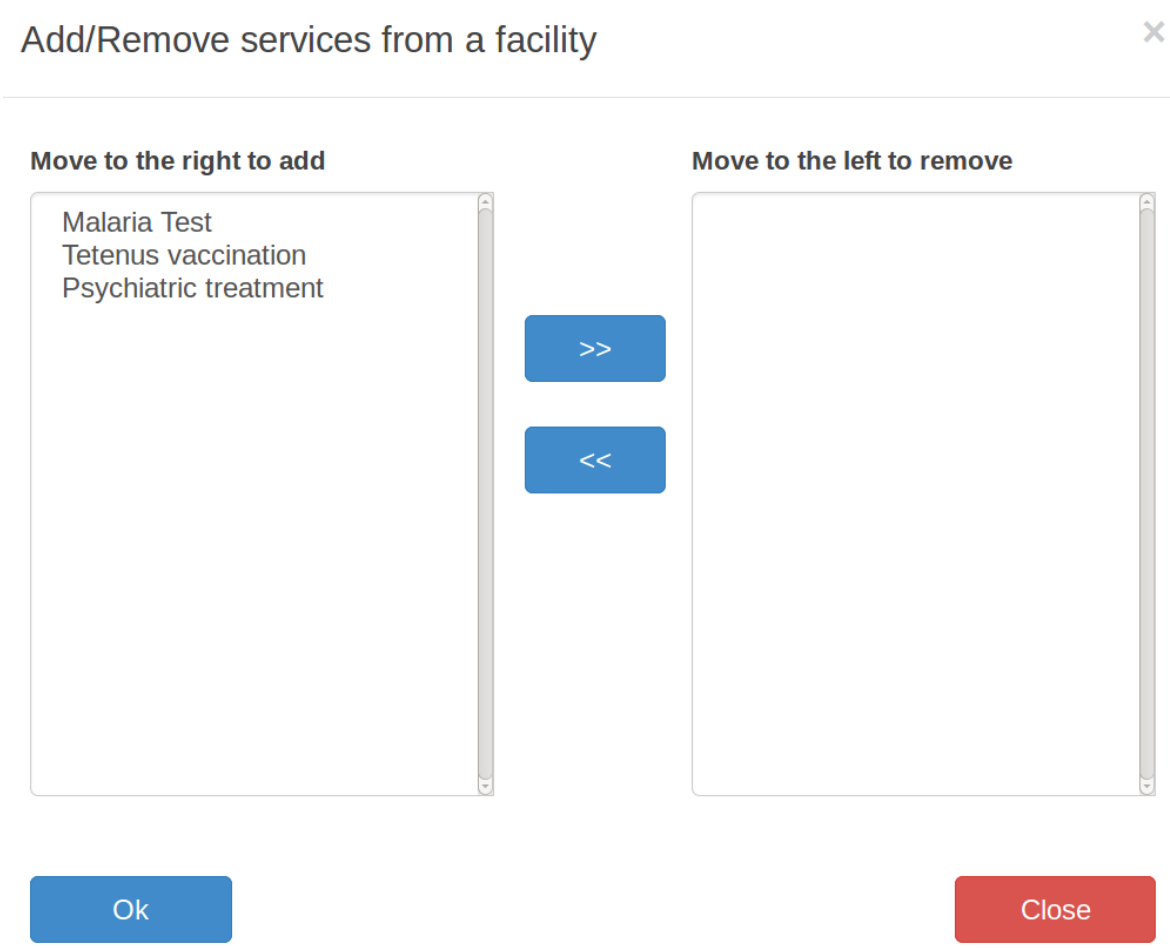

Figure 7.43: Adding and removing services

7.9.4 Accessing Facilities and Services Using the HealthMessenger

The features that require testing are searching for a facility or facilities; searching for services by name; displaying services offered by a facility. Figure 7.44 is a screenshot showing a list of health facilities accessed using the HealthMessenger.

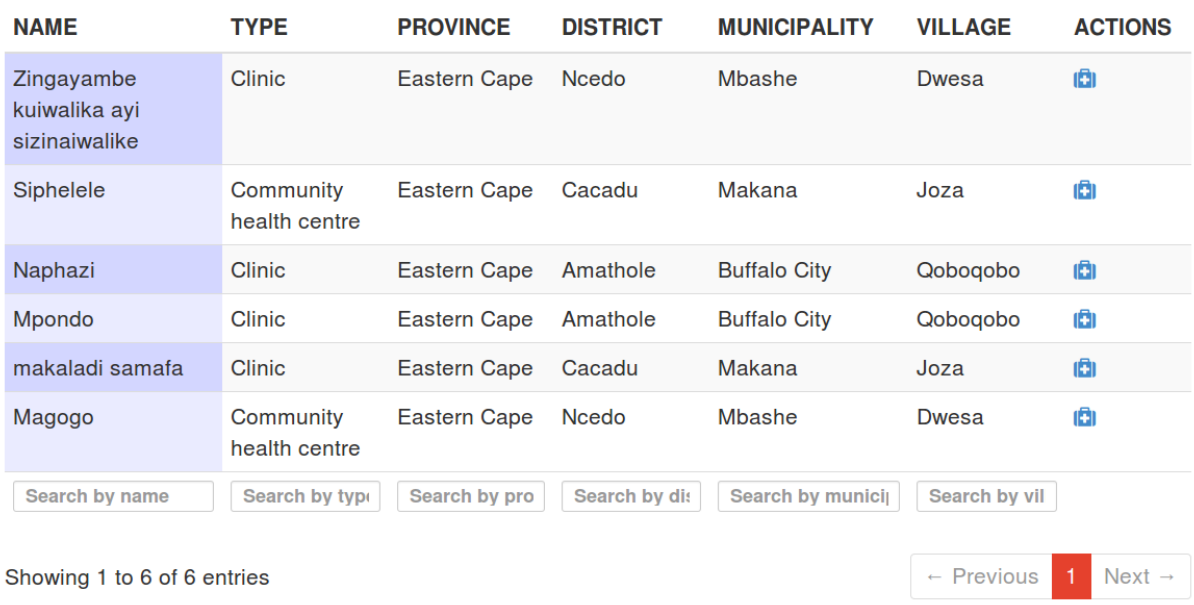

Figure 7.44: List of health facilities

A user can search for a health facility by `name`, `type`, `province`, `district`, `municipality`, and `village`. Figure 7.45 is a screenshot showing a user searching for a facility by district (Amathole).

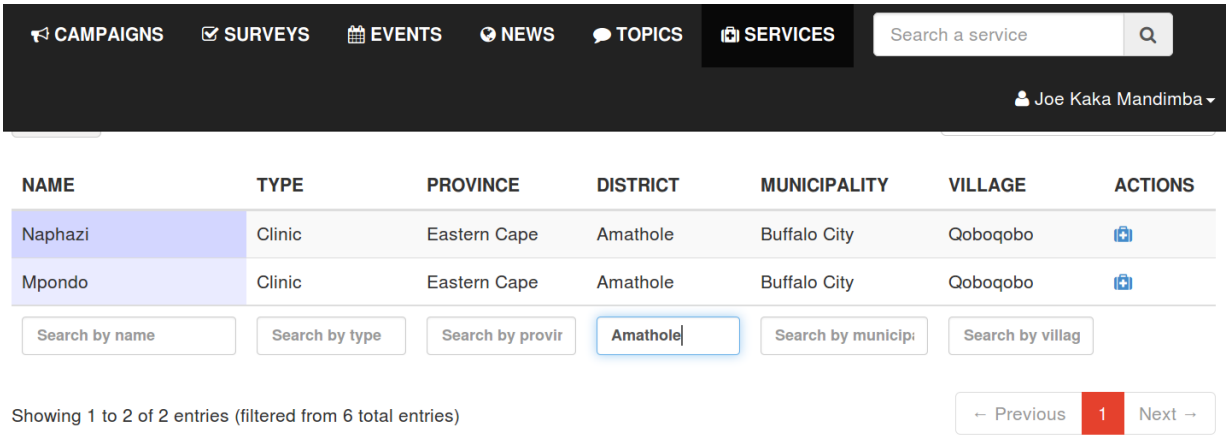

Figure 7.45: A user searching for facilities by district (Amathole)

When a user clicks on the icon in the **Actions** column, services offered by the facility are displayed (Figure 7.46). A user can also search for a particular service by name using the search field provided on the menu bar.

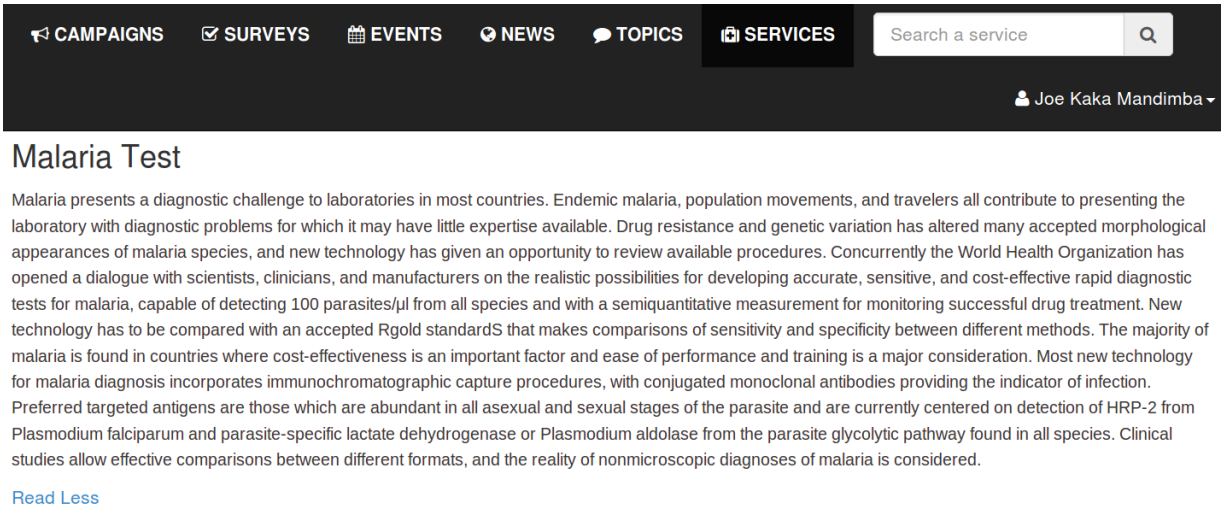

Figure 7.46: Displaying services offered by a facility

7.10 Testing the User Management Function

The features of the *User Management* function that require testing are creating, deactivating, activating, locking, unlocking and deleting a user account, logging in and logging out a user.

7.10.1 Creating a User

The feature that requires testing when creating a user is that the new user account should be in a *Logged Out* state. A new user account can be edited, deleted, deactivated and have its password reset. Figure 7.47 is a screenshot that was taken when a user named Michael Chiswe was created. As shown in the screenshot, the user account is in the logged out state. The status of the account and the icons in the **Actions** column confirm that the function passes the test.

7.10.2 Activating and Deactivating a User

The feature that requires testing when deactivating a user account is that its state should change to *Inactive*. A user account that has been deactivated can be deleted or activated. Figure 7.48 is a screenshot taken when the user that was created in Sub-section 7.10.1 was

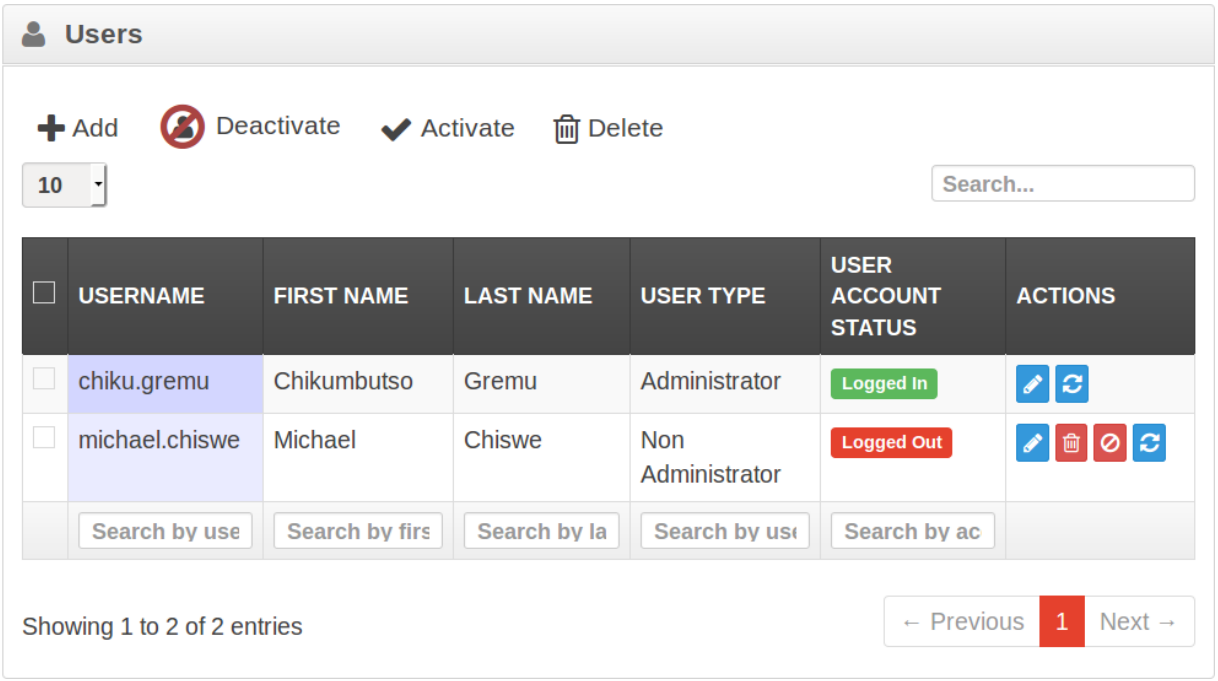

Figure 7.47: New user created

deactivated. As shown in the screenshot, the state of the account changed to *Inactive*. The status of the user account and the icons in the **Actions** column confirm that the function passes the test.

The feature that requires testing when a user account is activated is that its state should change to *Logged Out*. When the user account of Michael Chiswe is activated, the screenshot is as shown in Figure 7.47. This confirms that the function passess the test.

7.10.3 Testing Locking and Unlocking a User Account

The feature that requires testing when an user account is locked is that it state should change to *Locked*. A user account that is locked can be deleted or unlocked. The user account of Michael Chiswe was used to log in to the HealthAware with a wrong password. On the fifth attempt, the user account locked. Figure 7.49 is a screenshot taken when the user account was locked. The icon to the far right in the row containing the user account details of Michael Chiswe is for unlocking a user account. The state of the account together with the icons in the **Actions** column confirm that the function passes the test.

The feature that requires testing when a unlocking a user account is that its status should change to *Logged Out*. When the user account of Michael Chiswe is unlocked, the

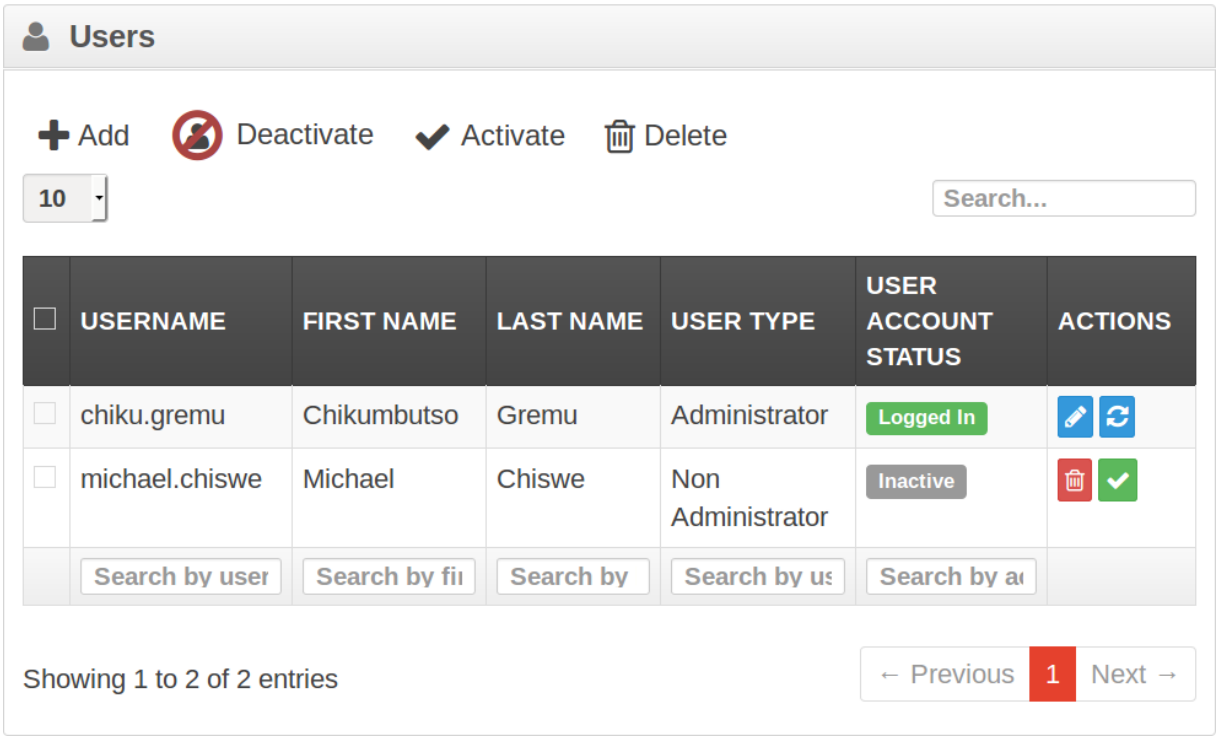

Figure 7.48: User deactivated

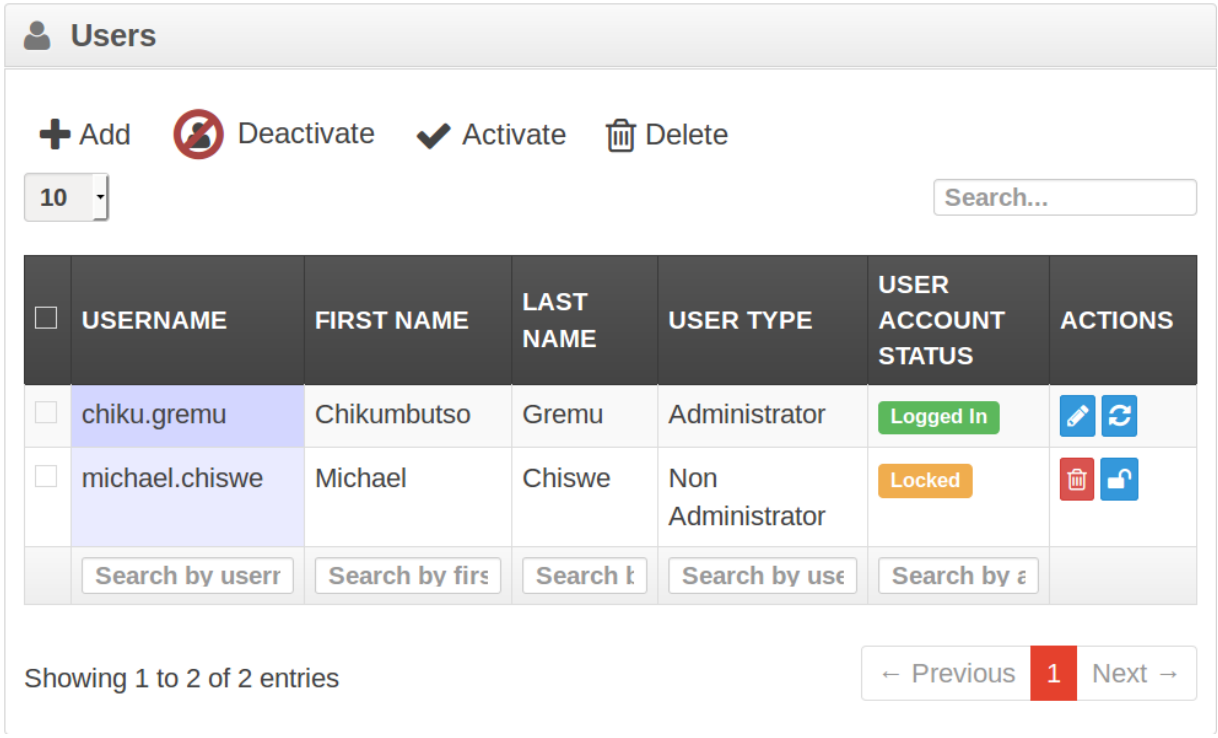

Figure 7.49: Locked user account

screenshot is as shown in Figure 7.47. This also confirms that the function passes the test.

7.10.4 Logging in and Logging out a User

The feature that requires testing when a user is logged in is that the state of the account should change to *Logged In*. When a user account is in *Logged In* state, the system administrator can only deactivate it. Figure 7.50 is a screenshot taken after the user account of Michael Chiswe was used to login to HealthAware. The screenshot confirms that the function passess the test.

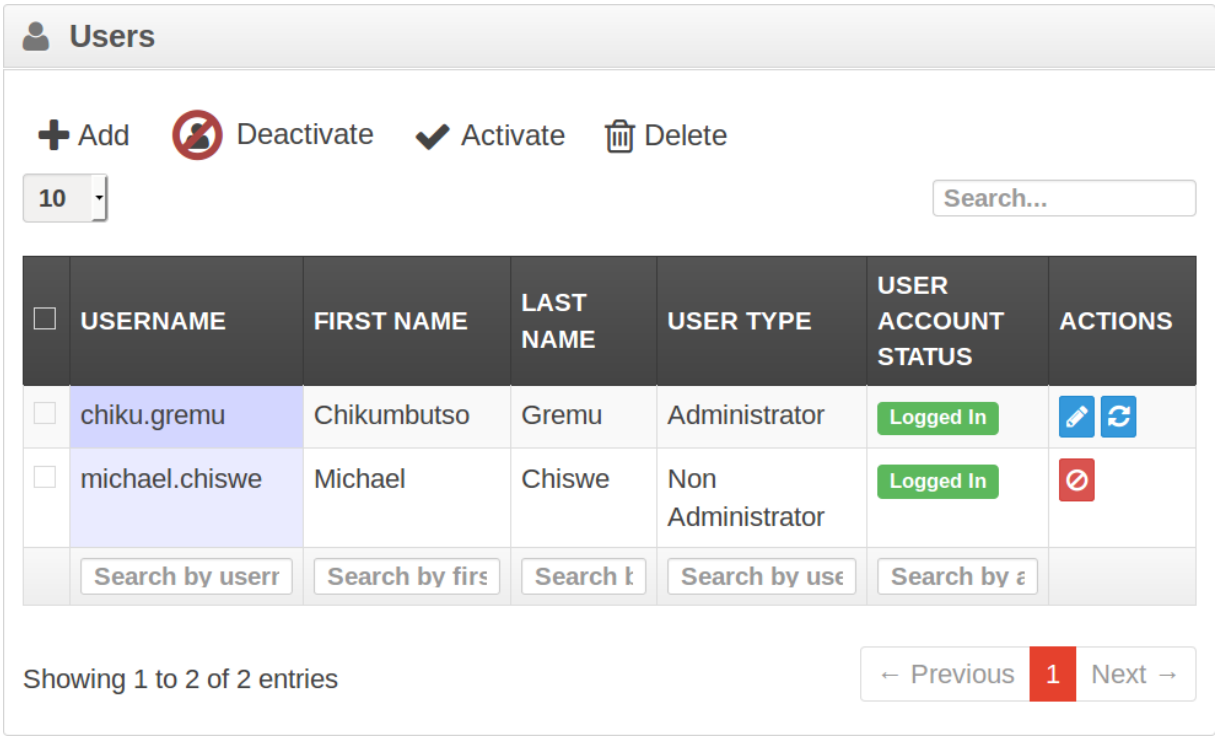

Figure 7.50: User logged in

7.11 Testing Language Switching

The feature that requires testing is switching of languages on the web interface. Due to lack of content in other languages, only the content in English was used. Figures 7.51 and 7.52 are screenshots showing a user switching languages from English to isiXhosa and back.

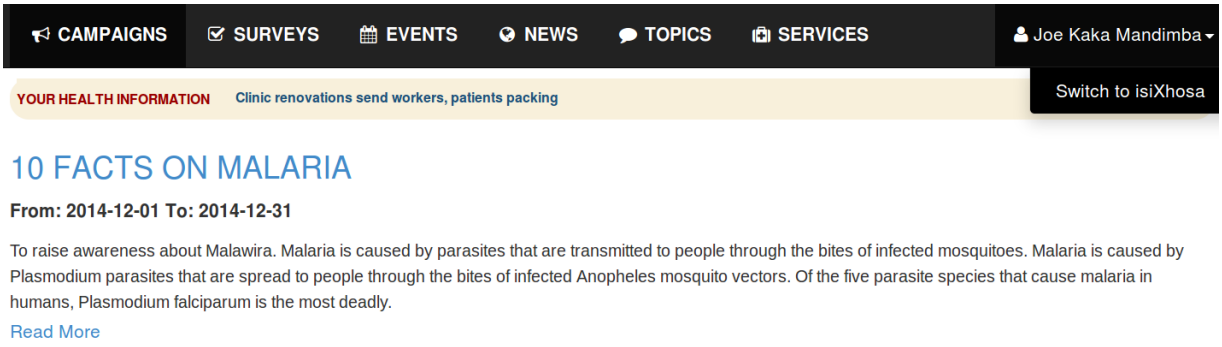

Figure 7.51: Switching language to isiXhosa

Figure 7.52 is a screenshot showing a user accessing the HealthMessenger in isiXhosa. The user can switch back to access the HealthMessenger in English.

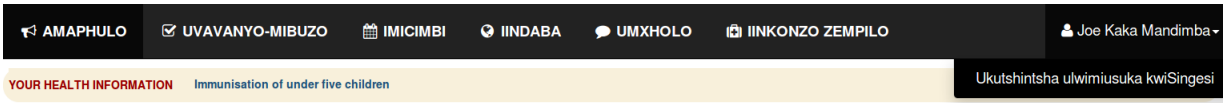

Figure 7.52: HealthMessenger accessible in isiXhosa

7.12 Testing Store and Forward

The feature that requires testing is the connectivity of a network of ActiveMQ brokers when one of the is offline and when both are online. Figure 7.53 is a screenshot taken when only one of the brokers was online. The screenshot shows the broker that was online continuously polling the other broker in order to establish connection, which was continuously being refused. Any data to be exchanged between the brokers is queued.

Figure 7.54 is a screenshot taken when both brokers were online. A communication was established between the brokers and messages can be exchanged.

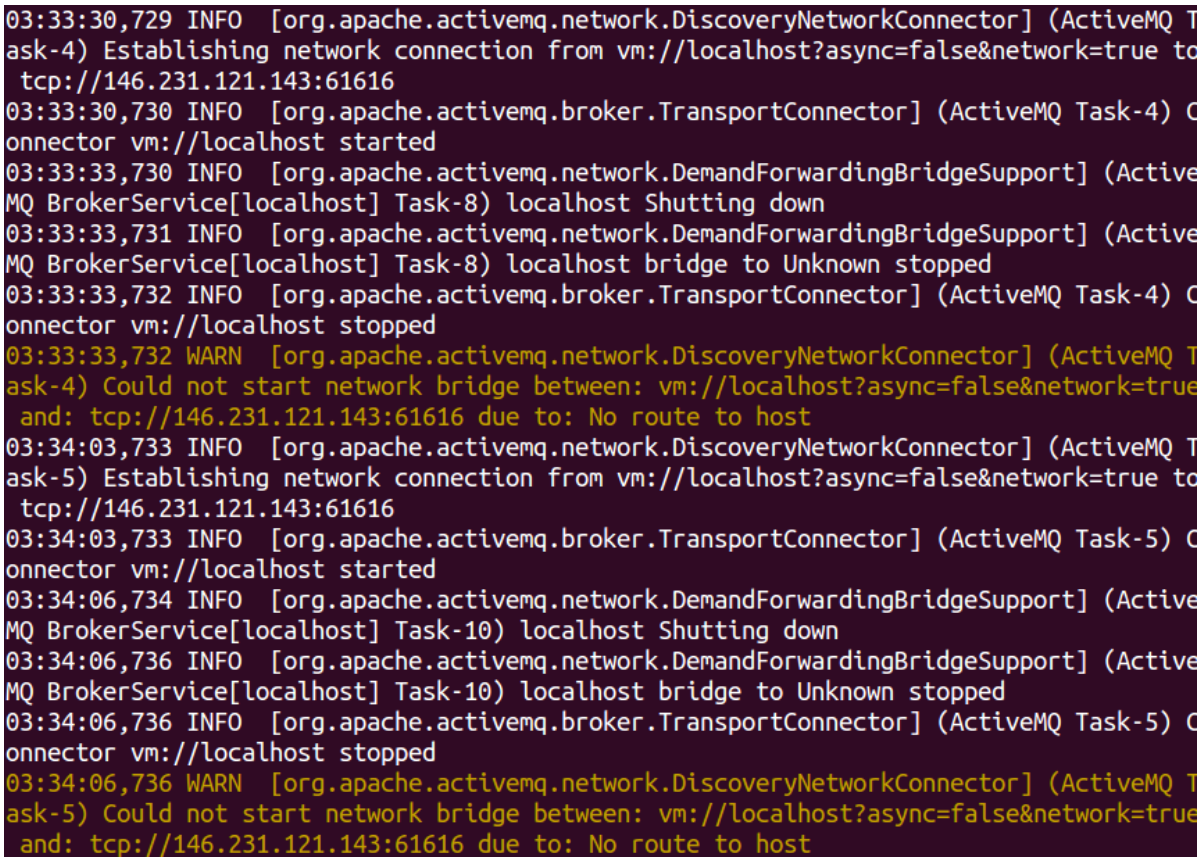

Figure 7.53: Communication when one broker is offline

```
03:37:48,764 WARN  [org.apache.activemq.network.DiscoveryNetworkConnector] (ActiveMQ T
ask-12) Could not start network bridge between: vm://localhost?async=false&network=tru
e and: tcp://146.231.121.143:61616 due to: Connection refused
03:38:18,765 INFO  [org.apache.activemq.network.DiscoveryNetworkConnector] (ActiveMQ T
ask-13) Establishing network connection from vm://localhost?async=false&network=true t
o tcp://146.231.121.143:61616
03:38:18,766 INFO  [org.apache.activemq.broker.TransportConnector] (ActiveMQ Task-13)
Connector vm://localhost started
03:38:18,783 INFO  [org.apache.activemq.network.DemandForwardingBridgeSupport] (trigge
rStartAsyncNetworkBridgeCreation: remoteBroker=tcp:///146.231.121.143:61616@49727, loc
alBroker= vm://localhost#34) Network connection between vm://localhost#34 and tcp:///1
46.231.121.143:61616@49727 (HealthMessenger) has been established.
```

Figure 7.54: Communication when both brokers are online

The test results shows that the store and forward function passed the test.

7.13 Discussion

Section 1.4 discussed the challenges this research aims at addressing. These include the limited source of health information for the people living in poor areas; the requirement of consumers to sift through the general information that is found online before arriving at the required information; most information that is found online being in a language that consumers do not understand with no option of translating it to the language they understand; and the limited sources of funds to sustain the digital infrastructure that is installed in poor areas to bridge the digital divide.

The results of the tests done to verify that HealthAware is working as per specifications confirm that these challenges have been addressed. The targeted strategy that is used to select the awareness campaigns accessed by an individual makes the selection using demographic characteristics and geographical location of an individual. This ensures that the individual access only the campaigs that are relevant to him or her. A consumer can switch the language of presentation on the interface on HealthAware when accessing awareness campaigns in a pull mode. This translates presentation texts and also selects appropriate campaigns for an individual that are in the selected language. The organisations running awareness campaigns using HealthAware can create content in multiple languages which can be matched with the language preference of a consumer. The logging of all user interactions with HealthAware produces data that can be used to bill organisations using HealthAware to run awareness campaigns. The revenue obtained from billing

the organisations can be used to support the development of the digital infrastructure that is installed in poor areas.

In addition to addressing the challenges faced when accessing general health information that is found online and that of sustaining the digital infrastructure that is installed in poor areas, the work being discussed in these thesis ensures that the transfer of data between the components of HealthAware is done reliably. I also ensures that users use HealthAware even when there is no link between its components without them noticing the missing link. This eliminates the challenge of failing to publish awareness campaigns or send back responses of surveys when there is no link.

7.14 Summary

In this chapter the tests done to verify that HealthAware meets the requirements discussed in Chapter 5 were presented. According to the results, HealthAware meets the requirements. The next chapter presents a a summary of what has been discussed in this thesis, assesses the the research objectives and makes a conclusion.

Chapter 8

Summary and Conclusion

8.1 Introduction

This thesis discussed the work to develop HealthAware, an e-Health system to facilitate the implementation of awareness campaigns by health service organisations that target people living in poor areas, and to generate revenue to support the development and upkeep of ICT infrastructure in the areas. This chapter summarises the work and reviews the research objectives introduced in Section 1.5 to assess whether they have been met.

8.2 Summary

This section summarises of the work discussed in this thesis.

8.2.1 Situation Analysis and Related Work

The thesis started with an analysis of the South African public health system in order to understand its challenges and how they affect the people who depend on public health services. The public health system was chosen because the majority of the target population depend on public health services. The thesis also analysed initiatives to provide the South African general public with health information. The analysis helped with the design of HealthAware to ensure that users access only the awareness campaigns that are relevant to them. A discussion of how countries are financing the development of

ICT in poor areas identified several challenges, which include lack of sustainable plans for ICT infrastructure invested in such areas. The work discussed in this thesis used the TeleWeaver business model to extend the functions of HealthAware to generate revenue to support the development and upkeep of infrastructure in poor areas.

8.2.2 Review of Related Literature

The review of related literature analysed, among other topics, the strategies that are used to communicate health messages. These strategies include mass, targeted, and tailored communication. Mass communication has the advantage of reaching a large target population, at a relatively low cost, but at the expense of ensuring that they access only the information that is relevant to them. Targeted communication improves on the weaknesses of mass communication by segmenting a population into subgroups and developing communication messages that are relevant to each subgroup. Tailored communication makes further improvements by developing messages that are relevant to an individual. The disadvantage of tailored communication strategy is that developing and customising messages to fit the situation of an individual is more expensive. The targeted communication strategy was used to implement the algorithm to disseminate awareness campaign messages in HealthAware to ensure that the target population access only those that relevant to them. The strategy was chosen because it is relatively less expensive as compared to tailored communication.

8.2.3 Review of Consumer Health Websites

Websites developed to provide consumers with access to health information were reviewed. The websites included those for local and international organisations. A number of common functions in the websites were identified and analysed to determine if they were appropriate for the HealthAware. The functions that were considered appropriate were selected and included in HealthAware. The functions that were excluded were considered either challenging to implement for the proposed design of HealthAware or not critical to achieve the research objectives.

8.2.4 Design and Implementation of HealthAware

The functions selected during the review of consumer health websites were used to define the requirements of HealthAware. These functions included *Campaigns*, *Events*, *News*, *Topics*, *Facilities*, and *Services*. In order to enable health service organisations collect data from the target population, the *Survey* function was included. A distributed architecture was used implement the HealthAware. The architecture was used to split the system into two independent component applications. A layered architecture that splits the implementation of a function into presentation, business, and data access layers was used to implement specific functions of HealthAware.

8.2.5 Testing of HealthAware

The functions of HealthAware were tested during and after the development work was completed. The tests after the development work were done in an experimental environment. Black box testing method was used to test the functions of HealthAware to determine if they meet the requirements specification. The test results showed the application to meet the requirements.

8.3 Meeting the Research Objectives

The previous section presented a summary of what was discussed in this thesis. This section analyses the research objectives introduced in Section 1.5 to assess whether they have been met.

8.3.1 e-Health System to Facilitate Targeted Awareness Campaigns

Target users access awareness campaigns in either a pull or a push mode. In each mode, a targeted strategy is used to select the campaigns that match a user as a target. As discussed in 5.2.5, when creating a campaign, demographic characteristics and geographical locations are used to specify the target population. When a user is authenticated, a search for campaigns that are in a running state and the user match as a target is

done. Matching is done by comparing the demographic characteristics and geographical locations of a user with the target of a campaign. When the demographic characteristics and geographical locations set on a campaign are a subset or same as that of a user, it is considered a match.

The *Campaigns* function was used to test if the targeting function met the requirements. A test schedule containing campaigns and their targets, user accounts and their demographic and geographical locations and the expected outputs was prepared and used to test the function. The campaigns had different targets and the user accounts had different demographic and geographical locations. The test results showed that the expected output and the actual output matched confirming that this objective was met.

8.3.2 A Function to Generate Revenue to Support the Development and Upkeep of ICT Infrastructure

Revenue generation in HealthAware was implemented using a charging framework that is similar to the pay-per-click that is used by Google, Yahoo, and Bing to charge online advertisers. User interactions with awareness campaigns and surveys are logged for billing purposes. The interactions are clicking to view a campaign message, completing a survey questionnaire, registering for an event, and reading a news article. In the absence of the billing application, an object containing billing data is created but not persisted to the database. When the billing application becomes available, the billing data will be persisted to the database of the billing application.

Only white box testing was used with this function to ensure that the object containing the billing data generates the data as required. A black box test will be done when the billing application becomes available. The white box testing confirmed that the function meets the requirements, therefore, this objective was also met.

8.4 Fulfilling Solution Requirements

The system had to fulfill the requirements discussed in the following sections.

8.4.1 Hosted on TeleWeaver

The tools that were used to develop the HealthMessenger were constrained by TeleWeaver. As introduced in Section 1.9, TeleWeaver is based on Wildfly 8.2.0.Final, which was developed to host Java applications. As discussed in Section 6.2, Java programming language and associated Java tools were used to develop the HealthMessenger. The same tools were used to develop the Dashboard for simplicity. An experimental environment was prepared using Wildfly 8.2.0.Final, which is the version TeleWeaver is based on and the HealthMessenger was deployed and tested successfully in the environment. It is expected that deployment of HealthAware on TeleWeaver will require minimal or no challenges at all. This requirement was also fulfilled.

8.4.2 Accessible in Multiple Languages

HealthAware is accessible in English and isiXhosa as shown in Section 7.11. Any number of languages can be included if translations are done. This requirement was fulfilled as shown by the test results of the function.

8.4.3 Usable in Low Speed and Intermittent Internet Connectivity Environments

As discussed in Sub-section 5.2.1, HealthAware was split into two application components. The split ensures that both components are usable even when the HealthMessenger is not accessible through the Internet. The challenge with this setup is that publishing of information from the Dashboard to the HealthMessenger is not possible when the HealthMessenger is offline. To solve this challenge, a store and forward technology using ActiveMQ was used as discussed in Section 6.6. The store and forward reliably publishes awareness campaigns as shown by the test results in Section 7.12 indicating that the requirement was fulfilled.

8.4.4 Logging of User Interactions for Billing

As discussed in Sub-section 8.3.2, this requirement is fulfilled.

8.5 Contributions

The work discussed in this thesis makes contributions to the field of Consumer Health Informatics and ICT for Developement. The contribution to the field of Consumer Health Informatics is the use of communication strategies to ensure that consumers access only the health information that is relevant them. This solves the challenge imposed by generic materials of sifting through potentially irrelevant information before arriving at the important information [25]. As discussed by [45], communities are heterogeneous with varied cultural, educational, social backgrounds, and information needs, which render disseminating general information less effective.

The contribution to the field of ICT for Development is the use of HealthAware to generate revenue to support the development of digital infrastructure that is installed in poor areas. The function to generate revenue to support the development digital infrastructure in poor areas uses the TeleWeaver business model. The model was developed following the challenges that were faced with the development of the digital infrastructure installed under the SLL. This is the first time the model has been implemented in an application.

In addition to the contributions listed above, HealthAware will alleviate the challenges health organisations face when they try to reach out to people in poor areas. When the HealthAware is implemented, health service organisations will be required tp upload the health messages using the Dashboard and the HealthMessenger will be responsible for the delivery to the intended audience.

8.6 Future Work

The HealthMessenger was not implemented and tested on TeleWeaver production environment. Future work should ensure that this is done. Usability tests should also be done with the target users. The legal requirements to process personal information should be met before testing with the users. The relationship with the Eastern Cape Department of Health should also be consolidated so that the tool is deployed for real life use by the department.

8.7 Summary and Conclusion

This chapter summarised the work discussed in this thesis and assessed the objectives of the research to determine if they had been met. The assessment results show that the objectives were met. This research contributes knowledge to develop e-health applications that ensure that users access only the health information that is relevant to them and that can generate revenue to support ICT infrastructure in poor areas.

References

[1] S. S. Abdool Karim, G. J. Churchyard, Q. Abdool Karim, and A. D. Lawn. HIV infection and tuberculosis in South Africa: an urgent need to escalate the public health response. http://www.sudafrica.cooperazione.esteri.it/utlSudafrica/EN/download/pdf/HivInfection_TB_UrgentNeed_PubblicHealthResponse.pdf. Online [Last accessed: 09/08/2014], August 2009.

[2] A. Akue-Kpakpo. Study on international Internet connectivity in sub-Saharan Africa. Technical report, International Telecommunication Union, March 2013.

[3] E. Almirall, M. Lee, and J. Wareham. Mapping Living Labs in the Landscape of Innovation Methodologies. http://timreview.ca/sites/default/files/article_PDF/Almirall_et_al_TIMReview-_September2012.pdf. Online [Last accessed: 14/02/2014], Septermber 2012.

[4] L. Alpay, J. Verhoef, B. Xie, Te'eni D., and J. H. M. Zwetsloot-Schonk. Current Challenge in Consumer Health Informatics: Bridging the Gap between Access to Information and Information Understanding. *Biomedical Informatics Insights*, 1:1–10, 2009.

[5] P. Anetos. US-sponsored MomConnect app provides free antenatal information. http://www.bdlive.co.za/national/health/2014/08/21/us-sponsored-momconnect-app-provides-free-antenatal-information. Online [Last accessed: 12/09/2014], August 2014.

[6] H. Attwood, K. Diga, E. Braathen, and J. Mat. Telecentre Functionality in South Africa: Re-Enabling The Community ICT Access Environment. *The Journal of Community Informatics*, 9, 2013.

[7] A. Barreiro. Authenticating users: Going beyond the password. http://www.techrepublic.com/blog/it-security/authenticating-users-going-beyond-the-password/. Online [Last accessed: 31/07/2014], November 2011.

[8] C. Bauer and G. King. *Java Persistence with Hibernate*. Manning Publications Co., 2007.

[9] D. Bell. *Software Engineering for Students for Students: A Programming Approach*. Addison Wesley Longman, 2005.

[10] S. Benatar. The challenges of health disparities in South Africa. *The South African Medical Journal*, 103:154–155, 2013.

[11] G. G. Bennett and R. E. Glasgow. The Delivery of Public Health Interventions via the Internet: Actualizing Their Potential. *The Annual Review of Public Health*, 30:273–292, 2009.

[12] T. W. Bickmore and M. K. Paasche-Orlow. The Role of Information Technology in Health Literacy Research. *Journal of Health Communication*, 17:23–29, 2012.

[13] Bing ads. What is pay-per-click (PPC) advertising? http://advertise.bingads.microsoft.com/en-us/help-topic/how-to/51031/what-is-pay-per-click-ppc-advertising. Online [Last accessed: 20/08/2013].

[14] A. M. Bowen, K. Horvath, and M. L. Williams. A randomized control trial of Internet-delivered HIV prevention targeting rural MSM. *Health Education R*, 22:120–127, 2007.

[15] R. Boyd. *Getting started with OAuth 2.0*. OŔeilly Media, Inc., 2012.

[16] S. Budweg, H. Schaffers, R. Ruland, K. Kristensen, and W. Prinz. Enhancing collaboration in communities of professionals using a Living Lab approach. *Production Planning & Control*, 22:594–609, 2011.

[17] California STD/HIV Prevention Training Center. What Is Social Marketing? http://www.stdhivtraining.org/YSMT_socmarketing.html. Online [Last accessed: 10/02/2014].

[18] M. M. Cassel, C. Jackson, and B. Cheuvront. Health Communication on the Internet: An effective Channel for Health Behavior Change? *Journal of Health Communication*, 3:71–79, 1998.

[19] Centre for Addiction and Mental Health. Gather and Analyze Information: How to get the information you need. http://www.camh.ca/en/hospital/about_camh/health_promotion/culture_counts-/Pages/culture_counts_chap3_howto.aspx. Online [Last accessed: 25/02/2014].

[20] M. J. Clark, W. McLaughlin, and L. A Straub. Health Communication: A Strategy for Reaching Rural Residents. *Rural Research Report*, 13:1–8, 2002.

[21] E. Coiera. Communication Systems in Healthcare. *The Clinical Biochemist Reviews*, 27:89–98, 2006.

[22] Community Tool Box. Understanding Social Marketing: Encouraging Adoption and Use of Valued Products and Practices. http://ctb.ku.edu/en/sustain/social-marketing/overview/main. Online [Last accessed: 10/02/2014].

[23] K. Cullinan. Health services in South Africa: A basic introduction. http://www.health-e.org.za/2006/01/29/health-services-in-south-africa-a-basic-introduction. Online [Last accessed: 20/02/2013], January 2006.

[24] L. Dalvit, I. Siebörger, and H. Thinyane. The Expansion of the Siyakhula Living Lab: A Holistic Perspective. *e-Infrastructure and e-Services for Developing Countries Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 92:228–238, 2012.

[25] S. Davis. Internet-Based Tailored Health Communications: History and Theoretical Foundations. *The Journal of Education, Community and Values*, 7:1–9, June 2007.

[26] I. de Lanerolle. The New Wave. Technical report, University of Witwatersrand, 2012.

[27] Department of Health. eHealth Strategy South Africa. http://www.health.gov.za/docs/strategic/2012/eHealth_Strategy_South_Africa_2012-2016.pdf. Online [Last accessed: 02/02/2014], 2012.

[28] Directorate-General for the Information Society and Media. Living Labs for user-driven open innovation. http://www.eurosportello.eu/sites/default/files/LivingOnline [Last accessed: 14/02/2014], January 2009.

[29] H. Dugmore. The Siyakhula Living Lab: An important step forward for South Africa and Africa. Technical report, Rhodes University, 2012.

[30] D. Esposito. Pros and Cons of Data Transfer Objects. http://msdn.microsoft.com/en-us/magazine/ee236638.aspx. Online [Last accessed: 29/08/2014], August 2009.

[31] G. Eysenbach. Consumer Health Informatics. *British Medical Journal*, 320:1713–1716, 2000.

[32] G. Eysenbach. What is e-health? *Journal of Medical Internet Research*, 3:1–2, 2001.

[33] G. Eysenbach and T. L. Diepgen. The Role of e-Health and Consumer Health Informatics for Evidence-Based Patient Choice in the 21st Century. *Clinics in Dermatology*, 18:11–17, 2001.

[34] C. D. Gremu, A. Terzoli, and M. Tsietsi. A User-Oriented, Bi-Directional Information Channel for e-Health Interventions in Marginalised Rural Areas. This paper was presented at the Southern African Telecommunications Networks and Applications conference in 2011 which was held at Spier Hotel in Stellembosch, South Africa, September 2013.

[35] A. Grier and C. A. Bryant. Social Marketing in Public Health. *Annual Review Public Health*, 26:319–339, 2005.

[36] S. Gumbo, H. Thinyane, M. Thinyane, A. Terzoli, and S. Hansen. Living Lab Methodology as an Approach to Innovation in ICT4D: The Siyakhula Living Lab Experience. http://www.siyakhulall.com/sites/default/files/ISTAfrica_Paper_ref_18_doc_4809_0.pdf. Online [Last accessed: 13/12/2014], 2012.

[37] D. H. Gustafson, R. Hawkins, E. Boberg, S. Pingree, R. E. Serlin, F. Graziano, and C. L. Chan. Impact of a Patient-Centered, Computer-Based Health Information/-Support System. *American Journal of Preventive Medicine*, 16(1):1–9, 1999.

[38] L. Haerens, L. Maes, J. Brug, C. Vandelanotte, and I. D. Bourdeaudhuij. A Computer-Tailored Dietary Fat Intake Intervention for Adolescents: Results of a Randomized Controlled Trial. *Annals of Behavioral Medicine*, 34:253–262, 2007.

[39] R. P. Hawkins, M. Kreuter, K. Resnicow, M. Fishbein, and A. Dijkstra. Understanding tailoring in communicating about health. *Health Education Research*, (3):23, 2008.

[40] T. K. Houston and H. E. Ehrenberger. The Potential of Consumer Health Informatics:. *Seminars in Oncology Nursing*, 17(1):41–47, February 2001.

[41] J. T. Huber and M. Snyder. Facilitating Access to Consumer Health Information. *Medical Reference Services Quarterly*, 21(2):39–46, October 2002.

[42] International Federation of Red Cross and Red Crescert Societies. Methodologies for information-gathering. https://www-secure.ifrc.org/dmis/response/humanresources/Gender_web_Version/Tools/programmes/different_methods.htm. Online [Last accessed: 25/02/2014].

[43] N. Jenkins. *A Software Testing Primer: An Introduction to Software Testing*. Creative Commons, 2008.

[44] A. A. Keller and D. R Lehmann. Designing Effective Health Communications: A Meta-Analysis. *American Marketing Association*, 27:117–130, 2008.

[45] A. Keselman, Logan R., Smith C. A., G. Leroy, and Q. Zeng-Treitler. Developing Informatics Tools and Strategies for Consumer-centered Health Communication. *Journal of the American Medical Informatics Association*, 15(4):473–483, July/August 2008.

[46] M. S. Klein-Fedyshin. Consumer Health Informatics:Integrating Patients, Providers, and Professionals Online. *Medical Reference Services Quarterly*, 3(3):35–50, October 2002.

[47] G. L. Kreps and L. Neuhauser. New directions in eHealth communication: Opportunities and challenges. *Patient Education and Counseling*, 78:329–336, 2010.

[48] M. W. Kreuter and S. M McClure. The Roles of Culture om Health Communication. *Annual Review of Public Health*, 25:439–455, 2004.

[49] M. W. Kreuter, D. L. Oswald, F. C. Bull, and E. M Clark. Are tailored health education materials always more effective than non-tailored materials? *Health Education Research*, 15:305–315, 2000.

[50] M. W. Kreuter, V. J. Strecher, and B Glassman. One Size Does Not Fit All: The Case for Tailoring Print Materials. *The Society of Behavioral Medicine*, 21:276–283, 1999.

[51] D. Lewis, G. Eysenbach, R. Kukafka, P. Z. Stavri, and H B Jimison. *Consumer Health Informatics: Informing Consumers and Improving Health Care*. Springer Science + Business Media Inc., 2005.

[52] love4life. love4life. http://www.lovelife.org.za/lovelife-programmes/love4life-healthy-sexuality-programme/. Online [Last accessed: 01/07/2014].

[53] J. Lowy. *Programming .NET Components*. OŔeilly Media, 2005.

[54] J. Mantas. Education and Consumer Health Informatics. *IMIA Yearbook of Medical Informatics*, pages 90–94, 2007.

[55] B. M. Mayosi, J. W. Lawn, A. van Niekerk, D. Bradshaw, S. S. Abdool Karim, and H. M. Coovadia. Health in South Africa: changes and challenges since 2009. http://uir.unisa.ac.za/bitstream/handle/10500/9521/HealthOnline [Last accessed: 09/08/2014], November 2012.

[56] Media Club South Africa. Healthcare in South Africa. http://www.mediaclubsouthafrica.com/component/content/article/34-democracy/developmentbg/102-healthcare. Online [Last accessed: 10/02/2014], February 2014.

[57] J. D. Meier, D. Hill, H. Homer, J. Taylor, P. Bansode, L. Wall, Rob Boucher. R. Jr., and A. Bogawat. *Microsoft Application Architecture Guide: Patterns and Practices*. Microsoft Corporation, 2nd. ed. edition, October 2009.

[58] O. Mirza. What is a Newsletter. http://www.writeawriting.com/business/what-is-a-newsletter. Online [Last accessed: 09/02/2014], March 2011.

[59] National Cancer Institute. Pink Book - Making Health Communication Programs Work. http://www.cancer.gov. Online [Last accessed: 21/02/2013].

[60] S. M. Noar, N. G. Harrington, and R. S. Aldrich. *Communication Yearbook 33*. Routledge, 2009.

[61] S. M. Noar, N. G. Harrington, S. K. Van Stee, and R. S. Aldrich. Tailored Health Communication to Change Lifestyle Behaviors. *American Journal of Lifestyle Medicine*, 5:112–122, 2011.

[62] Oak Ridge Institute for Science and education. Choosing the right communication channel. http://www.orau.gov/cdcynergy/erc/content/activeinformation/essential_principles/EP-channels_content.htm. Online [Last accessed: 22/02/2013].

[63] Oak Ridge Institute for Science and education. The 4 P's of Social Marketing. http://www.orau.gov/cdcynergy/soc2web/Content/phase03/phase03_step05_deeper_4ps.htm. Online [Last accessed: 10/02/2014].

[64] Open University. Stages of Health Communication. http://labspace.open.ac.uk/mod/oucontent/view.php?id=452840&section=83.6. Online [Last accessed: 11/02/1014].

[65] Opentracker. Pay-per-click (PPC) advertising and campaign management. http://www.opentracker.net/article/pay-click-ppc-advertising-and-campaign-management. Online [Last accessed: 20/08/2014], May 2009.

[66] R. Parrott. Emphasizing Communication in Health Communication. *Journal of Communication*, 54:751–787, 2004.

[67] PopTech Accelerator. Project Masiluleke: A Breakthrough Initiative to Combat HIV/AIDS Utilizing Mobile Technology & HIV Self-Testing in South Africa. https://poptech.org/project_m. Online [Last accessed: 21/04/2014].

[68] Praekelt Foundation. Momconnect. http://praekeltfoundation.org/momconnect.html. Online [Last accessed: 12/09/2014].

[69] Reed House Systems. TeleWeaver: An integrated enabler for rural telecenters and access nodes. http://reedhousesystems.com/?q=solutions. Online. [Last accessed: 21/04/2014].

[70] Republic of South Africa. Protection of Personal Information Act, 2013. http://www.issafrica.org/uploads/SA-POPI-Act-2013.pdf. Online [Last accessed: 30/10/2013], November 2013.

[71] A. J. Roberto, R. S. Zimmerman, K. E. Carlyle, and E. L. Abner. A Computer-based Approach to Preventing Pregnancy, STD, and HIV in Rural Adolescents. *Journal of Health Communication*, 12:53–76, 2007.

[72] R. Roinem, A. Ohinmaa, and D. Hailey. Assessing telemedicine: a systematic review of the literature. *Canadian Medical Association Journal*, 18:765–771, 2001.

[73] Siyakhula Living Lab. Project Overview. http://www.siyakhulall.org. Online [Last accessed: 10/02/2014].

[74] B. Snyder, D. Bosanac, and R. Davies. *ActiveMQ in Action*. Manning Publications Co., 2011.

[75] L. B. Snyder. Health Communication Campaigns and Their Impact on Behavior. *Journal of Nutrition Education and Behavior*, 39:S32–S40, 2007.

[76] South Africa the Good News. 41% of South Africans use the Internet. http://www.sagoodnews.co.za/global_connectedness/41_of_south_africans_use_the_internet.html. Online [Last accessed: 08/02/2013], September 2013.

[77] Soweto Connection. The Raphael Centre. http://www.sowetoconnection.org/content/raphael-centre. Online [Last accessed: 04/07/2014].

[78] Technology Solutions for Teaching and Research. Asynchronous vs Synchronous Communication. https://academictech.doit.wisc.edu/blend/facilitate/communicate. Online [Last accessed: 28/01/2014].

[79] A. Terzoli. A model for sustainable broadband in poor communities. This is a presentation which was done by Prof. Alfredo Terzoli.

[80] The Lancet. Health in South Africa: An Executive Summary for Th e Lancet Series. http://download.thelancet.com/flatcontentassets/series/sa/sa_execsum.pdf. Online [Last accessed: 09/08/2014], 2009.

[81] The Open Web Application Security Project. Data Validation. https://www.owasp.org/index.php/Data_Validation. Online [Last accessed: 5/09/2014], August 2014.

[82] Tutorialspoint. Spring Framework 3.1 Tutorial. http://www.tutorialspoint.com/spring/spring_tutorial.pdf. Online [Last accessed: 22/08/2014].

[83] UNAIDS. The Gap Report. http://www.unaids.org/en/media/unaids/contentassets/documents/unaidspublication/2014/UNAIDS_Gap_report_en.pdf. Online [Last accessed: 11/08/2011], July 2014.

[84] T. Unwin. Challenges of financing ICT in marginal contexts. http://www.commonwealthministers.com/images/uploads/documents/191-194.pdf. Online [Last accessed: 01/08/2014], September 2012.

[85] A. van Kesteren. Cross-Origin Resource Sharing. http://www.w3.org/TR/cors/. Online [11/09/2014], January 2014.

[86] R. Visser, R. Bhana, and F. Monticelli. The National Health Care Facilities Baseline Audit National Summary Report. Technical report, National Department of Health, September 2012.

[87] C. Walls. *Spring in Action*. Manning Publications Co., 2011.

[88] N. Weymann, M. Harter, and J. Dirmaier. A tailored, interactive health communication application for patients with type 2 diabetes: study protocol of a randomised controlled trial. *BMC Medical Informatics and Decision Making*, 13:1–9, 2013.

[89] W. Wheeler and J. White. *Spring in Practice*. Manning Publications Co., 2013.

[90] World Wide Web Consortium. Token Based Authentication - Implementation Demonstration. http://www.w3.org/2001/sw/Europe/events/foaf-galway/papers/fp/token_based_authentication/. Online [Last accessed: 31/07/2014].

Appendix A

Spring

It simplifies Java development by providing developers with a component model and a set of simplified and consistent APIs that insulate them from the complexity and error-prone low level code in complex applications. As discussed by Walls [87], it achieves this by employing the following key strategies:

- it is lightweight and ensures minimal invasive development with Plain Old Java Objects (POJOs);
- it ensures loose coupling through dependency injection and interface orientation;
- it uses declarative programming through aspects and common conventions; and
- it reduces low level code through aspects and templates.

A.1 Spring Functional Areas

The framework has 20 modules, which are grouped into 6 functional areas namely Data Access/Integration, Web, Aspect-Oriented Programming (AOP), Instrumentation, the Core Container, and Test (see Figure A.1). The modularity of the framework gives developers freedom to chose which parts of the framework to use in their applications without the need to include the entire framework. The Core container provides inversion of control (IoC) and Dependency injection (DI) capabilities on which all the other modules are built. It provides a facility to decouple creation, configuration, and management of beans

from application code. The AOP encapsulates cross-cutting concerns (security, logging, transaction management etc.) into aspects to retain modularity and reusability. The Data Access/Integration module provides support for Java Database Connectivity API (JDBC), object-relation mapping (ORM), Object/XML mapping (OXM), Java Message Service (JMS), and transactional support. The Web module provides common web infrastructure code for integrating Spring into web application, multipart file upload, and web-based remoting capabilities.

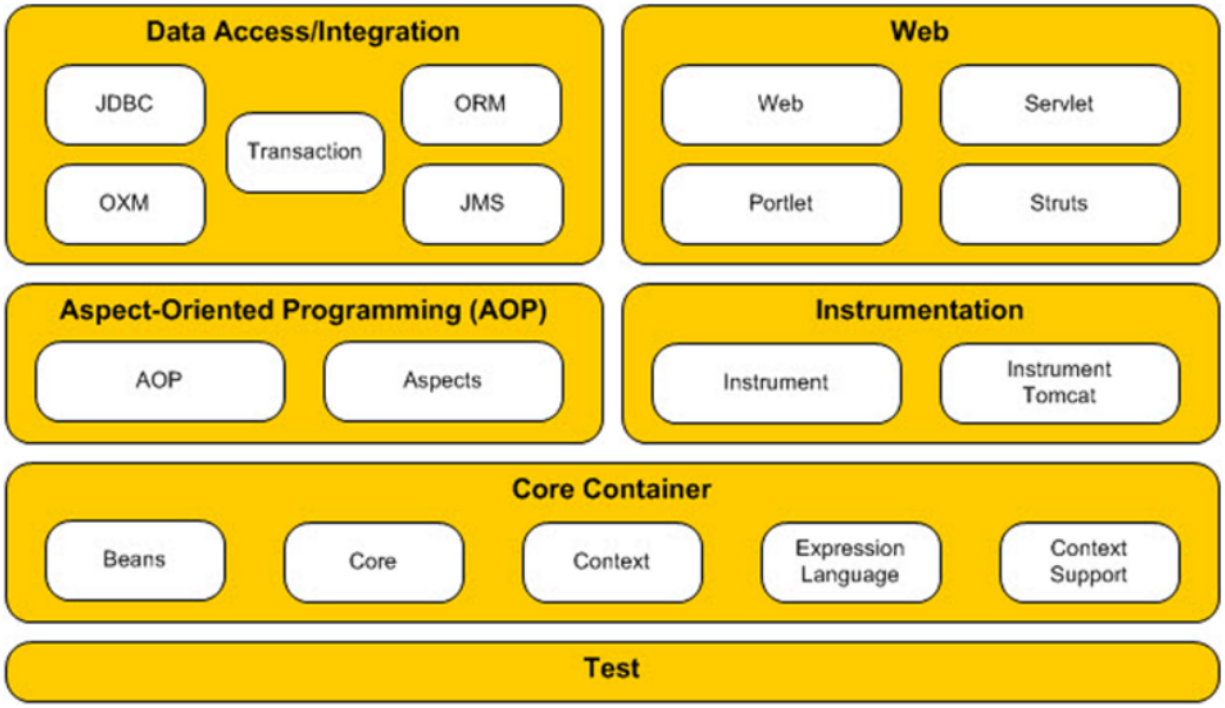

Figure A.1: Spring framework's basic functional areas [89].

A.2 Dependency Injection

In a sizeable application, two or more classes collaborate with each other to perform some business logic. This is done via construction or lookup, which creates an object. Traditionally, an object is responsible for obtaining its own references to objects it collaborates with. This leads to highly coupled and hard-to-test code [87]. Spring solves this by using DI, where objects are given references to the objects they collaborate with by the container at creation time. The act of creating associations between application objects to inject dependencies is called wiring. There are two ways that objects can be given their dependencies or wired, which are through a constructor or a setter method. Constructor-based DI is accomplished when a container invokes a constructor of a class with a number

of arguments, each representing a dependency of another class, while setter-based DI is accomplished by the container calling setter methods on the beans after invoking a no-argument constructor to instantiate the bean [82]. Wiring can be done via XML or using annotations in Java code.

According to Hawkins *et al.* [39], there are three strategies that are used to tailor messages. These are personalisation, feedback, and content matching:

Appendix B

Message Tailoring

B.1 Personalisation

This strategy is used to increase users' attention, interest, and motivation to process messages by implicitly or explicitly conveying to the recipient that the communication is specifically designed for them. It makes the messages seem more relevant and meaningful to the recipients. There are three tactics that are used to personalise messages, which are discussed in the following subsections.

B.1.1 Identification

this tactic uses tactics similar to direct mail marketing where the recipient of a message is identified by name or any other personal information like birthdays. An example is mentioning the name of the recipient in a welcome message. Identification is believed to make exposure to messages more likely and increases attention paid by the user to the message.

B.1.2 Raising Expectation

this tactic uses overt claims of customisation that indicate that a particular message was created specifically for the recipient. For example, by saying "The following health information has been created especially for you", it is believed that it will create positive expectations for the information that follow.

B.1.3 Contextualisation

this tactic frames a message in the context that is meaningful to the recipient. Some of the variables that are used to contextualise messages are family structure, residential status, ethnicity/culture, and personal interests. An example of this tactic is framing health messages differently for day scholars and borders in the contexts that make sense for either of them. The aim is to increase attention, interest, and motivation to process the message.

B.2 Feedback

In this strategy, personal information that is obtained during assessment or through other sources is presented them. In addition to increasing a user's attention, interest, and motivation to process messages, it targets psychosocial determinants of health behaviour. There are three types of feedback, which are discussed in the following subsections.

B.2.1 Descriptive Feedback

The feedback ranges from simply restating or acknowledging the recipient's information (e.g. "You said you are ready to quit smoking") to providing information based on complex processing of recipient's responses (e.g. "Based on your responses, we believe that it cannot be difficult for you to quit smoking"). This type of feedback may influence determinants of health behaviour by stimulating self-referential thinking. The recipient of the message feels being acknowledged and understood which helps to build rapport and may favourably lower resistance to persuasion.

B.2.2 Comparative Feedback

The feedback is created by comparing the recipient's attitudes, beliefs, or behaviours with those of other recipients. Alternatively, the recipient's information is gathered over time, which is then used to provide a report on the recipient's progress over time. As suggested by Hawkins *et al.* [39], an example of comparing an individual to others is "You tried to quit smoking but you went back. On average, ex-smokers try to quit at least three times before succeeding".

B.2.3 Evaluative Feedback

The feedback is given to an individual based on the interpretation, judgement, and/or reference to an individual's attitudes, beliefs, or behaviours. An example could be "You said you want to quit smoking. That could lower the risks of getting cancer".

B.3 Content Matching

This strategy attempts to address key theoretical determinants of behaviours of interest on which change is needed most or likely to produce success. These determinants are knowledge, outcome expectations, normative beliefs, efficacy, and/or skills. It works by first assessing the determinants for a given behaviour in an individual and then retrieving and presenting to him or her a set of matching content that can directly address the determinants. For example, if a person has the intention to perform a given behaviour but is not doing so, this strategy focuses on building his/her skills and knowledge to avoid, remove, or overcome the barriers.

Appendix C

Configuration Details for pom.xml

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
       XMLSchema-instance"
2    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
       v4_0_0.xsd">
3    <modelVersion>4.0.0</modelVersion>
4    <groupId>com.reedhousesystems.services.core.dashboard</groupId>
5    <artifactId>reedhousesystems-dashboard</artifactId>
6    <packaging>war</packaging>
7    <version>1.2.2</version>
8    <name>reedhousesystems-dashboard Maven Webapp</name>
9    <url>http://maven.apache.org</url>
10
11   <!-- Shared version number properties -->
12   <properties>
13     <!-- Aspect J version -->
14     <aspectj.version>1.5.4</aspectj.version>
15     <!-- cglib version -->
16     <cglib.version>2.2.2</cglib.version>
17     <!-- commons-cli version -->
18     <commons.cli.version>1.2</commons.cli.version>
19     <!-- commons-codec version -->
20     <commons.codec.version>1.8</commons.codec.version>
21     <!-- common dbcp version -->
22     <commons.dbcp.version>1.4</commons.dbcp.version>
23     <!-- commons io version -->
24     <commons.io.version>1.4</commons.io.version>
25     <!-- commons lang version -->
26     <commons.lang.version>2.6</commons.lang.version>
27     <!-- guava version -->
28     <guava.version>12.0</guava.version>
29     <!-- hibernate persistence version -->
30     <hibernate.persistence.version>1.0.1.Final</hibernate.persistence.version>
31     <!-- Hibernate version -->
32     <hibernate.version>4.0.1.Final</hibernate.version>
33     <!-- Hibernate validator version -->
34     <hibernate-validator.version>4.1.0.Final</hibernate-validator.version>
35     <!-- jackson version -->
```

```
36      <jackson.version>1.9.3</jackson.version>
37      <!-- jackson data type version -->
38      <jackson.datatype.hibernate4.version>2.2.2</jackson.datatype.hibernate4.version>
39      <!-- Java version -->
40      <java.version>1.6</java.version>
41      <!-- jaxb version -->
42      <jaxb.version>2.1.12</jaxb.version>
43      <!-- Jersey version -->
44      <jersey.version>1.12</jersey.version>
45      <!-- joda time version -->
46      <joda.time.version>2.1</joda.time.version>
47      <!-- jsr250 api version -->
48      <jsr250.api.version>1.0</jsr250.api.version>
49      <!-- jsr 311 api version -->
50      <jsr311.api.version>1.1</jsr311.api.version>
51      <!-- Junit version -->
52      <junit.version>4.8.2</junit.version>
53      <!-- logback version -->
54      <logback.version>1.0.1</logback.version>
55      <!-- MySQL version -->
56      <mysql.version>5.1.25</mysql.version>
57      <!-- RESTEasy version -->
58      <resteasy.version>2.3.2.Final</resteasy.version>
59      <!-- slf4j version -->
60      <slf4j.version>1.6.1</slf4j.version>
61      <!-- Spring version -->
62      <spring.version>3.2.8.RELEASE</spring.version>
63      <!-- Spring integration version -->
64      <spring.integration.version>2.2.1.RELEASE</spring.integration.version>
65      <!-- Spring security version -->
66      <spring.security.version>3.2.3.RELEASE</spring.security.version>
67      <!-- validation version -->
68      <javax.validation.version>1.1.0.Final</javax.validation.version>
69      <!-- velocity version -->
70      <velocity.version>1.7</velocity.version>
71    </properties>
72    <dependencies>
73      <!-- CGLIB dependency -->
74      <dependency>
75        <groupId>cglib</groupId>
76        <artifactId>cglib</artifactId>
77        <version>${cglib.version}</version>
78      </dependency>
79
80      <!-- Commons-cli dependency -->
81      <dependency>
82        <groupId>commons-cli</groupId>
83        <artifactId>commons-cli</artifactId>
84        <version>${commons.cli.version}</version>
85      </dependency>
86
87      <!-- Commons codec -->
88      <dependency>
89        <groupId>commons-codec</groupId>
90        <artifactId>commons-codec</artifactId>
91        <version>${commons.codec.version}</version>
```

```
92      </dependency>
93
94      <!-- Common DBCP dependency -->
95      <dependency>
96        <groupId>commons-dbcp</groupId>
97        <artifactId>commons-dbcp</artifactId>
98        <version>${commons.dbcp.version}</version>
99      </dependency>
100
101     <!-- Commons IO dependencies -->
102     <dependency>
103       <groupId>commons-io</groupId>
104       <artifactId>commons-io</artifactId>
105       <version>${commons.io.version}</version>
106     </dependency>
107
108     <!-- Common-lang dependency -->
109     <dependency>
110       <groupId>commons-lang</groupId>
111       <artifactId>commons-lang</artifactId>
112       <version>${commons.lang.version}</version>
113     </dependency>
114
115     <!-- CORS Filter dependency -->
116     <dependency>
117       <groupId>com.thetransactioncompany</groupId>
118       <artifactId>cors-filter</artifactId>
119       <version>2.1.2</version>
120     </dependency>
121
122     <dependency>
123       <groupId>com.thetransactioncompany</groupId>
124       <artifactId>java-property-utils</artifactId>
125       <version>1.9.1</version>
126     </dependency>
127
128     <!-- Hibernate dependencies -->
129     <dependency>
130       <groupId>org.hibernate</groupId>
131       <artifactId>hibernate-core</artifactId>
132       <version>${hibernate.version}</version>
133     </dependency>
134
135     <dependency>
136       <groupId>org.hibernate</groupId>
137       <artifactId>hibernate-entitymanager</artifactId>
138       <version>${hibernate.version}</version>
139     </dependency>
140
141     <dependency>
142       <groupId>org.hibernate</groupId>
143       <artifactId>hibernate-validator</artifactId>
144       <version>${hibernate-validator.version}</version>
145     </dependency>
146
147         <!-- Jackson dependency -->
```

```
148         <dependency>
149             <groupId>org.codehaus.jackson</groupId>
150             <artifactId>jackson-xc</artifactId>
151             <version>${jackson.version}</version>
152         </dependency>
153 
154         <dependency>
155                 <groupId>com.fasterxml.jackson.datatype</groupId>
156                 <artifactId>jackson-datatype-hibernate4</artifactId>
157                 <version>${jackson.datatype.hibernate4.version}</version>
158         </dependency>
159 
160         <!-- JAXB dependency -->
161         <dependency>
162             <groupId>com.sun.xml.bind</groupId>
163             <artifactId>jaxb-xjc</artifactId>
164             <version>${jaxb.version}</version>
165         </dependency>
166 
167         <!-- Jersey dependency -->
168         <dependency>
169             <groupId>com.sun.jersey</groupId>
170             <artifactId>jersey-core</artifactId>
171             <version>${jersey.version}</version>
172         </dependency>
173 
174         <dependency>
175             <groupId>com.sun.jersey</groupId>
176             <artifactId>jersey-server</artifactId>
177             <version>${jersey.version}</version>
178         </dependency>
179 
180         <dependency>
181             <groupId>com.sun.jersey.contribs</groupId>
182             <artifactId>jersey-spring</artifactId>
183             <version>${jersey.version}</version>
184             <exclusions>
185                         <exclusion>
186                                 <groupId>org.springframework</groupId>
187                                 <artifactId>spring</artifactId>
188                         </exclusion>
189                         <exclusion>
190                                 <groupId>org.springframework</groupId>
191                                 <artifactId>spring-core</artifactId>
192                         </exclusion>
193                         <exclusion>
194                                 <groupId>org.springframework</groupId>
195                                 <artifactId>spring-web</artifactId>
196                         </exclusion>
197                         <exclusion>
198                                 <groupId>org.springframework</groupId>
199                                 <artifactId>spring-beans</artifactId>
200                         </exclusion>
201                         <exclusion>
202                                 <groupId>org.springframework</groupId>
203                                 <artifactId>spring-context</artifactId>
```

```
204                         </exclusion>
205                 </exclusions>
206         </dependency>
207 
208         <dependency>
209             <groupId>com.sun.jersey</groupId>
210             <artifactId>jersey-json</artifactId>
211             <version>${jersey.version}</version>
212         </dependency>
213 
214         <!-- Joda time dependency -->
215         <dependency>
216             <groupId>joda-time</groupId>
217             <artifactId>joda-time</artifactId>
218             <version>${joda.time.version}</version>
219         </dependency>
220 
221         <!-- JSR 250 API -->
222         <dependency>
223                 <groupId>javax.annotation</groupId>
224                 <artifactId>jsr250-api</artifactId>
225                 <version>${jsr250.api.version}</version>
226         </dependency>
227 
228         <!-- JSR 311 dependency -->
229         <dependency>
230                 <groupId>javax.ws.rs</groupId>
231                 <artifactId>jsr311-api</artifactId>
232                 <version>${jsr311.api.version}</version>
233         </dependency>
234 
235         <!-- Logback dependency -->
236         <dependency>
237             <groupId>ch.qos.logback</groupId>
238             <artifactId>logback-classic</artifactId>
239             <version>${logback.version}</version>
240         </dependency>
241 
242         <!-- MySQL connector -->
243     <dependency>
244         <groupId>mysql</groupId>
245         <artifactId>mysql-connector-java</artifactId>
246         <version>${mysql.version}</version>
247         <scope>privided</scope>
248     </dependency>
249 
250     <!-- Spring dependencies -->
251         <dependency>
252             <groupId>org.springframework</groupId>
253             <artifactId>spring-core</artifactId>
254             <version>${spring.version}</version>
255         </dependency>
256 
257         <dependency>
258             <groupId>org.springframework</groupId>
259             <artifactId>spring-beans</artifactId>
```

```
260             <version>${spring.version}</version>
261         </dependency>
262 
263         <dependency>
264             <groupId>org.springframework</groupId>
265             <artifactId>spring-context</artifactId>
266             <version>${spring.version}</version>
267         </dependency>
268         <dependency>
269             <groupId>org.springframework</groupId>
270             <artifactId>spring-aop</artifactId>
271             <version>${spring.version}</version>
272         </dependency>
273 
274         <dependency>
275             <groupId>org.springframework</groupId>
276             <artifactId>spring-webmvc</artifactId>
277             <version>${spring.version}</version>
278         </dependency>
279 
280         <dependency>
281             <groupId>org.springframework</groupId>
282             <artifactId>spring-tx</artifactId>
283             <version>${spring.version}</version>
284         </dependency>
285 
286         <dependency>
287             <groupId>org.springframework</groupId>
288             <artifactId>spring-orm</artifactId>
289             <version>${spring.version}</version>
290         </dependency>
291 
292         <dependency>
293             <groupId>org.springframework</groupId>
294             <artifactId>spring-jdbc</artifactId>
295             <version>${spring.version}</version>
296         </dependency>
297 
298         <dependency>
299             <groupId>org.springframework</groupId>
300             <artifactId>spring-aspects</artifactId>
301             <version>${spring.version}</version>
302         </dependency>
303 
304         <!-- Spring integration dependency -->
305         <dependency>
306             <groupId>org.springframework.integration</groupId>
307             <artifactId>spring-integration-core</artifactId>
308             <version>${spring.integration.version}</version>
309         </dependency>
310 
311         <dependency>
312             <groupId>org.springframework.integration</groupId>
313             <artifactId>spring-integration-jdbc</artifactId>
314             <version>${spring.integration.version}</version>
315         </dependency>
```

```
316
317         <dependency>
318             <groupId>org.springframework.integration</groupId>
319             <artifactId>spring-integration-mail</artifactId>
320             <version>${spring.integration.version}</version>
321         </dependency>
322
323         <dependency>
324             <groupId>org.springframework.integration</groupId>
325             <artifactId>spring-integration-http</artifactId>
326             <version>${spring.integration.version}</version>
327         </dependency>
328
329         <!-- Spring security dependency -->
330         <dependency>
331             <groupId>org.springframework.security</groupId>
332             <artifactId>spring-security-core</artifactId>
333             <version>${spring.security.version}</version>
334         </dependency>
335
336         <!-- Velocity dependency -->
337         <dependency>
338             <groupId>org.apache.velocity</groupId>
339             <artifactId>velocity</artifactId>
340             <version>${velocity.version}</version>
341         </dependency>
342
343         <!-- Unit testing dependencies -->
344     <dependency>
345             <groupId>junit</groupId>
346             <artifactId>junit</artifactId>
347             <version>${junit.version}</version>
348             <scope>test</scope>
349     </dependency>
350   </dependencies>
351
352   <build>
353         <!--  Directory where source code is located -->
354         <scriptSourceDirectory>src</scriptSourceDirectory>
355         <resources>
356                 <resource>
357                         <directory>src</directory>
358                         <includes>
359                                 <include>**/*.properties</include>
360                                 <include>**/*.xml</include>
361                         </includes>
362                 </resource>
363         </resources>
364
365         <plugins>
366                 <!-- Plugin for compiling Java code -->
367                 <plugin>
368                         <groupId>org.apache.maven.plugins</groupId>
369                         <artifactId>maven-compiler-plugin</artifactId>
370                         <configuration>
371                                 <verbose>true</verbose>
```

```
372                             <source>${java.version}</source>
373                             <target>${java.version}</target>
374                             <showWarnings>true</showWarnings>
375                     </configuration>
376             </plugin>
377
378             <plugin>
379                     <groupId>org.apache.maven.plugins</groupId>
380                     <artifactId>maven-surefire-plugin</artifactId>
381                     <configuration>
382                             <includes>
383                                     <include>**/*Tests.java</include>
384                             </includes>
385                     </configuration>
386             </plugin>
387
388             <!-- Maven Plugin used to build WAR archive -->
389             <plugin>
390                     <artifactId>maven-war-plugin</artifactId>
391                     <configuration>
392                             <webappDirectory>webapp</webappDirectory>
393                             <warName>dashboard</warName>
394                     </configuration>
395             </plugin>
396
397             <!-- Hibernate3-Maven plugin -->
398             <plugin>
399                     <!-- run "mvn hibernate3:hbm2ddl" to generate a schema -->
400             <groupId>org.codehaus.mojo</groupId>
401         <artifactId>hibernate3-maven-plugin</artifactId>
402         <version>3.0</version>
403         <configuration>
404                             <hibernatetool>
405                     <annotationconfiguration configurationfile="src/main/resources/
                            hibernate/hibernate.cfg.xml" />
406                     <hbm2ddl drop="true"
407                                                     create="true"
408                                                     export="false"
409                                                     outputfilename="dashboard.sql"
410                                                     delimiter=";"
411                                                     format="true"
412                                                     console="true" />
413             </hibernatetool>
414                     </configuration>
415       </plugin>
416       </plugins>
417     <finalName>reedhousesystems-dashboard</finalName>
418   </build>
419 </project>
```

Listing C.1: Configuration details for pom.xml