# Contents

Bibliography viii

1 Introduction 1
   1.1 The course 1
   1.2 The group 1
   1.3 Constraints 1
   1.4 Definitions 2
   1.5 Chapter structure 2

2 Project description 3
   2.1 The task 3
   2.2 The Parent Application 3
   2.3 Vision of Life History 4
      2.3.1 Contents and use 4
      2.3.2 Dementia 4
   2.4 Stakeholders 4
      2.4.1 Customer 5
      2.4.2 Developer 5
      2.4.3 Supervisor 5
      2.4.4 Users 5
      2.4.5 Application outline 6

3 Prestudy 8
   3.1 The situation today 8
   3.2 Existing software 8
      3.2.1 Family Tree Maker 8
      3.2.2 Memolane 9
      3.2.3 Mémoires 9
   3.3 Development approach 9
      3.3.1 Native application 11
      3.3.2 Web based application 12
      3.3.3 Summary 12
   3.4 Voice 13
      3.4.1 Voice recognition 13
      3.4.2 Voice input 13
4 Technology used

4.1 HTML

4.2 Ruby on Rails

4.2.1 RubyGems

4.3 SQLite 3

4.4 CSS

4.5 Javascript

4.6 Git

4.7 Programming editors

4.8 Other software

5 Project management

5.1 Agile programming

5.1.1 Scrum

5.1.2 XP - Extreme Programming

5.1.3 FDD - Feature Driven Development

5.2 Choice of process model

5.2.1 How Scrum will be used

5.3 Team organisation

5.3.1 Internal team structure

5.3.2 Scrum Master

5.3.3 Product Owner

5.3.4 Document Manager

5.3.5 Customer Contact

5.3.6 Developers

6 Project requirements

6.1 Functional requirements

6.2 Non-functional requirements

6.2.1 User friendliness

6.2.2 Local storage or external storage

6.3 Use cases

6.3.1 Use Case 1

6.3.2 Use case 2

6.3.3 Use case 3

6.3.4 Use case 4

6.3.5 Use case 5

6.3.6 Use case 6

6.3.7 Use case 7

6.3.8 Use case 8

6.3.9 Use case 9

6.3.10 Use case 10

6.3.11 Use case 11

6.3.12 Use case 12

6.3.13 Use case 13

6.3.14 Use case 14

6.3.15 Use case 15
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Functional requirements</td>
<td>27</td>
</tr>
<tr>
<td>6.2</td>
<td>Use case 1</td>
<td>29</td>
</tr>
<tr>
<td>6.3</td>
<td>Use case 2</td>
<td>30</td>
</tr>
<tr>
<td>6.4</td>
<td>Use case 3</td>
<td>30</td>
</tr>
<tr>
<td>6.5</td>
<td>Use case 4</td>
<td>31</td>
</tr>
<tr>
<td>6.6</td>
<td>Use case 5</td>
<td>32</td>
</tr>
<tr>
<td>6.7</td>
<td>Use case 6</td>
<td>32</td>
</tr>
<tr>
<td>6.8</td>
<td>Use case 7</td>
<td>32</td>
</tr>
<tr>
<td>6.9</td>
<td>Use case 8</td>
<td>33</td>
</tr>
<tr>
<td>6.10</td>
<td>Use case 9</td>
<td>33</td>
</tr>
<tr>
<td>6.11</td>
<td>Use case 10</td>
<td>33</td>
</tr>
<tr>
<td>6.12</td>
<td>Use case 11</td>
<td>34</td>
</tr>
<tr>
<td>6.13</td>
<td>Use case 12</td>
<td>34</td>
</tr>
<tr>
<td>6.14</td>
<td>Use case 13</td>
<td>34</td>
</tr>
<tr>
<td>6.15</td>
<td>Use case 14</td>
<td>35</td>
</tr>
<tr>
<td>6.16</td>
<td>Use case 16</td>
<td>36</td>
</tr>
<tr>
<td>9.2</td>
<td>System testing (part 2)</td>
<td>77</td>
</tr>
<tr>
<td>9.1</td>
<td>System testing (part 1)</td>
<td>82</td>
</tr>
<tr>
<td>9.3</td>
<td>Test scenarios</td>
<td>83</td>
</tr>
<tr>
<td>10.1</td>
<td>Functional requirements met.</td>
<td>85</td>
</tr>
<tr>
<td>10.2</td>
<td>Functional requirements not met.</td>
<td>86</td>
</tr>
</tbody>
</table>
List of Figures

3.1 Screenshot of Family Tree Maker .......................... 9
3.2 Screenshot of Memolane ..................................... 10
3.3 Screenshot of Mémoires .................................... 10

4.1 MVC diagram .................................................. 17

5.1 The Scrum process ........................................... 23
5.2 Planning and feedback loops in Extreme Programming ... 24

6.1 Use case diagram describing Use Case 1-3 .................... 29
6.2 Use case diagram describing Use Case 4-14 .................. 31
6.3 Use case diagram describing Use Case 15 .................... 35

7.1 Work Breakdown Structure for the project .................... 39
7.2 Gantt diagram .................................................. 40
7.3 Product and release backlog .................................. 41
7.4 Backlog for sprint 1 .......................................... 42
7.5 Burn-down chart for sprint 1 ................................ 42
7.6 Backlog for sprint two ....................................... 43
7.7 Burn-down chart for sprint two .............................. 43
7.8 Backlog for sprint three ..................................... 44
7.9 Burn-down chart for sprint three ............................ 44
7.10 Backlog for sprint four ..................................... 45
7.11 Burn-down chart for sprint four ............................ 45
7.12 Backlog for sprint five ..................................... 46
7.13 Burn-down chart for sprint five ............................ 46
7.14 Backlog for sprint six ...................................... 48
7.15 Burn-down chart for sprint six ............................. 48
7.16 Backlog for sprint seven ................................... 49
7.17 Burn-down chart for sprint seven .......................... 49
7.18 Backlog for sprint eight .................................... 50
7.19 Burn-down chart for sprint eight .......................... 50

8.1 Life History Site Map ....................................... 52
8.2 Complete class diagram .................................... 54
8.3 Model ......................................................... 55
8.4 View .......................................................... 56
8.5 Controller .................................................... 57
8.6 ER-diagram .................................................... 60
Chapter 1

Introduction

This report will be detailing the design and implementation of an application from a set of requirements set in cooperation with the Registered Nursing Home Association (RNHA). The application has been named Life History. Life History is developed by students of the course IT2901 at the Norwegian University of Technology (NTNU).

1.1 The course

IT2901 Project work in informatics II is a course mandatory for students following the Bachelor’s degree programme of Informatics at NTNU. The objective of the course is to give students hands-on experience with team software development for a real customer. IT2901 is a 15 credit course, which estimates an average work load of 20 hours per week per person. There are some deadlines throughout the semester that the students need to meet with regards to the process of writing this report (see section 7.1).

1.2 The group

The team consists of six students registered for the IT2901 course, in the spring semester of 2011. All of the members are approximately equal in experience. The students are taking several other courses in addition to IT2901, and work will therefore mostly be limited to the estimated 20 hours a week.

1.3 Constraints

The amount of time available for the project is approximately 20 hours per person per week which will be balanced according to the workload of other courses the participants are taking. The work is also limited by the available tools and software, i.e what the students or the institute own. There are no financial resources available for the development. The team is in possession of only one Apple iPad for testing purposes.
1.4 Definitions

- The application will be referred to as Life History.
- The customer will be referred to as RNHA or Customer.
- Objects of different content forms such as photographs, video clips and sound bytes are referred to as media or medium (singular).

1.5 Chapter structure

This report will describe the process, purpose as well as the problems and solutions the group has faced during the building of Life History.

Chapter 1 - Introduction is a general overview of the basic information concerning the course and assignment.

Chapter 2 - Project Description describes in detail what the assignment is, how the end product should work, how it will be used and who will be using it.

Chapter 3 - Prestudy will discuss alternatives that were researched and how the group approached the assignment.

Chapter 4 - Technology Used will explain what technology was used and why we chose this technology.

Chapter 5 - Project Management covers the methods used to organise the team and what kind of development model was chosen.

Chapter 6 - Project Requirements contains an overview of the requirements the finished project should meet.

Chapter 7 - Time plan chronicles the milestones, goals and sprints the group has gone through during the development process.

Chapter 8 - Design describes how the product is structured, how data flows and the underlying architecture. It also gives an overview of the user interface. It describes how the program user will experience the program and how it is designed.

Chapter 9 - Testing contains the testing plan for the system, as well as describing the feedback we received during the tests.

Chapter 10 - Evaluation is an evaluation of the process and end product of the assignment by the group.
Chapter 2

Project description

This chapter will give an overview of the work to be done as part of this project. It will describe the initial information received by the customer, and the team’s overall vision for the *Life History* application, as well as a brief outline of the different parts that constitutes the application.

2.1 The task

*RNHA* is organising an ongoing project where new technology is being visualised, that will be used to aid carers taking care of people suffering from different stages of dementia in a nursing home. *RNHA* has thus been planning a suite of applications (henceforth referred to as the *parent application*) to be used on rugged Apple iPad-like devices. *Life History* is set to accomplish specific parts of this suite and will be a memory bank containing stories, pictures, important family and friends, among other things important to the resident. This information is then meant to help the carers get to know the residents in their care better, as well as help the residents remember things about their earlier life. The customer has provided a storyboard document detailing the envisioned use of the parent application. See appendix D for the storyboard.

2.2 The Parent Application

The storyboard document is a set of documents given to us by *RNHA* that describes the parent application and the way it is thought to be used. These documents includes specific use cases with user stories that describe how a carer comes to work, turns on the device and how he/she uses the application throughout the day. The parent application consists of several smaller applications, all with the primary goal of assisting the carers in taking care of people who are suffering from dementia. The purpose of these applications ranges from documenting information such as medical history, to registering more dynamic aspects such as the current mood and health of the resident.
2.3 Vision of Life History

2.3.1 Contents and use

Life History is to be deployed in a nursing home taking care of people suffering from different stages of dementia. A simplified way of looking at Life History is that it is designed to be a content management system\(^1\). In the Life History CMS, the information to be stored is the memories and family of a resident living in a nursing home.

As for the contents of Life History, Life History will display memories of a resident, presented in a simple fashion. These memories will primarily include photos and texts, but can be extended to also include video and audio. It is of great importance that both the registering of memories should be intuitive, as well as the navigation in the application itself.

Other regular uses of the application is thought to be to give assistance to a carer covering for another carer, in the instance of a carer being on a sick leave. It may also prove useful for training new carers. The carers should be able to look up the profile of a resident to learn about the resident’s personal information, such as interests, relatives, likes and dislikes. Life History will help the carers remember the residents and details about their personality.

The overall vision for Life History is that it should be simple, with an intuitive user interface that does not require much user training, is secure in terms of privacy (as it may involve sensitive information about residents) and that it will constitute a solid base for further development into a usable and useful application.

2.3.2 Dementia

Dementia is a general term describing a group of disorders in which memory and thought processes (cognition) become impaired for a period of at least 6 months. Unlike mental retardation, dementia involves a change in thinking abilities relative to a set baseline. People suffering from dementia have a hard time learning new concepts, and therefore have a special need for precise and unambiguous information on which specific actions will produce a specific outcome. If Life History is to be used by such an audience, it is necessary for the application to include this information on every step needing the resident’s attention. See appendix A for more information on dementia, and section 8.4.6 for the discussion on how residents are thought to interact with Life History.

2.4 Stakeholders

This section will list the stakeholders for Life History and briefly explain their role in the design and how their intended use of the Life History application has influenced the decisions taken.

\(^1\)A content management system, or CMS, is system devised to store and retrieve information. A well designed CMS is both easy to use for a reader (here: the carers or in some cases the residents) and for the writer (here: the carers, system administrators or the resident’s family). [9]
2.4.1 Customer

The customer of this project is RNHA and is represented by Kevin Pudney. Mr. Pudney is the group’s contact person, and is the one to be overseeing the development of the application. The RNHA is a Trade Association for nursing home owners, and RNHA is according to themselves campaigning strongly for high standards in nursing home care.

2.4.2 Developer

The IT2901 project group are the main developers of the Life History application prototype. The group is responsible for planning, designing and implementing the application, as well as providing thorough documentation and testing the product.

2.4.3 Supervisor

Supervising the development of Life History is a designated supervisor who will be providing the development team with feedback and advice on a regular basis throughout the entire project. The development team and the supervisor meet regularly to discuss the progress. The supervisor of the project is Mamdouh Eljueidi.

2.4.4 Users

Life History is designed mainly with two different types of users in mind: carers of a nursing home taking care of people suffering from dementia; and the residents on the nursing home themselves. In addition to these users, Life History is to be used by two other groups of users: administrators; and family members.

Carers

The staff working at the nursing home will throughout the report be referred to as carers. The carers will be the primary users of Life History and it is these users that Life History is mainly designed and tailored for. A carer will log onto Life History from his/her mobile tablet at the beginning of each shift, and use it throughout the work day as an aid in the care taking of the residents. According to Mr. Pudney this group of people often has limited experience in using digital equipment, and the Life History should be carefully designed to be as user friendly as possible. This is why there has been taken great care in choosing a fitting metaphor for the design of the application. See chapter 8.4.3 for the discussion regarding metaphors. Additionally, a carer can also be an administrator, but need not be.

Residents

The people in care of the nursing home will be referred to as residents, and will be the secondary users of Life History. In cooperation with Mr. Pudney the group is designing a Resident mode (see section 8.4.6), which is a slightly different graphical user interface where a minimal set of Life History’s functions will be available to the user. This is to make Resident mode as easy to use as
possible, as well as preventing the user from making unintended, accidental changes to the system. Seeing as it is the carers who will have access to edit, add and remove information from the application database, it is important to ensure that edits are not done unintentionally by anyone unqualified and/or unprivileged. A locked down Resident mode is visualised as containing nothing more than the ability to browse through the stored histories and corresponding media.

**Administrator**

Some administrative tasks need to be accommodated by the implemented system. Administrators will be able to add new carers and residents to the application database, as well as edit and delete them. The administrator also has to be able to give and revoke user privileges\(^2\). An administrator can be a carer, but need not be.

**The resident’s family**

The task of entering new data for a resident is thought to be rather extensive, especially for carers who probably are overworked as it is. It is thought that this job can be eased by letting the family of the resident being able to enter this data.

The only way a family member’s user role in the system differs from the one of a carer is that the family members are only able to enter data for the single resident that they are in relation to (a carer on the other hand can edit data on all residents).

### 2.4.5 Application outline

*Life History* consists of three main parts, the *Resident profile*, the *Memory* and *Family and Friends*. In addition to these there are some basic overview screens that connect these parts of the application. Below is a description of what *Life History* is, and how it is envisioned to be used.

**Resident profile**

The resident profile will be the main page for viewing information about a given resident. This page will contain a picture of a given resident, a few snippets of important personal information, as well as buttons for navigating to the *Memory* overview and the *Friends and Family* overview.

**Memory**

The *Memory* part of *Life History* will contain visual representations of memories experienced by a resident. These memories will consist of:

- a title describing the memory.
- the time (date, year, decade or era) the memory was experienced.

\(^2\)An example of a user privilege is the administrator role itself. An administrator has to be able to make another person an administrator.
• a textual story or description of the memory.
• one or more piece of media, each with its own description.
• one or more categories for grouping similar memories.

The memory title will be mandatory. The Memory part of Life History will be displayed like a digital photo album where one memory is displayed at a time with the ability to go either backward or forward between memories. A more detailed description of the visual representation of Memory will be provided in a later section.

Family and friends
The Family and Friends part of the application will display people important to the resident, both family and friends. An entry for Friends and Family will consist of:

• the name of the person.
• a photograph of the person.
• a textual story or description of the person.
• a category describing the relation to the resident, e.g. wife, parent, daughter or friend.

The only thing mandatory about the Family and friends entry is the name of the person.

Log in screen
The log in screen is the first part of the application one will encounter. In order to access the rest of the application one would have to enter a username and password in the relevant text fields found on the log in screen.
Chapter 3

Prestudy

This section of the report will be listing software and solutions that in some way accomplish some of the assignments that Life History will, and have a look at the paths that could have been taken in designing Life History. It will also address the technologies that were considered for the realisation of Life History, though it will be explained in more detail in the next chapter.

3.1 The situation today

Today, the goals of the application is fulfilled by making and maintaining an analog photo album which the residents and the carers fill with information they find useful, such as images, journals, comments, lists of favourite things, likes and dislikes. This works well in general, but has some clear disadvantages: as a single physical object, the life history book is easily misplaced or physically damaged. It also limits the availability of the information, as only one person can be using the album at a time. This makes it hard for carers to take it somewhere else to study and learn things about the residents. A digital version of this information will be backed on a remote location, and several users will be able to access the information at once.

3.2 Existing software

There are several applications already on the market that perform some or many of the tasks that Life History will perform. Below some of the alternative solutions are listed, along with a discussion of the advantages and disadvantages of these solutions.

3.2.1 Family Tree Maker

Family Tree Maker claims to be the number one bestselling genealogy software on the market today. It allows the user to create charts, as well as make reports and add media such as photos. This application is a good inspiration on how a family tree can look like, but it does not support any additional memories or information regarding the memories to be stored. See figure 3.1 for a screen shot of Family Tree Maker.
3.2.2 Memolane

Memolane is a web based application that show you your memories on a linear time line. The memories used by Memolane is the media you upload to various file and status sharing services on the Internet, such as Flickr and Facebook for photos or Twitter and Facebook for status messages. See figure 3.2 for a screenshot of Memolane. The scrollable timeline representation of the entries are a great way of representing lots of data, but might be too complex for Life History.

3.2.3 Mémoires

Mémoires[3] is a program for the Apple Mac which can be used for creating a personal journal. It lets the user add text and pictures to a specific date, to be able to look it up and remember it later. See figure 3.3 for a screen shot of Mémoires. Mémoires, while being very close to what is required from the project, is very well suited for writing an ongoing journal, which is outside the scope of Life History.

3.3 Development approach

This section of the report will detail the technologies and environments that were considered for the realisation of the Life History application.

The choice on how to develop Life History was a process that took a lot of discussion and consideration, and it was discovered early in the project phase that the main choice was between building a native application or a web based application. The choice to be made for the native application approach was
Figure 3.2: Screenshot of Memolane.

Figure 3.3: Screenshot of Mémories.
limited to decide between developing it for either the Apple iOS operating system or the Google Android operating system, or to both through middleware solutions. For a web based application, it was found that a dedicated framework was needed to effectively store and display the application, and the choice was limited to choose between either Django or Ruby on Rails.

3.3.1 Native application

For making a native application to run on the tablet device, the decision to be made was between developing Life History for either the Apple iPad or for Google Android fitted devices.

The main advantage with building a native application lies in the fidelity the application gets. A native application has the ability to communicate directly with the system hardware, unlike web based applications that has to communicate with the hardware through a web browser. By doing that, the web based applications gets more limited access than their native counterparts.

Multiple phone application frameworks (MPAF) such as Appcelerator Titanium promises to make it possible to develop for both the iOS and Android operating system through the use of common code which in turn is translated into native code for compilation.

Apple iOS

iOS is the operating system running on the Apple iPhone, iPod Touch and iPad, as well as the Apple TV. As mentioned in the storyboard that was received from the client (see D), Life History is thought to be used on an iPad or iPad-like device. Applications developed for the Apple iPad is written in the programming language Objective-C.

Originally designed to operate touch screen smart phones, the iOS operating system has been tailored to fit a larger tablet style device. According to Apple, the metaphor used to determine the size and form factor of the device is a digital note pad, which fits the application and is one of the main reasons the iPad’s dimensions was considered as being the ones to develop for.

To develop for the iPad (as well as the iPhone and Mac), one would typically use XCode\(^1\) as the IDE\(^1\) to write and compile the applications. XCode is the only IDE to be able and allowed to compile Objective-C applications in order to run on iOS devices, and is only available for the Apple Mac.

Google Android

Google’s Android operating system is a freely available operating system for mobile telephones and similar devices, including tablets. Android is available for free to use for developers and device manufacturers, which has paved the way for myriads of devices running Android, and Android fitted devices now exist in a multitude of sizes, including tablet sized devices. Applications written for Android devices are typically written in Java, which all of the group members have good experience in.

\(^1\)Integrated development environment, a tool used to write, compile, build and debug software. [21]
Google provides packages for the Eclipse IDE that simplifies the process of installing the components necessary for developing for the operating system [10]. The Eclipse IDE is available to every operating system used by the group members.

The availability of Android based tablet devices are limited to the Samsung Galaxy Tab at the planning stages of this project, a device which is considerably smaller both in size and in screen resolution than the Apple iPad. As it is considered crucial to test and develop for a device that is as close to the requirements received from the customer\(^2\), this is considered a big disadvantage.

**Multiple phone application frameworks (MPAFs)**

The need for MPAFs has risen due to the popularity and differences between the popular Android and iOS operating systems, and the wish (and need) to quickly develop software for all platforms with limited resources. Instead of tailoring the software to one specific operating system, one can instead write the code once and deploy it anywhere.

The options considered for MPAFs had to be limited due to the group’s lack of knowledge of the quality of such software, and to limit the scope of the project.

### 3.3.2 Web based application

Seeing that tablet devices are equipped with powerful web browsers, it is very much possible to deploy *Life History* as an interactive web page, often called a web application. The application would then be run in the web browser. A web application is normally developed using technologies such as HTML, CSS and with a framework such as Ruby on Rails on the server.

The biggest advantage in building *Life History* as a web based application is that it will be deployable to any device fitted with a web browser. *Life History* would thus be viewable on modern mobile phones and laptops. Both the Apple iPad and Google Android devices have excellent web browsers capable of displaying high fidelity web applications with ease.

There is a minor disadvantage with running *Life History* in the browser: the address bar is always visible and takes up considerable vertical space. To remove it, a simple, however costly solution has been proposed: making a simple, bare bones iOS native application, consisting by nothing more than a browser window, locked to a single page. In this case, the page would be the *Life History* application. For this project, this has been deemed unsuitable, as the developer licence fee\(^3\) is too great for a project without a budget.

### 3.3.3 Summary

To conclude this section we will be listing our chosen technologies for the development of the application. The code base will be written for a tablet device as a web application. This means that we will be deploying *Life History* as a series of HTML files, dynamically generated by a centralised server. To generate these

---

\(^2\)The customer specified an iPad or iPad-like device, which has been interpreted as the screen size and form factor.

\(^3\)As of the writing of this document, the developer fee is $99 per year
HTML-files dynamically we are using the *Ruby on Rails* framework. The code we produce will be version controlled with *Git*.

For a thorough description of technologies used to realise *Life History*, see section 4.

### 3.4 Voice

The ability to use voice to control the application has been a feature requested from the customer. Two approaches that were considered for *Life History* for achieving this were *voice recognition* and *voice input*.

#### 3.4.1 Voice recognition

The *voice recognition* approach makes it possible for the users to read data into *Life History*, which in turn gets translated into text. Examples of use is that the user reads the name of the relation that is to be stored in the application instead of typing it.

During the project Google updated its Webkit based web browser Chrome to version 11 with a new HTML5 specified feature: voice input on text fields [8]. This works by pressing a button on the web page, and the web browser in turn lets the user talk into the microphone. This recording is then sent to Google’s servers for processing.

There has been some tests with such fields in *Life History*. It was during these tests that the decision that such fields, for the time being had to be omitted. This due to several reasons: the voice recognition did not work as well as would be required; it is still an experimental feature not yet implemented in (Mobile) Safari; and RoR did not natively work with such input fields.

#### 3.4.2 Voice input

The *voice input* gives the possibility for the users to read data into the device and upload voice clips to *Life History*. Compared to the voice recognition approach discussed above the voice data is stored and played as audio clips rather than text. In the case of *Life History*, these clips would occupy the place of a media added to a Memory.

HTML 5 promises to provide new and improved data input fields, where one can stream audio (and video) directly to the web page [38]. Traditionally when adding files to web pages, one choose a file that is stored on ones desktop and uploads it. The key here is that this file has to be created in advance. The new options for file input promises to allow the web browser to get this input

---

4The distinction between these two are the group’s for the context of this report, and is not an universally agreed upon distinction.

5The procedure how to implement voice recognition is up to the browser developers, and it is fully possible to do the calculations on the device itself. Google’s procedure however makes it possible to have voice recognition on small low-cost devices with weaker processors, as the hard computation takes place on Google’s server farm instead of the comparatively measly hardware on ones (handheld) device. Barring a slow connection, the computation is likely to happen far quicker.

6No thorough testing was performed, but it seemed that it performed even worse on female voices.

7The test implementation was a makeshift hack of the existing work.
directly from the computer’s camera and/or microphone, which in the case of Life History would allow the users to be able to add stories to Memories and relations as sound clips with greater ease. Testing shows that Life History is able to allow such data input as soon as the specifications are finalised and support is present with very little additional work.

In the end neither of the solutions were implemented, mainly due to the immaturity of the technology itself, and compatibility issues with the supporting technologies used to build Life History. There are however a great possibility to later extend the functionality of Life History to later include the voice technologies mentioned above.
Chapter 4

Technology used

This chapter details the technologies that were chosen to realise *Life History*, based on the discussions mentioned in the prestudy in section 3. To build *Life History*, HTML has been chosen as a platform due to its flexibility and portability. *Life History* is not set to include features that is considered impossible to implement with the limitations that follows this choice.

Some technologies are added more naturally due to the nature of the platform to be developed for, such as CSS for styling the generated HTML documents (such as fonts and colours), and JavaScript to make the generated HTML documents more interactive (such as buttons and menus). Several other technologies, such as the specific gems (see section 4.2.1 below) imported JavaScript libraries has been added as needed as the development proceeded.

4.1 HTML

Hypertext Markup Language (HTML)[41] is a traditional hypertext document with similarities to plain text files\(^1\), where the text is encoded character by character using a system such as ASCII or Unicode. What sets them apart is that a hypertext document also contains special symbols, called tags, which describe how the document should behave and be displayed on a computer screen, the multimedia resources that belong to the document and which elements in the document that have reference to other documents. The HTML language started its development in 1989[2].

All known tablet devices, being based on iOS or Android have access to the Internet and are equipped with powerful web browsers. With the advent of HTML5, powerful web applications are more and more capable of achieving the same as native counterparts. By developing *Life History* with open technologies like HTML5 and CSS3 we have full portability between devices, and we are easily able to port the application to other classes of devices such as laptops and smart phones\(^2\). By using HTML5 we will however be sacrificing some fidelity compared to native applications, as it is not as easy to communicate with the device’s special hardware such as GPS and motion sensors. Considering that

---

\(^1\)Text files without formatting. Usually saved with the file extension txt.
\(^2\)http://en.wikipedia.org/wiki/HTML5
Life History will be a tailored content management system, the need for such hardware is limited if present at all.

HTML documents are traditionally written and stored on a server and delivered to a client on request. Web sites written with more flexible frameworks will dynamically generate these HTML files based upon data stored in a database, and the input data from the user\(^3\). The main advantage with this procedure is that such sites are more flexible and dynamically updatable. The main disadvantage is the increased complexity when designing and writing the page. When considering what Life History is set to accomplish, and the time allotted for the project, the choice of using a framework fell naturally. Ruby on Rails were chosen as the preferred framework to build Life History.

### 4.2 Ruby on Rails

Ruby on Rails[33] (RoR), is an open source web application framework, written in the programming language Ruby. The main goal of the RoR framework is to make it easier and faster for web developers to develop database driven websites, including making the generation of code to introduce, modify and delete information in a database automated. RoR was first released in July 2004 as an open source project, but the author did not share the commit rights to the project until 2005.

Ruby on Rails includes tools that make common development tasks easier, for example, by using scaffold that can automatically construct some of the models and views needed for a site to work properly. Ruby on Rails includes WEBrick, which is a simple Ruby server that is distributed with Ruby, and Rake, a system that is distributed as a gem. By using these different tools under active construction of a site, you will have everything you need for a basic development environment.

Ruby on Rails was chosen to realise Life History due to its promise of providing a quick and easy way to make prototypes, because of its inclusion of the MVC architecture. It is also a popular framework that the group feels could be a valuable tool to learn to use.

The Norwegian browser development company Opera has during the planning stages of Life History visited NTNU and evangelised HTML development. This sold the group on HTML and the web as a platform on early onset of the development. The Norwegian software company Bekk\(^4\) has had lectures in Ruby on Rails development that the group has attended, which was thought to be a resource to help the learning process.

**REST**

Representational State Transfer[27] (REST) defines a set of architectural principles by which you can design web services that focus on a systems resources distributed as states. At any given time, a client can either be in transition between application states, or at rest. The client can preform tasks within the

\(^3\)Examples of such data can be the address typed in the address bar of the browser or input data in a text field.

\(^4\)http://www.bekk.no/
limitation of the state it is in, and does not need to use the network, or contact the server side until it is ready to change state. Ruby on Rails uses REST.

MVC

Model-view-controller (MVC) is a software architecture where one separates the application into three parts: the model; the view; and the controller. The idea is to isolate the logic, the user interface and the database from each other and thus make it easier to work on the different parts separately, and makes the system architecture easier to navigate and gives a standardised method for sorting the code. It is considered a general solution to the problem of users controlling a large and complex data set. Figure 4.1 illustrates the MVC architecture.

The model manages data, responds to requests, and handles the state of the system. It is through the models the application stores and extract stored information, which is then handled in the controllers. In Life History, the model constitutes the database where all the data, such as data about the residents and the memories of these residents, are stored. These models also provides restrictions on which data is permitted to be stored.

The view shows the data contained in the model to the user and receives and sends user actions such as button clicks to the controller. In Life History, the view is the HTML documents.

The controller receives input from the user through the view, and instructs the model to execute. It provides and structures the received model data to the view. In Life History

Ruby on Rails uses the MVC architecture extensively, which is one of the main reasons RoR is being used in this project.

4.2.1 RubyGems

RubyGems is the packaging system employed by Ruby on Rails where one can use so-called Gems, packaged Ruby applications, to extend the functionality of an application. Below are the gems used to realise Life History.
**Paperclip** [36] Handles attachments of files to entries, such as media added to a memory.

**RailRoad** [29] Automatically generates diagrams, such as the ER-diagrams and class diagrams in section 8 reverse engineered from the code written. An extension called Railroadly was also used. This extension makes it possible to use Railroad functions in Rails version 3.0, which Railroad otherwise does not permit.

**Crummy** [18] Generates a breadcrumb path, as seen on the top menu bar (see section 8.4.6).

**SQLite3-ruby** Enables the use of SQLite as a database engine. SQLite is bundled with Ruby on Rails. See below for a short description of SQLite.

**jQuery-rails** [1] Simplifies the use of jQuery in Ruby on Rails. See section 4.5 below for a short description of jQuery.

### 4.3 SQLite 3

SQLite 3 is a "self-contained, serverless, zero-configuration, transactional SQL database engine"[31], and is according to themselves the most widely deployed SQL database engine in the world[32]. Compared to other database systems, SQLite does not employ a separate server process for reading and writing data, but instead reads and writes directly to ordinary disk files[31].

**Ruby on Rails** used SQLite by default, but is interchangeable with other systems such as MySQL and Oracle with little effort5, due to the fact that **Ruby on Rails** do not require the programmer to write engine specific syntax to read and write data, but instead uses it’s own syntax and translates these to database queries.

### 4.4 CSS

CSS[40] was a big breakthrough in web design when it was released in 1996, as it lets the author take control over the style and layout over several webpages in only one operation.

To make web development easier, the World Wide Web Consortium (W3C) created Cascading Style Sheets, mostly known as CSS. CSS is a style sheet language used to define how HTML elements on a Web page should look in a browser. These style sheets can be embedded in each of the HTML documents, but are usually stored as external stylesheet files. This lets the file with styles be used by several separate HTML documents, which makes development faster (one only has to write the style of a button only once), easier (in the case of a change, one need not change this on each page, or worse: every button) and more manageable (one developer can write the HTML files, while another works with the stylesheets).

All major browsers support CSS today. [12].

5Not much more than installing an alternative SQL engine on the server and replacing the SQLite gem with its appropriate alternative.
4.5 Javascript

JavaScript[42] is a programming language used to make web pages more interactive. These scripts\(^6\) runs in the user’s browser. JavaScript was originally developed by Netscape under the name LiveScript, for Netscape Navigator version 2 in 1995 by Brendan Eich, and has since 1996 appeared in all major browsers. Many mistake JavaScript for Java, but they do not have much in common, except for the syntax which is inherited from the programming language C. Javascript can be used to do several things, such as storing and retrieving information from the computer used by a visitor on a webpage, read and write HTML element and validate data.

JavaScript is used to handle some of the more dynamic parts of Life History, such as loading menus without refreshing the entire page. Some of the JavaScript used in Life History is written in RJS, which is Ruby on Rails’ abstraction of JavaScript. The developers write Rails code which in turn is translated into JavaScript by the server on page request.

In addition to scripts written Life History uses other JavaScript libraries, listed below.

**jQuery** jQuery[24] is a JavaScript library that makes it easier to create interactive Web pages. It contains several completed components that can in a simple way be used, without being able to program JavaScript. It consists of a simplified basis for JavaScript programming - JQuery API. It provides the capability for web page developers to create plugins on top of the JavaScript library. jQuery was developed by John Resig and was first released in 2006 and is now used by more than 41% of the 10,000 most visited websites. It is also the most popular JavaScript library used to this day.

jQuery is free open source software, and its syntax is designed to make it easier to navigate a document, create animations, handle events and to develop AJAX\(^7\) applications.

As of the writing of this document, Ruby on Rails delivered with Prototype[25], an alternative JavaScript library that accomplishes much the same as jQuery. Prototype will however be replaced by the more popular jQuery with the delivery of the next major version, Ruby on Rails version 3.1[26].

**jQuery TokenInput** jQuery TokenInput is an extension to jQuery which provides an easy to understand interface to add and remove categories to a Memory. A many-to-many relationship as it exists in the categorisation of memories (see 8.2)\(^8\) can be hard to represent in an intuitive fashion. This is handled by jQuery TokenInput, by that if a category already exists (and is used by another memory), the interface will immediately start displaying the matching items as the user enters characters. If an item does

---

\(^6\)A popular distinction between scripts and code (such as Java or C) is that scripts are not compiled while code is. Scripts are conversely interpreted impromptu by the client.

\(^7\)AJAX, or Asynchronous JavaScript and XML, is a group of technologies that enables a loaded web page to send and receive requests asynchronously in the background.[35]

\(^8\)A memory can be categorised with several categories. Each of the categories can be assigned to several different memories.
not exist, the system will add it to the database. An alternative would be for the user to enter the values as comma separated strings. What jQuery TokenInput achieves is to make this process easier for the user to understand how to operate the fields, as well as to know what categories already exists to avoid duplicates.

The latest version of jQuery TokenInput as of the writing of this report does not yet support adding new items, which initially limited the application to use a pre-defined list of categories for the memories. During the development of Life History there were however developed a forked version of jQuery TokenInput that supported this, which has been used instead. The forked version used in Life History was developed by Vinicius Depizzol.[6]

**iScroll 4** iScroll 4[30] is an open source JavaScript library for Mobile Webkit that provides a native way to scroll through content inside a fixed width/height element. What iScroll 4 does for Life History is for instance to make it possible for the user to scroll through the textual description of a Memory, while the other elements that constitutes the Memory (such as the headline and the media) does not move.

iScroll 4 is beta software, and has proven to be a bit of work to make it work smoothly. Being optimised for mobile Webkit-based browsers, there are occasions in Life History where hacks[10] has had to be devised to fix it for desktop browsers. It has however been thoroughly tested, and is deemed functional.

### 4.6 Git

GIT[4] is a distributed revision control system. A revision control system keeps track of software revisions and allows several developers to work on the same project concurrently without being connected to the same network. Being distributed means that it can operate without a centralised repository, but keeps synchronised by communicating with all of the participators. Git is written in C, Bourne Shell and Perl. It’s primarily written for Linux, and is extremely fast on UNIX based system, including Apple OS X. Modern versions can however run without problems on other operating systems, such as Microsoft Windows.

Git is employed to make it possible for all the members of the group to work on Life History concurrently. Git has been chosen over its alternatives (such as Subversion and Mercurial) because of the increasing adoption of it, and this project was considered a great opportunity to learn it.

### 4.7 Programming editors

**TextMate** Textmate[20] is a text editor for Mac OS X created by Allan Odgaard, first released in 2004. It’s very popular among programmers using Mac OS X. TextMate allows the users to create their own syntax highlighting.

---

[9] In software engineering, a project fork happens when developers take a legal copy of source code from one software package and start independent development on it, creating a distinct piece of software.[44]

[10] Hack here means that the code use undocumented features to make it behave properly.
**NetBeans** NetBeans[22] is a popular IDE and can be used to create diverse end-to-end computing solutions for multiple execution environments. NetBeans is written in Java and because of this works across all popular operating systems, including Microsoft Windows and Mac OS X. NetBeans is pre-fitted with plugins and tools ready for Ruby on Rails development\(^\text{11}\).

**Notepad++** Notepad++[13] is a simple Notepad replacement for Microsoft Windows. It features syntax highlighting and line numbering.

### 4.8 Other software

**Dropbox** is, put simply, a document folder shared between every member of the group. Every document that is placed in this folder will instantaneously be transferred and mirrored on the rest of the group member’s computers. Dropbox has also got a rudimentary version control system.[7]

**Google Docs** is among other things an online word processor with the capability for several people to edit the same document at the same time and seeing the updates in real time. The group is using Google Docs for for documents where this is needed.[11]

**Skype** is an internet telephony service, and an instant message client. The group maintains a private chat room for communication and discussing the project progress.[17]

**\LaTeX** is a powerful document preparation, and is according to themselves the de facto solution for communication and publication of scientific documents. Documents are written in plain text files with special syntax (not unlike HTML), which then in turn are typeset into professional looking documents. This report is written in \LaTeX.[16]

**TexWorks** is an editor optimised for writing \LaTeX documents, complete with syntax highlighting and a built in typesetter.[15]

---

\(^\text{11}\)The latest version as of the writing of *Life History*, NetBeans 7.0, is not delivered pre-fitted with RoR tools[34]. Therefore, version 6.9.1 and earlier versions were used.
Chapter 5

Project management

This section will describe and evaluate a few different methodologies for organising and managing software development projects, and how they are intended to be used in the project. There are many different ways of managing the process of developing software. The decision has been to approach project management through agile development. As opposed to the waterfall-model, in which the process is broken down to well defined sequential phases\(^1\), agile programming provides a more dynamic and flexible framework which will conform to the project.

5.1 Agile programming

The term agile software development, or *agile programming*, denotes a group of methodologies that approach iterative and incremental software development. The methods described by the methodologies are usually “light weight”, in order to minimise the overhead and bureaucracy involved in managing a project, and focuses on early deliveries of functioning software. This can be of importance when it comes to the communication between the customer and the development team\(^5\). Unveiling conflicts related to the requirements as early as possible could reduce the time and effort needed to correct the misunderstandings or create alternative solutions.

5.1.1 Scrum

*Scrum*\(^{[28]}\) is a widely used agile software development process. It is intended for team-based software development, and works well for projects where requirements and goals are not set in stone, but may change as the situation develops. It is based on so-called “sprints”, which have some fixed duration. It is up to the *Scrum* team to decide what should be done during each sprint. The list of things to be done during each iteration is put into a “sprint backlog”, and at the end of each sprint, everything in the backlog should be completed. The *Scrum* team has meetings on a regular basis (e.g. each day), where each participant updates the group on what he/she has done since the last meeting, what needs

\(^{1}\)E.g. requirements, design, implementation, verification, maintenance
to be done and potential problems are addressed. One obvious advantage of using Scrum is that it will allow the customer to change their requirements later in the process, when new things might be discovered. By using Scrum, minor changes in the product requirements could be implemented by the Scrum team in a relatively short amount of time. Due to the customer being located in England, thus making communication a bit more complicated than if it was possible to meet and discuss with the customer in person, it might be prudent to expect and accommodate for misunderstandings and sudden change of plans. Everyone in the group has used Scrum to some extent in previous similar projects. As the group members feel fairly comfortable in creating and utilising the different Scrum-artefacts, this will reduce or eliminate time spent on acquiring knowledge about a (new) process model.

Roles and responsibilities specified in Scrum

- **Product owner** - the person representing the interests of the stakeholders, responsible for maintaining the product backlog.

- **Scrum master** - the team leader, responsible for enforcing the process rules and maximising the benefits of using Scrum.

- **Team** - the group of people responsible for developing the software according to the project requirements.

- **Scrum team** - product owner, scrum master and team.

Artefacts specified in Scrum

- **Product backlog** - a list of high-level requirements, ordered by importance.

- **Sprint backlog** - a list of tasks to be completed in a given sprint.

- **Sprint burn down chart** - a visualisation of the progress over time in a given sprint.

Figure 5.1: The Scrum process.
5.1.2 XP - Extreme Programming

*Extreme Programming*\(^{[43]}\) is another popular agile development methodology, which emphasises the importance of creating tests for individual units of source code before writing the actual code to pass the tests. Releasing code without tests is not allowed. As most agile processes, XP advocates frequent releases of the product in short development cycles. Each iteration in the XP development cycles includes all the members of the team as well as the customer, as communication is of great importance. The heavy focus on unit testing might not be as beneficial for this project, as opposed to other, more technically challenging projects. It has been decided to focus more on testing the usability and user-friendliness of the application.

![Extreme Programming Project](image)

Figure 5.2: Planning and feedback loops in Extreme Programming.

5.1.3 FDD - Feature Driven Development

*Feature Driven Development*\(^{[19]}\) is also an agile way of developing software, very similar to Extreme Programming. In FDD, teams of three to five developers are usually assigned the implementation of one or more features in each iteration. This provides better scalability, as having a lot of people working on the same thing can sometimes prove difficult to handle in terms of communication and management. FDD also offers progress tracking and reporting capabilities, which is extremely useful, if not a necessity, in larger-scale projects.

5.2 Choice of process model

It has been decided through careful consideration that the *Scrum* method is the most suitable choice for this project. It will help limiting the complications that may arise due to not being able to meet the customer in person, as well as allowing both the group and the customer to make alterations to the requirements as time passes and new things are discovered and thought of.

5.2.1 How Scrum will be used

The Sprints will have a duration of 14 days. The chosen team leader will be the functioning Scrum Master. As each group member has his or her own set of
scheduled lectures, it’s been decided to have plenary sessions three days a week: each Tuesday, Wednesday and Friday. Each of these meetings starts with the above mentioned meeting protocol.

5.3 Team organisation

This section will go through how the development team has been organized and which roles has been distributed to the various members of the group.

5.3.1 Internal team structure

To make the team structure as plastic as possible, two main roles was chosen for the team besides the Scrum roles: the Document Manager and the Customer Contact Responsible.

5.3.2 Scrum Master

The Scrum Master’s main responsibility will be to make sure that everyone in the group has something to work with, and that the project is moving in the right direction. The Scrum specific tasks of the Scrum Master is described in section 5.1.1.

The group Scrum Master will be Ragnar A. T. Morken

5.3.3 Product Owner

The Scrum Product Owner is responsible for prioritising and maintaining the Product Backlog. Although this task is mostly accomplished through discussion within the team, the Product Owner has the final say in the matter.

The Product Owner will be Andreas Helland

5.3.4 Document Manager

The Document Manager’s main responsibility will be to make sure everything in this project is properly documented. The scope of documentation responsibility ranges from making sure all functions and methods are properly documented in the code, diagrams and related documents, as well as making sure that proper meeting minutes are taken on each meeting. In addition to the above mentioned responsibilities, the document manager will assume the Scrum Master’s responsibilities when said person is unable to attend the meetings.

The Document Manager will be Merethe Hegset

5.3.5 Customer Contact

The Customer Contact is mainly responsible for all communication with the customer. This includes communication by email, setting up Skype meetings, logging the communication with the costumer and keeping the customer up to date with the project progress at all times.

The group Customer Contact will be Kristian Winther.
5.3.6 Developers

The rest of the group will have the task of being developers. The developer group consists of:

Espen M. Bratberg
Trond K. Bremnes

It should be noted that all members of the group function as developers to a certain degree.
# Chapter 6

## Project requirements

### 6.1 Functional requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Priority</th>
<th>Use Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1</td>
<td>High</td>
<td>1</td>
<td>View personal details about a resident, e.g profile picture, name, date of birth, work history, education history, hobbies, likes/dislikes</td>
</tr>
<tr>
<td>FR2</td>
<td>High</td>
<td>4-6</td>
<td>Manage personal details about a resident, e.g profile picture, name, date of birth, work history, education history, hobbies, likes/dislikes</td>
</tr>
<tr>
<td>FR3</td>
<td>High</td>
<td>2</td>
<td>View resident’s Life History, including text and multi-media artefacts such as pictures, video, sound clips etc.</td>
</tr>
<tr>
<td>FR4</td>
<td>High</td>
<td>12-14</td>
<td>Administer residents Life History, including text and pieces of media such as pictures, video, sound clips etc.</td>
</tr>
<tr>
<td>FR5</td>
<td>Low</td>
<td>-</td>
<td>View generic content related to the resident’s age and interests</td>
</tr>
<tr>
<td>FR6</td>
<td>Low</td>
<td>-</td>
<td>Administer generic content related to the resident’s age and interests</td>
</tr>
<tr>
<td>FR7</td>
<td>Medium</td>
<td>15</td>
<td>Resident mode where the resident is allowed to view his/her own life history, either through navigating a very simple user interface, or through random display of media</td>
</tr>
<tr>
<td>FR8</td>
<td>High</td>
<td>4-7</td>
<td>Administer residents</td>
</tr>
<tr>
<td>FR9</td>
<td>High</td>
<td>3</td>
<td>View information about residents family.</td>
</tr>
<tr>
<td>FR10</td>
<td>High</td>
<td>9-11</td>
<td>Administer information about residents family.</td>
</tr>
</tbody>
</table>

Table 6.1: Functional requirements
6.2 Non-functional requirements

6.2.1 User friendliness

As Life History is mainly intended to be used by people with limited technical insight, hence it is important to make a system that is easy to use and navigate. In order to accomplish this, the interface should be as clean and minimalistic as possible. A balance has to be achieved between too much and too little explanation of the functionality. It would be optimal to avoid showing the user a wall of text, and the buttons and links should be as self-explanatory as possible.

6.2.2 Local storage or external storage

As Life History is to be implemented as a web application (see section 3.3.3), all of the data will be stored on a centralised server. HTML5 does open for data to be stored semi-permanently on a device[39]. This will likely not be used in Life History as we envision the web server to be installed in the nursing home Life History is to be deployed in. This means that the need for external Internet access is limited, and uptime will thereby be increased.

6.3 Use cases

This section contains use cases, demonstrating the actions performed by the user in order to perform certain tasks. The use cases are divided into two main groups, i.e those with admin as the main actor, and those with carer as main actor. In addition, Use Case 15 describes the Resident’s possible actions when the system is set to ”Resident Mode”. As denoted in the textual use cases, a carer can also perform some of the tasks mainly linked to the admin role, this is however left out of the use case diagram for simplicity.
6.3.1 Use Case 1

<table>
<thead>
<tr>
<th>ID</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Select resident</td>
</tr>
<tr>
<td>Goal</td>
<td>To view resident’s profile page</td>
</tr>
<tr>
<td>Main actors</td>
<td>Carer</td>
</tr>
<tr>
<td>Start requirements</td>
<td>The user is logged on to the system.</td>
</tr>
<tr>
<td>End requirements</td>
<td>The user can view profile page of the selected resident</td>
</tr>
<tr>
<td>Main flow</td>
<td>1. Browse residents</td>
</tr>
<tr>
<td></td>
<td>2. Click on a resident’s image</td>
</tr>
<tr>
<td>Alternative flow</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 6.2: Use case 1

Figure 6.1: Use case diagram describing Use Case 1-3.
6.3.2 Use case 2

<table>
<thead>
<tr>
<th>ID</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>View memory</td>
</tr>
<tr>
<td>Goal</td>
<td>To view a memory</td>
</tr>
<tr>
<td>Main actors</td>
<td>Carer</td>
</tr>
<tr>
<td>Start requirements</td>
<td>The user is logged on to the system.</td>
</tr>
<tr>
<td>End requirements</td>
<td>The user is viewing a resident’s profile page</td>
</tr>
</tbody>
</table>
| Main flow | 1. Click on memories  
2. Select category  
3. Click on memory box |
| Alternative flow | 1. Click on memories  
2. Click on memory box |

Table 6.3: Use case 2

6.3.3 Use case 3

<table>
<thead>
<tr>
<th>ID</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>View family member/friend</td>
</tr>
<tr>
<td>Goal</td>
<td>To view information about a family member/friend</td>
</tr>
<tr>
<td>Main actors</td>
<td>Carer</td>
</tr>
</tbody>
</table>
| Start requirements | The user is logged on to the system.  
The user is viewing a resident’s profile page |
| End requirements | Information about a family member/friend is displayed |
| Main flow | 1. Click the "family and friends" button  
2. Click the family member’s card to display information |
| Alternative flow | None |

Table 6.4: Use case 3
6.3.4 Use case 4

<table>
<thead>
<tr>
<th>ID</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Add new resident</td>
</tr>
<tr>
<td>Goal</td>
<td>To add a new resident to the database</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin</td>
</tr>
<tr>
<td>Start requirements</td>
<td>The user is logged on to the system, The user has admin privileges</td>
</tr>
<tr>
<td>End requirements</td>
<td>A new resident has been added to the database, and the new profile page is displayed</td>
</tr>
</tbody>
</table>
| Main flow | 1. Click on "Menu" and choose "New resident"
2. Fill in the form
3. Upload multimedia (optional)
4. Click on "add new resident" |
| Alternative flow | None |

Table 6.5: Use case 4

Figure 6.2: Use case diagram describing Use Case 4-14.
### 6.3.5 Use case 5

<table>
<thead>
<tr>
<th>ID</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Edit resident</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>To change personal information about a resident</td>
</tr>
<tr>
<td><strong>Main actors</strong></td>
<td>Admin, carer</td>
</tr>
</tbody>
</table>
| **Start requirements** | The user is logged on to the system.  
The user is viewing a resident’s profile |
| **End requirements** | Personal information about resident is changed  
Updated profile page is displayed |
| **Main flow** | 1. Click "Menu" and choose "Edit page"  
2. Edit, add or delete information  
3. Click "Update resident" |
| **Alternative flow** | None |

Table 6.6: Use case 5

### 6.3.6 Use case 6

<table>
<thead>
<tr>
<th>ID</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Delete resident</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>To delete a resident from the database</td>
</tr>
<tr>
<td><strong>Main actors</strong></td>
<td>Admin</td>
</tr>
</tbody>
</table>
| **Start requirements** | The user is logged on to the system  
The user has admin privileges |
| **End requirements** | The resident is removed from the database |
| **Main flow** | 1. Click "Menu" and choose "Admin panel"  
2. Click the "Delete" button next to the relevant resident  
3. Click "Yes" to confirm your decision |
| **Alternative flow** | None |

Table 6.7: Use case 6

### 6.3.7 Use case 7

<table>
<thead>
<tr>
<th>ID</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Add resident to carer</td>
</tr>
<tr>
<td><strong>Goal</strong></td>
<td>To allocate a resident to a specific user</td>
</tr>
<tr>
<td><strong>Main actors</strong></td>
<td>Admin</td>
</tr>
</tbody>
</table>
| **Start requirements** | The user is logged on to the system.  
The user has admin privileges |
| **End requirements** | Resident is shown in the user’s "My residents” view |
| **Main flow** | 1. Click on "Menu" and choose "Admin panel"  
2. Click on "Show user"  
3. Click on the "Add" button next to the relevant resident |
| **Alternative flow** | 1. None |

Table 6.8: Use case 7
### 6.3.8 Use case 8

<table>
<thead>
<tr>
<th>ID</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>New User</td>
</tr>
<tr>
<td>Goal</td>
<td>To add a new user</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin</td>
</tr>
<tr>
<td>Start requirements</td>
<td>The user is logged on to the system. The user has admin privileges</td>
</tr>
<tr>
<td>End requirements</td>
<td>A new user is added to the database</td>
</tr>
</tbody>
</table>
| Main flow | 1. Click on "Menu" and choose "Admin panel"  
2. Click on "New user"  
3. Add information and click "Create user" |
| Alternative flow | None |

Table 6.9: Use case 8

### 6.3.9 Use case 9

<table>
<thead>
<tr>
<th>ID</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Add family member or friend</td>
</tr>
<tr>
<td>Goal</td>
<td>A relation page is added</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin, carer</td>
</tr>
</tbody>
</table>
| Start requirements | The user is logged on to the system  
The user is viewing the family and friends overview |
| End requirements | A new relation page is added  
Information about the new relation is displayed |
| Main flow | 1. Click on "Menu" and choose "new relation"  
2. Fill in form with information  
3. Upload media (optional)  
4. Click "create" |
| Alternative flow | None |

Table 6.10: Use case 9

### 6.3.10 Use case 10

<table>
<thead>
<tr>
<th>ID</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Edit relation</td>
</tr>
<tr>
<td>Goal</td>
<td>To change information about a family member or friend</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin, carer</td>
</tr>
</tbody>
</table>
| Start requirements | The user is logged on to the system  
The user is viewing person page |
| End requirements | Information about relation is changed |
| Main flow | 1. Click on "Menu" and choose "Edit page"  
2. Add/remove desired information  
3. Click "Update" |
| Alternative flow | None |

Table 6.11: Use case 10
6.3.11 Use case 11

<table>
<thead>
<tr>
<th>ID</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Delete relation</td>
</tr>
<tr>
<td>Goal</td>
<td>To delete a family member/friend</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin, carer</td>
</tr>
</tbody>
</table>
| Start requirements| The user is logged on to the system  
The user is viewing relation page |
| End requirements| The relation is deleted from the database  
The friends and family overview is displayed |
| Main flow| 1. Click "Menu" and choose "Delete relation" |
| Alternative flow| None |

Table 6.12: Use case 11

6.3.12 Use case 12

<table>
<thead>
<tr>
<th>ID</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Add memory</td>
</tr>
<tr>
<td>Goal</td>
<td>To add a memory to a resident</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin, Carer</td>
</tr>
</tbody>
</table>
| Start requirements| The user is logged on to the system.  
The user is viewing a resident’s memory index |
| End requirements| A memory is added  
New memory page is displayed |
| Main flow| 1. Click "menu" and choose "New memory"  
2. Add information and click "Create"  
3. Upload media (optional)  
3. Click "Save" |
| Alternative flow| None |

Table 6.13: Use case 12

6.3.13 Use case 13

<table>
<thead>
<tr>
<th>ID</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Edit memory</td>
</tr>
<tr>
<td>Goal</td>
<td>To change information about a memory</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin, Carer</td>
</tr>
</tbody>
</table>
| Start requirements| The user is logged on to the system.  
The user is viewing a resident’s memory |
| End requirements| A memory is changed  
Updated memory page is displayed |
| Main flow| 1. Click "menu" and choose "Edit page"  
2. Add information and click "Update" |
| Alternative flow| None |

Table 6.14: Use case 13
6.3.14 Use case 14

<table>
<thead>
<tr>
<th>ID</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Delete memory</td>
</tr>
<tr>
<td>Goal</td>
<td>To delete a memory from the database</td>
</tr>
<tr>
<td>Main actors</td>
<td>Admin, Carer</td>
</tr>
<tr>
<td>Start requirements</td>
<td>The user is logged on to the system.</td>
</tr>
<tr>
<td></td>
<td>The user is viewing a resident's memory</td>
</tr>
<tr>
<td>End requirements</td>
<td>A memory is deleted</td>
</tr>
<tr>
<td></td>
<td>Memory index is displayed</td>
</tr>
<tr>
<td>Main flow</td>
<td>1. Click on &quot;Menu&quot; and choose &quot;Delete memory&quot;</td>
</tr>
<tr>
<td>Alternative flow</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 6.15: Use case 14

Figure 6.3: Use case diagram describing Use Case 15
6.3.15 Use case 15

<table>
<thead>
<tr>
<th>ID</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Browse and select memory</td>
</tr>
<tr>
<td>Goal</td>
<td>To view a memory page</td>
</tr>
<tr>
<td>Main actors</td>
<td>Resident</td>
</tr>
<tr>
<td>Start requirements</td>
<td>The system is in &quot;Resident Mode&quot;</td>
</tr>
<tr>
<td>End requirements</td>
<td>A memory page is shown</td>
</tr>
</tbody>
</table>
| Main flow| 1. Browse memories  
| | 2. Click memory |
| Alternative flow| None |

Table 6.16: Use case 16
Chapter 7

Time plan

This section will list the main milestones and sprint goals for the project.

7.1 Milestones

- Project report due - preliminary version (31 January 2011)
- Project report due - midterm version (29 February 2011)
- Project report due - for final comments from supervisor (8 April 2011)
- Project report due - final version (15 May 2011)

The project has been divided into 8 sprints of roughly 2 weeks each.
7.2 Work breakdown structure & Gantt diagram

Figure 7.1: Work Breakdown Structure for the project
Figure 7.2: Gantt diagram

<table>
<thead>
<tr>
<th>ID</th>
<th>Task/Task Name</th>
<th>Duration</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Delivery: Preliminary report</td>
<td>0 days</td>
<td>Mon 31.01.11</td>
<td>Mon 31.01.11</td>
</tr>
<tr>
<td>2</td>
<td>Delivery: Midterm version of report</td>
<td>0 days</td>
<td>Mon 28.02.11</td>
<td>Mon 28.02.11</td>
</tr>
<tr>
<td>3</td>
<td>Delivery: Project report (for final comments)</td>
<td>0 days</td>
<td>Fri 08.04.11</td>
<td>Fri 08.04.11</td>
</tr>
<tr>
<td>4</td>
<td>Delivery: Project report</td>
<td>0 days</td>
<td>Sun 15.05.11</td>
<td>Sun 15.05.11</td>
</tr>
<tr>
<td>5</td>
<td>WP1: Planning</td>
<td>16 days</td>
<td>Mon 17.05.11</td>
<td>Sun 06.02.11</td>
</tr>
<tr>
<td>6</td>
<td>Sprint 1</td>
<td>10 days</td>
<td>Mon 27.01.11</td>
<td>Fri 29.01.11</td>
</tr>
<tr>
<td>7</td>
<td>WP2: Documentation</td>
<td>14 days</td>
<td>Mon 07.02.11</td>
<td>Mon 18.02.11</td>
</tr>
<tr>
<td>8</td>
<td>Sprint 2</td>
<td>10 days</td>
<td>Mon 31.01.11</td>
<td>Fri 11.02.11</td>
</tr>
<tr>
<td>9</td>
<td>Sprint 3</td>
<td>10 days</td>
<td>Mon 14.02.11</td>
<td>Fri 25.02.11</td>
</tr>
<tr>
<td>10</td>
<td>WP3: Programming</td>
<td>15 days</td>
<td>Mon 28.02.11</td>
<td>Thu 15.04.11</td>
</tr>
<tr>
<td>11</td>
<td>Sprint 4</td>
<td>10 days</td>
<td>Mon 28.02.11</td>
<td>Fri 11.03.11</td>
</tr>
<tr>
<td>12</td>
<td>Sprint 5</td>
<td>10 days</td>
<td>Mon 26.02.11</td>
<td>Fri 25.03.11</td>
</tr>
<tr>
<td>13</td>
<td>Sprint 6</td>
<td>10 days</td>
<td>Mon 28.02.11</td>
<td>Fri 08.04.11</td>
</tr>
<tr>
<td>14</td>
<td>WP4: Implementation</td>
<td>21 days</td>
<td>Mon 28.05.11</td>
<td>Sun 24.04.11</td>
</tr>
<tr>
<td>15</td>
<td>Sprint 7</td>
<td>15 days</td>
<td>Mon 11.04.11</td>
<td>Fri 29.04.11</td>
</tr>
<tr>
<td>16</td>
<td>WP5: Finalization</td>
<td>21 days</td>
<td>Sat 16.04.11</td>
<td>Sun 16.05.11</td>
</tr>
<tr>
<td>17</td>
<td>Sprint 8</td>
<td>10 days</td>
<td>Mon 02.05.11</td>
<td>Fri 13.05.11</td>
</tr>
</tbody>
</table>

Project: gent
7.3 Sprint diary

7.3.1 Product & Release backlog

<table>
<thead>
<tr>
<th>Description</th>
<th>EST.</th>
<th>By</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Very High Priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Residents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 View a specific resident</td>
<td>40</td>
<td>M</td>
</tr>
<tr>
<td>2 View all residents</td>
<td>30</td>
<td>TK</td>
</tr>
<tr>
<td>3 Create the resident model</td>
<td>20</td>
<td>A</td>
</tr>
<tr>
<td>Memoriai</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Create the memory model</td>
<td>20</td>
<td>E</td>
</tr>
<tr>
<td>5 Upload media</td>
<td>20</td>
<td>K</td>
</tr>
<tr>
<td>6 Display memory</td>
<td>12</td>
<td>M</td>
</tr>
<tr>
<td>7 Navigate in memories</td>
<td>30</td>
<td>RA</td>
</tr>
<tr>
<td>8 Display media</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Family and friends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Display family and friends</td>
<td>12</td>
<td>E</td>
</tr>
<tr>
<td>10 Create the Family and friends model</td>
<td>20</td>
<td>A</td>
</tr>
<tr>
<td>Layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 Implement user friendly interface for interacting with data stored</td>
<td>180</td>
<td>All</td>
</tr>
<tr>
<td>12 Design a user friendly overall site theme</td>
<td>180</td>
<td>All</td>
</tr>
<tr>
<td><strong>High Priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 Login page</td>
<td>20</td>
<td>E</td>
</tr>
<tr>
<td><strong>Medium Priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 Navigation bar</td>
<td>24</td>
<td>A</td>
</tr>
<tr>
<td><strong>Low Priority</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Family and friends/Resident user</td>
<td>24</td>
<td>RA</td>
</tr>
<tr>
<td>Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search function</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>Generic memory</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.3: Product and release backlog

7.3.2 Sprint 1 (week 3 and 4)

This sprint mainly focused on getting to know each other, the project itself and start the planning phase. The preliminary version of the project report was due by the end of the sprint. By the end of this sprint, some work on discussing what development platform to choose, creating the preliminary product backlog, early drafting of UML diagrams, and plan a meeting with the customer remained. This was transferred to the next sprint.

Note that the development model (Scrum) was not yet chosen at the absolute beginning of the sprint. The first sprint was planned to coincide with preliminary report delivery, and thus were not the full two weeks (120 hours) as “ordinary” sprints are. The first sprint spans from the day we first met (the day after the project groups were finalised), 19 January to 30 January.

While the project report has gotten its own task, some crucial parts of the report has their own tasks, as they represent diagrams that will be used to model the application itself.
7.3.3 Sprint 2 (week 5 and 6)

This sprint mainly focused on finalising the planning phase of the project, as well as finalise the decision on what platform that Life History should be built for. Early design and mock-ups was started and given a lot of work. Together with the update of UML-diagrams, Gantt-diagram, and Version control system setup, it had to be transferred to the next print due to remaining work.
### 7.3.4 Sprint 3 (week 7 and 8)

This sprint mainly focused on finishing up the sketches of the design and mock-ups, the mid-term report, and starting to read up on how to work with Ruby on Rails. This was also transferred to the next sprint, since learning how to work
with Ruby on Rails is a big process and not done in a couple of weeks.

<table>
<thead>
<tr>
<th>Story ID</th>
<th>Story/task</th>
<th>M</th>
<th>T</th>
<th>W</th>
<th>T</th>
<th>F</th>
<th>S</th>
<th>S</th>
<th>M</th>
<th>W</th>
<th>T</th>
<th>F</th>
<th>S</th>
<th>S</th>
<th>Days in sprint / effort left</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Backlog for sprint three</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Burn-down chart for sprint three</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.8: Backlog for sprint three.

Figure 7.9: Burn-down chart for sprint three.
7.3.5 Sprint 4 (week 9 and 10)

This sprint mainly focused on a new sprint planning meeting, meeting with the customer, and programming the views and functionality for Memory, Residents, Family and Friends, and Login. The remaining work on functionality of the different views was transferred to the next sprint.

<table>
<thead>
<tr>
<th>Story ID</th>
<th>Story/task</th>
<th>Days in sprint / effort left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Project management/planning</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Workshops</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Project report</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Work on midterm report</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Updated class diagram</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Technical/implementation</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Version control system setup</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Functional Memory</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Resident modeling</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Memory modeling</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Family and Friends modeling</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Login issue</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Meetings</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Meeting with a contact at the local resource center for Dementia</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Sprint planning meeting</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Self-study/increase competency</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Rails tutorials</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 7.10: Backlog for sprint four.

Figure 7.11: Burn-down chart for sprint four.
7.3.6 Sprint 5 (week 11 and 12)

This sprint mainly focused on continuing with the programming of the different views. Some work remained on the functionality of every view, and was given a lot of attention. Attaching pictures to memories to different memories, adding memories, and the navigation bar still needed some work, and was transferred to the next sprint.

<table>
<thead>
<tr>
<th>Story ID</th>
<th>Story/task</th>
<th>Days in sprint / effort left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Technical/Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Media box in Ruby on Rails</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Add memories</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Attaching pictures to memories</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Working resident profile</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Working resident overview</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Working Family and friends view</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relating memories to resident</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Relating family and friends to resident</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changed from using Prototype to using Query</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Papersilo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Self-study/ increase competency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attending course in Ruby on Rails</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ruby on Rails / plugins tutorials</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Learning Query</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Remove prototypes</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Tactician</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project report</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Meditations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sprint opening meeting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Meeting with supervisor</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.12: Backlog for sprint five.

![Burndown chart for sprint 5](chart.png)

Figure 7.13: Burn-down chart for sprint five.
7.3.7  Sprint 6 (week 13 and 14)

This sprint mainly focused on working on the project report, implementing the
different views and the ability to navigate from one view to another, generating
breadcrumbs, implementing a functional swipe-function for the ipad, and meet-
ing with the customer and supervisor. Another sprint planning meeting was
held. The remaining work on the Resident mode prototype was transferred to
the next sprint together with the remaining work on the Breadcrumb function.
<table>
<thead>
<tr>
<th>Story ID</th>
<th>Story/Task</th>
<th>Days in sprint / effort left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>Technical/Implementation</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Test</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Resident mode prototype</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Market</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11 &amp; 12 Redesign av memory/PS</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>11 &amp; 12 Redesign av residente</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>Breakumbe</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Memories</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Add ability to navigate between memories</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Edit media given to memory</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Add Media to memories</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Working media switcher</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Project report</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>New Introduction</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Remaining chapter structure</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>User manual/Installation guide</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Upgrading diagrams</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Meeting notes</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Meetings</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sprint planning meeting</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Meeting with supervisor x 2</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Meeting with customer</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 7.14: Backlog for sprint six.

![Burndown chart for sprint 6](image)

Figure 7.15: Burn-down chart for sprint six.
7.3.8 Sprint 7 (week 15 and 17)

Week 16 was Easter holiday. This sprint mainly focused on finalising the redesign, and implement the functionality for each view to be able to work together and the possibility to easy navigate between. Much work was put in to the final report, and the remaining work was transferred to the next sprint.

<table>
<thead>
<tr>
<th>Story ID</th>
<th>Story/Task</th>
<th>Days in sprint / effort left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>11 &amp; 12</td>
<td>Finalising redesign</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Log in implementation</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Memory conservnation</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Resistant recipe function</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Show photo of resident</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Reestimations</td>
<td>12</td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Run user testing</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Project report</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Work on final report</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>Meetings</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Sprint planning meeting</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 7.16: Backlog for sprint seven.

Figure 7.17: Burn-down chart for sprint seven.

7.3.9 Sprint 8 (week 18 and 19)

This sprint mainly focused on finalising the project for the delivery date of 15 May. This included finding and fixing the remaining bugs in the system, as well
as do a few proper user tests. Finalising the rest of this report were also highly prioritised.

<table>
<thead>
<tr>
<th>Story ID</th>
<th>Story/task</th>
<th>Days in sprint / effort left</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Plan a visit to a senior centre and perform user testing.</td>
<td>18 18 12 12 12 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Analyse test result data from user testing.</td>
<td>6 6 6 6 6 6 6 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Perform acceptance testing.</td>
<td>12 12 12 12 12 12 12 12 12 12 12 0 0</td>
</tr>
<tr>
<td></td>
<td>10 Implementing new functionality.</td>
<td>4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Implementing the resident mode.</td>
<td>4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>Bug-fixing</td>
<td>10 10 10 10 10 10 10 10 10 10 10 10</td>
</tr>
<tr>
<td></td>
<td>Mitigate faults revealed in user testing.</td>
<td>20 20 20 20 20 20 20 20 20 20 20 20</td>
</tr>
<tr>
<td></td>
<td>Remove inconsistencies in the application GUI.</td>
<td>20 20 20 20 20 20 20 20 20 20 20 20</td>
</tr>
<tr>
<td></td>
<td>Project report</td>
<td>Finalize the project report.</td>
</tr>
<tr>
<td></td>
<td>Meetings</td>
<td>Sprint planning meeting</td>
</tr>
<tr>
<td></td>
<td>Meeting with supervisor</td>
<td>8 8 8 8 8 8 8 8 8 8 8 8 8 8 8</td>
</tr>
</tbody>
</table>

Figure 7.18: Backlog for sprint eight.

Figure 7.19: Burn-down chart for sprint eight.
Chapter 8

Design

This section of the report will detail the different design levels of Life History. This includes detailing the architectural outline of the application, the database model implemented in the application, the different user levels and roles of the application, and the security design of the application, as well as an exhaustive discussion of the visual design of Life History. Special care has been taken with the visual representation of the application, seeing as Life History is designed to be used by an audience with little to no experience with computers.

8.1 Architectural outline

Life History is constituted of mainly three parts: the Resident; Memories; and Family and Friends. Each of the entries in the Memories part is connected to exactly one resident, and each of the entries in the Family and Friends part is connected to exactly one resident. For a more thorough discussion on how this is achieved in the database used in Life History (also referred to Life History’s model). The navigation between these parts of the application is shown in the site map in figure 8.1, the actual actions required to navigate between the screens are described in greater detail in section 8.4.
8.1.1 Package structure

*Life History* uses the *MVC architecture* for the implementation, as *Ruby on Rails* uses this software architecture by default. A description of the MVC architecture can be found in section 4.2. The encouraged use of MVC in RoR, has a significant impact on the package structure of *Life History*: namely that near each and every element (as in a *memory* or a *resident*) has a *model*, *views* and a *controller*.

From the log in screen the user is directed to a screen showing the residents stored in the system. From this screen, the user is able to choose a resident. On the resident page, the user is able to enter either the *Memories* or the *Family and Friends* part of the application. When entering the *Memories* part of the application the user is presented with the memories stored in the system for the resident that the user has accessed. The user is able to navigate through these memories.

When the user selects the *Family and Friends* part of the application, the user is presented with every person stored for the selected user. From here, the user can enter a profile page for this person.

*Life History* has an administration page that is accessible from every page on the system for a user with administrator privilege.

---

1Exceptions to this "norm" includes the *admin* element which has no model, as its function is to administer the other models.
Class Diagram

The class diagram in figure 8.2 shows how the models, views and controllers are connected.

To aid the comprehension of the class diagram better, the full class diagram has been divided into three parts, each representing its own part of the MVC architecture. The full class diagram in figure 8.2 shows how the three are connected. The views are here seen linked to each other where one is able navigate between them in the application. The controllers are linked to the models where they extract data from, and to the views where their functions are used.

The models are linked in between themselves to models that An example is the Memory which is linked to a Medium and a Categorisation, as a memory is supposed to have media and categories.
Figure 8.2: Complete class diagram
Figure 8.4: View
Controllers
8.2 Database design

The database system there will be three principal tables of information: the Memory; the Ff; and the Resident. Connected to these are the remaining tables.

Memory and Ff

Each memory and family/friend are owned by one single resident. A connection between the Memory and Ff tables and the Resident table exists to handle this.

Medium and FfMedium

Connected to the memories and family/friends are media files. The tables Medium and FfMedium handles this tasks, and connects media files to a Memory and an Ff, respectively. Each Memory and Ff can be assigned several media files, but each media file can only be attached to exactly one Memory or Ff.

Category and Categorization

Exists to handle categorisation of memories as a many-to-many relation. Each Category can be attached to several Memories and each Memory can be categorised with several Categories.

Resident

The residents has, as mentioned above a set of memories and relations connected to itself. Connected to a resident is a collection of stored facts in the table ResidentDatum.

ResidentDatum

An example of a data tuple would be Basic description or nickname. Should a nickname be stored, the Description field would say nickname, while the text field would be the actual nickname.

Rmedium

Handles the attachment of a picture to a Resident.

Users and Belongs

Exists to handle the user and user roles of the application. The Belongs table handles the connection between a User and a Resident. Each carer can be assigned several residents, and each resident can be under the care of several carers.

Figure 8.6 shows a conceptual model of the database used by Life History as an ER-diagram\(^3\).

\(^2\)Our abbreviation of Family and Friends used throughout the application code and database.

\(^3\)Entity-relationship modelling is a database modelling method, used to produce a type
8.2.1 Database overview

- Categorization(created_at, updated_at)
- Category(name, created_at, updated_at)
- Memory(title, time, text, created_at, updated_at)
- Resident(name, created_at, updated_at, photo_file_name, photo_content_type, photo_file_size, photo_updated_at)
- Ff(text, name, relation, created_at, Updated_at)
- FfMedium(description, created_at, updated_at, photo_file_name, photo_content_type, photo_file_size, photo_updated_at)
- ResidentDatum(description, text, created_at, updated_at)
- DataDescription(description, created_at, updated_at)
- Belongs(user_id, resident_id, created_at, updated_at)
- Users(username, password, role, created_at, updated_at, salt)
- Medium(description, filepath, filetype, created_at, updated_at, photo_file_name, photo_content_type, photo_file_size, photo_updated_at)
- Rmedium(description, created_at, updated_at, photo_file_name, photo_content_type, photo_file_size, photo_updated_at)

of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.
8.3 Users and user roles

*Life History* is designed to be used by different types of users, each with its own role and accompanying privileges. The roles devised for *Life History* is the **administrator** and the **carer**. A third user role, the **resident** has been considered and even partially implemented, but had to be omitted in the late stages of the project due to security issues and time constraints. This user was thought to be used by the resident’s kin to allow them to view, add, edit and remove information about a resident.

Note that in this section the words *user* and *users* are describing *user accounts* in the application, rather than the actual human user as it is used in the rest of this report.

8.3.1 Carer

A **carer** has access to the complete system, except that a carer can not add or remove neither users nor residents. Even though there are systems in place for assigning residents to a carer’s care, there are no restrictions on what residents a carer can access. *Life History* will however be featuring the residents under
ones care more prominently than not. See section 8.4.6 for the implementation of such filtering.

8.3.2 Administrator

The administrator has all the privileges that of the carer with some additions: the administrator can add and remove users as well as assigning roles; and the administrator can add and remove residents. The administration interface is accessible from anywhere in the application through the Menu-bar. See section 8.4.6 for the implementation of the navigation bar and menus.

8.4 The graphical user interface

This part of the report will detail the visual representation of Life History. This includes a discussion on the suitable metaphors that have been considered for making Life History easy to use for computer illiterate people, as well as the actual design and layout of Life History complete with sketches and screen shots.

The graphical interface is the part which the users of Life History is actually seeing and using when operating the application. Great care has been taken to make it as visually pleasing, and still usable as possible.

No special care has been taken to make Life History operate well with screen readers or similar tools. Life History is thus highly unlikely to be usable by blind or visually handicapped people, but is by the developers not considered any less usable than a physical story book.

8.4.1 Use with people with dementia

Unfortunately there has as of today been little scientific research on interface design on people suffering of dementia, and thus has not been produced a set of guidelines that universally can be put to use when designing Life History for this audience. The guidelines that do exists therefore must be used as mere suggestions rather than a silver bullet. Extensive user testing in the intended environment of Life History is needed to get a good understanding of what works and what does not. Additionally, the primary user of Life History are the carers of people suffering from dementia, the residents are secondary users. Depending on the individual severity of the condition, it may be unfeasible to even hope to achieve a usable design that can be used by all. It is then up to the carers to assess the mental state of the resident and give the resident the opportunity to try Life History if he or she shows interest. The tablet computer device is a fairly new technology whose software has not been extensively tested on all publics, especially not publics with special needs such as those suffering from dementia.

8.4.2 Minimalism

It is of major importance that Life History is as easy to use as possible. This is accomplished by limiting the number of possible actions the user is able to perform at every step, to limit the potential for confusion (as in not knowing
what action produces what result). This means no superfluous buttons and no superfluous text.

Every button has to be as self explanatory as possible, considering the computer literacy of the users, and even greater care has to be taken for the parts of Life History that is to be used by the residents, considering their mental state. To exemplify this, one can consider the action of navigating between memories in the Memory part of Life History. There should only need one button for going to the next Memory, and one button for going to the previous Memory. Any other buttons, links or action enabling objects that is needed should be added with great care, and only if completely necessary.

8.4.3 Metaphor

As a metaphor the book has been deemed suitable to simplify the concepts and functions that Life History is to convey and accomplish. As mentioned in the prestudy in chapter 3, nursing homes today often use analog books with pictures to learn to know the resident. By emulating a known concept that is already employed, Life History hopes to achieve a sense of familiarity in the user. Anyone who has seen and learned how to use a book will know that it consists of turnable pages.

To illustrate this, one can consider the Memory part of Life History. For the display of a memory the device is to be held on its side in landscape mode, where the screen then will be divided into two parts on the middle. On the left side on this divide Life History will display the title of the memory, media belonging to the memory and text belonging to the displayed media. On the right side of the screen Life History will display the time the memory took place and a text belonging to the displayed media.

Attached to each memory there can be a collection of media. These are to be represented as Polaroid pictures placed on the left hand side of the spread book. The description connected to each medium is to be printed on each medium.

See section refsec:navigation for details about navigation between the screens of Life History.

One special occasion where abandoning the book metaphor is considered, is when the user is entering larger amounts of data\(^4\) into the application. This function is only accessible through a separate user interface only used by administrative personnel.

Here, no specific metaphor is being used, as the input screen is thought to be used seldom by trained personnel, and the need for anything except a simple collection of forms to fill is limited.

Life History should, however, have the ability to make small improvements and/or additions, without a complete context switch, as it may be easier for the user to enter data from the same context the data are displayed in. This might be accomplished by having an administrative edit button displayed for users logged in with the appropriate privileges.

A background texture reminiscent of paper has been tested to make Life History to be as close to a book as possible, as has a wooden texture to make a "table" for the "book" to rest upon. The paper texture for the book's paper was

\(^4\)Larger amounts can be adding several new residents complete with several memories, each with several media. This would in the book metaphor force the user to navigate through several different screens for each of the residents, which can be very ineffective.
in the end omitted due to a clear white background being sufficient to convey the idea. The wooden table texture was included as the background because of the clear distinction from the actual book (negative space) and how easy it was to recognise what the visual style is supposed to emulate.

Seeing as quite a few of the screens that constitutes Life History are divided into two parts down the middle, a shadowed line emulating the book binding has been chosen to make the metaphor even more distinguished.

To represent the resident overview page where the system’s user is supposed to choose which resident to view, the metaphor is a long horizontal row of books lying on the background table where the user can choose which book it wishes to open. These “books” has been designed as traditional looking books with a large picture of the relevant owning resident and his or her name clearly pasted on the book’s front cover. The system as a whole is thus represented as a collection of books, each containing the life history of each of the residents.

On the overview of all the residents in the system, the individual resident is presented as books, as though the books

During the user testing of Life History, it seemed that the test subject recognised it, and responded positively (see section 9.2.2).

8.4.4 Navigation

The primary users of Life History, the carers on the nursing homes, are thought to have limited computer literacy according to a meeting with Mr. Pudney. Navigation through a book is however a concept that is easy to recognise and it is thus thought that the carers should be able to handle navigating through Life History.

The residents are thought to be familiar with the concept of how books work, and how page turning provides new text at every turn until the end of this book. The method of navigation back and forth through imagined pages in the application can prove to be an obstacle to overcome, and has to be carefully considered. Considering the mental state of these secondary users, much care has to go into making the navigation as intuitive as possible.

An example the group has discussed on how to achieve this is to use the sides of the memory viewing part of the application (which should be able to be used by the resident unsupervised) to navigate back and forth on a chronological timeline of memories. The right side of the “page” will contain a button which bring you to the next memory, the left side of the “page” will contain a button which will bring you to the previous memory. The function of these buttons has to be as redundantly clear as possible, which can be done by using big arrows to point in the direction, as well as writing in clear text what the outcome of the button press will be. Whether or not this will be enough remains to be seen, and testing of various options in the field test of the application to determine the best one will be performed.

The interface for editing, deleting or adding content will always be accessed through a menu button located in the navigation bar at the top of the relevant interface. An example would be if you want to edit a memory, you then access the edit interface by going to this memory followed by going to the edit page through the menu.
On the *Memories* part of the application one is able to both the previous and next memory\(^6\).

On each memory that contains media there are buttons that lets the user change the medium to the next and previous medium\(^7\).

**Spatial navigation**

The human brain has evolved to let its owner be aware of its placement in a room. This puts a slight limitation to what is allowed to do when it comes to navigating, and what animations to display without confusing the user.

In the instance of the user turning the page forth from page 1 to page 2, it would be reasonable for the user to expect to see page 1 when turning the page back again. Similarly, if an object was put (for instance the *Settings* menu) on top of the book, it would be reasonable for the user to expect that the book is located underneath the *Settings* menu, and to go back to the book again, the “correct” dismissal of said menu would be to remove it from the top of the book, and not put the book on top of the menu again.

### 8.4.5 Screen size

The device chosen as the main device to use and test for, the Apple iPad, has a screen size of 10 inches across, and a screen resolution of 1024 times 768 pixels\(^8\).

The book metaphor will work best when the device is held sideways in landscape mode. The iPad is able to lock this screen orientation. The screen resolution operated with when designing *Life History* is approximately 1024x700 to accommodate the various menu and browser bars that the iPad has. If *Life History* were to be deployed as a native application, and not the web application that it is, it would be possible to remove these menu bars and thus have more vertical space.

### 8.4.6 Page layout

In his section the layout of the main screens that constitutes of the application will be described, complete with screenshots and early mockups to describe the screens' evolution.

*Life History* is optimised for the screen size and resolution of the Apple iPad, locked in a horizontal orientation layout. Most of *Life History*’s screens, the elements of the screen is laid out over two pages, that, with the tablet device turned on its side horizontally, emulate an open book as per the chosen metaphor.

**Resident overview**

The resident overview screen is the first screen a carer sees after logging in to the application. Here, the residents are represented as owners of books with their picture and name prominently featured on the book’s front page. These books are placed on a scrollable horizontal line which the carer can swipe

\(^6\)The order of the memories are defined by the order the memories are added to *Life History*

\(^7\)The order of the media is defined by the order the media are added to *Life History*

\(^8\)Commonly referred to as a 10” screen with a resolution of 1024x768
through with their fingers. Touching a book will open it, and the user is directed to the resident profile page. The carer can also choose if he/she wants to see all residents, or just residents under his/hers care.

See figure 8.16 for the final design for the resident overview screen.

**Resident Profile**

The **resident profile** screen is the central hub for all the information *Life History* stores about a given resident. On this screen the resident’s name, his/her picture and selected information is displayed. The screen is laid out like a book, with the picture on the right hand side of this book, and the information on the right. Below the content of the *Resident overview* screen there are two buttons: one to bring the user to the *Memory overview* screen; and one to bring the user to the *Family and Friends overview* screen.

See 8.7 for a more detailed sketch that illustrates this design. Figure 8.8 shows the final design.

![Figure 8.7: Mockup of the resident profile](image)
Memory

The Memory screen is where the data connected to a memory entity is displayed. This Memory is spread across the two pages, with associated media on the left hand side and the associated text on the right side. See 8.9 for a more detailed sketch that illustrates this design.

The reasoning behind the placement of the screen elements is based on the reading direction the target audience operates with: left towards right. The main part of the history is thought to be the media collection, and will therefore be prominently displayed at the beginning of each page.

The left hand side of the Memory screen contains a media box. This box contains all the associated media connected to the displayed Memory, and has two buttons, previous and next that lets the viewer navigate through all the associated media. This navigation loops: when the viewer press next on the last medium, the first one will be displayed. Accompanying the medium is a description. This description is displayed under the associated medium. The media box is designed to look like a Polaroid photograph to emulate the physical nature of the metaphor. The right hand side of the Memory screen contains a textual short story describing the Memory. This text is scrollable. Below the contents of the Memory there is another set of previous and next buttons. These buttons will send the viewer to the previous and next Memory, respectively. By touching the medium in the media box, a bigger version of the medium will fill the screen. One is able to navigate between media in the same manner one is able to in the media box. By touching the big medium, the big medium closes, and the user will find itself on the Memory screen again. The prototype version of Life History does not have the ability to display anything other than pictures in place of a medium. It is near trivial to implement support for other types of media, but it had to be omitted due to time constraints.

At the top of the screen there has been placed two buttons reminiscing
bookmarks, labeled with arrows pointing left and right. Touching these buttons will change the displayed memory to the previous and next memory, respectively. What constitutes the "previous" and "next" memory is the chronological order the memory has been added. Discussions were had whether or not to have these buttons take the user to the next and previous memories (again chronological) within a category, but this idea was not put in place due to uncertainties on how to then navigate to other categories.

While in resient mode (see 8.4.6) the navigation bar on top displays a button which takes the viewer to a random memory when touched.

Figure 8.9: Mockup of the Memory page
We spent a lovely 3 days at the cabin and taking in the scenic beauty of all the surroundings. Myself, my dear husband and our son was there.

This trip was special, as it was my birthday. We had been there before a couple years ago as well, but that year it was still quite hot and we didn’t feel like loading up on food. But this year it was beautiful weather-wise and we were going to enjoy it! We arrived around 11 a.m. Wednesday and found our cabin was already ready and waiting for us. We were in the 2500 loop and just like last time we were lucky as the bus stop was right outside our door. We went inside and it was very clean. We did find a few of the flooring sections peeling up by the sink, but that was really the only thing we noticed that needed fixing...for now.

Friday was AK day and finishing up our food journey in Epcot. Everything we ate was very good and some of the wines were amazing! I could have spent a bit more and I went bottomless...
Family and friends

The Family and friends pages are modelled after the Memory pages as to not require the users of Life History to learn to use yet another interface. As with the Memory pages, the screen is divided into two parts on the middle. The left side contains the name of the person viewed along with a medium added for the person. The right side contains a text related to the person viewed.

A relation contains only one piece of media without a description. Unlike the media box present in Memory, it is thus not possible to browse between media, nor is it possible to enlarge a medium by touching it. Similarly to the media box present in Memory, support for other media types than pictures is not present in the prototype version of Life History. Again, the implementation of this feature is near trivial, but had to be omitted due to time constraints.
Memory overview

The Memory overview screen displays the memories connected to a given resident, organised by category. The screen is divided into two parts in the middle, where the left part contains the categories presented as pushable buttons. When a category on the left side is selected, the right side is populated by memories categorised as the chosen category.

The memories that populates the right hand side of the screen is presented as pushable buttons with the title of the Memory and an excerpt of the text if available.
Family and Friends overview

The Family and Friends overview screen displays the relations connected to a given resident, organised by relationship. As with the Memory overview screen, the screen is divided into to parts down the middle, with the relationships (analogous to the categories of memories) as buttons on the left hand part of the screen. When a relationship button on the left side is pressed, the right hand side is populated by relations categorised as the chosen relationship.

The relations that populates the right hand side of the screen are presented as pushable buttons, with a the name and a picture if available to quick and easy recognise the relation that is to be viewed.
Figure 8.15: Family and friends overview

Log in screen

The Log in screen displays two simple text field where one must enter ones username and password to gain access to Life History.

Resident overview

The Resident overview screen shows an overview of all the residents registered in the system. The residents are placed beside each other. To see through all residents the users can use the swipe function to scroll through them. Here one can choose to filter out residents not assigned to the relevant user account and only browse residents in the users care by clicking the Your residents button, or one can browse them all by clicking the button titled All residents Figure 8.16
Resident mode

The resident mode is thought to be a simplified way of looking through the Memory part of Life History. As mentioned in section 2.3.2, the residents of the home that Life History will be deployed in has special needs, which has to be reflected when it comes to interface design. That is why the resident mode has been designed. The resident mode will have most of the features of Memory deactivated, both to reduce the complexity of the application by minimising the number of distracting buttons, as well as minimising the potential of both intended and unintended wrongdoings by the handler of the device.

With the device in resident mode the user will only have the ability to look at memories and navigating forwards and backwards through these, or go to a random memory. The user will have the ability to scroll through and view the media associated with each memory. The only administrative button available in resident mode will be a button that enables a carer to disengage resident mode and return to the main carer mode.

Navigation bar

Excluding the Log in screen, there is a navigation bar placed above every screen. This navigation bar is designed to be an extension of the menu bar already present on the device and contains a prominent chronological back button9 and a forward button.

It is thought that the intended users have some experience in using the Internet in traditional web browsers. The back button in such application is

---

9A chronological back button is a button whose action takes you back to the previously displayed screen. Contrasting the chronological back button is a hierarchical back button, a button whose action brings you back to the level above the one displayed. A Memory’s parent is an overview of the memories, whose parent in turn is the patient profile page.
always chronological, and it is presumed that the user will expect our back and forward buttons to do the same.

**Administrative screens**

In addition to the screens mentioned above, the need for an administrative user interface was deemed necessary. This due to two main factors: there are some actions that, for safety reasons, are not suitable to be carried out by everyone in the system; and to further limit the number of actions to reduce the application’s complexity. The administrative panel will only be accessible for users registered as administrator.

In this interface an administrator can add and remove residents, add, remove and administer user rights of a user, as well as a more streamlined interface for mass removal of data.

The creation and deletion of users has been hidden away in this interface, and can only be achieved by a system administrator. The creation and deletion of a resident entity in the application is thought to be an action that do not happen so often to warrant the need for this action to be too prominent in Life History. It has therefore been placed on an administrative interface in the application, to limit the complexity and to limit the possibility for a user to perform an unintended action.

The administrative screens are to be used by trained personnel, and thus these screens are not designed with extreme usability in mind. This is not to say that they are impossible to use, but rather that more care has been taken designing the previously mentioned screens than the administrative screens. As mentioned above, tasks that it has been deemed that

The administrative interface is accessible from the navigation bar, but only for logged in users with administrative rights.

### 8.5 Security

Security is always an issue when it comes to designing web applications. Web applications are relatively easy to attack, as they are simple to understand and manipulate. It is trivial to intercept data passed through the server and the client, and thus gain insight into how the application is built. This can in turn be used to spot exploit weaknesses in the application itself and spot and exploit known weaknesses in the application framework (Ruby on Rails).

In Ruby on Rails by default helper methods to prevent SQL injection\(^\text{10}\) and Cross-Site Request Forgery\(^\text{11}\), so these threats are considered mitigated. The biggest threats to Life History are considered to be user account hijacking, and bypassing the access control itself.

For authorising and authenticating there has been built a log in system that requires a username and password. The password is then stored as a salted hash\(^\text{12}\) to both prevent brute force dictionary attacks, and to limit the information someone can gather from a direct view into what’s stored in the

\(^{10}\)Entering SQL code syntax in the place of input field data to force the SQL server to run unauthorised queries.

\(^{11}\)Injection of malicious code.

\(^{12}\)Salting a password is the act of attaching a random value to a password for added security.
database. By not storing the passwords as plain text\textsuperscript{13} an intruder would be none the wiser in the case of a partial data break, as the intruder would then only see an undecipherable hash value. A good hash function makes an unique hash value for every input, and the initial string is impossible\textsuperscript{14}.

The biggest concern is the user itself compromising his/her own account by actions such as forgetting to log out, choosing a password which is easy to guess or writing down the password and store this somewhere insecure. The two latter ones are not something that can be mitigated by system design.

### 8.5.1 OWASP Top 10

The Open Web Application Security Project (OWASP) each year compiles a Top 10 list of the most exploited web application weaknesses\textsuperscript{[23]}. This list has been used in some extent to secure Life History, but seeing as the application is a small-scale application with only a limited amount of personal data, choices have been made on what of these mitigation strategies have been implemented.

Out of the OWASP Top 10, protections against Cross-Site Request Forgery (CSRF), Injection and Insecure Cryptographic Storage has been put in place, which was chosen due to the relative ease of mitigation and relative impact on such a small scale web application Life History is. Future improvements of Life History could mitigate more of these weaknesses.

\textsuperscript{13}Instead of storing the password \texttt{banana} as the text string \texttt{banana} in the database, it would be stored as \texttt{72b302bf297a228a75730123efef7c41}.

\textsuperscript{14}Disclaimer: Nothing is impossible, but rather either improbable or extremely inconvenient (as in taking years to decipher).
Chapter 9

Testing

This chapter will detail the testing environment that has been decided upon, as well as detailing what tests that will constitute the testing strategy, including system testing and usability testing.

9.1 Testing Environment

Today, users have a tremendous choice when it comes to web browsers, with the major browsers being Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Apple Safari and Opera. Each of these are based on their own browser engines, except for Chrome and Safari, that share the Webkit browser engine. There is no way of knowing for sure whether a web page will display consistently on every browser, without testing said web page on each browser for browser specific bugs. When developing an application that is to be run inside a web browser, great care must be taken to ensure that it runs well on all target browsers.

As this project focuses on making Life History run on 10" tablet devices, the testing of Life History has been limited to such a device: the Apple iPad. The iPad uses Mobile Safari, which again uses the Webkit browser engine.

It was shown early on in the course of the project that even simple applications can look distinctly different on the iPad screen compared to a desktop computer screen. This is due to the fact that Apple applies a few custom WebKit tweaks to make a web page more aesthetically pleasing on smaller screens, as well as making web pages more consistent with each other. While this certainly can make a web page prettier, it makes for some unpredictability when programming a web application for the iPad on a desktop computer. A specific example of this was found early in the project, namely the rendering of buttons on web pages. While the preferred visual style was quickly achieved, the issue was only discovered by testing the views on a iOS device. This adds to the certainty that thorough testing is crucial to ensure that the application performs as expected.
9.2 Test plan

9.2.1 System testing

System testing is an important part of the testing process, wherein the application is tested as a whole through so-called “Black box testing”. An important aspect of the system test is that the test person should not have to know anything about the technical parts of the system (as if it was hidden inside a black box) in order to be able to perform the tests. System tests are highly connected to the functional requirements, and should be designed to make sure these are being met in an efficient and correct manner. Below are the system tests that have been devised to test the functional requirements, and other important parts of the system:

<table>
<thead>
<tr>
<th>Administrative</th>
<th>Navigation bar</th>
<th>Ok</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Test 39: Enter admin panel</td>
<td>• Test 44: Back button functional</td>
<td></td>
</tr>
<tr>
<td>• Test 40: Create user</td>
<td>• Test 45: Forward button functional</td>
<td></td>
</tr>
<tr>
<td>• Test 40: Delete user</td>
<td>• Test 46: Breadcrumbs from resident to resident overview</td>
<td></td>
</tr>
<tr>
<td>• Test 41: Assign user rights when created</td>
<td>• Test 47: Breadcrumbs from memory overview to resident overview</td>
<td></td>
</tr>
<tr>
<td>• Test 42: Assign residents to a user</td>
<td>• Test 48: Breadcrumbs from memory to resident overview</td>
<td></td>
</tr>
<tr>
<td>• Test 43: Un-assign residents from a user</td>
<td>• Test 49: Breadcrumbs from family and friends overview to resident overview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 50: Breadcrumbs from memory overview to resident</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 51: Breadcrumbs from memory to resident</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 52: Breadcrumbs from family and friends overview to resident</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 53: Breadcrumbs from memory to Memory overview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 54: Breadcrumbs from resident to resident overview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 55: Breadcrumbs from Family or friend to Family and friends overview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 56: Breadcrumbs from Family or friend to Resident</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test 57: Breadcrumbs from Family or friend to Resident overview</td>
<td></td>
</tr>
</tbody>
</table>

Table 9.2: System testing (part 2)
9.2.2 Usability testing

Since the system mostly will be used by people with limited technical experience, the usability is very important. The Usability testing will be part of the ”black box” system testing.

9.3 Results

In this section the results of the usability testing will be discussed.

9.3.1 What was used during testing

In preparation of the test, several scenarios based on the system test in the preceding section were devised. These scenarios were made to be presented to the test subjects as instructions, ideally without any help from the test observers. These scenarios can be found in the table below 9.3.1. A questionnaire was also prepared in order to get feedback about the test procedure itself from the test person. The questionnaire is presented below.

Both an iPad and a PC were prepared for the test. Ideally, the test would take place on an iPad, but seeing as an iPad requires a persistent Wi-Fi connection, a backup PC was prepared as well. The PC, unlike the iPad, has the advantage of having the Life History application stored and run locally without an Internet connection.

There were discussions on whether or not to translate the application from English to Norwegian for the duration of the testing, to limit the impact of the test subject’s mother tongue not being English. This was decided against, due to the extra time needed for translation.
9.3.2 How the testing was performed

The testing was performed with three test observers and one test subject at a time. The test person was in all cases a person employed within the health
sector. While one test observer gave the test person instructions, the others would observe and write down the progress and feedback of the test person. In some cases, the test persons would run the application on a PC instead of the intended iPad due to the lack of an accessible Wi-Fi connection.

9.3.3 Testing

The test subjects chosen were people with average to limited computer literacy, and hence within the target audience. Despite this, the subjects seemed to understand the metaphor and appearance of the application quite well.

By observing the test subjects, many useful observations were made. It was very obvious that the actions needed to choose a resident, browse his or her profile and access memories or family and friends were simple and intuitive. In some of the subjects a slight hesitation was noticed, but it became clear that this was due to language restrictions rather than the user interface.

The main issue with navigation was discovered to be the breadcrumbs. When the subjects were asked to go back to the resident or memory overview, all of the test subjects chose to return by pushing the back button several, which caused some problems when updating information about residents. The breadcrumbs tool was not at all discovered by the subjects until they were told about it. After a short explanation however, it was easily understood and used.

Regarding the process of adding and editing information, a few more issues were discovered. When trying to edit a resident profile, more specifically editing the resident name, clicking the menu button to find an edit function was not as intuitively obvious as intended. Instead of using the menu button at the top of the page, the test subjects looked for ways of editing the information they saw directly, i.e by clicking on the name itself.

Once the test subject were pointed to the edit page, they understood how it worked quite intuitively, with only minor hesitations.

In general it was observed that the subjects were unsure at first, but ultimately managed to understand button navigation and placement of information. The major issue with the GUI was clearly the menu button. The pages for editing residents proved quite intuitive except for some minor trial and error, while the pages for editing relations and memories proved to be even easier. Once the test subject had been shown once how the breadcrumbs and various other functions worked, they used the correct functions with ease.

The testing showed that Life History seems to be a usable application, also for a less computer literate audience.

9.3.4 Feedback received

As mentioned above, a questionnaire was used to get feedback from the test subjects. Below you can see the result average among those who answered it. A translated version of the questionnaire can be found in the appendix.

- **Question 1:** 4
- **Question 2:** 1,5
- **Question 3:** 2

80
• Question 4: 5
• Question 5: 2
• Question 6: 2,5
• Question 7: 4,5
• Question 8: 1,5
• Question 9: 3,5
• Question 10: 2,5

Comments: Some of the test subjects explained their low score on question 3 and 6 as related specifically to the breadcrumbs, as opposed to the system as a whole, and made it clear that they felt the breadcrumbs as well were immediately intuitive after being directed to them once.

9.3.5 Conclusion

It was clear that not all of the functions was immediately intuitive to the test subjects. They did, however, manage to repeat operations with much more ease after having had the functions explained briefly. From this it can be concluded that users might need a short demonstration of the system in order to ensure that the users are comfortable using the software and don’t feel threatened by something unfamiliar.

9.4 Hardware testing

Testing on the first generation iPad has shown that Life History unfortunately is slow and feel sluggish on modest hardware. The second generation hardware, however, is delivered with a better processor and, more importantly, more memory. This hardware buff has proven to be very beneficial for running Life History.

With more development, speeding up and optimising the application further is thought to be possible. Life History runs fine on desktop computer browsers.
<table>
<thead>
<tr>
<th>Requirements</th>
<th>Tests</th>
<th>Result</th>
</tr>
</thead>
</table>
| Log in and Log out      | • Test 1: Log in  
• Test 2: Log out                                              | Ok     |
| Handle resident         | • Test 3: Create new resident  
• Test 4: Edit a residents picture  
• Test 5: Edit personal information  
• Test 6: Edit a residents name  
• Test 7: Remove resident  
• Test 8: View Resident  
• Test 9: Display all residents  
• Test 10: Display your residents  
• Test 14: Scroll through residents | Ok     |
| Handle Categorisation   | • Test 11: Display all memories  
• Test 12: Display memories of a specific category  
• Test 13: Scroll visible memories  
• Test 15: Scroll visible categories | Ok     |
| Handle memories         | • Test 17: Create memory  
• Test 18: Display memory  
• Test 19: Navigate between memories  
• Test 20: Delete memory  
• Test 21: Edit fields in a memory  
• Test 38: Scroll information for a memory | Ok     |
| Handle media            | • Test 22: Add media to memory  
• Test 23: Remove media from a memory  
• Test 24: Display media in memory  
• Test 25: Navigate through different medias within a memory  
• Test 26: Maximise picture when clicking on it | Ok     |
| Resident mode           | • Test 27: Activate Resident mode  
• Test 28: Navigate through randomly through memories  
• Test 29: Exit resident mode  
• Test 30: Same navigation possibilities as in memory | Ok     |
| Family and friends      | • Test 31: Create new family or friend  
• Test 32: View all family and friends  
• Test 33: Scroll through family and friends  
• Test 34: Delete family or friend  
• Test 35: Edit Family or friend  
• Test 36: Display a family or friend  
• Test 37: Scroll information for family or friend. | Ok     |

Table 9.1: System testing (part 1)
<table>
<thead>
<tr>
<th>Scenario</th>
<th>#1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Find resident and change resident name</td>
</tr>
</tbody>
</table>
| Instructions | • Go to resident Ola Nordmann  
  • Change resident name to Ola Johan Nordmann  
  • Save changes  
  • Go back to resident overview |

<table>
<thead>
<tr>
<th>Scenario</th>
<th>#2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Find and edit memory for resident</td>
</tr>
</tbody>
</table>
| Instructions | • Go to resident Ola Johan Nordmann  
  • Go to the resident’s memory overview  
  • Find the holiday memory where the resident went to Spain the summer of 1995  
  • Change the memory to take place in Greece instead  
  • Save changes  
  • Go back to memory overview  
  • Find all memories  
  • Go back to resident overview |

<table>
<thead>
<tr>
<th>Scenario</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Add new resident</td>
</tr>
</tbody>
</table>
| Instructions | • Click the menu and add a new resident called Kari Nordmann  
  • Add information to the new resident (date of birth, place of birth, profession and spouse)  
  • Save information  
  • Go back to resident overview |

<table>
<thead>
<tr>
<th>Scenario</th>
<th>#4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Add a family member to resident</td>
</tr>
</tbody>
</table>
| Instructions | • Go to resident Kari Nordmann  
  • Go to overview of family and friends  
  • Add information for Espen Nordmann, Kari’s son (date of birth, wife Sofie, children Terje and Silje, Lives in Harstad and work as a project leader)  
  • Save information  
  • Go back to resident overview |

Table 9.3: Test scenarios
Chapter 10

Evaluation

This chapter will evaluate the quality of the final application and whether it meets the functional and non-functional requirements set in chapter 6. The test results in chapter 9 will also be taken in consideration and reflected upon. This chapter will also evaluate the project process itself.

10.1 The final product

10.1.1 Requirements met

Functional

The functional requirements that has been successfully met are listed in table 10.1.1.
<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1</td>
<td>View personal details about a resident, e.g profile picture, name, date of birth, work history, education history, hobbies, likes/dislikes.</td>
<td>A resident profile page has been created with the option to add or edit facts consisting of both the label (such as &quot;Date of birth&quot;) and the data-entry (for instance &quot;01.01.1901&quot;), enabling users to freely customise the information shown.</td>
</tr>
<tr>
<td>FR2</td>
<td>Manage personal details about a resident, e.g profile picture, name, date of birth, work history, education history, hobbies, likes/dislikes.</td>
<td>Personal details, or facts, about the resident can easily added, edited or removed, and the resident profile can be accessed through &quot;Menu - Edit&quot; at the resident profile page.</td>
</tr>
<tr>
<td>FR3</td>
<td>View resident’s Life History, including text and multi-media artefacts such as pictures, video, sound clips etc.</td>
<td>A resident’s life history is implemented through memories containing a title, time/date, text and media. Navigation between memories and between pictures added to a memory is implemented.</td>
</tr>
<tr>
<td>FR4</td>
<td>Administer residents Life History, including text and pieces of media such as pictures, video, sound clips etc.</td>
<td>A resident’s life history can be recorded by adding memories. Several pictures can be added to a memory, but other media such as video and sound clips are currently not supported. This takes a relatively small effort to implement, but were not prioritised.</td>
</tr>
<tr>
<td>FR7</td>
<td>Resident mode where the resident is allowed to view his/hers own life history, either through navigating a very simple user interface, or through random display of media.</td>
<td>In resident mode only navigation between memories is allowed.</td>
</tr>
<tr>
<td>FR8</td>
<td>Administer residents.</td>
<td>Residents and other users can be administered through the admin panel. Functionality has been prioritised over aesthetics in the admin panel.</td>
</tr>
<tr>
<td>FR9</td>
<td>View information about residents family.</td>
<td>It is possible to view family members and friends associated to a resident.</td>
</tr>
<tr>
<td>FR10</td>
<td>Administer information about residents family.</td>
<td>Family members and friends only have three attributes; name, picture and a description of how the person is associated to the resident.</td>
</tr>
</tbody>
</table>

Table 10.1: Functional requirements met.
FR5
View generic content related to the resident’s age and interests.

This requirement was considered much too comprehensive and time-consuming by the team members, and therefore not prioritised. As a result, it was not implemented.

FR6
Administer generic content related to the resident’s age and interests.

Same reason for incompletion as FR5.

FR7
Resident mode where the resident is allowed to view his/her own life history, either through navigating a very simple user interface, or through random display of media.

"Slide show"-functionality has not been implemented, but it is possible to retrieve a random memory by pressing the "random memory" button. Lack of time and low priority is the reason why this requirement was not fully met.

Table 10.2: Functional requirements not met.

Non-functional

The application has been designed with user-friendliness in mind. Through testing and constant re-evaluation of the design, the system has been made quite user friendly and easy to use. This view is supported by the test results, except for when it comes to some of the admin pages, that has been proven somewhat less user friendly. When it comes to the question of local vs external storage, both options are open. Since the application has been implemented as a web application, the application could be deployed on both external and local servers.

10.1.2 Requirements not met

Only two functional requirements have not been fully met, and one has only partially been met. The primary reason why these and other features were not included was because of the time constraint. Which function would most likely be used and how much they would be used compared with how much time it would take to implement them was also an important consideration. There is however, nothing preventing such features to be added in the future if one wished to expand the application’s functions. Table 10.1.2 shows the requirements and why they were not implemented.

10.1.3 Quality and test result analysis

As mentioned, the test results from the usability testing required some re-evaluation of the application. The test-subjects had some difficulties using the resident- and category-filters, as they did not receive sufficient feedback on which button had been pressed last. In order to make this more apparent,
we changed the visual appearance of the selected button in the resident- and memory-overview.

The overall design has been up to evaluation several times, and now fits the metaphor. The design is also considered to be relatively easy to use and understand, which was to some extent proven during the usability testing. The test-subject had some trouble navigating between certain pages at the beginning of the testing session, but after a quick demonstration had little difficulties managing on their own. Training or demonstration of certain use of the application should be considered for future users of this program.

Using Ruby on Rails and conforming to the MVC-architecture has resulted in a generally well structured and modular application. Most of the applications functionality has been commented, and is included in the source code, which makes further development easier.

As mentioned in section 9.4, the application may feel a bit slow, or "sluggish", at times on the first generation iPad. This is however not an issue on the second generation of iPad.

10.1.4 Future improvements

The following suggestions are some of the features that could be implemented in the future to improve the application:

Generic content In the course of the application development there has, unfortunately, been certain features the application should have had that was left out for one reason or another. One such feature was a way to browse through generic media relevant to the resident’s age. For example; someone who was born prior to World War II would most likely have memories related to this event. Having a repository of World War II in this generic media aggregate might be considered interesting to anyone who experienced this event, especially if they don’t have any memories of their own during this particular event. Another considered feature that didn’t make it to the final product was a search function that would allow the user to search for a specific memory through keywords.

Resident/Family and friends user The thought behind this was to make a user that family and friends can use to log in and add memories to the resident. This would relieve the carers of some of the work of having to add content, and if the application is deployed on the web, the residents family and friends could easily do this from their home. This user would only have access to one resident, and would not be able to access information about other residents. The logic for this has been added to the system, however there are some security issues with this, so we decided to not add it as a future.

Resident/Memory search If the number of residents, or memories added to a resident, grows very large, implementing a search field would be very beneficial.

Video/audio Uploading video and audio works fine at the current state of the application, but the memory displayer currently only supports pictures. Expanding this to support video and audio should not pose a challenge.
Voice input/recognition By using voice input/recognition, content could be added more efficiently. Stories and information may be narrated by the residents and inserted (with the assistance of the carer) in the respective text input fields.

Media added at the same page as memory Currently there is one page where the memory information is filled out, and a different page to add media to the given memory. This could be done on the same page, which would make it easier for the user to understand and use.

Slideshow The resident mode can be extended to display the memories as a slideshow. Since a resident most likely has a limited technical understanding, this feature will increase usability.

Auto logout For good security measures, users should automatically get logged out if no action has been performed during a certain time interval.

10.2 The process

10.2.1 Progress

Generally, the process went quite well. The group met to work together every week day, except Mondays and Thursdays, which normally were used to work on other courses, except for the weeks close to report deliveries.

The group focused a lot on making the application as user friendly as possible. This was reflected in the process by the planning stage at the start of the project being rather lengthy. In this period a lot of sketches was produced to capture the general feel of the application-to-be, and the navigation between the different elements that would constitute the application. In hindsight, the planning stage might have been too long, and too substantial.

Seeing as the entire group had to learn a new programming language, there was a lot of time set to learning Ruby on Rails through self study. At the end of the project period, the experience of this is that this time was not spent efficient enough. Higher demands on what people should learn per day could have been very beneficial.

Contact with the customer was had rather infrequently during the project period, however, without either the group or the customer himself feeling that the meetings were to too few and far apart. The group and the customer came to an agreement that there was no need for unnecessary meetings, so the group contacted the customer if there were questions. Also, in order to update the customer, a copy was sent the customer at every report delivery, for review and feedback, something that the group felt was very helpful.

The application was available for the customer to view during the last third of the project period, which let the customer try out the application and give us feedback.

Meetings with the supervisor were had approximately once every sprint, and the group received good information on how to write and structure the report, as well as giving great feedback on the project progress. A status report (see appendix E for an example of a status report) mailed to the supervisor one day prior to each supervisor meeting served as a meeting agenda.
Communication within the group happened over Skype and by meeting in person. A mailing-list was created, but only used when contacting the supervisor and the customer. This worked very well, and all of the group members were kept informed at all times. There were, of course, occasions of bad communication, but no more than one would expect with any other method of communication.

10.2.2 Management

The group roles that were decided upon early in the project (see section 5.1.1) were not used very much in practice. As the group consists of students, there wasn’t really a strong “group leader”, but rather a flat group structure even though we had assigned the role of Scrum master to one of the team members. The only role that really persisted through the whole project was the customer contact, to prevent that the customer would have to deal with different people every time he was contacted. The risk list was changed only once during the project, to increase the likelihood of illness. This was because the group struggled a lot early on with nearly all group members being sick within a short time interval.

As this project required a lot of report writing, the group quickly ended up with a structure of four people focusing on writing code, and two people focusing more on making diagrams and writing the report. While this approach resulted in the work on the report happening in a steady fashion, this also resulted in these two people not participating in programming Life History as much as the rest of the group and vice versa.

Seeing as the group as a whole did not have opportunity to meet at the same time every day, and the meeting place differed from day to day, the idea of a Scrum-board was not implemented. The daily meetings also ceased to take place after a few weeks because group members had other classes to attend. In the early reports we needed to provide several diagrams, which pushed the group out of the Scrum model for some time, and it took a long time for the first proper prototype to be produced. After the mid-term report was delivered, the process became a lot more agile.

10.2.3 Summary

Almost all of the functional requirements have been met, and the team members are confident that the functional prototype implemented will satisfy the customers needs. The feedback from the customer has been positive, and it seems that Life History will be considered for use in nursing homes, possibly with some alterations and further development. As a team, the members have been working well together. Naturally, some of the team members have been working more and a some less than others, but this is mainly due to differences in the amounts of external studying and work obligations. However, in general, the work has been fairly well balanced between group members.

The task itself has been very interesting to work with, and the group feels that the customer and thus the requirements have been very easy to work with. All in all a great experience.
Bibliography


[18] London Made. Crummy - Tasty breadcrumbs!


Appendix A

Dementia

Dementia is a general term describing a group of disorders in which memory and thought processes (cognition) become impaired for a period of at least 6 months. Unlike mental retardation, dementia involves a change in thinking abilities relative to baseline.

Dementia can be caused by about 50 different disorders, but 50% to 70% of cases are caused by Alzheimer’s disease, and 20% to 30% by vascular disease. In many forms of dementia, such as Alzheimer’s disease, Lewy body disease, and other neurodegenerative disorders, symptoms develop slowly, are relatively stable rather than fluctuating, and continue into a slow decline. However, other forms of dementia, such as vascular dementia associated with small strokes (multi-infarct dementia) may begin abruptly and worsen in stepwise fashion, with relative stability between each decline. In dementia secondary to head trauma or encephalitis, memory problems are worst at the outset, and remain relatively stable or may even improve with time.

Impaired memory is a prominent and early symptom of dementia. New skills and knowledge are difficult to learn, while old skills and knowledge are eventually lost. Valuables may be lost, such as a wallet or keys. The person may become lost, even in familiar surroundings. Late in dementia, individuals may forget their occupation, family members, or even their name. Other symptoms are difficulty naming objects or people (anomia), rambling speech, difficulty performing certain activities (apraxia), or failure to recognise certain objects (agnosia). Executive functions, such as thinking abstractly, planning, and initiating complex activities, can be impaired. Poor judgment and insight are common. The individual usually has little or no awareness of memory loss or other abnormalities. Individuals have an unrealistic view of their capabilities or
their future. For example, they may talk of starting a business or driving. There can be mood and sleep disturbances. False beliefs (delusions) are common, especially paranoid delusions involving others stealing from them or conspiring against them. Individuals with dementia may have further deterioration of cognitive abilities with stress, either physical stress such as a viral illness or minor surgery, or psychological stress such as bereavement.

Source: http://www.mdguidelines.com/dementia/definition
Appendix B

User Manual

The following pages contain the software user guide. This will normally be available through “help” on the software home page.

B.1 About

*Life History* is a web based application to help carers working with patients suffering with dementia. Using the application, the user will be able to browse through residents, display memories, containing photographs and other media, and view the resident’s family and friends.

B.2 Software requirements

*Life History* has been developed primarily for use on the Apple iPad, with Safari web browser. Some web browsers may not support it.

B.3 Disclaimer

The content on the *Life History* application is the sole responsibility of its users and the users will take full responsibility, liability, and blame for any libel or litigation that results from content added in or as a direct result of content added.

B.4 Installation

See the installation guide (C) for direction on how to get *Life History* running on your server and device.

B.5 The different users

In the system, there are 3 different roles a user can have

- *Administrator* - Has all privileges.
• *Carer* - Has most privileges, but can not add a new resident or assign residents to users.

• *Family and friends of the resident* - Can only view as well as edit the specific resident they are related to.

### B.6 Getting started

#### B.6.1 Logging into the system

In order to log into the system, you must have a valid account.

- *Type in your username.*
- *Type in your password.*

You are now logged into the system.

#### B.6.2 Logging out of the system

- *Click the menu button in the top right screen.*
- *Click on "log out"*

A message will show up, asking you if you for confirmation.

- *Click "Yes" to proceed with the log out, or click "No" to cancel the log out session.*

You are now logged out of the system.

#### B.6.3 The navigation bar

The navigation bar is located at the topmost part of the screen and will follow you through all the different pages, except for the administrative screens. Here you can find the following:

- *Back and Forward button* - These buttons are used to browse through memories (in the same way you would do on a web browser)*

- *Breadcrumbs* - The breadcrumbs will aid you by showing where in the program you are, and enables you to navigate back to where you were previously. These work as individual links that takes you to whichever step you click.

- **Menu** - After clicking the menu, you will find several options, depending on which page you are on. This is where you will typically go if you wish to edit or change the current page in some way.
B.7 Residents

B.7.1 Creating a new resident

In order to create a new resident, you must have administrator privileges. First, log into your administrator account on the home page:

- Click "Menu"
- Click "New resident"

A new page will be displayed in your browser, with several fields available for editing. The following field needs to be filled in order to create the resident.

- Name

If you want to upload a picture for the resident:

- Click the "Select file" button
- Select the photograph you wish to use as profile picture

The image selected will now be saved in the resident profile when you are finished creating the resident. Next, you have the option of filling out an unlimited amount of Personal information in the fields on the right side of the page. The first field is the description of the field, and the second (longer) field is the actual information. In order to add more fields like this do the following.

- Click the "Add more facts" button

You will then see another two fields appearing below the one that is currently there. To remove one of these personal information fields.

- Click the "Remove" button below the fact you wish to remove.

After you have finished filling out all the fields:

- click the "Create new resident" button at the bottom of the screen.

A new resident is now created, and is visible next to all the other residents.

B.7.2 Delete resident

First you need to be logged into the application with administrative privileges.

- Click the "Menu button" in the top right side
- Click the "Admin panel" button.

A new page will appear in your browser, which lists all users and all residents.

- Click the "delete" button next to the resident you wish to delete.

The resident will be deleted from the application.
B.7.3 Display an existing resident

In order to display an existing resident, you must have the privileges to view that certain resident, and be logged into the system. After you have logged in, you will see all residents you have access to. In order to view one specific resident:

- **Swipe your fingers to the left, or to the right, until you find the resident you wish to view.**
- **Click the resident you wish to view.**

A new page will now show up in your browser, which shows the resident, with a name and picture to the left, and whichever personal information that was written about the resident to the right.

B.7.4 Edit an existing resident

In order to edit an existing resident, you must be logged on to the system. First, you need to display the resident you wish to edit (see previous section).

- **Click the "Menu" button on the top right side**
- **Click "Edit page"**

A new page will show in your browser, which looks identical to the "New resident" page where you create a new resident, only with the existing data already filled in. At this page you can edit what you wish to change. When you are done:

- **Click the "Update resident" button, at the bottom of the screen.**

The information should now be changed and you will be forwarded to the resident profile where you can see the changes.

B.8 Memories

B.8.1 Creating a new memory

In order to create a new memory, you need to be logged on to the system, and you must have access to the resident you wish to add a memory to.

- **Click the resident you wish to add a new memory to**
- **Click the "Memories" button**
- **Click the "Menu" button**
- **Click "New memory"**

A new page will appear in your browser, with several fields that you can fill out. These include "Title", "Time/Date", "description text" and categories.

The title is the only mandatory field. The Time/Date field is meant to describe when this memory took place. It could be a specific date or a general period, e.g. "childhood". The description text is the field where you fill in the primary
textual content of the memory.. In order to find your memory more easily when searching through memories in the overview section, it would be prudent to add categories to it. You can add as many categories as you want to a memory. If you want to upload a picture or media file to this memory, you will need to proceed:

- Click the "Create" button
- Click the "Select file" button
- Select the photograph you wish to choose.
- Optional: write a description of the medium in the field below.
- Click on the "Add new medium" button

The image selected will now be displayed on the right side of the page and will be saved with the memory when you are finished creating it. After you have finished adding media files:

- Click the "Save" button at the bottom of the screen.

A new memory is now created, and is visible in the memory overview.

B.8.2 Display an existing memory

In order to view an existing memory, you must have the privileges to view that certain resident which the memory belongs to, and be logged into the system. From the main screen:

- Swipe your fingers to the right, or to the left, until you find the resident you wish to view.
- Click the resident whose memory you wish to display.
- Click "Memories".
- Click the memory you wish to display.

The memory you have selected, should now be displayed in your browser.

B.8.3 Browse through Photographs within a memory

While you are displaying a memory, you will see that under the picture, there are two arrows. These are used for browsing through the added photographs, within a memory

- Click the right button to show next photo.
- Click the left button to show previous photograph.

B.8.4 Navigate between memories

When viewing a memory, there are two arrows located on top of the book. These are used to navigate between the memories:

- Click the arrow button on the top right side to view next memory
- Click the arrow button on the top left side to view the previous memory.
B.8.5 Delete memory
First you need to display the memory you wish to delete.

- **Click the "Menu" button in the top right side**
- **Click the "Delete memory" button.**

The memory is now removed and the application will send you to the memory overview screen.

B.8.6 Edit an existing memory
In order to edit an existing memory, you first need to display the existing memory you wish to edit (see above section).

- **Click on the "Menu" button in the top right side**
- **Click on the "Edit memory" button**

A new page will be displayed in your browser identical to the "new memory" page, but with the existing data already filled in. Here you can edit what you wish to change. When you are done:

- **Click the "Update" button, at the bottom of the screen.**
- **Click the "Save" button, at the left side of the screen.**

Adding a new photograph to an existing memory
To add new photographs or other media to an existing memory, follow the instructions in the previous section, but after you have clicked the "update" button you:

- **Click the "Select file" button**
- **Select the photograph you wish to add.**
- **Optional: write a description of the medium in the field below.**
- **Click on the "Add new medium" button**

When you have added all the photographs or media files you wish to add to the memory proceed by:

- **Click the "save" button, at the left side of the screen**

New photograph(s) or media files have now been added to the existing memory.

B.8.7 Enter resident mode
First you need to Display an existing memory (see above section). From there you have to:

- **Click the "Menu" button in the top right corner**
- **Choose "Resident mode"**
The resident can now browse back and forth through the memories without being able to access other parts of the application. A button for browsing through memories at random will also be displayed in the navigation bar while in resident mode. All navigation in the navigation bar will be disabled, with the exception for an "Exit resident mode" link in the Menu, which will take you back to the normal memory browser.

B.9 Family and friends

B.9.1 Create new person in "Family and Friends"

First you need to display an existing resident (see above section). From there you have to:

- Click the "Family and friends" button.

A new window will be shown in your browser, showing all existing people that have been added to the resident’s family and friends.

- Click the "Menu" button in the top right corner.
- Click "New relation"

A new page will show up in your browser, with several fields that you can fill out, including "Name", "Relation" and "Description". The "Name" is the only mandatory field. "Relation" is a short description of what this person is to the resident, for example an "Uncle" or "Childhood friend". If you want to upload a picture to this person, then:

- Click on the "Select file" button
- Select the photograph you wish to choose.

The image selected will now be saved in the resident’s profile when you are finished creating the new relation. After you have finished filling out all the fields:

- click the "Create" button at the bottom of the screen.

A new person is now created, and is visible in the "Family and friends" overview.

B.9.2 Display an existing family or friend

First you need to display an existing resident (see section above). From there you have to:

- Click the "Family and friends" button.
- Click on the field where the person you wish to display is shown.

The browser will now show the profile of the person whose field was clicked on.
B.9.3 Delete family member/friend

First you need to display the person you wish to delete from family and friends. Then:

- Click the "Menu button" in the top right side
- Click the "Delete relation" button.

The memory is now removed and the application will send you to the memory overview screen.

B.9.4 Edit an existing family or friend

First you need to view the person profile you wish to edit. From there you have to:

- Click the "Menu" button in the top right side
- Click the "Edit" button

A new page will show in your browser identical to the "new relations" page, only with the existing data already filled out. Here you can edit what you wish to change. When you are done:

- Click the "Update" button, at the bottom of the screen.

B.10 Administrative functions

B.10.1 Create new user

First you need to be logged into the application with administrative privileges.

- Click the "Menu" button in the top right side
- Click the "Admin panel" button.

A new page will appear in your browser which lists all users and all residents.

- Click the "New user" button, below the list of users.

A new page with three fields will appear. Fill in the username of the new user and password in the following two fields. You also need to assign privileges to the user ("Role"), which can be either "Administrator", "Carer" or "Resident". An Administrator has all privileges, a Carer has all rights except creating and deleting users as well as deleting residents. The "Resident" role only has the right to display the one resident it is connected to, as well as adding and deleting memories, family and friends and editing the profile of the resident. When you are finished:

- Click the "Create User" button.

The new user should now be operable.
B.10.2 Delete a user

First you need to be logged into the application with administrative privileges.

- Click the "Menu" button in the top right side
- Click the "Admin panel" button.

A new page will appear in your browser which lists all users and all residents.

- Click the "Delete" button next to the user you wish to delete.

The user should no longer appear in the list, and it should now be impossible to log in to the application using it’s credentials.

B.10.3 Assign resident to a user

First you need to be logged into the application.

- Click the "Menu" button in the top right side
- Click the "Admin panel" button.

A new page will appear in your browser which lists all residents in two groups, those who are assigned to the user and those who are not. To move a resident to the list of residents assigned to the user do the following:

- Click the "add" button next to the resident you wish to assign.

The resident should now be displayed in the list of assigned residents and no longer appear in the list of those who are not assigned. When the user later click the "My residents" button in the resident overview page, this resident should be displayed in the list.
Appendix C

Installation guide

C.1 Install the application server

This application uses Ruby as its dynamic programming language, more specifically Ruby on Rails, which is a web application framework built for and with Ruby. To get the application running, it is important that the server from which the application is to be run has installed the relevant packages, which also includes SQLite3.

C.1.1 Install Ruby on Rails and its needed gems

The first thing you will need after you’ve copied the application to the location where you want it stored, is to get Ruby up and running. Ruby is a dynamic, open source programming language which can be downloaded for free from their official page\(^1\). Another easy way to install Ruby on Rails and the Gems Life History require to work, is to enter the commands below in a Terminal window:

1. `sudo su`
2. `apt-get install build-essential`
3. `apt-get install ruby rdoc libopenssl-ruby`
4. `wget http://production.cf.rubygems.org/rubygems/rubygems-1.3.7.tgz`
5. `tar zxvf rubygems-1.3.7.tgz`
6. `cd rubygems-1.3.7`
7. `ruby setup.rb`
8. `ln -s /usr/bin/gem1.8 /usr/local/bin/gem`
9. `gem install rails`
10. Installing MySQL gem
11. `apt-get install ruby-dev libmysql-ruby libmysqlclient-dev`
12. `gem install mysql`

C.1.2 Install Life History

Ruby on Rails comes with built-in support for SQLite3, which is a lightweight server-less database application. All you need to do after you’ve got Ruby and Ruby on Rails installed, is type the command "bundle install" to get any other support programs you might need up and running. When you want your application to start running you then type "rails server -c -p 80". Again, make sure this is done in the root directory of the application.

C.2 Run Life History

Since the application is to be accessed remotely, all you have to do in order to run it, is open a web browser and type in the server address where the application is running (default: http://localhost:3000/). Any browser new enough to be able to run HTML5 and CSS3 will do. On the apple iPad, which has been the platform the application is modelled to fit, will typically use The Safari browser. Once you’ve typed in the address it should take you directly to the log-in page and is ready to be used.
Appendix D

Storyboard
RNHA Storyboard Elements

Introduction

This Story covers a range of genuine business needs, where a gap in knowledge, or a discrepancy between the expectation of the staff and the reality, can cause or ‘trigger’ a time of reflection – from an individual’s moment of thought (‘is this how I should be doing it?’), to a collaborative management meeting to review a policy or procedure in the light of events. This picture, below, emerged after just one hour, but does contain the key elements of our needs and many of the technical solutions on offer during the Creativity Workshop.

This note expands and explains what our art was trying to convey!
RNHA Storyboard in text

We incorporated our 5 ‘business requirements’ in this story, building up from a simple personal file in digital form, to solving a complex business problem, using a combination of business data and reflection.

The ‘solutions’ highlighted and incorporated in this Storyboard are aimed primarily at staff – carers and managers of carers in Care Homes. However, where the same content can be ‘re-used’ for the benefit of Residents, this is also explored.

The solutions offered by technical partners included some duplication and functional overlap. This provided a convenient way of grouping the solutions – data capture, data manipulation, information display, and others. Many picture blocks will highlight these functions, and list the relevant technical solutions – which we assume would be ‘rationalised’ to eliminate any duplication and benefit from cooperative working between partners.

We have made no assumptions about hardware solutions. In current technology we would expect to use some form of rugged iPad for mobile staff - but do not preclude breakthroughs offering alternative technologies in the future. However, our focus is on the software and the User. Simply put – if the software is not genuinely useful, and if the output is of lesser value than the cost of inputting, then it won’t be used.

The 9 Pictures

Picture 1 – Life History / Data Capture

Social Care in a Nursing or Residential Care Home is focused on the individual. Each resident or patient is a unique individual, with their own background, experiences, preferences, and ways of reflecting.

All carers will need to know about the people they care for. However, for an average Care Home, with some 50 residents, even to learn the names (‘put a name to a face’) of all the residents is not an easy task for a new member of staff.

Two sorts of information are currently held about a resident.

1. the official records of the Care Home, usually held on paper or a Care Planning System (a database of residents), with support for basic professional tasks, e.g. updating care records, holding status reports, medication records, health records from the National Health Service, social care records from Social Services.
2. A resident’s Life History book. This is a personal album of photos of family, friends, pets, ex-colleagues, etc, with documents of significance (e.g. letters), mementoes (e.g. tickets, programmes to events). It will also hold a record about personal preferences, likes and dislikes, gleaned from the resident themselves, and family and visitors over time.

Both sorts of information are relevant to the carer.

However, currently, the first sort is difficult to access in practice, with the information held on a centralised database requiring access to a PC, or in an office filing system. The Life History Book is useful, but as a single physical object it is not always available, and can’t be taken away to study easily.

At the end of a successful MIRROR implementation, we envisage that information about all key actors – particularly the resident and the carer – can be collected and stored digitally in a variety of multimedia. The collection of information about a resident – a ‘Biog’ – could contain formal NHS and Social Services record information, personal information from the resident, friends and family, and information gained by care staff in the process of caring, who would continually update the Biog. All information that helps the staff to get to know the resident better, is relevant.

With daily updates for some 50 residents and by some 50 staff, as well as collecting and maintaining a plethora of background information, the importance of easily capturing and storing digital information in a mobile environment is clear.

And ‘Biogs’ could be developed for staff as well, to help sympathetic matching with Residents, and general ‘HR’ usage.

Once digitised, the Life Histories and Biogs may include photos, favourite music, videos, audio recordings, and other digital media, from a variety of sources.

**Picture 2 – Data Structure and Interpretation**

This second slide recognises that all this multimedia data needs storing. Formerly, this information would be stored in one very large, standardised and centralised database. Now, we expect, that the information will be held in smaller portable units with common interface standards, which will be usable and re-useable by a variety of Apps, which will combine or manipulate the data in a specific way, to be available by mobile devices.
Although our Storyboard in the picture was developed for RNHA needs, in principle, almost all of the activity painted in the pictures has general applicability – with each profession or business able to paint their own activities.

Such development work will need a business ‘template’ of requirements which will be specific to the business. In our case, the template can be provided by the social care business (RNHA), specifying the key fields of information to be collected and stored.

The information could be selected and combined in many different ways, to provide different information for different functions, which is shown in the following Pictures 3, 4, 5, 6, and 7, looking at how this functionality could be used in a Care Home.

**Picture 3 – How it is Used – By Resident & Carer**

Resident’s Look Touch and Say ‘Rummage Box’

This picture shows that different Apps could be developed to access the information collected in a Life History or Biog App, and can be accessed and presented according to the User’s need at the time.

Ultimately, it should be possible for the information to be presented (including the choice of GUI) to the Carer as a series of Carer Apps, and to the Resident as a Life History, and later, as part of their ‘Rummage Box’. The picture shows a Carer and Resident looking at part of the contents of a Resident’s Rummage Box, building on their own Life History or Biog as a core, with the choice of personal photos and documents, and also supplemented by favourite music, books, films, TV shows, and a variety of cultural images appropriate for this age, background, and gender.

The Rummage Box idea uses and extends a real therapeutic activity, reminiscence, into the digital world. It would have considerable relevance and application across all European care homes and organisations caring for the elderly.

**Picture 4 – Training**

*Caption - Experienced & New Staff Learn about the Resident*

Picture 4 focuses on a core function for Mirror Apps – training. This picture shows a simple example training session.

Using a variety of devices – from the TV screen in the room, controlled by an App on a mobile hand-held device, both the new and experienced carers can access relevant information about the Resident and their care, from their Life History or Biog records.
This is part of the new Induction process, when a range of essential formal and informal information about a Resident can be reviewed with an experienced carer.

Where relevant, access to knowledge about specialist resources can be shown according to needs identified in the review, or in supervision.

**Picture 5 – Data Visualisation/Aggregation**

**The Management/Organisational/Sector View**

This Picture tries to portray a wide variety of display options, which should be related to the use of the aggregated data, (collected by the new diaries, pens, and other sensors) such that it can be used by staff in a variety of situations, and at a number of levels of management, up to the overall sector level.

New Apps that can present complex data in a visual, easy and clear, way will be useful.

For most of management in such care organisations, a modernised ‘Executive Information System’ is needed, incorporating the simplicity of Apps, with the power of modern data processing and communication and presentation.

Once ‘information’ is extracted from the mass of data and understood, managers at all levels will want to communicate the implications of the information in an appropriate and effective way, to different categories and levels of staff.

Milestones, and parameters are also set from review of such data. From these, notifications, alerts, and reminders are set up.

**Picture 6 – Editing/Updating e.g. Shift Handover**

**Diary consultation between staff on Residents mood, health & medication, activities.**

This Picture depicts one of the crucial times of the day, when one set of carers ends a shift and a ‘handover’ is required. Although personal interaction between carers will continue to be essential, more information could be recorded easily, and be accessible for joint review.

The Picture highlights a need for easy access to data, which can be kept up to date (editable) by a mobile worker. The device and interfaces of the Apps should be User focused, offering the best way of ‘doing the task’.
Picture 7 – Reflection – Using the information

- Organisational & Collaborative Reflection, using ‘virtual facilitator’
- Serious games for group and team training.
- Individual Apps for Reflection, including personal mood-maps, and diaries.

Reflection here is seen as an enabler. It is seen as a practice which can be applied at all levels of management, and both individually and collectively. It recognises that analysed/prepared business data (which may be a twitter or a chart) can be a vital prompt for reflection, and a subsequent change of practice.

Reflection here is seen as necessary, but not sufficient. It is the process which links all the available evidence for a human assessment. It turns data into information, and presents that information in a context-sensitive way.

Such prompts for reflection can also be generated by simulation. Using real or constructed data and scenarios, a wide variety of real-life situations can be played out, and replayed, in safety.

Training guides and Apps to aid this process would be vital.

Picture 8 – Challenging Behaviour – the ‘Trigger’

Documented in the diary

This picture captures the moment which was then logged as ‘challenging behaviour’ when a Resident physically hits a carer, apparently without reason.

This behaviour is the trigger, a discrepancy between expectation and reality, which forces the carer to retreat and consider why this happened. Was it a pure random act? Was it something she said? Was it a misunderstanding? What was actually said? The carer reflects, replays the previous interaction with the Resident, and with access to their Biog, sees that this Resident is nearly blind in her left eye and hates being approached from that side.

Understanding the situation better, the carer now avoids that situation.

Picture 9 – Smiley Faces – of Residents, reflected by the Staff.

The challenging behaviour, in this case associated with approaching the Resident was avoided in future. The incident was identified, logged, and communicated to others.

Carers feel the technology is a help not a hindrance, and are open to new ideas in the future.
A Day in the Work-life of a Carer – a post-MIRROR Scenario

Joanne is a 20 year old Senior Care Assistant at The Elms, a Nursing Home for the Elderly. A pretty Irish woman, she left school at 16 and always wanted to work with people, which she finds rewarding. She recently passed her National Vocational Qualification in Health and Social Care at Level 3, and is now a senior care assistant, looking after a team of 8 care assistants.

She starts early, at 7:30am. On the bus on the way to work she checks her phone for messages, and checks the Elms Staff Bulletin Board. A resident died last night, unexpectedly. It always lowers the mood of the staff for a while. She prepares herself.

On entry to the Elms, Joanne collects her iCare device and logs on. A mobile device, the ‘iCare’ is standard issue for Care Staff. (It’s also available to Residents and used in therapeutic sessions.) The device is similar to an iPad, with a rugged case, and with a variety of GUIs, according to the type of User. She turns on her digiPen and collects her Diary, and goes to meet her colleague, Lidia, for the ‘hand-over’ session.

Using the Handover App, Lidia highlights a number of issues to be aware of – Residents who knew the dead resident well, exceptions of medication, the status of current illnesses, and staffing issues. Lidia passes over a number of notes to Joanne’s iCare, and a voice memo from a memo of staff. She listens to it and sets a Reminder. She feels in control, with access to good information and an easy handover. Fortunately all the care staff are in today.

She goes to assist in the ‘getting up’ and dressing of residents before their breakfast. She has 6 residents to help today. She’s seen their Biogs, and seen their preferences about when and how to get up.

She starts with Mr Friend. She picks up Mr Friend’s ID from the barcode on his bedframe with her digiPen before helping him up. She is keen to support the Care Home policy to maximise the time spent in direct contact with the Residents, and finds this an easy way to monitor her work. During the 17 minutes spent with Mr Friend, they talked amiably, and he told her of a visitor he was to receive tomorrow.

When Mr Friend went to Breakfast in the Dining Room, Joanne ended the contact time recording, ticked the checklist of tasks done, made a note of the visitor, and noted his mood, and her own. 35 seconds – and all logged.

Joanne finished her round slightly early, and was asked to help a new Resident, Mrs Richards. Joanne called up the Biog of Mrs Richards and quickly scanned a few notes. She was already seen as irritable by staff, and “a handful” on a note left by a Carer. Joanne tries to be as friendly as possible, but can feel her stress levels rising as her help does not seem appreciated. Joanne thinks that it will be interesting to see her stress graph later. In a new research experiment, Joanne and other Carers are wearing sensors to record their stress.
levels during the working day, (and also by some Residents, with their or their relative’s permission).

At one point, Mrs Richards asks Joanne “Have you been out for a walk with Rex?” Thinking Rex was her dog, she innocently says “yes”, believing that such an answer would calm Mrs Smith who was getting agitated. Then Mrs Richards hit her and started calling her names. Joanne retreats.

Within a minute, a manager appears, alerted by a stress sensor on Mrs Richards. Joanne explains the situation, and the manager calms down Mrs Richards. Only later, during her lunch when she was reflecting on the incident, did Joanne look more carefully at Mrs Richard’s Biog. When she searches for ‘Rex’, she finds that Rex was her first husband, who ran away with an Irish nurse ‘during the War’. She notes the ‘coincidence’ in her log, copies it to her manager and suggests monitoring Mrs Richards with female Irish nurses, and annotates the incident of ‘challenging behaviour’ with Rex in the Biog.

After her break, Joanne helps in the Induction of a new member of staff, reviewing the Residents using the Biogs on her iCare. Photos and videos of them now, and then, help to bring the resident ‘to life’ for the carer, and show the person, not just a task. The mood-maps, now a regular feature of assessment, and looking at their preferences and favourite artefacts shows the Resident’s emotional history as well as their situation.

After lunch Joanne attends a Care Home Management meeting. The Agenda included:

1. a Review of Reflection in Care Training,
2. the weekly review of the Management Information Scorecard – extracted from a range of individual Apps,
3. a review of Communications.

1. Reflection

The Care Home Manager said that ‘Reflection’ was now recognised as a necessary part of learning. In UK, it has become a mandatory element in the Health & Social Care NVQ. She said that Reflection was already being supported by a range of Reflection Apps on the iCare. The Apps were those which prompted a mental check, to monitor, fix, or improve a situation. They often used analysed data, collected from diverse sources, but when combined and manipulated, was able to produce new professional information. She noted:

- The iCare was chosen because it offered a range of specialist Apps in a Carers Library, and others in a Residents Library. The Carers Library included simple recording and communicating options. Other Apps help format, store, manipulate, and display data
extracts on request, for many different uses. It was ideal for busy mobile carers who needed to focus on interacting with people – with the intelligence in the Apps hidden, providing a simple clear and quick interface.

- Reflection Apps were now available for the group, particularly using the simulation options of a Virtual Care Home. Groups had begun using the latest emergency simulations (‘Nightmares on Elm’s Treat’) to recreate scenarios of emergencies, and simulations setting multiple demands to help prioritisation training. As a result, new procedures and checklists were being introduced.
- A range of Reflection Apps for the Individual were now available, including accessing analyses of their own stress readings, correlated with personal interaction and tasks.

2. Management Information Scorecard
The Manager took the meeting through the range of indicators available via the Apps. She displayed a number of graphs, controlled from her iCare, on the large screen on the meeting. She displayed

- The latest ‘Social interaction’ results, looking at contact times of staff with Residents, and with other Staff, with a policy objective to increase contact time with Residents.
- The results of the latest stress survey of staff, and of Residents. Experiments to pair staff and Residents identified by pattern matching were going well.
- Mood-maps for Staff and Residents over the past month, quarter, and year. The change in staff morale was noted as the new manager began to have an impact over the quarter. Research into the effect of Staff Moods on those of Residents was ongoing.
- Only overviews were shown, with individual analyses taken into supervision.

It was also noted by the group that the ‘spikes’ of stress, often identifying challenging behaviour and triggering an alert for help, were working well set at the current levels.

3. The Communication ‘Good Practice Guide’ was to be revised to take account of the multimedia options now available via the iCare – instant or recorded, voice or text, Diary, Twitter or formal documents, or leave a note on the bulletin boards. Now all is logged and sorted by categories.

Joanne finished off her workday with Residents, stimulating them via Reminiscence sessions. Changing the iCare into a ‘Resident GUI’ mode, she starts with elements from their Life Histories, and extends their interest into the events of their youth with photos and the popular music of the age – their personalised Rummage Box. After a while, for some residents, Joanne can give the iCare to the Resident, who continues to ‘surf’ their memories unaided. She is amazed at how often a favourite film or piece of music will be replayed. But
the importance and pleasure gained by giving control back to a Resident, is worth hearing ‘Singing in the Rain’ one more time!

Joanne goes home tired, but satisfied. She’s pleased she works with people. She has also entered and used data extensively in a complex information system that could only be dreamed about 20 years earlier. But she’s not really aware of it. As it should be.

Kevin Pudney & Malcolm Rose
RNHA
18th October 2010
Appendix E

Status report example
PROJECT RNHA

SUMMARY STATUS REPORT
Week 15
1 Introduction

We have tried to finish up the application views (GUI) in order to perform usability testing, as well as fixing some minor bugs. In addition, a lot of work has been done on the report, trying to make it as complete as possible before the last evaluation by our supervisor.

2 Progress summary

We still have a lot of work left to do, but our application is beginning to look like what we expected it to look like during the planning phase of the project. The report is almost done, the application views are nearly finished, but we still have some functionality left to implement.

3 Open / closed problems

Illness and unavailability

4 Planned work for next period

- Functionality
  - Menu not fully implemented yet
  - Add user model/authentication + user management view
  - View for assigning residents to carers
  - Add multiple pictures to a memory
  - View for adding categories to a memory
  - Administrator-functionality like “new” and “edit” in the views
  - Fixing fonts/layout

- Testing
  - Unit testing
  - Usability testing

- Report
  - Not finished:
    - Chapter: testing
    - Chapter: evaluation

5 Updated risks analysis

No changes
Appendix F

Miscellaneous

Figure F.1: Translated questionnaire.