

IN3405
Final Report Bachelor Project
Visual Voicemail and Greetings

Version 2

Computer Science
Faculty of Electrical Engineering, Mathematics and Computer Science



<u>Date:</u>	July 8, 2008
<u>University:</u>	Delft University of Technology
<u>Course:</u>	IN3405 Bachelor Project
<u>Company:</u>	Greetingq B.V.
<u>Authors:</u>	Nathan Bruning 1274465 Medya Amidi 1289780 Michelle Cheung 1262912
<u>Supervisor:</u>	Ir. H.J.A.M. Geers
<u>Coordinator:</u>	Ir. B.R. Sodoyer

Preface

This report is the result of a project done in the period of almost three months by N. Bruning, M. Cheung and M. Amidi at Greeting B.V. The project was intended to accomplish the bachelor Computer Science program at the TU Delft University of Technology. According to the TU Delft bachelor curriculum, each undergraduate computer science student has to do an internship within a group of at least two students to be able to obtain his/her BSc. degree.

The purpose of this program is to give students a chance to experience actual work in a company and make them ready for entering the career world.

The assignment was to create an application for mobile phones enabling people to use the Greeting service at any time and location.

Late in March, we did a pre-research for the project about what techniques and methods are needed to develop the application. We studied the different aspects of the project, trying to anticipate possible upcoming problems in the process of development, the current possibilities and so forth¹. Beginning of May 2008, we started our internship at Greeting, a start-up high-tech company located in Delft. The duration of the internship was 10 weeks.

The project was to develop a software application (visual voicemail & greetings) called “myGreeting mobile” to be written in Java Micro Edition (Java ME) for mobile phone devices.

During this period we have gained many valuable experiences. Our team work skills were enhanced, we have got familiar with the principles of working in a company and on top of that we have enjoyed working on the project and got familiar with mobile application development. All these experiences were not possible without the support from TU Delft and Greeting Company.

Willingly, we would like to thank Greeting for providing us a pleasant work environment and facilities. The ease of communication with them accelerated the decision making process. We have enjoyed working with them.

Thanks to Mr. Sodoyer for coordinating and arranging the project. And finally thanks to Mr. Geers for his support and supervision during the project. His valuable comments and critiques provided great improvement for our project regarding both technical and managerial aspects.

July 2008, Delft

Medya Amidi
Michelle Cheung
Nathan Bruning

¹ All this information is gathered in our Orientation Document, attached as appendix A.

Summary

Having launched this year a web-base service managing voicemails and personal greetings through the Internet, Greeting Company has the intention to expand its services. The company introduced its new project - the development of a visual voicemail and greetings application, as a bachelor project to the department of software engineering of TU Delft University.

We were assigned to develop a mobile application that enables users to browse through a list of voicemail messages and personal greetings; select, play and in the case of greetings record them, in a user friendly GUI environment. The product is called “myGreeting Mobile”.

According to the TU Delft curriculum the bachelor project begins in April and will last for three months. Due to workplace problems related to the Greeting Company, our internship was postponed to the beginning of May. Putting much effort in our work we finally could manage to compensate two weeks of the delay and finish the project in less than three months.

Before the project started, we focussed on the pre-research. The purpose of this phase was to get to know what kind of techniques we would meet and need during the project. When the implementation phase started, we still needed to do research for better solutions particularly for the Graphical User Interface, audio format support, and testing methods.

The design was mainly based on Greeting’s design principles². Both the Requirements and Design Document were evaluated by Greeting before starting the actual implementation. After discussing with the company, we decided to use the Model View Controller (MVC) architectural pattern to build the application. Choosing this pattern enabled us to implement different parts of the application independently.

The implementation phase began with a mock-up (GUI prototype having no actual functional implementation), which gave us a view of how the code works in the early phase. We noticed three main parts of the implementation which can be divided into phases, namely, Graphical User Interface, Voicemail and Greetings. During the implementation we faced different problems such as compatibility of different mobile devices and support of different audio formats.

Testing a Java application on small devices is different from testing on computers. There are several constraints set up by the device. It was hard to test our product thoroughly. Standard testing libraries do not work on a mobile phone and different mobile phones have different implementations of the Java environment. We applied the following tests: unit tests, functionality tests and system tests.

² See chapter 5

Although our final product did not meet all the requirements we initially agreed on, it works properly. One of the major problems is that Java ME is not a very mature platform. The main problem of Java ME is the relatively high amount of bugs in implementations. Sun provides a reference implementation, while phone manufacturers have to create their own implementations specifically optimized for the targeted mobile phone. Our recommendations for developing a mobile application contain some guidelines that we believe would have helped us and greatly simplified our process. There is a need of great attention for the platforms and support of mobile phones.

Glossary

- AMR:** Adaptive Multi-Rate, an audio data compression scheme optimized for speech coding.
- API:** A formalized set of software calls and routines that can be referenced by an application program in order to access supporting system or network services.
- CLDC:** The Connected Limited Device Configuration is a specification of a framework for Java ME applications targeted at devices with very limited resources such as pagers and mobile phones.
- Greetings:** voice greetings recorded by callee and listened by caller if the callee failed to answer his/her mobile phone.
- Greetingq** used interchangeably with greetings.
- MIDP:** Mobile Information Device Profile is a specification published for the use of Java on embedded devices such as mobile phones and PDAs. MIDP is part of the Java Platform, Micro Edition (Java ME) framework.
- MMAPI:** The Mobile Media API (MMAPI) extends the functionality of the J2ME platform by providing audio, video and other time-based multimedia support to resource constrained devices.
- MP3:** MPEG-1, Audio Layer 3, more commonly referred to as MP3, is a digital audio encoding format using a form of lossy data compression³.
- myGreeting:** the name of the web-service product.
- Platform:** describes hardware architecture or software framework (including application frameworks), that allows software to run. Typical platforms include a computer's architecture, operating system, programming languages and related runtime libraries or graphical user interface.
- RTP:** The Real-time Transport Protocol defines a standardized packet format for delivering audio and video over the Internet.
- Streaming:
Audio** is a variant of Streaming Media⁴, where the audio data is continuously transmitted on the computer network.

³ Lossy data compression is a method where compressing data and then decompressing it retrieves data that may well be different from the original, but is close enough to be useful in some way.

⁴ Streaming multimedia is multimedia that is constantly received by, and normally displayed to, the end-user while it is being delivered by the provider

Symbian OS: is an operating system, designed for mobile devices, with associated libraries, user interface frameworks and reference implementations of common tools, produced by Symbian Ltd.

Voicemail: audio messages spoken by the caller when a call is unanswered.

XML: The Extensible Markup Language is a general-purpose specification for creating custom markup languages.

Table of contents

Preface.....	2
Summary	3
Glossary	5
1. Introduction.....	9
1.1. The Company.....	9
1.2. The project.....	9
1.3. People involved	10
1.4. Description of the chapters	10
2. Project Organization.....	11
2.1. Project Plan.....	11
2.2. Project meetings.....	12
2.3. Documentation.....	13
2.4. Team work.....	13
3. Problem domain and Analysis.....	13
3.1. Problem domain	13
3.2. Analysis	13
3.2.1. <i>General analysis</i>	14
3.2.2. <i>Users</i>	14
4. Research.....	16
4.1. Pre-research	16
4.2. Requirements	16
4.3. Scenarios.....	16
4.4. During project.....	19
5. Design phase and its problems	20
5.1. Graphical User Interface.....	20
5.2. Application Logic.....	20
5.3. Data Model	21
5.4. Audio Processing	23
6. Implementation phase and its problems	24
6.1. Graphical User Interface.....	24
6.2. Application Logic.....	25
6.3. Voicemail	25
6.4. Greetings	26
7. Testing	27
7.1. Unit Testing.....	27
7.2. Functionality Testing	28
7.3. System Testing.....	28
7.4. Acceptance Testing.....	32
8. Final product	33
8.1. Requirements Satisfaction.....	34
8.2. Screenshots.....	35
8.3. Logging Functionality.....	38
8.4. Code Fragment	39
8.5. Future Plan.....	40
9. Conclusion and Recommendation	41

10.	Personal Reflection	43
10.1.	Medya Amidi	43
10.2.	Nathan Bruning	44
10.3.	Michelle Cheung	45
	References.....	47
	Attachment: task division.....	48
	Appendix A: Orientation Document.....	49
	Appendix B: Requirements Document.....	62
	Appendix C: System Design Document	84
	Appendix D: Project Plan Document.....	101
	Appendix E: Test Report.....	110

1.

1. Introduction

1.1. *The Company*

Greetingq B.V. is a high-tech start-up company, established in January 2007 and located in Yes!Delft⁵, Rotterdamseweg 145 in Delft. The company is started by three students; Arthur Tolsma (27), a technical management student, Richard Stronkman and Ruben van Eijnatten (both 24), computer science students, all studying at TU Delft University. The company started to develop a system for mobile phone users to personalize greetings⁶. The system provides users to record different greetings for different people or groups of people. Recorded greetings are coupled with phone numbers in the users contact list of his/her mobile phone. In this way the user can separate his/her business greetings from personal ones. Working almost one year on this product, the company wanted to expand its potential by adding extra features to its existing personal greeting service. One is a web-based service for managing voicemails and personal greetings and the other one is to visualize voicemails & personal greetings on the mobile devices.

The web-based service is currently being developed. The second project, the visual voicemail & greeting application, was introduced as a bachelor project to the department of software engineering of TU Delft University. After interviewing with the Greetingq staff the company decided to assign this project to us.

1.2. *The project*

Our assignment was to develop a mobile application that enables users to browse through their voicemail messages and personal greetings; select, play and in the case of greetings record them in 'any order' of their choice, in a user friendly GUI environment. The product is called "myGreetingq Mobile".

We chose this project for number of reasons. First of all none of us had experience in the field of mobile application development therefore it appeared very interesting to us. Secondly, mobile Internet technology and its related services have been greatly progressed in the past few years and are still being developed rapidly. This creates more job opportunities for one who is experienced in this field.

⁵ Yes!Delft is a 'Young Entrepreneurs Society' who inspires, stimulates, and supports students and young technologists to start their own business. See www.yesdelft.nl

⁶ See Glossary for the definition of greeting

1.3. People involved

The people and organizations involved in this project are as follows:

- Organization:
 - o Delft University of Technology:
 - *bachelor project mentor*: Ir. H.J.A.M. Geers, department of Software Engineering
 - *bachelor project coordinator*: Ir. B.R. Sodoyer, department of Software Engineering,
 - o Greeting:
 - *Principal*: Mr. A.D. Tolsma
 - *Principal*: Mr. R. Stronkman
 - *Principal/contact person*: Mr. R.van Eijnatten.

1.4. Description of the chapters

This report is presented in 8 chapters. In chapter two, Project Organization, we will explain the planning process consisting of the project plan (how we divided different tasks into phases), project meetings, documentation and finally the team work experience.

Chapter 3 analyzes the problem domain. This chapter mainly describes which user groups will use the final product and which factors are important to us to consider.

In the next three chapters (chapter 4 to 6), we explain the problems we have faced and consequently the changes we have made to our planning and the decisions we have taken during the process. Chapter 4 explains the pre-research phase and the research we had to do during the process; the difficulties that we did not have or could not predicted before starting our project and their effect on the process. The design phase is described in chapter 5 and the implementation phase is in chapter 6 with its related problems, boundaries and limitations.

Further, in chapter 7, we introduce the final product and explain, among others, whether it has realized and satisfied the requirements.

The conclusion and the recommendations are written in chapter 8. And finally, we finish the report with our personal reflections.

Note: here we tried to explain the process globally and hence have omitted details. For any detailed information you can refer to the corresponding documents in appendices attached to the end of this report.

2. Project Organization

2.1. Project Plan

According to the TU Delft curriculum the final bachelor project should started in April and would last for three months. Due to problems related to the Greetingq Company (moving and preparing suitable place to work) the start of our internship was postponed to the beginning of May. By working hard every day we could manage to compensate two weeks of the delay and finish the project in less than three months.

One of the most important factors in software development is to meet the deadline to deliver the final product - “time to the market”. Greetingq Company has planned to introduce its product to the market in September. However, according to the TU Delft curriculum our actual deadline is specified in mid July. Based on this, we planned to deliver our work in the beginning of July and spend the rest of the time to finalize our report and documentations and prepare our presentation. Despite of many unexpected incidences during the project we could successfully manage to catch our deadline. The following paragraph will give a brief explanation of our planning.

In the first two weeks of May we have finished the first version of the requirements, design and project planning documents. The design was mainly based on Greetingq’s design principles⁷. Both Requirements and Design Documents were evaluated by Greetingq before starting the actual implementation. After discussing with our contact person at Greetingq, we decided to use the Model View Controller (MVC) architectural pattern to build the application. Choosing this pattern enabled us to implement different parts of the application independently. We divided the project in four implementing phases: mock-up, voicemail, greetings, and error correction⁸. Next to these phases, we assigned one week for spare time and one and a half week for the final report. However after starting the actual implementation we faced different problems (mostly in the first phase⁹). This caused to change our planning: 1. instead of completing phases sequentially, we started implementing them in parallel. We did so to prevent potential delays in the process - since some phases needed more time than what we had expected. More importantly, we needed a mobile device to test which was not available for the first month of our internship. 2. We had planned to combine the development and testing of each phase. To test the application, we needed to communicate with the Greetingq’s server. Since Greetingq’s server was also being developed, there were still bugs in it while we wanted to use it. Therefore the major testing part was done later in the process. In figure 2.1 you find the process flow during 10 weeks of our internship.

⁷ See chapter 5

⁸ For more information please see Project Planning Document, Appendix D

⁹ Problems relating implementation phase are described in chapter 6

February	Get acquainted with the Greeting Attending Greeting pilot Administrative arrangements for the Project										
March	Get started with the pre-research Preparing Orientation Document										
April 25th	start off the project										
Tasks	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11
Planning											
Analysis											
Requirements Elicitations											
Requirements Document											
Design											
Implementation			Phase 1		Phase 2		Phase 3		Phase 4		
Testing				Unit Testing			System Testing				Acceptance Testing
Final report											

Figure 2.1: the process flow

2.2. Project meetings

During the first month of the internship we were working at the Civil Engineering Faculty where Greeting had temporary arranged us to work. During that period we were having lunch everyday at YES Delft and had a chance to discuss the problems, our progress in the project and to exchange ideas with Greeting. In the beginning of June we moved, together with the Greeting staff, to a new office at YES Delft! so we could discuss problems more frequently than before. This has accelerated the progress of the project. Besides, we could use the mobile phone of one of the Greeting members as a test device.

Next to our daily informal meeting with Greeting, we had an appointment almost every week with Mr. Geers, our TU Delft supervisor, to inform him about our progress and to evaluate our works. Among the team, we had a short informal meeting almost two times per week to inform each other about our individual progress in the project, make to do-lists and divide the remaining tasks among us.

2.3. Documentation

During the internship we created different types of documentation. We started with a Project Log document to keep track of the daily work and particularly the problems we encountered. Further, the requirements, design and project planning documents were made in the early stage of the internship but were constantly improved during the project. As requested by Greetingq, we made a user manual for the final product. The user manual and the Project Log are not included in this report. The rest are found in the appendices.

2.4. Team work

Tasks were divided based on preferences of the group members. We did not have to spend much time redoing each other's works. Fortunately, there were no conflicts among team members. We fully cooperated and discussed the problems with each other. This was one of the important success factors in speeding up the process. Using the SVN tool was another factor which eased the team work. We mainly worked individually although some tasks were done by a group of two persons. Everyone participated equally in the decision making process. This prevented conflicts in the process.

3. Problem domain and Analysis

The myGreetingq pilot web service was announced in April 2008. This service, a form of visual voicemail, provides mobile users to manage their voice messages and to use personal greetings. The Greetingq Company wanted to expand its services by providing a mobile phone version of myGreetingq. This allows users to access myGreetingq anywhere at any time.

In this chapter we introduce the problem domain and an analysis of it. This research is needed to understand our problem and to make the right decisions. In section 3.1 the problem domain is explained. The analysis in section 3.2 focuses on the potential market of the product.

3.1. Problem domain

Greetingq assigned us to develop a mobile service (visual voicemail & greetings) called "myGreetingq Mobile". Voicemail services provide phone users facilities to listen to messages left by callers when they fail to answer incoming calls. This service has been in use in most of the countries for over a decade. What is new in this field is the possibility of a 'visual' service to manage voicemails. Next to this, Greetingq wanted the users to be able to set personal greetings for different callers. Currently, there is no way to set different greetings.

3.2. Analysis

This section is divided into two parts. Part 3.2.1 describes a general analysis of the problem domain involving general knowledge and a short introduction to competing software. Part 3.2.2 gives an analysis of the users.

3.2.1. *General analysis*

Telephony and in particular mobile telephony is a service which has had a growing amount of users ever since its invention. Together with fast growing wireless technology, the number of services that a mobile phone can offer is increasing dramatically. This has opened a new area in software development that is evolving very fast. As mentioned in the previous section, our plan is to develop a visual voicemail and greeting service for mobile phones.

The traditional voicemail service is implemented by forwarding calls that are not answered, to a voicemail operator line. This line is connected to an advanced answering machine that handles the call; it asks the caller to leave his voice message, and records the message on some storage medium to be listened later by the callee.

Of course, in this strong competitive mobile phone market there are several telecom companies such as Apple [1] which provides a similar visual voicemail service. Currently, in the Netherlands there is no such service available yet. At this moment Greetingq has a great opportunity to introduce a visual voicemail and personal greetings service. An important feature making myGreetingq different from others is to combine web based service with mobile service, which other existing companies do not have.

3.2.2. *Users*

The number of mobile phone users is increasing rapidly. Currently, the number of mobile phones in use is more than there are inhabitants. Research has shown that within a few years the percentage of mobile phones with internet will be around the 70 percent of all mobile devices [4]. Further, it is remarkable that mobile internet is popular especially among educated people younger than 45 years. Only 7 percent of the old or low educated people use mobile internet [2]. Every mobile phone owner is a potential customer. People that already heavily using their voicemail are most likely interested in the improved service offered by Greetingq Company. However, people who do not use their voicemail currently- because of its bad service, could also be persuaded by the new service. We have categorized the potential customers in four groups:

- **Businessmen:** because of the nature of their work, they are always in contact with people. Sometimes they need to talk in different languages with their clients. Besides they are hard to reach because they have many meetings and appointments. It would be very practical for them if they can make different greetings for different groups of people.
- **Employees:** using one phone for both work and private life may not be favourable. It wouldn't be affordable either for many people to buy two different mobiles to separate their work from their personal life. Especially choosing an applicable greeting is not evident; a more formal greeting would be good for business, but not for kids. The possibility of making multiple greetings and selecting between voicemails comes in handy here.
- **Teenagers:** recently, a few companies have been very successful at selling ringtones for mobile phones. Many of these ringtones were popular because they are funny. Greetingqs could cause the same hype: selecting different funny Greetingqs for friends when they get your voicemail.

- People with foreign contacts: being anything from business partners to close family, it would be clear and polite to speak to people in a language they understand. With the Greeting service, one could configure the voicemail service to provide an English greeting to numbers calling from outside The Netherlands.

4. Research

4.1. Pre-research

End March, before the project started, we focussed on the pre-research. The purpose of this phase was to get familiar with techniques and tool needed during the project. Later in the design and implementation phase it became clear which of these techniques and tools were applicable and should be used. The final decisions made during the project are explained in chapter 5 and chapter 6.

Since the application must be implemented in Java Micro Edition (Java ME), we concentrated on the following techniques and tools with respect to Java ME: XML, HTTP, Symbian OS, CLDC, MIDP, MMAPAPI and Sun Java Wireless toolkit (for more details of these terms, refer to Glossary). We could not collect enough useful information of all these techniques. A good solution was to have a try to get familiar with them. We needed to actually use them to figure out whether they are applicable and useful. This strategy we adapted during the project.

4.2. Requirements

We have interviewed the Greeting company staff for the identification of requirements. These requirements have been classified in functional (functions that the program must be able to perform), constraint (a situation the program must be able to cope with, or a property the program must have), quality requirement (how is software quality asserted), platform requirement (in what environment the program must function) or process requirements (how must development be planned and managed).

For a detailed list of requirements, see appendix B section 3.1.

4.3. Scenarios

User scenarios are used to explain an interaction of a fictitious person with a system by simple logical stories, helping readers grasp complex ideas, non-technically. This section gives a brief introduction of the use case model, and the user scenarios we collected after interviewing the company. We describe here one of the user scenarios in details. The rest of the scenarios and their related use cases are explained in the Requirements Documents in Appendix B.

Figure 4.1 illustrates the interaction of a user with the system. The model describes a system's functional requirements in terms of use cases. Here it is shown which tasks the user can initiate when interacting with the system. For example if the user wishes to delete a message, he has to **login** first, select that particular **message** he wants to delete in the screen and select afterwards the **delete** option in the menu. The scenario related to this task is given below:

Deleting a message

Actor: Peter

Flow of events:

1. Peter successfully logged in and sees that he has a few new messages.
2. Peter notices that a person whom he does not like has left a message. He chooses this new voice message and goes to 'Menu'.
3. Here he selects the option 'Delete' and presses 'OK'.
4. A confirmation message is shown and asks whether he is sure to delete the message.
5. Peter chooses "Yes" and the message is deleted from the system.



Figure 4.1: Use Case Model

4.4. During project

When only studying techniques and tools and not applying them in an actual work, it is hard to find out whether they can be used in developing a particular application, and if so, how well they will work. We applied the strategy mentioned in the section 4.1, when we designed the system. First we discovered what Java ME can offer for the user interface environment of our product. With the standard of Java ME, it was not possible to implement the user interface compatible with the style that Greetingq had requested. The standard Java Me gave us only a very basic user interface. A research to additional Java ME GUI libraries was necessary. We found a number of open source libraries, J4ME¹⁰ and jMobileCore¹¹, but none of them fulfilled our requirements. The problems with the graphical user interface will be treated in section 6.1.

A second subject, which was also significant, was the support of different audio format on different mobile phones. To choose the right audio format we focussed on three criteria: Time/Size, Quality and Support. Time: streaming of an audio file from the server to the mobile phone must take less than a minute, thus the size of the file cannot be too large. Size should be less than 1 Mb for one minute audio. Even taking the size of the file into account, it should keep its quality at an acceptable level, which means that the user can hear the voice message well. And of course, the file format should be supported by Nokia S60 mobile phones. First we considered the mp3 format, but we found out later that it was not fully supported by JavaME. Our results showed that AMR and WAV¹² are the best audio formats we can use for our product.

Testing was an important part of the whole project. To do a thorough but not exhaustive test we needed some testing tools. For unit testing there were JUnit and J2MEUnit test frameworks available. Sun Java Wireless Toolkit provides us Debugger, Profiler, Network Monitor and Memory Monitor. Besides we had the JT Harness and ME framework. A problem was that they function below our expectations and so we needed to find another additional test. Besides, testing the system with these frameworks required more time than we had left. Problems with testing are discussed in section 6.4

¹⁰J4me, <http://code.google.com/p/j4me/>

¹¹ jMobileCore, <http://jmobilecore.sourceforge.net/>

¹² WAV or WAVE is a file format developed by Microsoft. WAV contains audio which can be compressed with a codec, but practically always no compression is used. [6]

5. Design phase and its problems

We started out by decomposing the application into a small number of subsystems. They are, loosely based on the System Design document (appendix C):

- Graphical User Interface
- Application Logic – including web service access
- Data storage
- Audio processing (playback / recording)

The first three subsystems relate to our decision of using the Model View Controller (MVC) design pattern [3]. This pattern decouples the data object model (model), graphical representation (view) and the application logic (controller). Greeting is planning to work together with different companies, in particular mobile phone service providers. In such cooperation, the service provider would want to distribute our application using their own corporate identity (their logo, colors, text style). Having a separate view “layer” reduces this adjustment to replacing only that layer.

5.1. *Graphical User Interface*

From our past experience with the Java programming language we knew that libraries are available (AWT, Swing) which greatly simplify the process of developing a graphical user interface (GUI). Therefore, we did not expect great problems to arise for the GUI (this hypothesis proved to be false later on, see chapter 6 for the encountered problems). We had already identified the various screens needed in the Requirements Document (appendix B), and appointed one class for each screen.

5.2. *Application Logic*

The application logic receives events from the user interface, and provides all data that the interface has to display. Events can update the data model, trigger a server operation, and possibly make the application change its current state and/or screen.

Web service access should be fully asynchronous, for reasons described in the Orientation Document (appendix A). While waiting for a response from the web service, the user interface should indicate that the system is busy, while not blocking the user interface. When the response is received and parsed, the application logic is notified and takes action appropriate for the response. Figure 5.1 illustrates the architecture of the application logic.

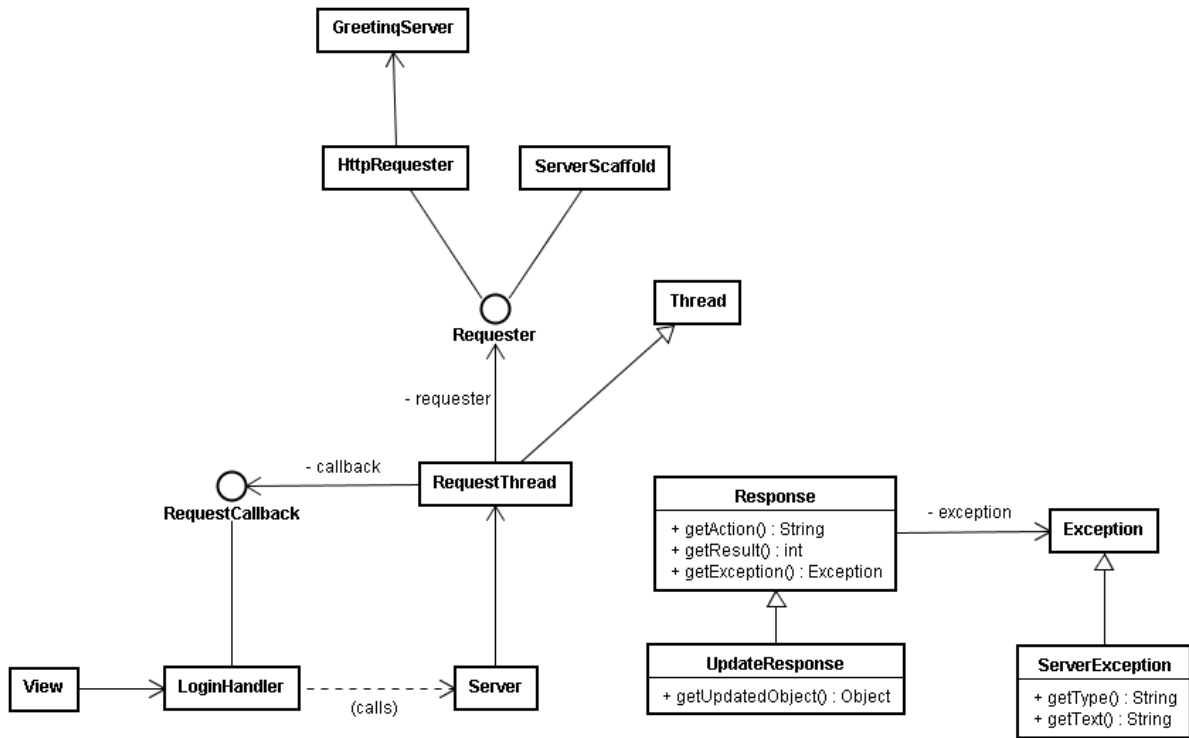


Figure 5.1: architecture of the controller

5.3. Data Model

The application communicates with the Greeting web server which stores and processes all account data. Therefore, the data model of the application domain is already fixed. We received the Greeting web service specifications document describing the high level interface to the web service. The data model used in the application is an exact copy of the XML objects structure used while communicating with the web service.

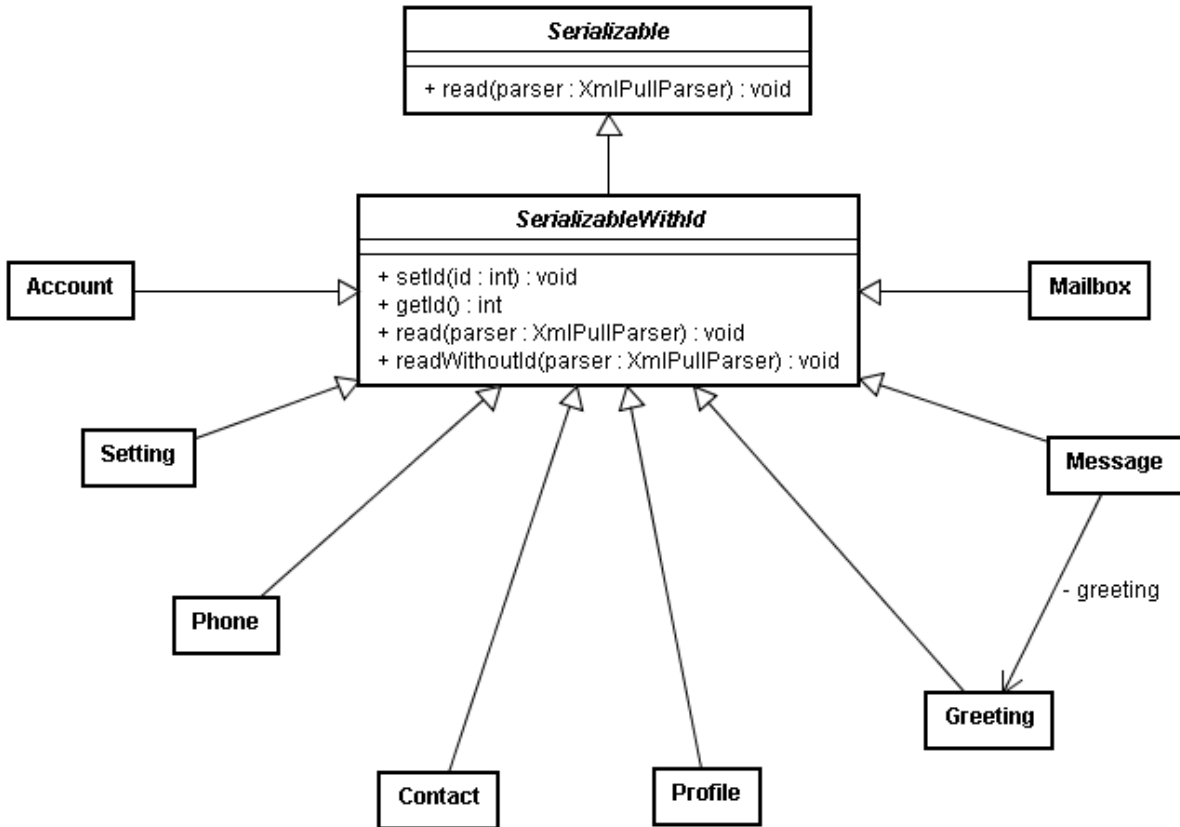


Figure 5.2: architecture of the Model

A short overview of the objects in the system:

Account: stores information about the users account, such as username, password, and personal information.

Contact: a contact of the user, consisting of a phone number and a name.

Greeting: consists of name, description, and audio file of a greeting.

Message: a received voicemail message. It includes data such as sender, date and the associated greeting.

Phone: a phone owned by the user, on which it can receive voicemail messages.

Profile: describes which contacts will hear a specific greeting when they get the voicemail of the user. Associates **Contact** objects with **Greeting** objects.

Setting: all the settings configured by the user.

Mailbox: configures the current voicemail service for the user. Options include enabling/disable the Greeting service and choosing the active profile.

Serializable is the base class for data storage. It provides a function to read and parse the object from a XML stream. Storage is implemented with a collection of elements of various types. We chose not to use specific attributes in the data classes for each data element, because the specification is likely to change. With the chosen approach we can easily adapt to such changes.

5.4. Audio Processing

Choosing an audio format supported by all mobile phones provided quite a challenge. We started looking at the well-known MP3 format but it seemed to be supported only on very new phones from Nokia. We then determined that we should support multiple audio formats, and choose one at runtime. This means the server has to be able to deliver audio files in various formats. Also, we chose to support multiple transport protocols. For details, see System Design document section E3. Because detailed information concerning which phones support which protocols was not available, we decided to determine the exact formats during the implementation phase.

6. Implementation phase and its problems

In this chapter we describe the implementation phase. From the principal we were advised to have a mock-up before we started with the implementation of the core of the system. The mock-up, which gave us a view of how code works in the early phase, was a basic user interface of the product. In section 6.1 we discuss the implementation of the graphical user interface and its problems we met during the process. Regarding to the divide and conquer principle¹³, we noticed two main parts of the implementation which we divided into two phases, namely Voicemail phase and Greetings phase. These two phases will be discussed in section 6.3 and 6.4, respectively.

6.1. Graphical User Interface

At the recommendation of the company we first started with a mock-up. The mock-up is a sketch of the GUI program without functionality. Quite early in the project we discovered that the UI was actually a notable problem. Different mobile phones provide different kinds of support for various graphical operations. Screen sizes, color depth, processor speed and interaction methods (keypad, pointer, touch screen) all differ to such an extent that making a uniform interface is very hard.

From the orientation phase we knew there are two forms of UI development for mobile phones: low level and high level. The former means that drawing of primitive components (labels, images, text boxes and so on) must be implemented by the developer, while the latter uses a default implementation provided by the mobile phone.

The product requirements state that the user interface must be in Greeting style (colors, fonts, icons) and that it must be adaptable to another company's style. Using the high level API the mobile phone implementation draws all components. This means that the exact looks of the application depend on the device. It also means there is no customization possible with respect to colors and background images / colors, which was one of the requirements. We therefore ruled out the option of using the high level API.

Low level development, on the other hand, requires us to create every component by hand (handling graphical representation and providing an interaction method), because mixing low level and high level components is not possible in the current version of Java ME. Creating all components equals developing a UI framework; this was simply too much work for the two months reserved for this project.

A third option was to use a readily-available UI library. We tested a few libraries, including Apime, LwVCL, J4ME, jMobileCore and LWUIT. Most of them were still in very early stage of development. Using the accompanying demos provided by the authors the libraries worked like a charm, but when we created a simple screen the result was one big mess. Three libraries were identified as possible candidates: TagsME GUI¹⁴, J2ME Polish¹⁵ and LWUIT¹⁶. Due to

¹³ Divide and conquer principle: Trying to deal with something big all at once is normally much harder than dealing with a series of smaller things. As software system can be divided in subsystems or in packages etc.[5]

¹⁴ TagsME homepage: <http://www.tagsme.com/>

time shortage we decided, together with Greetingq, not to further investigate these libraries. This will be included in the next version, released by a new team of developers.

Finally we decided to use the high level GUI. While not meeting the initial requirements, we saw no way to include a fancy GUI in the product while not failing to support some phones, or failing to meet the deadline. Therefore, the requirement stating the application should have the graphical Greetingq style was dropped.

6.2. Application Logic

Options for deleting a message or a greeting, and marking a message as listened / new were put in the message / greeting details screen. When showing the application to the Greetingq staff around half way the project, one employee noticed that these options would be better suited in the message / greeting list screens. We agreed and moved the options. However, because of limitations of the high level user interface API, it was not possible to show the difference between marking a message as listened and marking a message as new. The options was therefore renamed to just 'Mark'. When used on a new message, the message is marked as listened; when used on a listened message, the message is marked as new.

6.3. Voicemail

In this phase we had to implement the communication with the web service. Network traffic can block the user interface because the network is unreachable, the mobile phone has to ask permission for the network access, or the operation simply takes very long. Blocking the user interface prevents the user from using buttons, and no feedback can be provided on the screen. Therefore, network access (and also file access) is done in a separate thread, asynchronously with the GUI. While this prevents blocking issues, it gives rise to synchronization issues, the worst case being a deadlock. Luckily, the Java ME provides means to synchronize two threads. The implementation uses callbacks to indicate when the network operation is done. See the System Design Document, appendix C, for implementation details.

Next problem was the support of various audio formats in mobile phones. Uncompressed sound in a WAV file is supported on nearly all phones, but is far too large to be transferred over a slow wireless communication link. We chose for AMR files¹⁷ as they are supported by all S60 3rd edition Nokia phones [3]. These files are of low quality, but this is not an issue for voice. The low quality combined with compression makes the files extremely small. Downloading a voicemail message takes only little waiting time. As a backup method, WAV file playing is supported by the application, but has yet to be implemented at the server side.

As transport protocol we chose HTTP as it is the simplest and most supported protocol. RTP streaming is supported by the application (as decided in the Design Document), but is presently not supported at the server side.

Demo: <http://netbeans.tv/community/TagsMe-GUI-in-Action-352/>

¹⁵ j2mepolish <http://www.j2mepolish.org/>

¹⁶ LWUIT <https://lwuit.dev.java.net/>

¹⁷ See System Design Document, section E3.

A particularly hard problem was the synchronization of the user interface, which should display the progress of the audio player, and the audio player itself. Both are residing in different threads and are therefore executing asynchronously. They need to interact with messages in two directions: the UI should inform the player about buttons pressed (play, stop, and pause) and the player should inform the UI about the current status (playing, paused, current time position).

6.4. Greetings

In the Greetings phase, functions related to the “Greetings” were implemented. The main function in this phase was the ability to record voice with this mobile application. Therefore we had to take the possibilities and limitations of a mobile phone into account. We tried to store the audio we recorded in a byte array. This was an easy solution, because the audio was passed in plain wave form, we did not have to worry about mobiles that do not support a specific audio-format. But storing in a byte array had two disadvantages: memory usage (the whole byte array is allocated in memory) and size (no compression), which means long transfer times to the Greeting server. We noticed that all Nokia phones supported the AMR file format. For non-Nokia phones we chose the WAV file format as a backup file format. For the whole process we had two options:

- Record first in the main memory, create a file in the hard disk, save in the file, send to server, and delete the file.
- Create a file on the hard disk and store the audio while recording, send to server, delete the file.

To ensure that we do not lose any data (in case of memory shortage) we chose for the second option. The recorded audio file is only temporarily saved on the storage of the device. After sending it to the server the file will be automatically deleted.

We developed the system to be robust. It finds a match between supported audio types on the device and supported audio types of the server. The server then converts all supported input to AMR. In spite of this we found phones on which the application did not work. The phone simply did not accept audio formats it claimed to support. Our workaround for this situation is to let the mobile phone choose which file format to use if all our suggested formats fail. Then, the resulting format is checked for server support.

7. Testing

Testing a Java application on small devices is quite different from that of on a computer. There are several constraints set up by the device. For example, a mobile phone has a small screen and limited memory. There are also just a few threads, the connectivity is intermittent and there is no clear file system. Automating the testing does not work on mobile because multiple tests are hard to run. Besides, the growing number of different devices makes it more complicate to test a mobile application.

Although we restricted to mobile phones with Symbian S60, it was still hard to have our product thoroughly tested. While there was no standard to test Java mobile application we still needed the existing test methods to verify if our product fulfils the requirements.

Due to complexity of the code, it was not possible to write tests for all the packages in time. It would also be an exhaustive job. We decided to apply unit testing to packages where it can test one class at a time; in our case only the model package. After each phase we tested the functionality on the emulator and on our testing mobile device. The final or system testing was done manually. The unit testing, functionality and system testing are discussed in section 7.1, 7.2 and 7.3, respectively. We also planned to take an acceptance testing. But at this moment, this document it is not done yet. In section 7.4 we introduce the acceptance test.

7.1. Unit Testing

We have been familiar with JUnit during one of our course, Software Quality and Testing, at TU Delft. JUnit, a variant of JUnit for Java ME, was thus quick to learn and easy to use. After studying JUnit we realized that the testing in this framework may have limitations because of the simplicity of JUnit. So we were forced to keep the unit testing strict into the classes of the model package. Unit testing is to verify that a particular source code is working properly. Due to time limitation we were unable to do unit testing in other complex classes. Unit tests are not designed for testing multiple classes but for a single class in isolation.

JUnit has all basic assertions that JUnit has, but it does not allow us to insert assertions in the source code. We used primitive code to assert the pre- and post conditions. For example, for `assertNotNull()` we used `if(something!=null)`.

The whole unit testing resulted in three errors which we corrected right after. The test result after removing those errors was 100% passed as it is shown in figure 7.1. We can only say that the code fulfils the assertions. About the coverage we cannot say anything, because there are no coverage tools available for Java ME.

Class	Pass	Fail	Error	Total	Time
Account	11	0	0	11	31
Contact	5	0	0	5	12
Greeting	8	0	0	8	16
Mailbox	4	0	0	4	2
Message	9	0	0	9	41
Phone	3	0	0	3	1
Profile	6	0	0	6	18
Setting	3	0	0	3	1

Figure 7.1 Final results of the classes under unit testing

7.2. Functionality Testing

During each phase, a part of the product was implemented. Before we integrated these parts into the system, we tested whether it worked properly. We tested whether the functions worked properly in the emulator and device. Of course this did not happen without problems.

The first problem we met was that there was no mobile phone available for testing our graphical user interface implementation. We had indeed a mobile device with a Windows Mobile 5.0 platform, but unfortunately till today there is no Java virtual machine developed which works properly. Doing tests on Windows Mobile 5.0 was not working. We have propounded to our principal that it was necessary to have a mobile phone to test our product. They agreed and until the test phone was available we had to do our test using the emulator and Windows Mobile.

According to Sun Microsystems, when an application is running in the WTK emulator it is supposed to work on mobile devices too, but it was not the case. Some functions might not work on the mobile device but do work well in the emulator and visa versa. Note that we are developing a product for mobile devices and not for emulators. Therefore we give a greater weight to tests failing on the mobile phone then in the emulator.

7.3. System Testing

As mentioned in section 4.2 the currently available testing tools for Java ME were not enough for a thorough testing. Besides, the structure of the implementation was too complicated to write a simple test with one of the tools. We finally decided to write possible situations and test these manually. It required much more work and could not be automated too, but with our situation it was the best solution. We called this system testing and it was needed to detect defects that would occur only when the complete system was assembled. The testing verified functionality, usability and security of the system.

Taking use cases and scenarios of the Requirement Documents¹⁸ into account, we have the following test cases: Log In, Main Menu, Messages, Message Details, Greetings, Greeting Details, New Greeting and Settings.

¹⁸ Requirement Documents , Appendix B.

Each test case has a Pass/Fail column in the tables below. The green colour indicates that the test has passed. After debugging, all tests have passed. Therefore the tests which failed once have their row in the table coloured orange. These errors are described in the Test Report (Appendix E). Situations that will never happen are coloured grey and not tested.

The test cases we used are the following:

Log In

Username	Password	Expected	Result	Pass/Fail
correct	correct	Login Succeed, IF messages = 0 MainMenu Screen ELSE Messages Screen	Login Succeed IF messages = 0 MainMenu Screen ELSE Messages Screen	P
correct	incorrect	Login Failed, Warning	Login Failed, Warning	P
incorrect	incorrect	Login Failed, Warning	Login Failed, Warning	P
incorrect	correct	Login Failed, Warning	Login Failed, Warning	P
Null	null	Login Failed, Warning	Login Failed, Warning	P

Main Menu

	Option	Expected	Result	Pass/Fail
Messages = null	Messages	Alert	Alert	P
Messages ≠ null	Messages	Messages screen	Messages screen	P
Greetings = null	Greetings	Alert	Alert	P
Greetings ≠ null	Greetings	Greetings screen	Greetings screen	P
	New Greeting	GreetingNewType screen	GreetingNewType screen	P
	About	About screen	About screen	P
	Settings	Settings screen	Settings screen	P
	Exit	Application closed	Application closed	P

Messages

Messages	Expected	Result	Pass/Fail
0	Alert	Alert	P
1	Messages Screen	Messages Screen	P
>1	Messages Screen	Messages Screen	P

Message	Option		Real result	Pass/Fail
Null	Details	-	-	-
True && New	Details	Message details Screen , Old	Message details Screen, Old	P
True && Old	Details	Message details Screen, Old	Message details Screen, Old	P
Null	Delete	-	-	-
True	Delete, Confirmation = YES	Messages - Message Messages Screen	Messages – Message Messages Screen	P
True	Delete, Confirmation = NO	Messages Screen	Messages Screen	
Null	Mark	-	-	-
True && New	Mark	Messages Screen , Old	Messages Screen , Old	P
True && Old	Mark	Messages Screen , New	Messages Screen , New	P
	Main Menu	MainMenu Screen	MainMenu Screen	P

Message Details

	Option	Expect	Result	Pass/Fail
AudioState = Stopped	Play	AudioState = Playing	AudioState = Playing	P
AudioState = Playing	Pause	AudioState= Paused	AudioState= Paused	P
AudioState = Playing	Stop	AudioState=Stopped	AudioState=Stopped	P
-	Back	Messages Screen	Messages Screen	P

Greetingqs

Greetingqs	Expected	Result	Pass/Fail
0	Alert	Alert	P
1	Greetingqs	Greetingqs	P
>1	Greetingqs	Greetingqs	P

Greetingq	Menu	Expected	Result	Pass/Fail
Null	Details	-	-	-
True	Details	Greetingq Details	Greetingq Details	P
Null	Delete	-	-	-
True	Delete Confirmation=NO	Greetingqs Screen	Greetingqs Screen	P
True	Delete Confirmation= YES	Greetingqs - Greetingq Greetingqs Screen	Greetingqs - Greetingq Greetingqs Screen	P
	Main Menu	MainMenu Screen	MainMenu Screen	P

Greeting Details

Greeting Details	Menu	Expected	Result	Pass/ Fail
	Play	AudioState = Playing	AudioState = Playing	P
AudioState = Playing	Pause	AudioState = Paused	AudioState = Paused	P
AudioState = Playing	Stop	AudioState = Stopped	AudioState = Stopped	P
	Edit	Edit Greeting Screen	Edit Greeting Screen	P
	Add Contact	GreetingAddContact Screen	GreetingAddContact Screen	P
Contact = 0	Delete Contact	GreetingRemoveContact Screen, Contact = 0	GreetingRemoveContact Screen, Contact = 0	P
Contact >= 1	Delete Contact	GreetingRemoveContact Screen, Contact >=1	GreetingRemoveContact Screen, Contact >=1	P
	Back	Greetings Screen	Greetings Screen	P

Screen	Option	Expected	Result	Pass/Fail
Edit Greeting	Cancel	GreetingDetail Screen	GreetingDetail Screen	P
Edit Greeting	Save	GreetingDetail Screen + updated details	GreetingDetail Screen + updated details	P
GreetingAddContact	Cancel	GreetingDetail Screen	GreetingDetail Screen	P
GreetingAddContact	Add	GreetingDetailScreen + added Contacts	GreetingDetailScreen + added Contacts	P
GreetingRemoveContact	Cancel	GreetingDetail Screen	GreetingDetail Screen	P
GreetingRemoveContact	Save	GreetingDetail Screen – deleted Contact	GreetingDetail Screen – deleted Contact	P

Settings

	Option	Expected	Result	Pass/Fail
Maximum messageduration =null	Save	Maximum messageduration = 0	New Maximum messageduration = 0	
Maximum messageduration = t>0	Save	New Maximum messageduration =t>0	New Maximum messageduration=t>0	P
Voicemail Activation = No	Yes	Voicemail Activation = Yes	Voicemail Activation = Yes	P
Voicemail Activation = Yes	No	Voicemail Activation = No	Voicemail Activation = No	P
Default Greeting = null	Save	Default Greeting = null	Default Greeting = null	P
Default Greeting = Greeting	Save	Default Greeting = Greeting	Default Greeting = Greeting	P
Language = Dutch	Modify, English, Save	Language = English	Language = English	P
Language = English	Modify, Dutch, Save	Language = Dutch	Language = Dutch	P
	Main Menu	MainMenu Screen	MainMenu Screen	P

Most of the errors in this testing were easily found in the code and debugged. They were not hard to correct. They were rather human mistakes. For example, a wrong String is given as parameter. Other errors were caused by the server side. Because we were not responsible for what happens at the server side, we could not see what was wrong and why it was wrong. The only thing we could do is deliver these problems to the responsible department of the company and let them fix these problems. In this situation we only can ensure that the client side, thus the mobile application, is tested well. Taking into consideration the possibility of the server bugs delayed out testing process.

7.4. Acceptance Testing

The acceptance test will be conducted to enable a user/customer to determine whether to accept a software product. Normally it is performed to validate whether a software product meets a set of agreed acceptance criteria. Greeting has made an appointment on July 10th to test the final product. Therefore we cannot include the acceptance testing in this report. We will discuss the results during the presentation¹⁹.

¹⁹ Presentation at July 15th, 2008 at Mekelweg 4, Delft

8. Final product

The final product must fulfil all constraints and requirements demanded by Greeting. According to Greeting Company the delivered final product must include:

- The application
- Documentation
- User Manual
- The source code
- Generated Javadoc and UML diagrams

During the process some of the requirements and constraints were changed and even eliminated. The reasons of these changes are already mentioned in their corresponding parts in this report. In section 8.1 we have summarized all requirements in a table and give a description in the case of differences with the initial requirements document. Section 8.2 demonstrates what myGreeting mobile product looks like in an emulator.

8.1. Requirements Satisfaction

Functional Requirements ²⁰	Tasks	Description
Voicemail messages	Marking the messages as listened or new: Changed	Changed to “Mark”.
Greeting service	Successful	
Settings	Change cache policy: Eliminated	Not required because of the small file format of the audio
Mp3 support	Eliminated	No support on our target mobile device Instead the AMR format is chosen
Audio streaming	Eliminated	No server support
Quality Requirements		
User Manual	Successful	Delivered in Dutch
Source code	Successful	
Technical Documentations	Successful	Delivered in English
Java Logging API	Successful	
Platform Requirements		
Symbian OS, S60 series 3 rd edition	Successful	
Windows Mobile 5.0	Eliminated	No Java Virtual Machine available which works properly
Constraints		
Pretty designed GUI compatible with the Greeting style	Changed	Simple GUI compatible with the Greeting style.
Multiple language option	Successful	
Change the graphic style	Eliminated	No possibility

Figure 8.1 Table of (un)satisfied requirements

²⁰ The detailed requirements can be found in Requirements Documents (Appendix B)

8.2. Screenshots

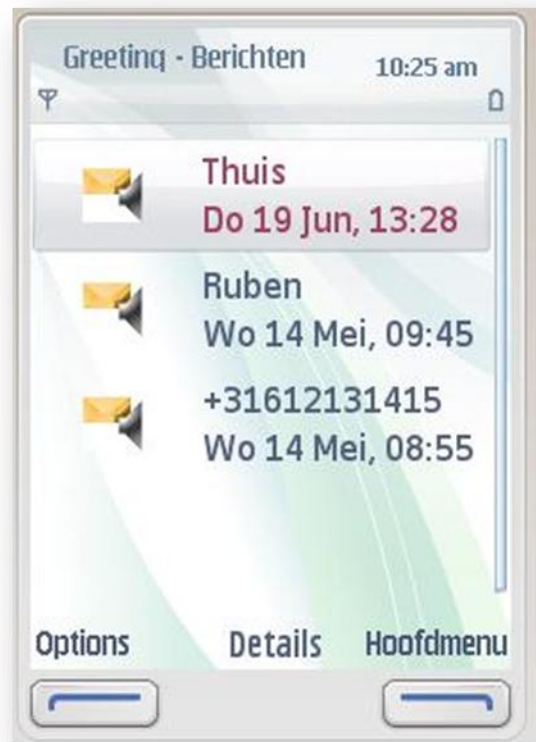
These screenshots show some of the main screens of the application. Because the high level graphical library of the target mobile phone is used²¹, the look and feel of the application differs from phone to phone. Screenshots below, unless noted otherwise, were taken from the Nokia S60 3rd edition Feature Pack 2 emulator for PC²².

²¹ For more information, see section 6.1

²² Downloadable from http://www.forum.nokia.com/info/sw.nokia.com/id/6e772b17-604b-4081-999c-31f1f0dc2dbb/S60_Platform_SDKs_for_Symbian_OS_for_Java.html



Loading screen



Message List



Message Details



Settings



Greeting List



Greeting Details



Add Contact to a Greeting



Trying to delete the default Greeting



Login on Windows Mobile 5.0 with IBM JVM v9



Message List on Windows Mobile v5.0 with IBM JVM v9

8.3. Logging Functionality

The application contains logging statements throughout the code. This has proved to be very useful when debugging on a mobile device. The output is stored in the device's memory and it is sent to the debugging output window of the development environment.

The text below illustrates debugging output when starting the application, playing a voice message and then recording a greeting. This example shows only lines tagged with 'info', but we also used 'warning' and 'error'.

```
[info] in com.greetingq.mobile.storage.Storage: RecordStore: 4 records
[info] in com.greetingq.mobile.Greetingq: Platform: SunMicrosystems_wtk
[info] in com.greetingq.mobile.Greetingq: Model: wtk-emulator
[info] in com.greetingq.mobile.Greetingq: Locale: nl-NL
[info] in com.greetingq.mobile.Greetingq: Timezone: UTC
[info] in com.greetingq.mobile.server.HttpRequester:
Requesting: ?action=Login&passwordhash=098f6bcd4621d373cade4e832627b4f6&use
rname=Nathan
[info] in com.greetingq.mobile.control.LoginHandler: Storing login data
[info] in com.greetingq.mobile.server.HttpRequester:
Requesting: ?action=GetMailboxData
[info] in com.greetingq.mobile.audio.player.Player: Supported protocols:
http file device capture
[info] in com.greetingq.mobile.audio.player.Player: Supported formats:
video/mpeg image/gif audio/x-wav audio/amr audio/x-tone-seq audio/sp-midi
audio/midi
[info] in com.greetingq.mobile.audio.player.Player: Using type audio/amr,
protocol = http
[info] in com.greetingq.mobile.audio.player.Thread: Loading audio from
http://asterix.greetingq.com/GreetingqWebservice/DownloadServlet?gtqItemType=
messages&gtqMailboxID=4&gtqFilename=C9D76870554743C7669CA97501E65F64&gtqFil
eType=amr&gtqHash=956463546fd7749576aa59b07f5560b7
[info] in com.greetingq.mobile.audio.player.Thread: Realized: duration =
27400000
[info] in com.greetingq.mobile.audio.player.Thread: Prefetched: duration =
27400000
[info] in com.greetingq.mobile.audio.player.Thread: Done.
[info] in com.greetingq.mobile.audio.recorder.Recorder: Chose
pcm&rate=8000&bits=8&channels=1 as record encoding.
[info] in com.greetingq.mobile.audio.recorder.Recorder: Possible recording
encodings: encoding=pcm encoding=pcm&rate=8000&bits=8&channels=1
encoding=pcm&rate=22050&bits=16&channels=2
[info] in com.greetingq.mobile.audio.recorder.Thread: Trying recording url:
capture://audio?encoding=pcm&rate=8000&bits=8&channels=1
[info] in com.greetingq.mobile.audio.recorder.Thread: Recording media in
content type: audio/x-wav
[info] in com.greetingq.mobile.audio.recorder.Thread: Recording audio at
file:///root1/testrecord.wav
[info] in com.greetingq.mobile.audio.recorder.Thread: Recording started.
```

8.4. Code Fragment

We provide here some code fragments to give the reader an idea of how the system design looks in practice.

Storing new greeting details in the handler class of the greeting details screen, `GreetingDetailsHandler`, first checks for an empty greeting name. When an empty name is passed, it instructs the view package to show an error message. The ‘translate’ comments hint the translation system that these strings ought to be translated.

```
public void store(String name, String desc) {

    if (name.equals("")) {
        BaseScreen.displayMessage(
            /* TRANSLATE */"Please fill in a name for this Greeting.",
            /* TRANSLATE */"Error", null, AlertType.ERROR, null);
        return;
    }

    screen.startWaiting();
    Server.getInstance().updateGreeting(gr.getId(), name, desc, this);
}
```

The method `updateGreeting` in the `Server` class simply converts its parameters to a format conforming to the web service specification. The code is as follows:

```
public void updateGreeting(int greID, String name, String description,
    RequestCallback callback) {

    Hashtable params = new Hashtable();

    params.put("greetingid", String.valueOf(greID));
    params.put("name", name);
    params.put("description", description);

    RequestThread r = new RequestThread(ACTION_UPDATEGREETING, params,
        callback, new Integer(greID));
    r.start();
}
```

This method returns immediately without waiting for the server to respond. The result of the operation will be provided through the following callback method of GreetingDetailsHandler:

```
public void respond(Response res, Object data) {

    switch (res.getResult()) {

        case Response.CMD_EXCEPTION:
        case Response.CMD_FAILED:
            screen.stopWaiting();
            screen.error(LanguageManager.get(
                "Error during request:") + "\n" +
                res.getException().getMessage());
            return;

        case Response.CMD_SUCCEEDED:

            screen.stopWaiting();
            BaseScreen.display(new GreetingDetailsScreen(gr));

            return;

    }

}
```

This method shows the error message if one occurred, and goes to the Greeting Details Screen if the operation succeeded.

8.5. Future Plan

In section 2.1 we described how Greeting planned to release the final version of the product to the market in September 2008. Time-to-market was an important factor for this project because the mobile phone market is developing fast and competition may be on their way.

Later in the project, as described in chapters 5 and 6, quite some unexpected difficulties arose. The scope of supported hardware was greatly narrowed down. The initial requirement of supporting Windows Mobile-based devices was dropped, and because only one phone was available for testing we are not able to completely verify the product for other devices.

At some point, Greeting decided not to launch the application in September. This is of course disappointing both for us as developers and for Greeting. However, this decision was taken regardless of our work. Our final application provides all of the functionality required and has been finished within the deadline of our bachelor project.

Greeting intends to continue development of the application after we finish our project. Development will be taken over either by Greeting staff, by hiring extra developers or by outsourcing it to another team of interims. The good practices of documenting design decision, commenting code and writing a development manual will in this case surely prove to be abundantly fruitful.

9. Conclusion and Recommendation

Developing an application for a mobile device has proven to be much more difficult than developing for a computer. The difference can be mainly attributed to two factors:

- A mobile phone has less capabilities
- Mobile phones differ more in capabilities

While personal computers also differ a lot (processor architecture, operating system and even localization) these differences seem to be negligible when using the Java programming language. The same cannot be said of the Java ME language.

Sun Microsystems tried to minimize the chaos by specifying a subset of the Java ME API which all mobile phones should support. The rest of the functionality is available in optional packages, which a mobile phone can support or not, and the API provides a way to determine which packages are available. However, the real problem is that the functionality common to all phones is not enough to build commercial applications. As mentioned in section 6.1, there is no way to implement a nice looking user interface, for example.

Another huge problem of the Java ME platform is the relatively high amount of bugs in implementations. Sun provides a reference implementation, while phone manufacturers have to create their own implementation specifically optimized for the phone. In general, Windows Mobile based phones do not even have a standard Java implementation. Various companies offer (commercial) Java implementation, but they are so buggy that we decided not to support them at all.

Nokia provides a bulk of information, consisting of a resource library, implementation notes (describing specific details about their implementations), a maintained wiki²³, a lot of examples and detailed information about the supported APIs on all phones. While this helped a lot, the many implementation bugs (although sometimes documented) make it particularly hard to develop a robust application.

While it was not easy, we did manage to complete the project in the reserved time. The pitfalls took much more time than we expected. This was compensated mainly by the following three decisions:

- spending less time than expected on things like web service access,
- having a full week of spare time planned in from the beginning,
- making sure that all hours are used; if for some reason one could not be present in the morning, it was possible to work in the evening

Our recommendations for developing a mobile application are some guidelines that we believe would help us and greatly simplify our process.

First of all, pay attention to the target platforms. Fully targeting and supporting all mobile phones will amount to testing, adapting and maintaining a separate version for almost every phone. This approach is taken by J2ME Polish, a software library trying to hide the differences between mobile phones.

If not every mobile phone is to be supported, choosing S60 3rd Edition as a minimum platform overcomes a lot of issues with unsupported features. S60 3rd edition and onwards phones provide a vast subset of the Java ME specification and the phones share a lot of particularities.

²³ A wiki is a library in which visitors are allowed to edit content. See <http://nl.wikipedia.org/wiki/Wiki>

Note that while this should also apply to non-Nokia phones, we did not try the application on these phones.

Something else we should have done much earlier is read through the Nokia Implementation Notes very carefully. Although the document is not on the main documentation index of the Nokia website, it provides detailed information essential for developing mobile applications on Nokia phones.

Concluding we can say that this project has been completely different than expected due to the various unexpected issues, but therefore also a big challenge. We have learned to overcome unforeseen limitations, and to adapt both the requirements (in consultation with Greeting) and our planning during the project.

Besides the troublesome development for mobile phones, the cooperation with Greeting and their technical staff has been excellent. The advantages were twofold. Greeting provided us with useful knowledge and tips, helped us find bugs, and provided us with the needed documentation. On the other hand, we used their web service, which is still in beta phase. We provided useful first-hand feedback and reported some otherwise hard-to-find bugs.

10. Personal Reflection

10.1. *Medya Amidi*

First of all, I have very much enjoyed working on the project. To me it was an interesting and informative project because I haven't had an experience in developing a mobile application before. I do have to admit that at first I did not have a clear vision of how we are going to tackle the project. By using the TU Delft guidelines for bachelor project and also the experiences we have gained during the projects we have done in our study, we could finally manage to figure out how we can approach the problem. This was of course not possible without lots of brainstorming with each other.

During the project, knowledge of a variety of courses I attended at TU Delft came of use. Software engineering, software quality and testing, project skills, data structure, and Java programming were the main relevant courses.

I had work experience in a Dutch company before I started my bachelor project. So I was more or less familiar with work sphere within a company. However there were some major differences between Greeting Company and the one I had worked in before. Greeting is a new, fresh company established by three enthusiastic students. This resulted in a friendlier working sphere. The communication with the Greeting staff was straightforward and we could frequently discuss with them about the problems we have encountered during the process. The other company, however, had a clear discipline and methodology of software development. When we started our project we could choose which methodology we are going to use during the process. During our internship, I had a chance to get to know basic principles to start a company. Because of close, daily contact with Greeting staff, I could see how they manage their work and cooperate with each other.

When working in a team, the most important factor is cooperation. During our study, I had done a few projects with both of my team members. Both are hard working and determined students. I could perfectly communicate with them. So when I chose my team work I was pretty convinced we can cooperate well. Fortunately, we could work with each other and discuss our problems with no conflicts. This has enhanced the progress of the process.

My major contribution in this project was in developing the GUI environment of the product. This phase, as it is also mentioned in the report, was problematic and needed the great deal of effort. I have participated in creating the documents and reports. Further, I have done researches regarding audio format support, searching for useful and available libraries for the GUI, tried out low level GUI APIs during the process. At the beginning of the project it was very difficult for me to figure out how the structure of the system should be. So I couldn't participate in the design of the system. It took me a while to understand the structure. The project seemed to be simpler before starting actual implementation. Later in the project, we have faced many problems which we had to work hard to figure them out. But at the end, despite of difficult moments, when I look back to the past few months, I can say I have gained valuable experience during this period.

Finally I would like to thank Greeting for providing us a pleasant work environment and facilities; Mr. Geers for his support and supervision during the project and Mr. Sodoyer for coordinating and arranging the project.

My special thanks to my team members, Nathan and Michelle, for their great effort, cooperation and dedication to the project.

10.2. Nathan Bruning

Looking back at the past half year, from the moment I first met Greetingq, a lot has happened. At the very beginning, I was not having any preference of which project to choose for my bachelor project. Greetingq was an option, but other companies were clearly very eager to find students for their projects. Even though, Greetingq managed to get me interested for this project in a way other companies had not been able to.

When one of the team members asked me to join her for the bachelor project, it took little time to agree. We had been doing a few projects together earlier in the study program and we get along too outside of university-related activities. The third project member was familiar (we all started our study in the same year) and I got to know her quite well along the project.

The project itself started very relaxed, just reading through the available information and getting familiar with the field of mobile application development in which we had no experience at all. It was not until we actually tried to code a simple application we found out that the documentation had presented us with an oversimplified view of reality.

Previous experience in using concepts like HTTP, XML, audio processing and multi-threaded programming proved to be absolutely necessary to complete the project within the planned time frame. Because we all followed the same curriculum we all had a solid theoretical knowledge base. Even though, differences were noticeable because of the different amounts of experience in programming. For me, programming reasonably large-scaled applications was not a completely new experience, while the other project members had more difficulties overseeing the application structure. On the other hand, the knowledge of the English language provided by the other team members was very valuable for writing the many documents.

We started working in a room separate from the Greetingq staff, but close enough to it that we could walk to their office for lunch every day. However, we very much missed the ability to ask for support at any time. Because our application uses the Greetingq web service, which is still in beta at the time writing, we needed thorough knowledge of its working. If not, we could be stuck trying to solve a bug in our application which actually is a bug in the web service.

Working in the same office as the Greetingq staff gave me a good look at the inner workings of a starting company. I noticed that every member of Greetingq had a fairly specialized function. They did not interfere with each other's work, neither did they understand each other's work at all times. This fact surprised me, because many of the work I have done involved close collaboration of multiple developers on the same piece of software. Also, at one time I remember two members of the staff, just returned from a meeting with a potential customer, telling they did not know what to say when the customer asked them about some technical issues.

The collaboration between our team and Greetingq worked very well, giving rise to a friendship which extends beyond the work floor, for example, when we watched the European soccer championship together.

My part in developing the application has involved mainly the backend of the system; for example, communicating with the server, determining audio formats and avoiding deadlocks and race conditions. Also, I spend much time cleaning up code, adding Javadoc and writing a developers manual. Finally, we all spend about the same time writing the documents that can be found in the appendices.

I would like to conclude saying that this bachelor project has been a good experience for me, exploring the ups and downs of both our application and a starting company. We have had fun on the project as well as difficult moments. I am very glad with my choice of team members and even so with the choice of assignment to work on. Greetingq has proven to possess the skills needed to supervise a team of young developers creating a complex application.

10.3. Michelle Cheung

Before the start of the project, there was an orientation phase where we had to do research in techniques that are appropriate for the product. Some techniques and tools were already pointed out by the principal.

The process from beginning to the end was constructive, each time I was getting more and more involved in the subject. In this phase, I was still at an unknown territory. Developing mobile application was a new challenge for me; even I was familiar with Java programming. It was confusing to research so many techniques and their information was a little bit superficial. They remained vague until I worked with them. Companies of all kind techniques claim that their products are functioning well and have the features to develop or to support a mobile application. During the implementation phase it turned to be the other way. Here I experienced what could happen when developing new software. For a next project I know there need more attention for a research to other developers' experiences. They often can provide objective more information about a technique than the companies do.

Different from other projects done in TU Delft, we had to find solutions ourselves and presenting these to the principal. Since the principal has certain knowledge in this area, we could easily discuss about possible solutions. The informal deliberation which took place almost every workday accelerated the process. We could nearly take decisions immediately.

As well as new experiences with the programming language Java ME, it was also my very first time working in a company of the software industry. Greetingq is an infant company which is still growing. After we moved from our first location at the Stevinweg 1 to the new office, there was the opportunity to see what their daily work was. It varied from such as obtaining new clients as adjusting legal affairs. Bringing a new product in the market does not only depend on a good application but also the whole organisation of the company. Most of the assignments for Bachelorproject are situated in experienced companies. Trainees then create a picture how a part of a company functions, in this case only the development department. In the Greetingq Company, I had the opportunity to have a look at the whole entrepreneurship. Which was also one of the reasons I chose for this assignment: there is a lot of chances to work in big companies, but not in a starting up company.

The collaboration with the other two team members was as expected good. Partly due to the same motivation and guideline we had; having this assignment done together and supporting

each other through this long process. Until we had the product finished, I was a little bit nervous about the deadline. Mainly due to the late start of our project. Of course it was exciting to take part of the developing of an interesting product. Perhaps this uncertainty was my incentive to work hard. And seeing my team members putting many efforts in this project stimulated me too. Also division of tasks was not a problem. Each of us took his or her task serious. It happened that one of us could not finish a task; there was someone willing to take over the responsibility.

This bachelor project is indeed a nice experience for an entrance in the real business world, but also an intensive process requiring patient, teamwork, endurance, and certain knowledge about developing software. I consider this project as a big practical where I can gather experiences which I cannot find in other practical of the bachelor study.

References

- [1] Apple, *Apple iPhone*
<http://www.apple.com/iphone>
- [2] Centraal Bureau voor de Statistiek. *Eén op de vijf internetters mobiel online*. March 17, 2008. Fetched June 20, 2008.
<http://www.cbs.nl/nl-NL/menu/themas/vrije-tijd/cultuur/publicaties/artikelen/archief/2008/2008-2415-wm.htm>
- [3] Nokia Forum, *Mobile Media API Support In Nokia Devices v2.0*
http://www.forum.nokia.com/info/sw.nokia.com/id/54200041-bdb3-4e25-817e-66ed9b56abb1/MIDP_Mobile_Media_API_Support_In_Nokia_Devices_v2_0_en.xls.html
- [4] Wikipedia, *Mobile Commerce*. Fetched June 19, 2008
http://en.wikipedia.org/wiki/Mobile_Commerce
- [5] Wikipedia, *Model-view-controller*. Fetched June 23, 2008
<http://en.wikipedia.org/wiki/Model-view-controller>
- [6] Wikipedia, *WAV*. Fetched June 19, 2008
<http://en.wikipedia.org/wiki/WAV>

Attachment: task division

Tasks	Medya Amidi	Nathan Bruning	Michelle Cheung
User interface	X	X	
Voicemail		X	
Server communication		X	
Greeting record/playback			X
Test	X		X
Documentations	X	X	X

Appendix A: Orientation Document

IN3405: Bachelor Project

Orientation: state of the art techniques

Bachelor Project at Greeting

Technische Informatica
Faculty Elektrotechniek, Wiskunde en Informatica



IN3405 – Bachelor Project

<u>Date:</u>	July 8, 2008
<u>University:</u>	Technische Universiteit Delft
<u>Course:</u>	IN3405 Bachelor Project
<u>Authors:</u>	Nathan Bruning Medya Midi Michelle Cheung
<u>Teacher:</u>	B.R. Sodoyer

1. Preconditions

The first condition is implicitly introduced by the product: the software package we are to create has to run on a mobile phone.

Furthermore, we are bound to support a specified number of platforms. The mobile device-platforms *Symbian S60 3rd Edition* and *Windows Mobile 5.0* are to be supported as minimal systems.

Because these two platforms are quite different in both operation and technical design, we are required to use the programming language Java which allows us to develop an application that runs on both platforms (and their successors).

2. Familiar Techniques & Tools

In this section we will shortly review known techniques and tools which will be used during the project. Most of the knowledge comes from lectures and practical experience.

2.1. Techniques

The familiar techniques used in this project are as follows:

- **Programming in Java**

Java is the programming language that we have used in all projects that we have done during our study. In this project we are going to write our application in this language. The major advantage of using Java in mobile application is its portability. This means it can be run on different mobile phones with various mobile operating systems.

- **Building a Graphical User Interface**

In order to build a mobile application with an interface for users to interact, we need to be familiar with graphical User Interfaces in Java. As in other projects that we have done before, we are quite familiar with GUI in Java SE and its features.

- **Using Databases**

To transfer data from a database to the mobile phone we need to have some knowledge of how a database works.

- **Using XML**

XML is the de facto standard for encoding messages when transferring them between various platforms. The messages from the database are encoded into XML by the server, and we need to handle this format in our application.

- **Using the Hyper Text Transfer Protocol (HTTP)**

For the messages to be transferred from the Greeting servers to a mobile phone, a communication protocol is needed. HTTP is used for its simplicity and wide spread implementations.

2.2. Tools

Because the project management wants to use our project right after its completion, we have been given a couple of guidelines about how to store the code and in which environment to develop. This ensures a smooth transition of code maintenance from us to Greeting.

- Storage and Version Control

To keep to project accessible from multiple locations, and to be able to work simultaneously we are going to use Subversion (SVN). This *version control system* is widely accepted as the successor of the well-known CVS (Concurrent Version System).

We are already familiar with the details of using SVN for storing application code because we have been using it in earlier projects.

- Development Environment

Although we are not obligated to use a specified development environment, we are advised to use a free and open-source IDE (Integrated Development Environment) called Eclipse. This software package has come a long way since its original start at IBM [7]. It has been given to an independent non-profit corporation called The Eclipse Foundation which has lead its development ever since the transition from IBM.

The Eclipse environment is also familiar to use because of previous projects.

3. New Techniques

In our study we have been learning to program for embedded devices, but not for mobile devices running Java. Therefore quite a few techniques are new to us. We have been orientating in the world of mobile programming, and we will describe both the techniques we are required to use and the techniques we suspect to be useful when implementing the application.

3.1. Symbian OS

Symbian OS is a proprietary operating system, designed for mobile devices. Symbian is currently owned by Nokia (47.9%), Ericsson (15.6%), Sony Ericsson (13.1%), Panasonic (10.5%), Siemens AG (8.4%) and Samsung (4.5%). Although BenQ acquired the mobile phone subsidiary of Siemens AG, the Siemens AG stake in Symbian did not pass to BenQ [14].

The Symbian OS MIDP (to be described later) runtime environment is an integrated component of the operating system. The platform allows Java applications to exploit the

advanced multitasking, graphics, and communication capabilities of the underlying OS. The platform includes similarly highly integrated implementations of many optional Java ME API packages, including Bluetooth, 3D graphics, Multimedia and Content Handling.

Symbian OS v9 supports very high Java ME standards such as:

- CLDC 1.1
- MIDP 2.0
- Large collection of optional APIs – such as WMA, File and PIM access, Bluetooth, 3D graphics
- On-device debugging

In order to write a mobile application, we need also to have SKD – a MIDP Software Development Kit – which allows us to write the program for the mobile phone. There are a number of MIDP SDKs available, depending on the phone. When writing programs for Symbian mobiles and no particular phone is targeted, the most recommended platform and SKD to use is S60 and UIQ respectively.

3.2. Sun Java ME

Java Micro Edition (Java ME) is a Java platform for micro devices. Micro devices are devices with limitation in processing power, in storage capacity, and devices with no or low bandwidth network connection. Mobile phones, PDAs and car navigation systems are such devices.

Java ME is subdivided by Configurations, Profiles and optional packages.

A **Configuration** is the specification of a JVM and a base set of necessary class libraries; commonly based on a subset of the J2SE APIs. Currently, there are two Java ME Configurations:

- Connected Limited Device Configuration (CLDC)
- Connected Device Configuration (CDC)

A **Profile** sits on top of a configuration to complete the runtime environment by adding more high-level APIs, thereby preparing a device for a specific device category. Currently, there are two common Java ME Profiles:

- Mobile Information Device Profile 1.0 (MIDP1)
- Mobile Information Device Profile 2.0 (MIDP2)
- Mobile Information Device Profile 3.0 (MIDP3)

The combination of MIDP/CLDC standard is widely used to provide a complete Java platform for mobile phones and other similar devices. It provides core application functionality required by mobile applications in the form of a standardized Java runtime environment and a rich set of Java APIs. Developers using MIDP can write applications once, and then deploy them quickly to a wide variety of mobile information devices.

The MIDP applications have many benefits. Among others is Rich User Interface Capabilities: The graphical user interface is optimized for the small display size, varied input methods, and other native features of modern mobile devices.

Optional packages add functionality to the Java ME platform by offering standard APIs for the use of various technologies, such as databases, connectivity, web services, 3D graphics and much more [16].

3.3. *Connected Limited Device Configuration*

Connected Limited Device Configuration (CLDC) is a specification set up by Sun in which a number of programming interfaces are defined, together with a (virtual) environment in which mobile application can be run [4].

It gives a developer the possibility of developing applications for devices with limited speeds, memory sizes and screen capabilities. The CLDC can be complemented with optional packages so that specific possibilities of more advanced phones can be exploited.

Sun Microsystems manages the specification and provides a reference implementation. This implementation can be used as a reference to build applications. But, this implementation is explicitly not meant for use in a production environment (on released phones) [7]; mobile phones developers have to create their own implementation of the standards.

3.4. *Mobile Information Device Profile*

The Mobile Information Device Profile, also known as MIDP, is a key element of J2ME. Combining with the previously mentioned CLDC, MIDP provides a standard Java runtime environment for cell phones and other mobile information devices.

Using MIDP an application can be written once and then being deployed quickly to a wide variety of mobile information devices. MIDP is deployed globally on phones of all popular brands and is supported in most leading development environments [9]. MIDP 2.0 is a backwards compatible revised version of MIDP 1.0. Some new features are added, like more extensive connectivity and end-to-end security.

By adhering to the MIDP, we can assure that our application runs on any commercially available mobile phone that is capable of running Java applications.

3.5. Sun Java Wireless Toolkit

This product, also released by Sun Microsystems, is a collection of tools that come in handy when developing for the CLDC platform. It includes, among other tools, an emulator environment and programs for analyzing performance, optimization and tuning [6].

In addition to tools it includes many examples, documentation and tips making it possible to manage developing applications for mobile phones more quickly.

3.6. Playing audio on mobile devices

In J2ME the package *javax.microedition.media* is needed to support audio and provides upward compatibility with all of the multimedia APIs available. The package was introduced in the MIDP2.0 profile.

3.6.1. Mobile Media API

Mobile Media API (MMAPI) defines the superset of the multimedia capabilities that are present in MIDP 2.0. MMAPI allows one to play mp3 files from a server on J2ME devices. It is built on a high level and its abstraction can be divided in three main classes that form the bulk of operations. These classes are the Player, Control and the Manager class.

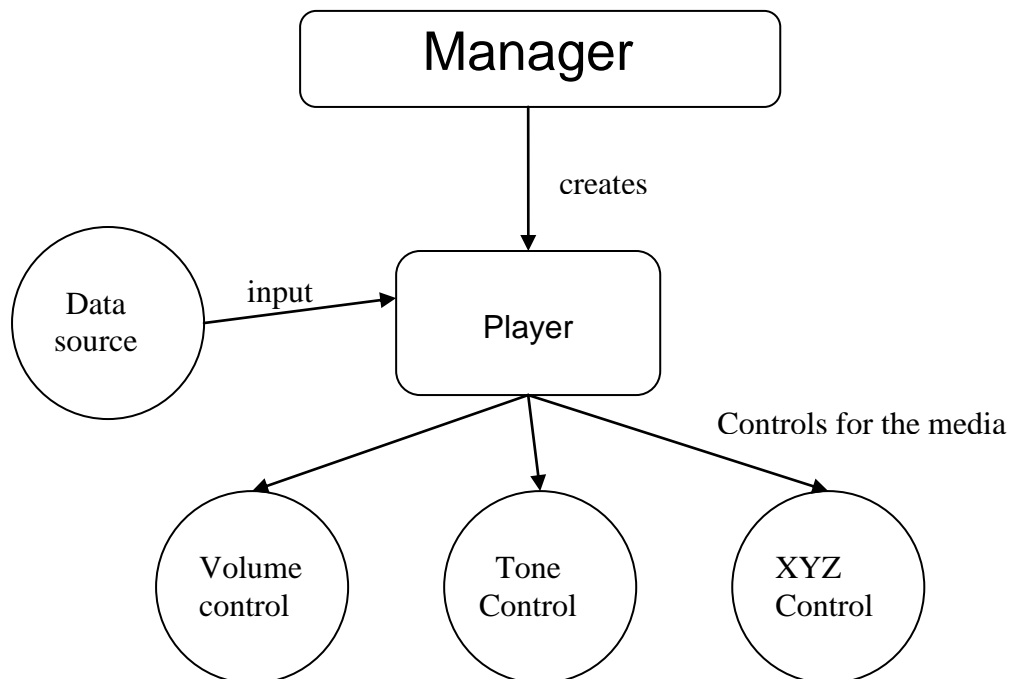


Figure 1 Player creation and management [10]

Figure 1 shows that the Manager class creates Player instances for different media by specifying DataSource instances. DataSource is another abstract class and is used to locate

resources. Though, with the known ways of reading data, there is no need to use it directly. Reminding that not all mobile devices support all the options and MMAPI is designed in such a way that it takes full advantage of the capabilities that are available, despite of those that it cannot support. MIDP 2.0 comes with a subset of the MMAPI which gives a certainty when a device does not support MMAPI, one can still use a scaled down version. This version only supports a minimal set of functions regarding streaming audio.

3.7. Streaming Audio

While the Mobile Media API makes it possible to play audio files on every device containing a speaker, it does not provide services for retrieving the actual audio data from a remote location over a network. Because the time needed for a complete transfer could be quite long, and because the device could have very limited storage capabilities, we need a way to stream audio data from a server to the device.

3.7.1. MIDlet

A MIDlet is an application which only works with devices that have implemented at least Java ME, CLDC and MIDP. The ability to stream media over the network is crucial to create a successful MIDlet application. The MIDlet can play media over the work easily by specifying an HTTP based file.

In this case there are two issues to be considered:

- 1) Media access over the network requires permission from the end user.
- 2) Media access over the network is an operation that can take a lot of time. It is then recommended that this operation should not be done in the main application thread but in a separate thread. Otherwise it will tie up the main user interface for a long period when communicating over the network, which is not permitted [11].

For the first issue, one can use Application Management Software (AMS) to solve the permission problem. The second issue is taken care of just by separating the network media access code in its own thread.

3.7.2. Red5Server

Red5Server is an open source and fast Java server. It is a project dedicated to the interaction between a Flash Player client and a Free Connection Oriented Server, for which the communication protocol RTMP is used [12]. As the most basic operation, Red5Server lets the server operator store audio files on the disc of the server for streaming them to clients. Furthermore, Red5Server supports the following which is useful for the Greeting project: Stream Audio (MP3), Stream Data, Live Stream Publishing, On Demand Streaming and Universal Java Integration. Red5Server will be useful for storing the recorded voice messages and streaming the audio files to and from the clients. This server is using a specialized protocol called RTMP that only can be used by a Flash Server. Thus access is only

possible made by Flash. It keeps a constant and persistent connection with the client, so that the server is able to deliver audio, video or other data types.

3.7.3. RTMP Protocol

RTMP is an acronym for Real Time Messaging Protocol. It is a proprietary protocol developed by Adobe Systems for streaming files over the Internet. The project Red5 aims to produce a feature-complete implementation written in Java [8]. To know which packets are successfully delivered and which have failed transmission, RTMP uses the TCP/IP protocol for packet transmission. The protocol has three variations:

- 1) RTMP protocol which works on top of TCP
- 2) RTMPT which basically is a HTTP wrapper around the RTMP.
- 3) RTMPS is just like RTMPT but works with HTTPS [13]

3.8. Web Service Access

The mobile device on which our application runs needs to access various web services provided by Greetingq. The interactions with their servers include:

- load a list of received voicemail messages
- load (stream) a received voicemail message
- mark messages as read
- load (stream) a voicemail greeting
- upload (send) a new voicemail greeting
- read information about contacts and their associated greetings
- update information about contacts

The interactions not specified as *streaming* will be implemented by sending messages in the modern and extendable language XML, over a standardized HTTP connection. Some, if not most of the operations, need to be secure; that is, they have to be done via an encrypted link. Therefore, we will use a HTTP connection encrypted with SSL.

3.8.1. Network communication with HTTP

The base implementation of Java ME, CLDC, provides a general interface for network communication [1]. Furthermore, we need readily available packages for:

- communication with HTTP (both GET and POST methods)
- composing and parsing messages in the XML format

Packages for communicating over HTTP protocol, both using an encrypted and an unencrypted link, are included in the Mobile Information Device Profile (MIDP) technology. The difficulties of maintaining a connection while roving from one access point (radio tower) to another, and overcoming the differences between packet-switched and circuit-based networks are hidden from the developer by this package.

The MIDP package is available on the all phones that ought to be supported for our software.

3.8.2. Parsing XML

Parsers, in particular XML parsers, are notorious for the size of their memory footprint and their memory usage at runtime. Because mobile phones do not have a great amount of memory available, we need to take care in selection our parser.

There are various free XML parsers to be found on the Internet, with different memory footprints. Here we present an overview of products that are readily available and supported on a standard Java ME configuration [2]:

<i>Product</i>	<i>Size</i>	<i>Type</i>
ASXMLP	6 kB	Push, model
kXML	9 kB	Pull
Xparse-J	6 kB	Model

When reading a document, the *model* type of XML parsers will completely parse the document before giving any results. This can use a lot of memory when the document is large. Therefore, we will use the kXML package for XML parsing. It is freely distributed by the Enhydra Public License, based on the Open Public License, which gives us the right to use the package royalty-free [3].

3.9. Graphical User Interface on mobile phones

Before Java ME technology there were not many choices available for developers for graphical API. At that time it was J2ME. In fact, the only API that was available was the standard LCD user interface packages of `javax.microedition.lcdui` of MIDP. MIDP was only at version 1.0, and there was no Sun Java Wireless Toolkit tool, Netbeans Mobility Pack, or Websphere Device Developer. Today, the current and upcoming APIs of the mobile Java platform allow the developer to create a wide experiences ranging from vector graphics manipulation to 3D object rendering. Nowadays, the Java SE applications even with a Swing user interface can run on some mobile devices.

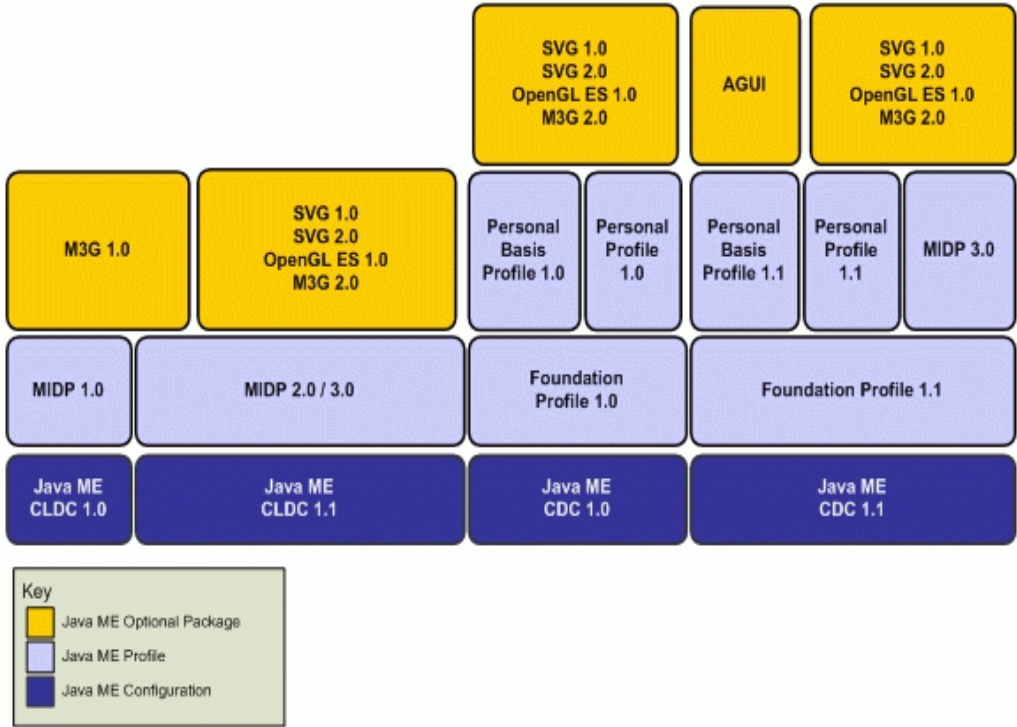
A majority of these APIs available today have been finalized in mobile devices, but some of them are still under specification in JCP²⁴.

The graphical capabilities of the MIDP 1.0 and 2.0 specifications were designed to support the multi-color displays of today's mobile phones. MIDP 3.0 devices will be equipped towards a richer mobile experience, and requires that mobile devices support at least 64k colors. All MIDP implementations support high-level and low-level user interface components. All user interface classes for MIDP devices can be found in the `javax.microedition.lcdui` package, and the high-level user interface components are subclasses of the `Screen` and `Item` classes.

²⁴ Java Community Process is a participative process to develop and revise Java technology specifications, reference implementations, and test suites, the Java Community Process (JCP) program has fostered the evolution of the Java platform in cooperation with the international Java developer community. Source: <http://jcp.org/en/introduction/faq>

One of the major advantages of the high-level API is the fact that high-level GUI components can be used in the mobile devices, thus allowing the device to present the components in an appropriate manner.

In contrast, the low-level APIs for MIDP applications allow the developer to have greater control of the user interface, and therefore the developer has the responsibility for drawing and rendering the display. The major low-level API classes are `Canvas` and `Graphics`: classes for creating gaming applications. But this does not mean that the low-level MIDP API is used only for gaming applications. In the figure bellow you can see Java ME graphical API in a glance [15].



Summary of the Java ME Graphical APIs

3.9.1. **GUI in Java ME and its differences with Java SE**

Java ME does not use AWT or Swing. Instead it defines its own GUI classes which are part of the MIDP API. Reasons for not using AWT or Swing are:

- AWT and Swing are designed with desktop computers. They are windowing elements for a screen with a higher resolution and more capabilities than a mobile phone's screen.
- AWT and Swing are supposed to work with a mouse. Most mobile phones however do not have such a pointing device.

- Most of the features that the AWT and Swing supports, such as the Window Management and overlapping windows, are impractical and impossible on a mobile device.
- AWT and Swing creates event objects during runtime for each GUI event such as mouse-move and button-click. These objects exist only until the event is processed and is later freed during garbage collection. However, the limited processing power and memory of mobile device cannot handle such a heavy process.

Due to these reasons MIDP decided to have its own set of GUI APIs. These APIs are classified as **high level** and **low level** APIs. Both the high level and low level APIs come in the **javax.microedition.lcdui** package.

One of the classes used in MIDP is *Displayable* class. Anything that can be displayed is represented by this class. The *Screen* class is a sub-class of the *Displayable* class and represents screens that take the full screen area. There are three types of screens:

- Screens that have a predefined structure and are for a specific purpose. This means we cannot add any other components to them. For example List and TextBox are of this type.
- General purpose screens made using the Form component. You can add other components to this Form creating custom screens of your desire.
- Screens created using low-level APIs; as a subclass of Canvas, GameCanvas etc. they are used mainly for game applications. [17]

References

- [1] MIDP Network Programming using HTTP and the Connection Framework, Qusay Mahmoud, fetched April 14
<http://developers.sun.com/mobility/midp/articles/network/>
- [2] Parsing XML in J2ME, Jonathan Knudsen, fetched on April 14
<http://developers.sun.com/mobility/midp/articles/parsingxml/>
- [3] Open Public License version 1.0, fetched April 14
<http://kxml.objectweb.org/software/license/opl.html>
- [4] Connected Limited Device Configuration (CLDC); JSR 30, JSR 139 Overview, Sun Microsystems, fetched April, 16
<http://java.sun.com/products/cldc/overview.html>
- [5] J2ME CLDC and K virtual machine: Frequently Asked Questions, Sun Microsystems, fetched April 16
<http://java.sun.com/products/cldc/faqs.html>
- [6] Sun Java Wireless Toolkit for CLDC, Sun Microsystems, fetched April 16
<http://java.sun.com/products/sjwtoolkit/>
- [7] About the Eclipse Foundation, The Eclipse Foundation, fetched April 16
<http://www.eclipse.org/org/>
- [8] Information priority within the RTMP protocol, fetched April 16
http://www.adobe.com/go/tn_16458
- [9] Mobile information Device Profile(MIDP); JSR 37, JSR 118, fetched April 17
<http://java.sun.com/products/midp/overview.html>
- [10] J2ME Tutorial, Part 4: Multimedia and MIDP 2.0, java.net: Vikram Goyal, fetched April 14
<http://today.java.net/pub/a/today/2005/09/27/j2me4.html>
- [11] Streaming media over the network, java.net: Vikram Goyal, fetched April 14
<http://today.java.net/pub/a/today/2005/09/27/j2me4.html?page=3#streaming-media>
- [12] RED5 Server mission, Red5 Server, fetched April 14
<http://www.red5server.com/mission.jsp>
- [13] RTMP protocol Draft, Open Source Flash, fetched April 17
<http://osflash.org/documentation/rtmp>
- [14] Symbian Software Limited
www.wikipedia.org/symbian
<http://developer.symbian.com/main/learning/press/books/pdf/JavaME.pdf>
- [15] GUI APIs in Java ME By Bruce Hopkins, June 2007
<http://developers.sun.com/mobility/midp/articles/guiapis/>
<http://developers.sun.com/mobility/midp/articles/guiapis/#1>
- [16] Symbian Software Limited
<http://developer.symbian.com/main/learning/press/books/pdf/JavaME.pdf>
- [17] GUI in Java ME and its differences with Java SE, Java ME Blog
<http://www.javameblog.com/2007/12/this-is-presentation-that-i-made-to.html>

Appendix B: Requirements Document

Bachelor Project

Visual Voicemail and Greetings

Requirements Analysis Document

Version 6

Computer Science
Faculty of Electrical Engineering, Mathematics and Computer Science



<u>Date:</u>	July 8, 2008
<u>University:</u>	Delft University of Technology
<u>Course:</u>	IN3405 Bachelor Project
<u>Company:</u>	Greetingq B.V.
<u>Authors:</u>	Nathan Bruning Medya Amidi Michelle Cheung
<u>Supervisor:</u>	Ir. H.J.A.M. Geers
<u>Coordinator:</u>	Ir. B.R. Sodoyer

Table of Contents

1.	Introduction	65
2.	Domain Analysis	65
2.1.	Glossary.....	65
2.2.	General Knowledge.....	66
2.3.	Customers and users.....	66
2.4.	Tasks and procedures currently performed	66
2.5.	Competing Software.....	67
2.6.	Similarities across domains and organizations.....	67
3.	Proposed Solution	67
3.1.	Requirements.....	68
3.1.1.	Functional Requirements.....	68
3.1.2.	Constraints.....	68
3.1.3.	Quality Requirements.....	69
3.1.4.	Platform Requirements.....	69
3.1.5.	Process Requirements	69
3.2.	Models.....	70
3.2.1.	Scenarios	70
3.2.2.	Use Case Models.....	74
3.3.	Use case and interaction with the System	75
State Diagram of the Graphical User Interface	80	
3.4.	User Interface	82

1. Introduction

Visual Voicemail or Random Access Voicemail is the standard phone voicemail service with an added visual aspect. Users can select their voicemails and play them in any order of their choice, in a user friendly GUI environment.

Recently several companies in the telecommunications world have integrated a visual element into their voicemail services. The most prominent usage of this technology is found in Apple's iPhone.

This document describes in details the technical requirements needed to realize such a visual voicemail services as a mobile phone application. The project is given to us by the newly started company Greeting B.V. – consisting of three under graduated students at TU Delft University of Technology - in the third quarter of 2008, as our bachelor project. This document is formulated based on information obtained from interviewing the company as our principal.

After studying this document one can attain the technical insight of what this product is capable to offer to its users before being released.

2. Domain Analysis

Telephony and in particular mobile telephony is a service which has a growing amount of users ever since its invention. Together with fast growing wireless technology, the number of services that a mobile phone can offer is increasing dramatically. This has opened a new area in software development that is evolving very fast. As we mentioned in our introduction, our plan is to develop a mobile service called visual voicemail. As everyone probably knows voicemail service provides facilities to the mobile phone users to be able to listen to their messages left by callers when they failed to answer their incoming calls. This service has been in use in most of the countries for the last decades. What is new in this field is the possibility of a ‘visual’ service to manage voicemails.

2.1. Glossary

- ***Greetings:*** voice greetings recorded by called and heard by caller if the called one failed to answer his/her mobile phone.
- ***Platform:*** describes some sort of hardware architecture or software framework (including application frameworks), that allows software to run. Typical platforms include a computer's architecture, operating system, programming languages and related runtime libraries or graphical user interface.
- ***SMS:*** Short Message Service, a mobile phone service which provides the ability to send and receive short text messages.
- ***Voicemail:*** audio messages spoken by the caller when a call is unanswered.

2.2. General Knowledge

Voicemail is implemented by forwarding calls that are not answered, to a voicemail operator line. This line is connected to an advanced answering machine that handles the call; it asks the caller to leave his voice message, and records the message on some storage medium to be listened later by the called person.

2.3. Customers and users

Every mobile phone owner is a potential customer. People that already heavily using their voicemail are most likely interested in the improved service offered by Greeting the company. Even though, people that currently do not use their voicemail because of its bad service could also be persuaded by the new service.

A few not non-disjoint categories of potential customers can be identified:

- **Businessmen:** because of the nature of their work, they are always in contact with people. Sometimes they need to talk in different language with their clients. Besides they are hard to reach because they have many meetings and appointments. It would be very practical for them if they can make different greetings for different groups of people.
- **Employees:** using one phone for both work and private life may not be favourable. It wouldn't be affordable either for many people to buy two different mobiles to separate their work from their personal life.
Especially choosing an applicable greeting is not evident; a more formal greeting would be good for business, but not for the kids. The possibility of making multiple greetings and selecting between voicemails comes in handy here.
- **Teenagers:** recently, a few companies have been very successful at selling ringtones for mobile phones. Many of these ringtones were popular because they are funny. Greetings could cause the same hype: selecting different funny Greetings for friends when they get your voicemail.
- **People with foreign contacts:** being anything from business partners to close family, it would be clear and polite to speak to people in a language they understand. With the Greeting service, one could configure the voicemail service to provide an English greeting to numbers calling from outside The Netherlands.

2.4. Tasks and procedures currently performed

The voicemail service currently being used works through a long procedure with little power given to the user. The user has to dial first a special number given by the provider and then listen to a standard operator- which is sometimes not in a favour of the user, and after that he can listen to the voicemails. During this process, the user does not know who the sender is before playing the voice message. Besides, the duration of a voice message is not specified. He cannot select which message to listen first and which one to ignore or remove either.

The user can record only one greeting message for all callers. There are three kinds of greetings the user can choose from:

- 1) A standard voicemail greeting provided by the mobile network.
- 2) A standard voicemail greeting with the users name spoken by him/her
- 3) A voicemail greeting completely spoken by the user.

2.5. Competing Software

Until now there are only Apple's iPhone and the Blackberry which have a visual voicemail service. Apple has control of the interface; the core of this application is provided by the phone network. Dividing responsibilities in this way creates difficulties with backwards and forwards compatibility when for example Apple wants to change their interface.

Currently, in the USA and Canada a company called Callwave provides users a voicemail service, allowing them to manage their voicemail via Internet on their own computers.

Another company called YouMail provides also extra voicemail facilities such as allowing users to listen to their voicemails both online and on their mobile phones. Another service YouMail presents is personalizing the greetings, i.e. the user can make different greetings for different callers.

GrandCentral, recently taken over by Google, provides among others, an online voicemail service. It collects voicemails from different phone numbers and presents them on a website.

Lastly, another company called GotVoice tries to convert voice messages into text messages by speech recognition technology and subsequently send an SMS message to the user's mobile phone's mailbox.

2.6. Similarities across domains and organizations

All Dutch telecom companies have a voicemail service that comes with the standard phone service. Recording a voicemail message costs just as much as a normal conversation, but listening to recorded voicemail messages differs in price from one provider to another.

In The Netherlands, listening to the voicemail of a land line cannot be free because of regulations of the OPTA (the independent postal and telephone postal authority). Some mobile telephony providers do not charge for using voicemail at all. Most of them inform the user of a new voicemail message by sending them an SMS.

3. Proposed Solution

To have a fast and easy-to-use service, we will develop a mobile application for both voicemail messages and voicemail service settings. Our system provides the user an intuitive interface to listen to received messages. The user is able to browse through a list of messages. Further the product can be used for configuring the Greeting voicemail service. This consists of allowing the user to record Greetings, and linking them to the contact list (choosing who gets which Greeting when they call).

3.1. Requirements

We interviewed the Greeting company staff for the identification of requirements. These requirements have been classified in functional (functions that the program must be able to perform), constraint (a situation the program must be able to cope with, or a property the program must have), quality requirement (how is software quality asserted), platform requirement (in what environment the program must function) or process requirements (how must development be planned and managed).

3.1.1. Functional Requirements

The user must be able to perform the following tasks with the application:

- Work with voicemail messages:
 - o Browse through a list of received voicemail messages
 - o Listen or remove a message
 - o View details²⁵ of a message
 - o Mark a message as listened or new
- Work with the Greeting service:
 - o Browse through a list of Greetings
 - o Listen or remove a Greeting
 - o View and edit the details²⁶ of a Greeting
 - o Change the assigned contacts of a Greeting
 - o Record a new Greeting
- Settings:
 - o Turn voicemail service on and off
 - o Change maximum time limit for new voicemail messages
 - o Choose the default Greeting
 - o Remember the login data
 - o Language settings²⁷
- View “About screen”:
 - o Greeting Logo
 - o Application Version
 - o Platform Information
 - o Website address
 - o Greeting service e-mail address
 - o Greeting hotline

3.1.2. Constraints

The following constraints have been imposed by the principal:

²⁵ Details screen of messages include: phone number of the caller, the date the message is sent and the Greeting (standard/personal) made for the caller

²⁶ Detail screen of greeting include: name of the greeting, date in which the greeting is recorded, list of contacts associated with the greeting, and description.

²⁷ Currently: Dutch and English.

- The application must communicate with the user in Dutch.
- The application graphics must be consistent with the house style of the company.
- The user interface must be designed to provide easy changing language or graphic style services.

3.1.3. Quality Requirements

We must provide documentation for end users as well as for other developers. The end user documentation must be in Dutch and easy understandable for non-technical readers.

As the application (source code) might eventually be sold to a foreign company or investor or further be developed by other English speaking developers, all code and comments must be in English. The technical documentation must be in English and use the formal Javadoc notation. Various UML diagrams should accompany the documentation for better clarification. For easy debugging and consistency with other Greetingq applications, we are required to use some form of application logging.

Finally we need to document and provide a rationale for each choice we make so that all of these choices can be easily re-evaluated.

3.1.4. Platform Requirements

The application must be able to run with complete functionality on the Symbian Operating System 9.3, running the S60 3rd edition platform. For more information about this platform, refer to the Orientation Document.

The version number indicates a minimum; that is, it should run on any newer version of this operating system as well.

3.1.5. Process Requirements

The hard deadline for the application to be completely finished is in the 3rd quarter of 2008. This is because the Greetingq service must be in production state at that time, and will be launched to the public (we however intend to finish it earlier, as indicated in our project schedule).

The first deliverable will be a mock-up (non-functional prototype), in which the user is able to navigate through all menus and select all options, but no functionality is actually implemented.

Listening to voicemail messages will be implemented in the second phase. This makes the application usable as soon as possible and will stress and test the Greetingq servers, which consequently will aid Greetingq in discovering errors on their side.

A third phase consists of implementing the management of greetings; recording, editing and listening greetings.

The last phase will make the system robust in the sense that it does not fail under erroneous conditions (for example, failing internet connection).

For easy file and version management, the principal will provide a SVN server for all artefacts (code, documentation, and graphics).

For easy and smooth communication, one contact person is appointed between the project members and the company, and between the project group and the TU Delft supervisor Hans Geers. Contact information of the people involved can be obtained from the **Project Schedule**.

Finally, we need to know what is done and what decision have been made during the process. For this reason, in a document called the **Project Log** we keep track of our daily work and put this document on the public SVN server.

3.2. Models

3.2.1. Scenarios

1) Log In

Actor: Peter

Flow of events:

1. Peter is in a meeting and is not able to answer his incoming calls. It is now after the meeting and he is in his office. He opens myGreeting mobile application and waits for the program to be loaded. He enters his username and password.
2. His login information is correctly verified and Peter has logged in. Peter sees the window screen with messages.

2) Playing a message

Actor: Peter

Flow of events:

1. Peter has successfully logged in.
2. Peter sees a new message and selects it. The voice message is now playing.

3) Deleting a message

Actor: Peter

Flow of events:

6. Peter successfully logged in and sees that he has a few new messages.
7. Peter notices that a person whom he does not like has left a message. He chooses this new voice message and goes to 'Menu'.
8. Here he selects the option 'Delete' and presses 'OK'.
9. A confirmation message is shown and asks if he is sure to delete the message.
10. Peter chooses "Yes" and the message is deleted from the system.

4) Marking a message as new

Actor: Peter

Flow of events:

1. Peter successfully logged in and has just listened to a new message. He returns to the window of messages
2. Peter chooses the listened message and goes to 'Menu'.
3. He selects the 'Mark' option. The system marks the message as new. Peter sees that the message has the "New Message" pictogram now.

5) Marking a message as listened

Actor: Peter

Flow of events:

1. Peter has logged in and sees his new messages.
2. He notices that a person who is not important for him has left a message. He chooses this message and goes to the 'Menu'
3. He selects the 'Mark' option and presses 'OK' on his phone. The system marks the message as listened. Peter sees that the message has the "Old Message" pictogram now.

6) Add new Greeting

Actor: Jan

Flow of events:

1. It is the first time Jan using the Greetingq mobile application. He has successfully logged in.
2. Jan selects 'Main Menu' and the Main Menu screen is shown
3. He is excited and he chooses the 'New Greetingq' option from the menu screen.
4. A list with all contact persons appears. John excitingly chooses some of his friends and presses 'OK'.
5. He can now record his Greetingq. He presses 'Record' to record.
6. When he finishes he presses on 'Stop'.
7. The first Greetingq is ready and Jan hears his new recorded Greetingq.
8. He is happy with the Greetingq and chooses to send. The system sends the Greetingq to the server and asks Jan to be patient.
9. The system has successfully sent the Greetingq and displays the window for the name and description of the Greetingq.
10. Jan fills 'First' and 'First Greetingq' as its name and description, respectively. And presses 'OK'. The system saves this new Greetingq, and returns to the list of Greetingqs.

7) Edit Greetingq

Actor: Jan

Flow of events:

1. Jan has logged in and has just added a new Greetingq.
2. Jan changes his mind and wants to edit the Greetingq. He selects the 'First' Greetingq and goes to the 'Menu'.
3. He selects the 'Edit' option and an 'Edit Greetingq' window is shown.
4. He changes the name 'First' into 'Friends' and presses on 'OK'. The system saves the change and returns to the details of the updated Greetingq.

8) Add contacts to a Greeting

Actor: Jan

Flow of events:

1. Jan successfully logged in and is now watching his 'Friends' Greeting. He wants to add a new friend into this group.
2. He goes to 'Menu' and selects 'Add contact'. He sees a list of all his contacts from his mobile phone now.
3. He selects the one he wanted to add and presses 'OK'. The system saves the contact.

9) Delete contacts from a Greeting

Actor: Jan

Flow of events:

1. Jan has just added contacts to his 'Friends' Greeting. And he discovers that one of the associated contacts 'Vincent' is his boss. He wants to delete this contact.
2. Jan selects in 'Menu' the 'Delete contact' option. He sees a list of the contacts for this Greeting.
3. Jan selects 'Vincent' and presses on 'OK'
4. The system deletes the contact from the Greeting. Jan now sees the 'Friends' Greeting without 'Vincent' as one of the contacts.

10) Playing a Greeting

Actor: Sandra

Flow of events:

1. Sandra has just got a new contact in her phone book. She logs in and selects 'Greetings' in the 'Main Menu'.
2. Sandra sees a list of Greetings and selects one of the Greeting named 'Colleagues'.
3. She wants to listen if this Greeting is suitable for the new contact. She presses 'Play'.
4. She listens to the Greeting and after 20 seconds she presses on 'Stop' and decides to link this Greeting to the new contact.

11) Remember password

Actor: Jan

Flow of events:

1. Jan has just received an SMS-notification about a new voice message. He logs in for the second time. Because he does not want to log in every time, he chooses 'Yes' for 'Remember Password' under the 'password' field.
2. Jan presses on 'OK'. The system logs in and recognizes this combination and saves this setting on the phone.

12) Edit settings

Actor: Monica

Flow of events:

1. Monica is using myGreeting application for the first time. She installs it and logs in.
2. She wants to be sure that the voicemail service is working as she wants it. So the first thing she does is selecting "Settings" in the "Main Menu".

3. In the “Settings” there are five option fields: the first one she sees is the “Maximum message length(in sec)”. She thinks that most of the people will not talk too long, thus she types “60” in the field.
4. Because this is the first time she is using this service, she chooses Yes at “Voicemail service activated?”. Now, she can use this voicemail application.
5. There is no change needed for the other options, and then she presses ‘OK’. The system saves this changes and returns to the message window.

13) View information

Actor: Jan

Flow of events:

1. Jan wants to know if he has got the latest version of the program. He logs in and goes to the ‘Main Menu’.
2. He selects ‘About’ and the system displays a window with information about Greeting (website, address, service hotline), the version number and platform.

3.2.2. Use Case Models



Figure 1. Use case diagram

3.3. Use case and interaction with the System

- 1) **Use case:** Log in
Related use cases: None
Actors: Mobile phone user
Goals: The user wants to use the Greeting service with the mobile application
Current situation: A login screen is shown.
Steps:

User action	System reaction
1. User fills in his username and password.	
2. User presses on "OK"	3. System validates the username and the password.
	4a. (success) The system shows the Main window (Message window). 4b. (failure) The system reports a failure of log in.
5b. User presses on "OK"	6b. System shows the login window again

- 2) **Use case:** Play message
Related use cases: Log in
Actors: Mobile phone user
Goals: The user wants to listen to a new voice message.
Current situation: A list of messages is shown.
Steps:

User action	System reaction
1. User selects a message.	2. The system plays the voice message and displays the details of it.
	3. The system marks the voice message as listened.

- 3) **Use case:** Delete message
Related use cases: Log in, Play message
Actors: Mobile phone user
Goals: The user wants to delete a message which he has already heard.
Current situation: The message list is currently shown on the screen.
Steps:

User action	System reaction
1. User presses on 'Menu'.	2. The system shows a pop up list of options.
3. User chooses a message and selects 'Delete'	4. The system shows a confirmation to delete the voice message.
5. User presses on 'Yes'.	6. The system deletes the voice message and shows the messages list display.

- 4) **Use case:** Mark message as new

Related use cases: Log in, Play message
Actors: Mobile phone user
Goals: The user has listened to a message and found that the message was important to him. He wants to mark this message to catch his attention the next time.

Current situation: A list of messages is shown.

Steps:

User action	System reaction
1. User presses on 'Menu'.	2. The system shows a pop up list of options.
3. User selects 'Mark'	4. The system marks the message as not listened and returns the updated list of messages. The message has a "New message" pictogram.

5) **Use case:** Mark message as listened

Related use cases: Log in, Play message

Actors: Mobile phone user

Current situation: A list of messages is shown.

Goals: The user think that a message is not important but not yet to be deleted. He wants to mark it as listened.

Steps:

User action	System reaction
1. User presses on 'Menu'.	2. The system shows a pop up list of options.
3. User selects 'Mark'	4. The system marks the message as listened and returns the updated list of messages. The message has a "Old message" pictogram.

6) **Use case:** Add Greeting

Related use cases: Log in

Actors: Mobile phone user

Goals: The user wants to add a personal Greeting for some (new) contacts.

Current situation: The main menu is shown on the display.

Steps:

User action	System reaction
1. User selects "New Greeting" in the 'Main Menu'.	2. The system shows two possible type for the new Greeting: Personal or Default Greeting. It asks the user to select one of them.
3. User selects 'Personal' and presses on 'OK'.	4. The system shows a list of contacts from the phone book.
5. User chooses the contacts he want to add and presses on 'OK'.	6. The system shows a voice recorder window.
7. User presses on 'Record'.	8. The system starts to record the voice of the user.
9. User presses on 'Stop'.	10. The system stops the recording. It

	displays the new Greeting with the associated contacts. The recorded Greeting is playing.
14. User listens to the Greeting and presses on 'Stop'.	14. The system plays the recorded audio file.
15. User presses on 'Save'	16. The system saves the audio file. Asks the user to wait and then displays the screen with fields for the name and description.
17. User fills the name for the Greeting and presses on 'Save'.	17. The system saves the name. And displays the 'Main Menu' window

7) **Use case:** Play Greeting
Related use cases: Log in
Actors: Mobile phone user
Goals: The user wants to listen to a Greeting he has added.
Current Situation: Message screen
Steps:

User action	System reaction
1. User presses on 'Main Menu'.	2. The system shows a popup list of options.
3. User selects 'Greetings'.	4. The system shows a list of the Greetings.
5. User selects a Greeting.	6. The system shows the details of the Greeting, plays the selected Greeting and displays its details.
7. User presses on the 'Menu'.	8. The system shows a popup list of options.
9. User selects "Play".	10. The system plays the Greeting.

8) **Use case:** Edit Greeting
Related use cases: Log in, Play Greeting
Actors: Mobile phone user
Goals: User wants to change information (title and/or description) of a Greeting.
Current Situation: A Greeting has been selected and is shown with details on the display.
Steps:

User action	System reaction
1. User presses on 'Menu'.	2. The system shows a popup list of options.
3. User selects 'Edit'.	4. The system shows an edit window of the Greeting.
5. User changes the title and its description.	
6. User presses on 'OK'.	7. The system saves the changes and return the updated Greeting.

9) **Use case:** Add contacts
Related use cases: Log in, Play Greeting
Actors: Mobile phone user
Goals: The user wants to add some (new) contacts to a greeting.
Current Situation: A Greeting has been selected and is shown with details on the display
Steps:

User action	System reaction
1. User presses on 'Menu'.	2. The system shows a popup list of options.
3. User selects 'Add Contacts'.	4. The system shows all the contacts stored in the mobile phone.
5. User selects of the contacts he wants to add.	
6. User presses on 'OK' .	7. The system links the selected contacts to the Greeting and returns the updated Greeting.

10) **Use case:** Delete contact
Related use cases: Log in, Play Greeting
Actors: Mobile phone user
Goals: The user wants to delete a contact of a Greeting
Current Situation: A Greeting has been selected and is shown with details on the display
Steps:

User action	System reaction
1. User presses on 'Options'.	2. The system shows a popup list of options.
3. User selects 'Delete Contacts'.	4. The system shows all the contacts linked to the Greeting.
5. User selects the contact he wants to delete.	
6. User presses on 'OK' .	7. The system deletes the selected contact from the Greeting and returns the updated Greeting without the deleted contact.

11) Use case: Settings
Related use cases: Log in
Actors: Mobile phone user
Goals: The user wants to change his preferences(voicemail service on, maximum length 120 seconds, 'General' as default Greeting and Language is Dutch).
Current situation: Message screen
Steps:

User action	System reaction
1. User presses on 'Menu'.	2. The system shows a popup list of options.
3. User selects 'Settings'.	4. The system shows all settings for the user.
5. User fills in 120 seconds for the maximum message length. Then he chooses 'Yes' for the voicemail service. Next, he selects 'General' as Default Greeting. And he switches the language from 'English' to 'Dutch'. And finally he presses on 'Save'.	6. The system saves the changes and returns the updated settings in Dutch.

12) Use case: About
Related use cases: Log in
Actors: Mobile phone user
Goals: The user wants to know where he can find more information about this mobile application.
Current situation: Main Menu
Steps:

User action	System reaction
1. User selects 'About'.	2. The system shows detailed information about the software. (Version number of the application, platform, Greeting company website, Greeting company service email address and hotline.

State Diagram of the Graphical User Interface

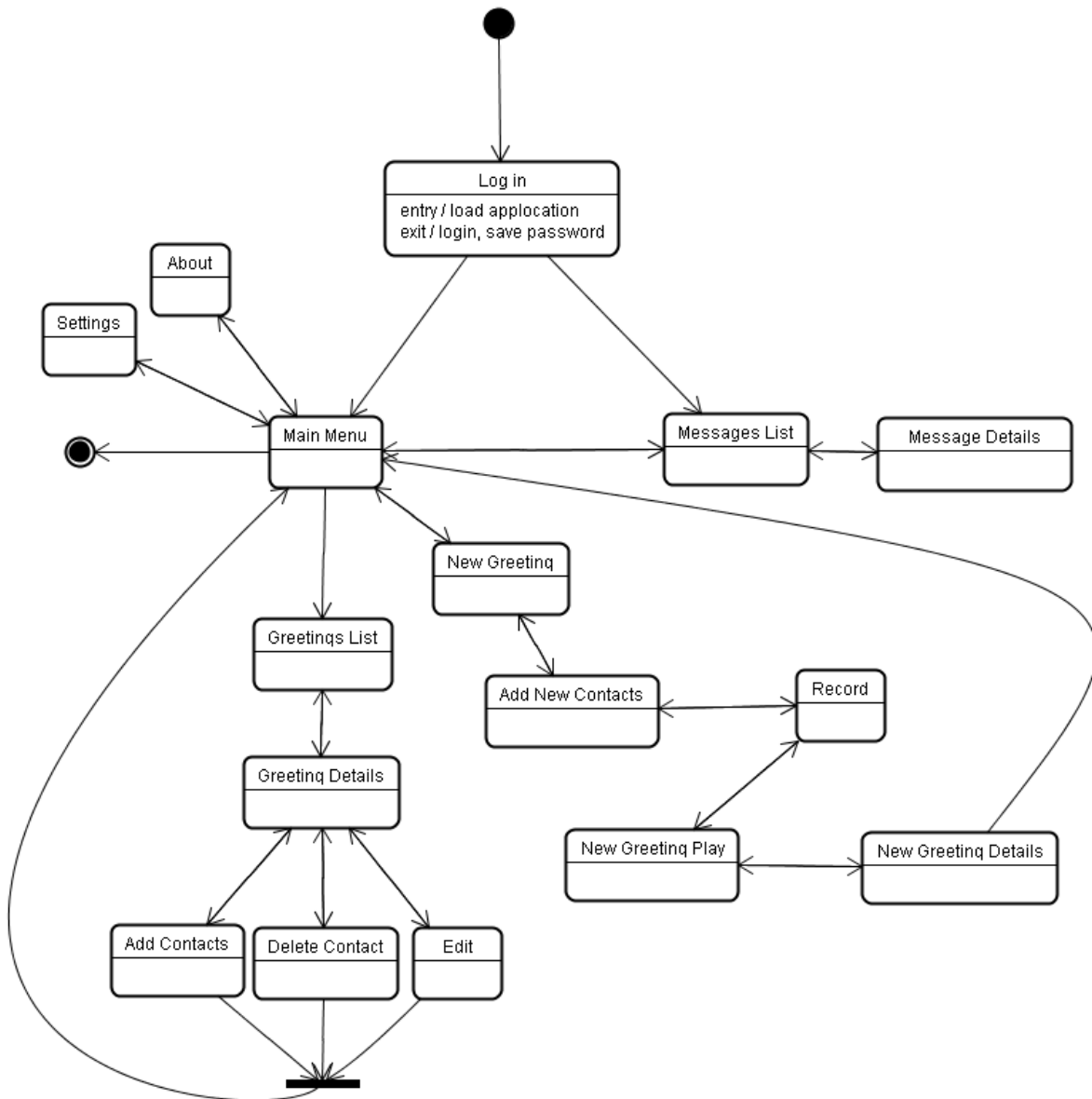


Figure 2. State diagram of the user interface

In the **Main Menu** state there are the following options:

- Messages, leading to the **Messages List** state (screenshot no.3 in section 3.4)
- Greetings, leading to the **Greetings List** state (screenshot no.5)
- New Greeting, leading to the **New Greeting** state
- About, leading to the **About** state (screenshot no. 11)
- Settings, leading to the **Settings** state (screenshot no. 10)

In the **Messages List** state, the Menu button will reveal a popup list with the following options:

- Details, leading to **Message Details** state (screenshot no.4)
- Delete
- Mark message (depending on the message state, mark the message new or old)

In the **Message Details** state, the Menu button will reveal a popup list with the following options:

- Play (only available when it is paused or stopped)
- Pause (only available when it is playing the Message)
- Stop (only available when it is playing the Message)

In the **Greetings List** state the following options are available:

- Details , leading to the **Greeting Details** state (screenshot no. 6)
- Delete

In the **Greeting Detail** state (screenshot no.6), the following options are available:

- Play (only available when Greeting is not playing)
- Pause (only available when it is playing)
- Stop (only available when it is playing)
- Edit, leading to the **Edit Greeting Details** state (screenshot no. 7)
- Add contacts, leading to the **Add Contacts** state (screenshot no.8)
- Delete contact, leading to the **Delete Contact** state

In the **New Greeting** state, the following option is available:

- OK, confirmation of the type, leading to the **Add New Contacts** state

In the **Add New Contacts** state, the following option is available:

- OK, confirmation of the contacts, leading to the **Record** state (screenshot no. 9)

In the **Record** state, the following options are available:

- Record
- Stop, (available when it is recording) leading to **New Greeting Play** state

In the **New Greeting Play** state, the following option is available:

- Save, leading to the **New Greeting Details** state

In the **New Greeting Details** state, the following option is available:

- Save, leading to the **Greetings list** state

In the four states in the centre column, a button named **Main Menu** (screenshot no. 12 in 3.3) is presented that leads to the **Main Menu** state.

In all states on the right side of the state diagram, a button named **Back** or **Cancel** is presented leading to the state to the left of it.

3.4. User Interface

<p style="text-align: center;">greetingq make it personal</p> <p style="text-align: center;">Welcome to myGreeting!</p>	<p style="text-align: center;">Log in</p> <p>Username <input type="text"/></p> <p>Password <input type="password"/></p> <p>Remember Password? Yes <input type="checkbox"/></p> <p style="text-align: right;">Exit OK</p>	<p style="text-align: center;">Messages</p> <table border="1"> <tbody> <tr><td>Richard Stronkman ▶</td></tr> <tr><td>Ruben van Eijnatten ▶</td></tr> <tr><td>Arthur Tolsma ▶</td></tr> <tr><td>Richard Stronkman ▶</td></tr> <tr><td>Ruben van Eijnatten ▶</td></tr> <tr><td>Richard Stronkman ▶</td></tr> <tr><td>Ruben van Eijnatten ▶</td></tr> </tbody> </table> <p style="text-align: right;"> <input type="button" value="Details"/> <input type="button" value="Delete"/> <input type="button" value="Mark"/> </p> <p style="text-align: center;">Main Menu Menu</p>	Richard Stronkman ▶	Ruben van Eijnatten ▶	Arthur Tolsma ▶	Richard Stronkman ▶	Ruben van Eijnatten ▶	Richard Stronkman ▶	Ruben van Eijnatten ▶
Richard Stronkman ▶									
Ruben van Eijnatten ▶									
Arthur Tolsma ▶									
Richard Stronkman ▶									
Ruben van Eijnatten ▶									
Richard Stronkman ▶									
Ruben van Eijnatten ▶									
<p style="text-align: center;">Message Details</p> <div style="text-align: center;">  Playing... </div> <p>From: Ruben van Eijnatten Recorded on: 31-01-2008 10:45</p> <p style="text-align: center;">Back Menu</p>	<p style="text-align: center;">Greetings</p> <table border="1"> <tbody> <tr><td>Voor Richard</td></tr> <tr><td>Johan Cruijff ' Da's logisch...'</td></tr> <tr><td>Standaard begroeting</td></tr> </tbody> </table> <p style="text-align: right;"> <input type="button" value="Details"/> <input type="button" value="Delete"/> </p> <p style="text-align: center;">Main Menu Menu</p>	Voor Richard	Johan Cruijff ' Da's logisch...'	Standaard begroeting	<p style="text-align: center;">Greeting 'Voor Richard'</p> <div style="text-align: center;">  </div> <p>Name: Voor Richard Description: bla bla bla Recorded at: 31-01-2008 10:45 uur</p> <p>Associated contacts:</p> <table border="1"> <tbody> <tr><td>Richard Stronkman</td></tr> <tr><td>Richard Stronkman (thuis)</td></tr> </tbody> </table> <p style="text-align: center;">Back Menu</p>	Richard Stronkman	Richard Stronkman (thuis)		
Voor Richard									
Johan Cruijff ' Da's logisch...'									
Standaard begroeting									
Richard Stronkman									
Richard Stronkman (thuis)									
No. 1	No. 2	No. 3							
No. 4	No. 5	No. 6							

Edit Greeting

Name

Description

Cancel OK

No. 7

Add Contacts


Select contacts for the Greeting:
 'Voor Richard'

<input checked="" type="checkbox"/>	Richard Stronkman
	Richard Stronkman (thuis)
<input checked="" type="checkbox"/>	Ruben van Eijnatten
	Arthur Tolsma
	Michelle Cheung

Cancel OK

No. 8

New Greeting



15 sec

Recording.

Back Stop

No. 9

Instellingen

Maximum message duration:

Activate voicemail service?
 Yes

Remember Pasword.
 Yes

Default Greeting:
 Hello

Language:
 English

Cancel Save

No. 10

About

greeting make it personal

Version: 1.1
 Platform: Windows Mobile 5.0
 © Copyright Greeting B.V. 2008

Visit: www.greeting.com
 Email
 address:support@greeting.com
 Service Hotline: 015 2400 200

OK

No. 11

MainMenu

Messages

Greetings

New Greeting

About

Settings

Exit Select

No.12

Appendix C: System Design Document

Bachelor Project

Visual Voicemail and Greetings

System Design Document

Version 4

Computer Science
Faculty of Electrical Engineering, Mathematics and Computer Science



<u>Date:</u>	July 8, 2008
<u>University:</u>	Delft University of Technology
<u>Course:</u>	IN3405 Bachelor Project
<u>Company:</u>	Greetingq B.V.
<u>Authors:</u>	Nathan Bruning Medya Amidi Michelle Cheung
<u>Supervisor:</u>	Ir. H.J.A.M. Geers
<u>Coordinator</u>	Ir. B.R. Sodoyer

Table of Contents

A.	Purpose	87
B.	General priorities	87
C.	Outline of the design	87
D.	Major design issues	87
1.	Subsystem Decomposition	87
2.	Hardware/Software Mapping	88
3.	Persistent Data Management	88
4.	Global Resource Handling and Access Control	88
E.	Detailed design.....	89
1.	Class Diagrams.....	89
2.	Sequence diagrams	94
3.	Algorithms.....	98
F.	MoSCoW	99
G.	Implementation Plan	99
H.	Code template.....	100

A. Purpose

This document describes the most important aspects of the design of the visual voicemail Greetingq system, presenting a framework in which the actual implementation can be build.

B. General priorities

The decisions made in this document are based on the following priorities: maintainability, simplicity, efficiency, safety, reliability and finally ergonomics.

C. Outline of the design

We chose to use the Model View Controller (MVC) architectural pattern to build the application. Using this pattern ensures a clear division between the presentation (view) and the application logic (controller). Therefore, adapting the layout for different phones or for different companies will only require changes in the view package.

The model package provides storage of data. It parses data received in XML format from the web services, and stores it in Java classes. When an object needs to be sent to the server, the package has to convert from a Java class to an XML object.

The controller package facilitates data retrieval and sending, responds to user stimuli and handles error conditions.

D. Major design issues

1. *Subsystem Decomposition*

We can divide the system to be built in the following subsystems:

- **Model**
 - o **Data Storage** (model): several classes are used to store and pass model data, such as greetings, messages and contacts
- **View**
 - o **GUI**: displays the controls on the screen
- **Controller**
 - o **Command Handlers**: react to actions in the GUI (such as keystrokes).
 - o **Server Access**: handling all communication with the server, including:
 - Formatting messages / responses
 - Handling Internet connection failure
 - o **Streaming Media**: real-time streaming of audio data from (playback) and to (recording) the Greetingq servers
- **Auxiliary Services**: providing easy access to various external data, for example:
 - o Language data (all displayed texts can be localized)
 - o Style data (all graphics and colours can be styled)
 - o User Settings (some preferences are stored persistently on the mobile phone)

2. Hardware/Software Mapping

There are two hardware devices involved with the application. First is the mobile phone which runs the Java code and communicates with the Internet. Secondly, the Greetingq servers keep all voicemail, greeting, contact and service settings data. Note that we are not going to implement the Greetingq servers. These servers runs software already created by Greetingq (and currently in beta phase, in which the server serves about 50 testers).

3. Persistent Data Management

As just mentioned, the account data (voicemail messages, user settings, recorded greetings) is stored on the Greetingq servers. The data that has to be stored on the mobile phone is limited to data that is important only for our application. This data is stored on the mobile phone by means of the Java Persistence Package (the actual storage location is mobile phone dependent; it could be on-device non-volatile memory, a storage card, or a hard drive).

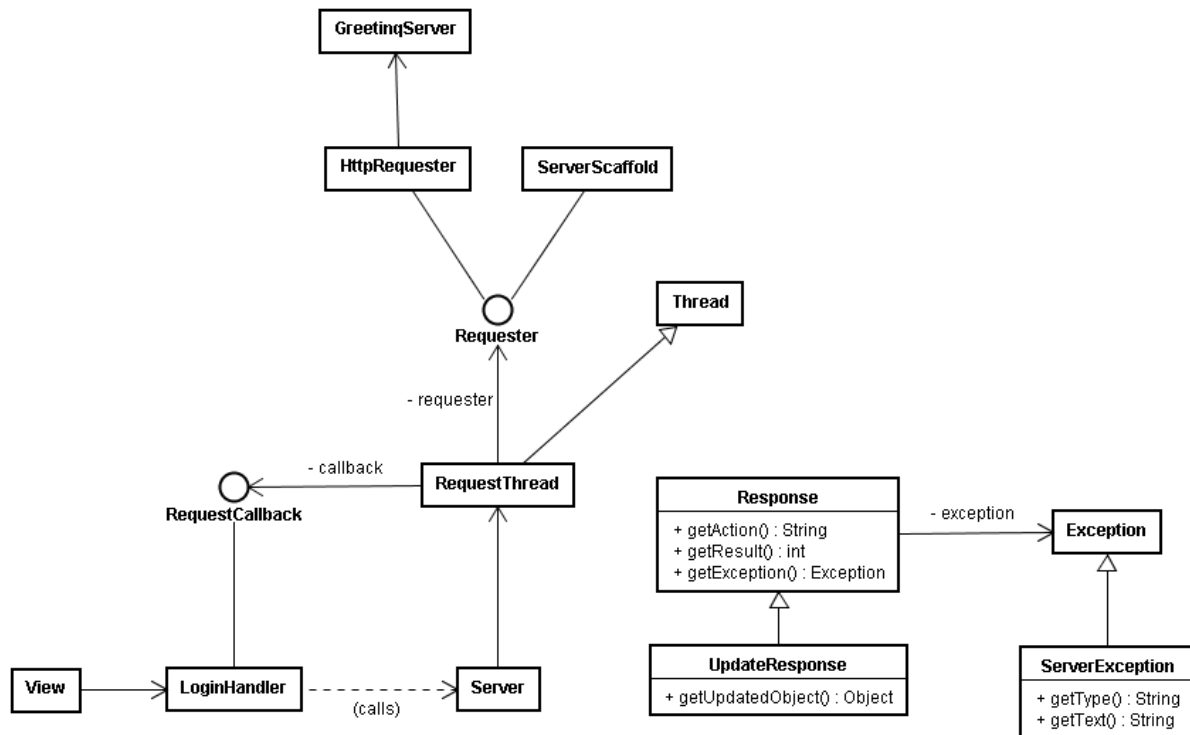
4. Global Resource Handling and Access Control

Global resources are all managed by the Greetingq servers and hence are not an issue. Access control is provided by letting the user enter a username and a password before using the application. When not logged in, the server denies any requests, thus making the application unusable.

E. Detailed design

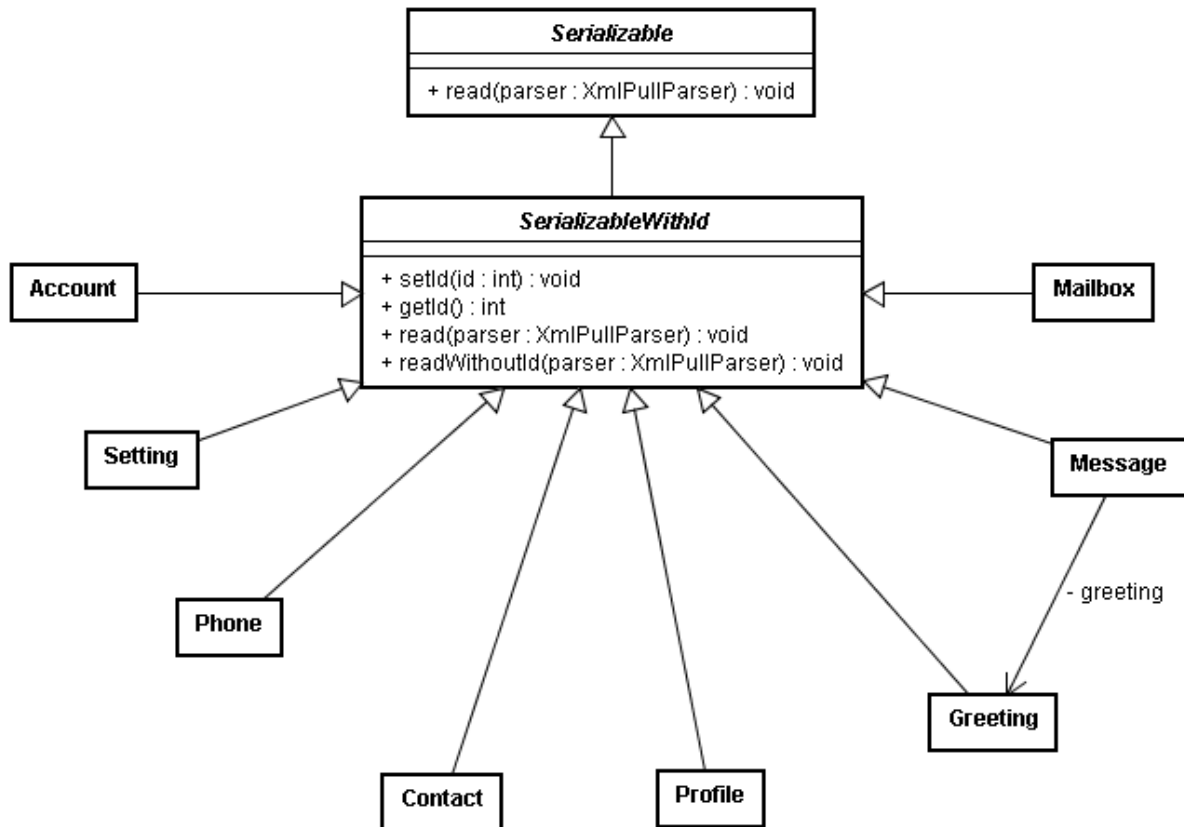
1. Class Diagrams

Controller



This figure presents the architecture of the controller. For network access, a handler calls the appropriate method on the `Server` object, which starts a `RequestThread` so the user interface will not be stalled. This thread will use `HttpRequester` for Internet access, or `ServerScaffold` (which simulates a connection to a server) if configured to do so. A response received in this worker thread will be converted to a `Response` object and is then queued in the main thread for handling by a `RequestCallback`-implementing class (handler). One such handler, `LoginHandler`, is shown here, but for every `Screen` class (see below) there is one handler.

Model



The data model is originating from the Greetingq webservice data model. This model is described as XML objects in the *Greetingq Webservice Specification* document.

A short overview of the objects in the system:

Account: stores information about the users account, such as username, password, and personal information.

Contact: a contact of the user, consisting of a phone number and a name.

Greeting: consists of name, description, and audio file of a greeting.

Message: a received voicemail message. It includes data such as sender, date and the associated greeting.

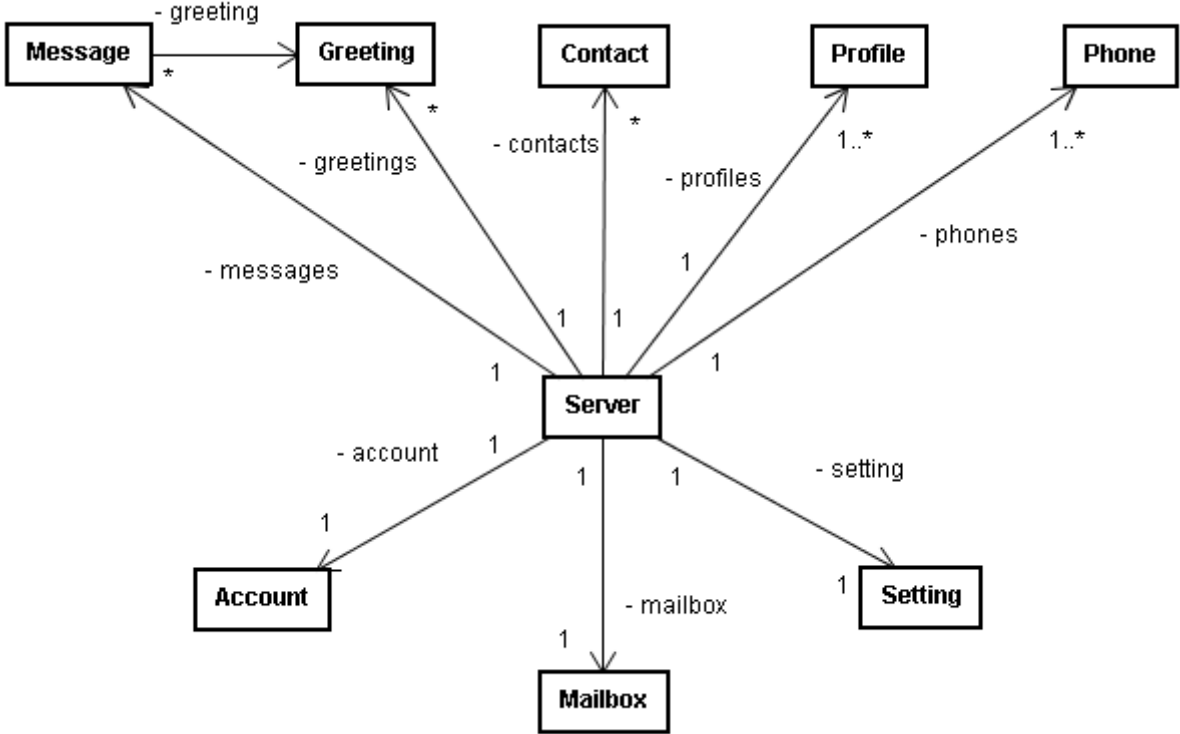
Phone: a phone owned by the user, on which it can receive voicemail messages.

Profile: describes which contacts will hear a specific greeting when they get the voicemail of the user. Associates **Contact** objects with **Greeting** objects.

Setting: all the settings configured by the user.

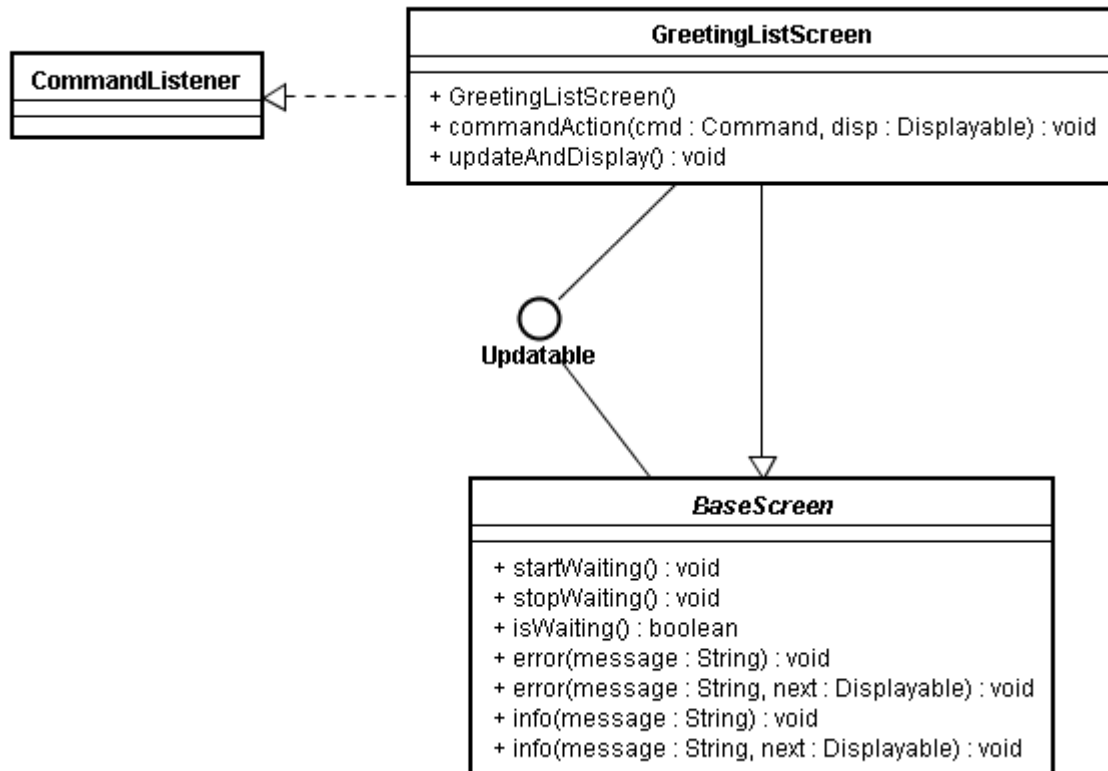
Mailbox: configures the current voicemail service for the user. Options include enabling/disable the Greetingq service and choosing the active profile.

Serializable is the base class for data storage. It provides a function to read and parse the object from a XML stream. Storage is implemented with a collection of elements of various types. We chose not to use specific attributes in the data classes for each data element, because the specification is likely to change. With the chosen approach we can easily adapt to such changes.



The `Server` class is implemented as a *singleton*. It stores all data objects and thus it represents the data model.

View

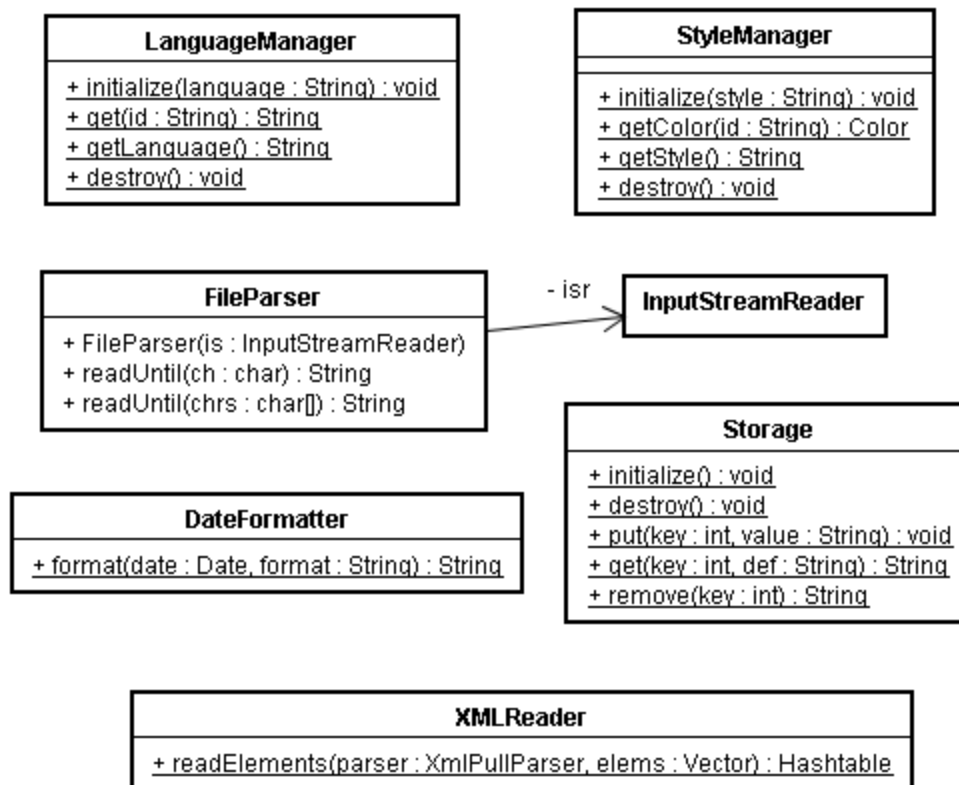


The user interface consists of a collection of screens that display information, and pass user stimuli (pressing a button) to the associated controller (handler) class. This decoupling of user interface and control logic classes allows for changing the graphical presentation later.

The interface *Updatable* specifies that all screens should have a method that refresh the screens content; this provides a way for some external event (i.e. receiving a web service response) to propagate an update of the data model to the user interface.

Please note that for clarity we displayed just one screen (*GreetingListScreen*) in the diagram, but there is one screen class for every user interface state described in the Requirements Document.

Auxiliary Classes



These classes handle language management (including date localization), style management, simple file parsing, XML reading and local persistent data storage.

2. Sequence diagrams

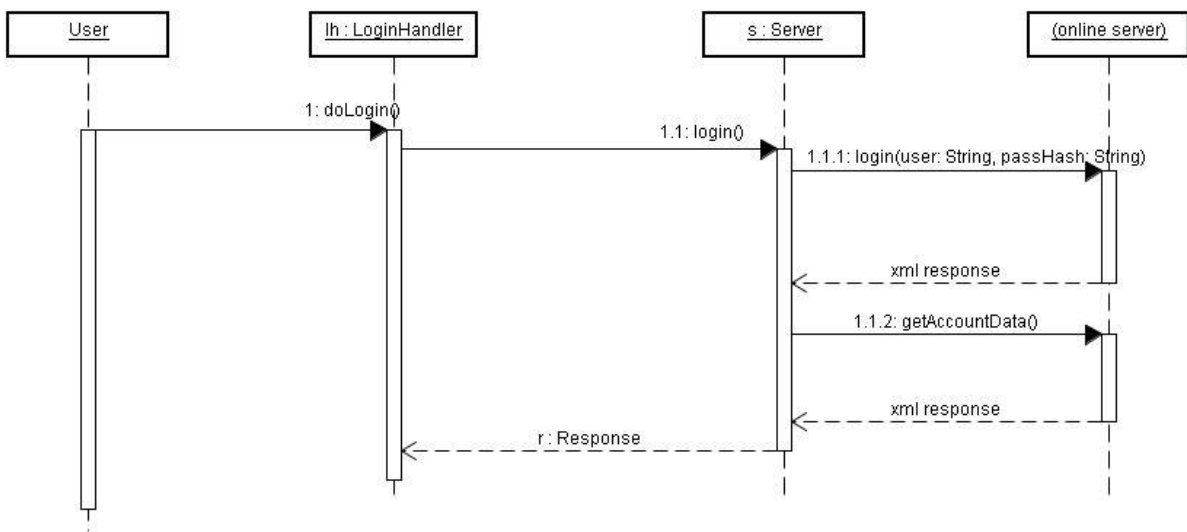


Figure 1 Login sequence diagram

Here we show the login process and the remote server access. XML responses are parsed into a `Response` object and then returned to the handler that requested the operation.

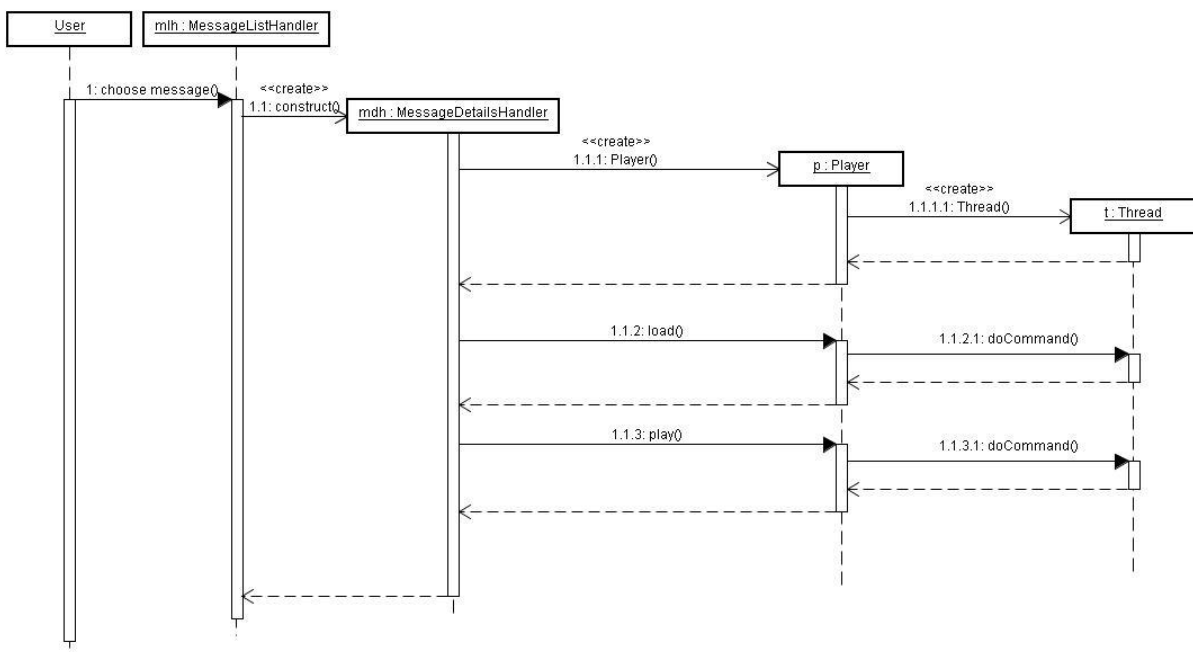


Figure 2 Play message sequence diagram

When a message is chosen from the list of messages, the user proceeds to the details screen. This screen creates a player (which creates an associated thread) that handles the audio playing. The `Player` communicates with the `Thread` by passing it commands which are then processed asynchronously.

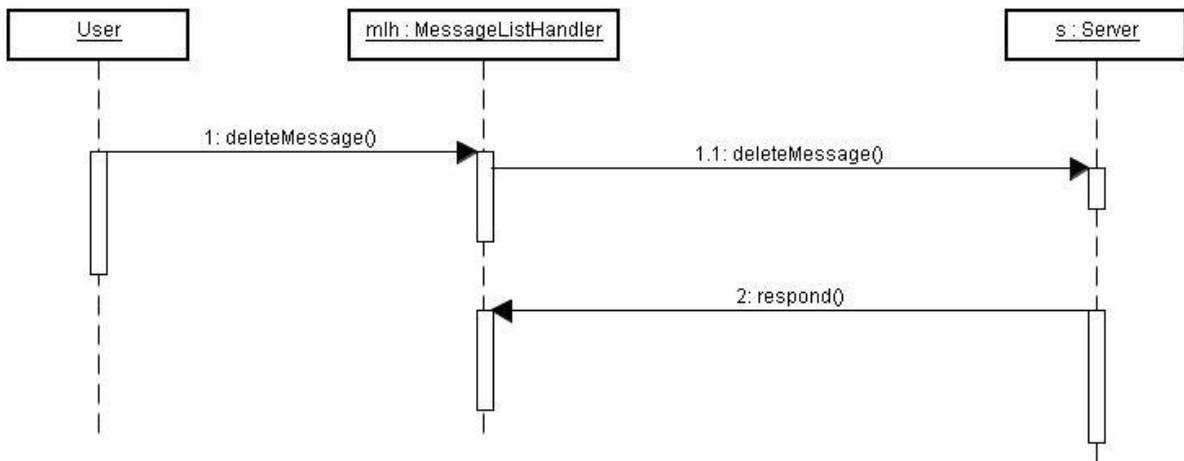


Figure 3 Delete message sequence diagram

Here we have shown in detail the asynchronous nature of the server requests.

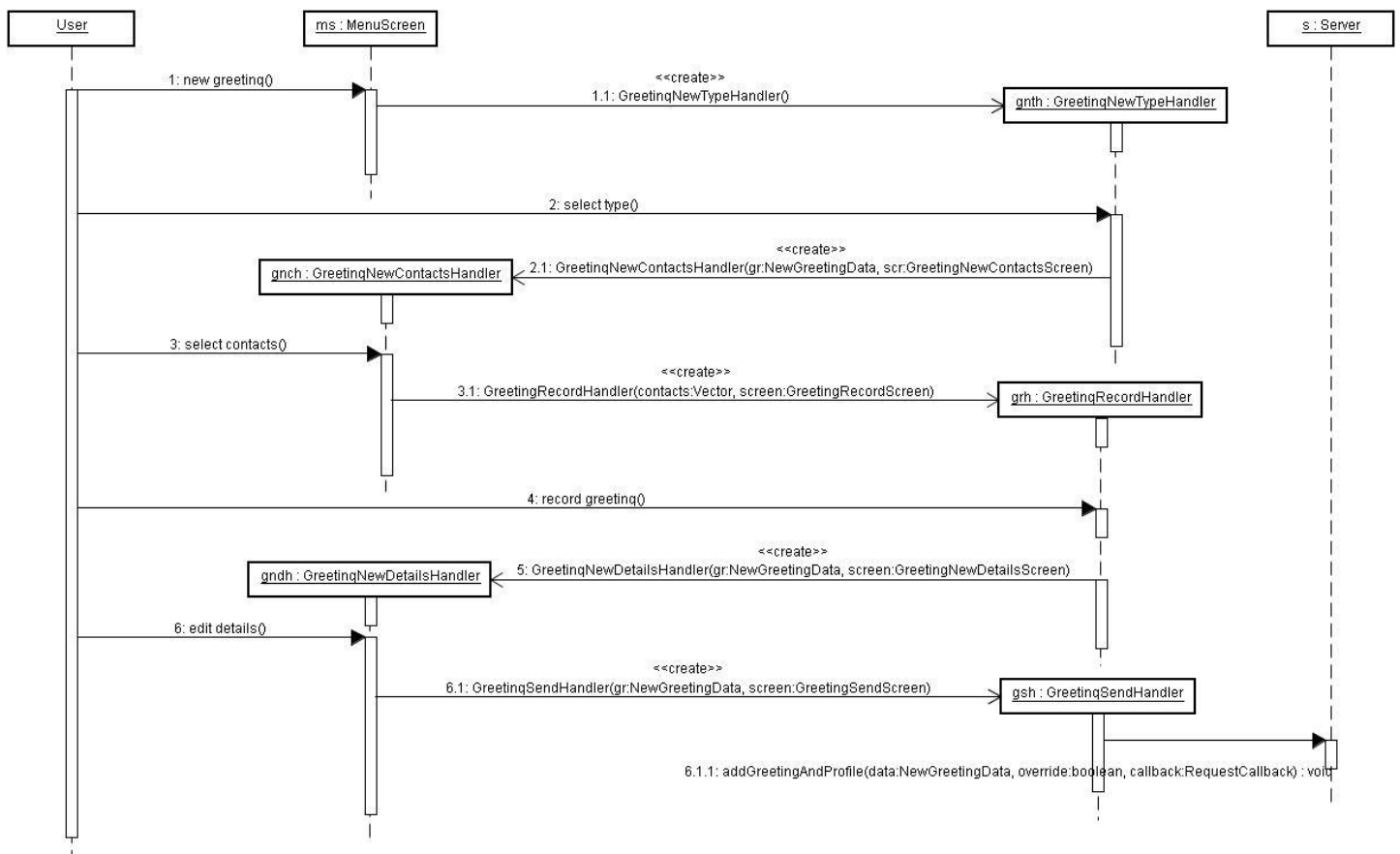


Figure 4 Add greeting sequence diagram

This shows the various steps (screens) involved when creating a greeting. The last step is sending the audio file and greeting data to the server.

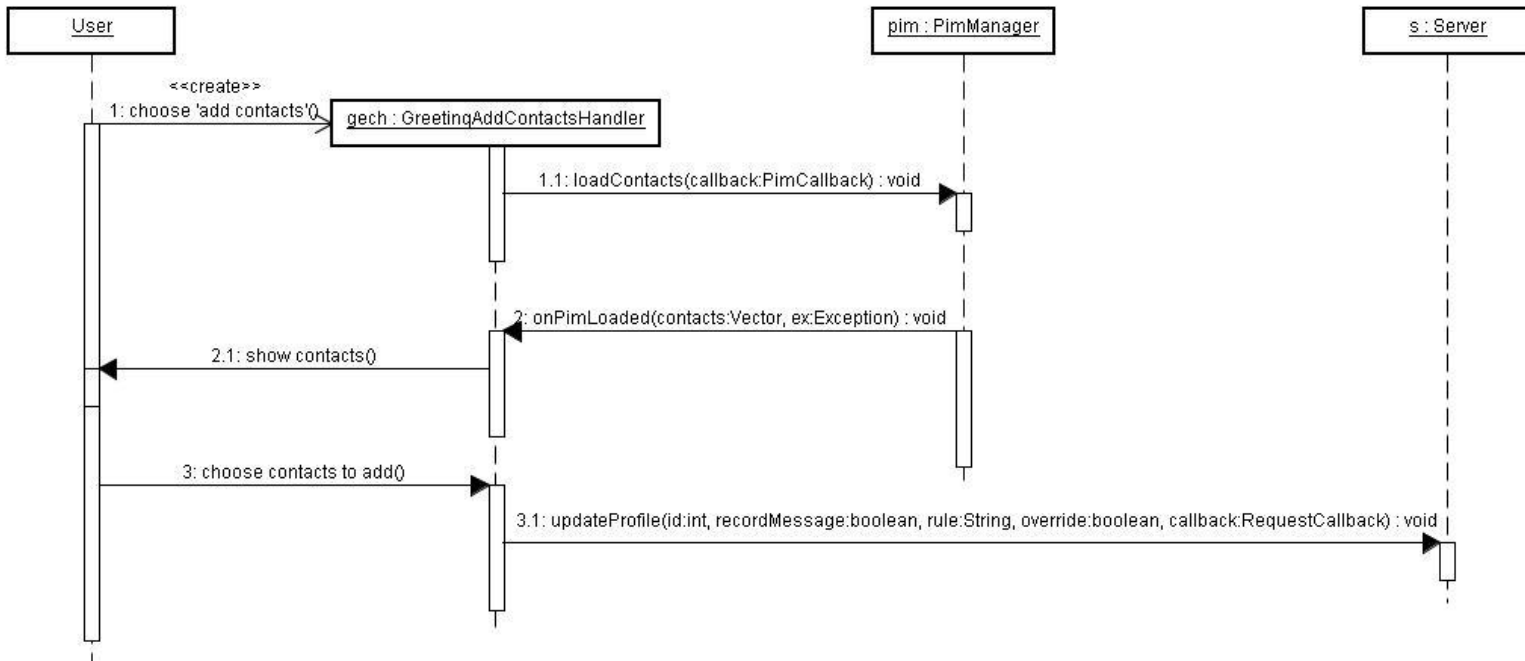


Figure 5 Add contact sequence diagram

Precondition: user has chosen a greeting to edit in the Greeting List screen, and then selected ‘Add Contacts’. The PIM manager²⁸ loads contacts from the mobile phones address book. The user can then select any number of contacts to associate to the greeting. The chosen contacts are converted into a so called rule string²⁹ and then passed on to the server.

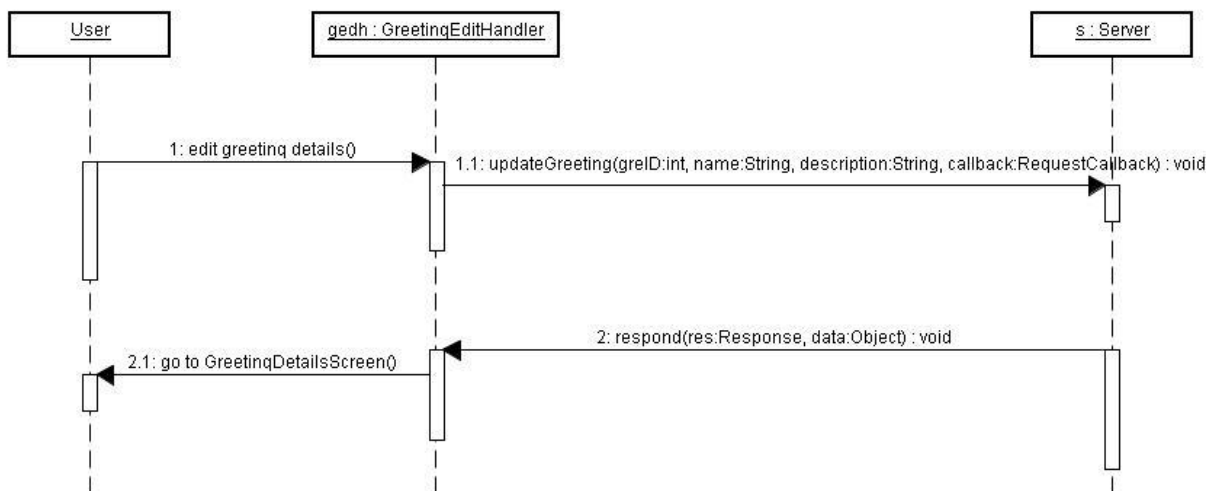


Figure 6 Edit greeting sequence diagram

Precondition: the user selected a greeting to edit in the Greeting List screen, and then selected ‘Edit Greeting’.

After giving the new name and description, the greeting is updated on the server and then the user returns to the Greeting Details screen.

²⁸ PIM stands for Personal Information Management, which is an optional package for the Java ME platform.

²⁹ As defined in the *Greeting Webservice Specification* document.

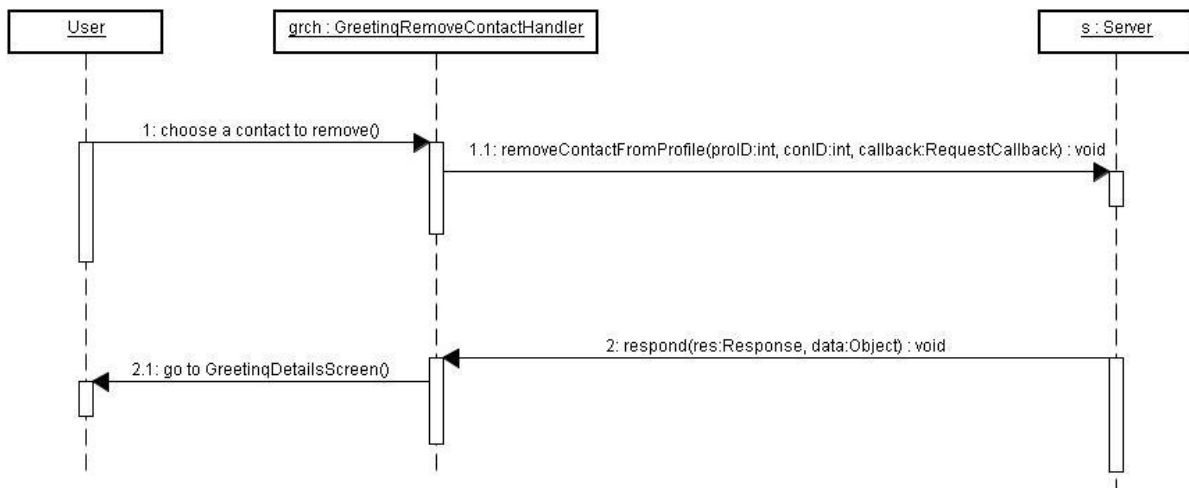


Figure 7 Delete contact sequence diagram

Precondition: the user selected a greeting to remove a contact from in the Greeting List screen, and then selected 'Remove contact'.

After selecting the contact to remove, the removeContactFromProfile operation is called on the server, and the user returns to the details of the greeting.

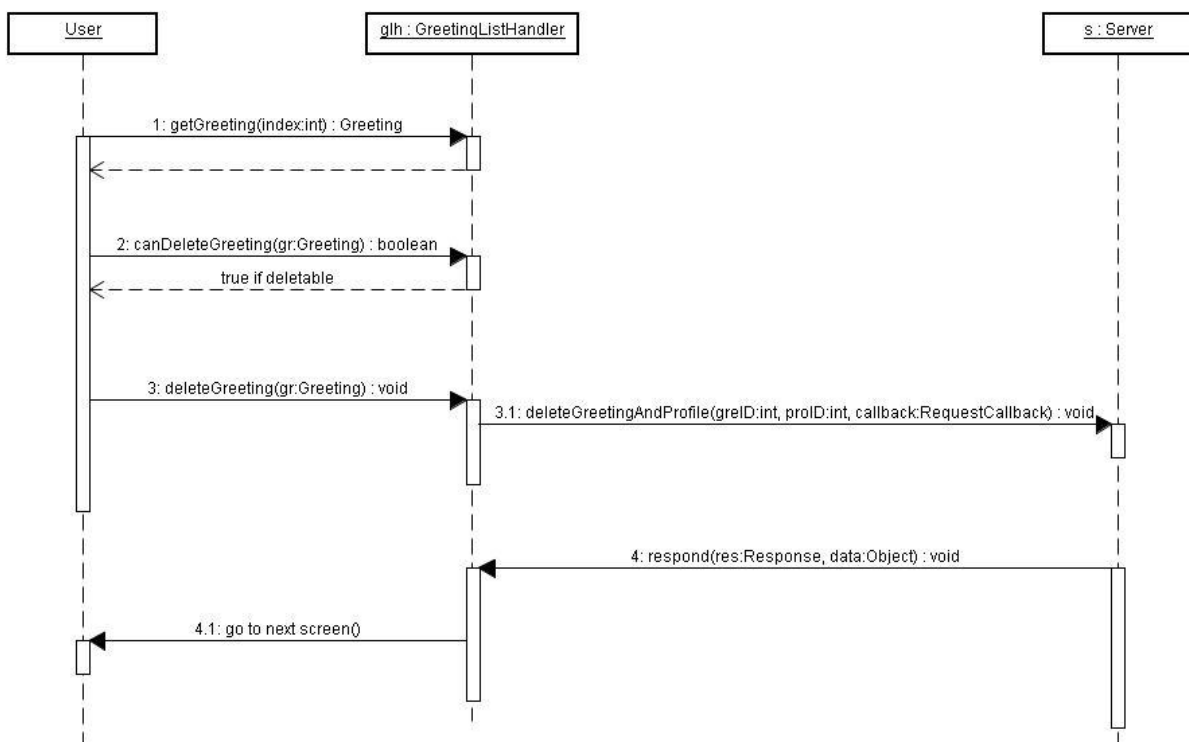


Figure 8 Delete greeting sequence diagram

The user interface checks with the handler if deleting is possible before trying; the default greeting cannot be deleted. The screen to show after the deletion depends whether there are greetings left; if all greetings are deleted, the Main Menu is shown, otherwise the Greeting List screen is shown.

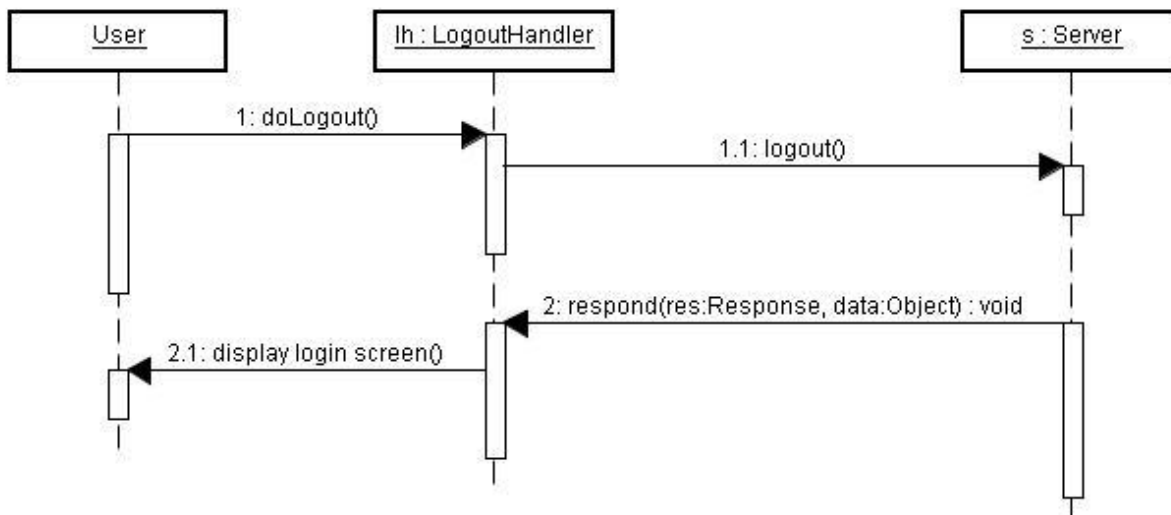


Figure 9 Logout sequence diagram

3. Algorithms

Streaming media with limited bandwidth is a difficult problem in general. It is even more problematic on mobile phones because of the inconsistent support for file formats and protocols. After comparing the support of various formats on mobile phones, we have identified AMR (Adaptive Multi-Rate³⁰ compression) as the most suitable. It is supported by a wide range of phones and needs little bandwidth, while still providing quality reasonable for voice.

Other candidates we chose not to use were:

- MP3, not supported by all phones not from Nokia
- WAVE, taking too much space & bandwidth
- WMA, supported by some Windows Mobile devices but not by any other brand.

While choosing a transport protocol we noticed two suitable candidates:

- HTTP (see Orientation Document)
 - o This protocol is availability on every mobile phone the application runs on.
- RTP (Real-time Transport Protocol³¹)
 - o It is specially constructed to guard against varying bandwidth availability, high latency and varying latency.

We have chosen to implement both protocols. The application will use RTP if it is supported, and will fall back to HTTP otherwise. When using HTTP, the application first tries to use progressive download (listen while downloading), but if this is not supported it will download the entire file before playing it.

We note that RTP streaming has yet to be implemented on the server side, while a HTTP server is already providing access to all audio data.

³⁰ See http://en.wikipedia.org/wiki/Adaptive_Multi-Rate

³¹ See http://en.wikipedia.org/wiki/Real-time_Transport_Protocol

F. MoSCoW

Must have (see Requirements Document):

- Listening and editing voicemail messages
- Listening, editing and creating greetings
- Changing settings of the Greeting voicemail service
- Work on Symbian OS 9.3

Should have:

- Possibility of adding / changing languages and styles
- Enable or disable SMS notifications for received messages and/or missed calls
- Ergonomic interface which suits to small displays.

Could have:

- Adaptive quality control for fluctuating bandwidth
- Send recorded Greetings in the background

Would have:

- Nothing currently known.

G. Implementation Plan

Phase 1:

- Operational mock-up: the GUI is fully done, calling only empty stubs on commands

Phase 2:

- Listening to voicemails:
 - o Retrieving list of voicemails from server
 - o Playback voicemail (streaming)

Phase 3:

- Adding other functionality (greetings, recording, storing settings)

Phase 4:

- Provide for fluctuating bandwidth
- Provide for losing the Internet connection (recovery)

H. Code template

Because of restrictions of the Java ME language, we will use the Java 1.3 language specification. We use the coding style borrowed from the Eclipse Formatter tools. An example follows:

```
/**
 * A sample source file for the code formatter preview
 */

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
    private final LinkedList fStack;

    public MyIntStack() {
        fStack = new LinkedList();
    }

    public int pop() {
        return ((Integer) fStack.removeFirst()).intValue();
    }

    public void push(int elem) {
        fStack.addFirst(new Integer(elem));
    }

    public boolean isEmpty() {
        return fStack.isEmpty();
    }
}
```

Appendix D: Project Plan Document

Bachelor Project

Visual Voicemail and Greetings

Project Plan Document

Version 2

Computer Science
Faculty of Electrical Engineering, Mathematics and Computer Science



<u>Date:</u>	July 8, 2008
<u>University:</u>	Delft University of Technology
<u>Course:</u>	IN3405 Bachelor Project
<u>Company:</u>	Greetingq B.V.
<u>Authors:</u>	Nathan Bruning Medya Amidi Michelle Cheung
<u>Supervisor:</u>	Ir. H.J.A.M. Geers
<u>Coordinator</u>	Ir. B.R. Sodoyer

Table of Contents

Foreword	104
1. Introduction	104
2. The Project Assignment	104
2.1. The project principal	104
2.2. The goal.....	105
2.3. The deliverable Product	105
2.4. Important success factors	105
3. Project Planning	106
3.1. Phase 0: Requirements Gathering & System Design	106
3.2. Phase 1: mock-up	106
3.3. Phase 2: Implementing and Testing: Voicemails	106
3.4. Phase 3: Implementing and Testing: Greetings.....	106
3.5. Phase4: Implementation and Testing: Error correction.....	107
4. Important factors in developing software product	107
5. Documentation	108
Attachment A: Project Planning.....	109

Foreword

This document describes the planning of the Bachelor of Science project at the TU Delft University of Technology. The project is to develop a software application written in Java Micro Edition (Java ME) for mobile phone devices.

First, the aspects of the project will be described and the document will be concluded with the actual plan.

1. Introduction

Fast growing technology brings many opportunities to realize many new ideas. Not while ago, the idea of making a mobile phone device was a dream. Today, every one can use them almost everywhere. This feature refers to one of the special characteristics of evolving technology: the constant demand and supply of new services. Mobile telephony is not excluded from this fact. As the wireless and hardware technology grows, the number of services that a mobile phone can offer increases accordingly. This has opened a new area in software development which is evolving very fast. Our project is to realize new features and services for mobile phone devices.

2. The Project Assignment

Voicemail is a service with which users of the mobile phones are able to listen to their messages left by callers. The service currently being used in the Netherlands has an inconvenient and tiresome procedure with little power given to the user. The challenge is how to develop a service which decreases the duration of the procedure while giving the users more freedom to manage their own voicemails.

Greetingq Company (see section 2.1) has decided to develop an application which enables users to manage their voicemails “visually”- i.e., users can “select” their voicemails and play them in “any order” of their choice, in a user friendly GUI environment.

For our bachelor project, we are assigned by Greetingq to develop this application.

2.1. *The project principal*

Greetingq B.V. is a new high-tech start-up company, established on January 2007 and located in Yes!Delft³², Rotterdamseweg 145 in Delft since September 2007.

The company started to develop a system for mobile phone users to personalize their greetings with which they can record different greetings for different people. In this way users can separate their business greetings from personal ones. The company has expanded its

³² Yes!Delft is a ‘Young Entrepreneurs Society’ who inspires, stimulates, and supports students and young technologists to start their own business. See www.yesdelft.nl

productions by offering personal greeting service through the Internet and visual voicemail & greeting service.

2.2. The goal

As mentioned before the voicemail service currently being used has some limitations. Here are the main disadvantages of this service:

1. Voicemails can only be listened sequentially with no alternative given to users to select them.
2. There is no information indicating who has left a voicemail, therefore user has to listen to the message in order to know who has called.
3. Listening to the voicemails is not directly possible. User has to dial first a special number given by the provider and then listen to a standard operator which is sometimes not in a favour of the user.

Currently there are only three possibilities available to make any voicemail greetings:

1. A standard voicemail greeting provided by mobile network provider
2. A standard voicemail greeting with the user's name spoken by him/her.
3. A voicemail greeting completely spoken by the user

The goal of our project is to develop an application with a user friendly graphical interface which solves all the issues mentioned above in an easy, reliable and fast way for the users to use.

2.3. The deliverable Product

The final product must fulfil all constraints and requirements demanded by Greetingq³³. Furthermore, from the software development perspective, it must accomplish the essential features such as functionality, scalability, efficiency and so on. For more information see section 4.

2.4. Important success factors

When developing a software product, one of the most important factors is to meet the deadline to deliver the final product- "time to the market". Greetingq's plan is to introduce the product to the market in September. However, according to the TU Delft curriculum our actual deadline will be in the first half of July. Hence our planning to deliver the final product will be at the beginning of July.

Another success factor is whether the final product has met all the requirements demanded by the company. With frequent feedback from Greetingq, and discussing important factors in

³³ For details, see Requirements Documents

developing the product, such as whether we have focus more on the performance and functionality than for example the customizability, we try to assure that the final product satisfies the requirements.

3. Project Planning

Late March, we have done a research about what techniques and methods are needed to develop the application. We have studied the different aspects of the project, trying to anticipate possible upcoming problems in the process of development, the current possibilities and so forth. All this information is gathered in an Orientation Document (for more detail, see Orientation Document).

3.1. Phase 0: Requirements Gathering & System Design

Officially bachelor projects at TU Delft University start in April. Due to some problems related to the Greeting Company- such as moving and preparing suitable place to work, our project is postponed to the beginning of May. In April we have had an interview to gather requirements and start writing documents.

3.2. Phase 1: mock-up

The first deliverable will be a mock-up (non-functional prototype), in which the user is able to navigate through all menus and select all options, but no functionality is actually implemented.

3.3. Phase 2: Implementing and Testing: Voicemails

The objective of this phase is to develop a part of the application which provides a user the ability to listen to his/her voicemails using a mobile device. After finishing this phase we are able to:

- Retrieve list of voicemails from the server
- Playback the voicemails (streaming)

3.4. Phase 3: Implementing and Testing: Greetings

Adding “greeting” functionalities to the application will be done in this phase. The goal is to achieve how to record and store a voice message on the mobile device, and finally how to send it to the server.

3.5. Phase 4: Implementation and Testing: Error correction

We need to consider all possible disturbances that may occur during the process. Such problems can be bandwidth fluctuation, loss of the Internet connection and so forth. This phase is assigned to deal with these problems. Further we have allocated an extra week for possible delays in any of the above mentioned phases.

Note: Depending on the process flow, the plan may change. We will keep track of daily activities and decision making in a separate document called Project Log. For more information about the plan, refer to the Project Log.

4. Important factors in developing software product

One of the problems when developing a software product is that developers generally have a particular kind of user in their mind as their target user. However it is not always easy to predict what kind of users will finally end up using the product. Different group of users may have different priorities when they use a specific product.

Further, since the first release of a specific product is usually immature (not all needed features are predicted in forehand); developers need to consider possible changes in the structure of their products for next releases. Below you can find some of these important factors that we need to take into account during the development of our product.

- Functionality
- Efficiency
- Portability
- Reliability
- Maintainability

In the first place, the final product must be functional and simple. Since this will be the first release of the product it is essential that it works perfectly while having all the goals achieved³⁴. The application must be simple so that users in different range of age can easily work with it. Simplicity of the program also increases its reliability.

Our final product must be portable, i.e. it has to run on multiple platforms. The product must work well on different mobile phones and must be easily adaptable to the environment of other mobile devices.

Maintainability is the *ease* with which the software system or component can be modified to correct faults, improve performance or to adapt to the changed environment³⁵. This is easily understood under the modularity, comments, naming conventions, documentation, design

³⁴ See section 2.2

³⁵ As time passes new hardware and software technologies come into existence: new version of operating systems, programming libraries etc. Therefore software products should be made in a way that they can adapt to and take advantage of changes in the environment. Changing in requirements is also common during developing software programs.

patterns, etc. Regarding this factor we are going to use, among others, the standard Javadoc and UML.

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. The high complexity of software is the major contributing factor of Software Reliability problems³⁶. Users in general don't like to use faulty programs. The service that our application offers should be available 24 hours per day. Of course external problems such as failing internet connection, is not the responsibility of this product, but the program should be able to reconnect the internet at most in a few minutes. Otherwise it has to report the error to the user.

And finally the efficiency plays an important role in our application. It is important to have a fast response to the server, i.e. fetching data from, and sending it to the server should be done in a few seconds to a minute. This depends on how big the data is.

5. Documentation

All the documentations will be written in English with the same layout. The end-user manual of the product will be published in Dutch. Development documentation will be provided in the form of the System Design document with UML diagrams of the system.

³⁶ Software Reliability, Carnegie Mellon University 18-849b Dependable Embedded Systems

Attachment A: Project Planning

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
31	April, 1 Orientation Document	April, 2	April, 3	April, 4	April, 5	April, 6
April, 7	April, 8	April, 9	April, 10	April, 11	April, 12	April, 13
April, 14	April, 15	April, 16	April, 17	April, 18	April, 19	April, 20
April, 21	April, 22	April, 23	April, 24	April, 25 Requirement Elicitation Interview at Greeting	April, 26	April, 27
April, 28	April, 29	April, 30	May, 1 Requirements Document	May, 2	May, 3	May, 4
May, 5 <i>Liberation Day</i>	May, 6	May, 7	May, 8	May, 9	May, 10	May, 11
May, 12 <i>Second Pentecost day</i>	May, 13	May, 14	May, 15	May, 16	May, 17	May, 18
May, 19	May, 20	May, 21 <i>Deliverable: mock-up</i>	May, 22 Implementation & Testing: phase 2 – voicemails	May, 23	May, 24	May, 25
May, 26	May, 27	May, 28	May, 29	May, 30	May, 31	June, 1
June, 2	June, 3 <i>Bachelor Seminarium</i>	June, 4	June, 5	June, 6	June, 7	June, 8
June, 9	June, 10	June, 11	June, 12	June, 13	June, 14	June, 15
June, 16	June, 17	June, 18	June, 19	June, 20	June, 21	June, 22
June, 23	June, 24	June, 25	June, 26	June, 27	June, 28	June, 29
June, 30	July, 1	July, 2	July, 3	July, 4	July, 5	July, 6
July, 7	July, 8	July, 9	July, 10	July, 11	July, 12	July, 13
July, 14	July, 15	July, 16				
Presentation						

Appendix E: Test Report

Bachelor Project

Visual Voicemail and Greetings

Test Report

Version 2

Computer Science
Faculty of Electrical Engineering, Mathematics and Computer Science



<u>Date:</u>	July 8, 2008
<u>University:</u>	Delft University of Technology
<u>Course:</u>	IN3405 Bachelor Project
<u>Company:</u>	Greetingq B.V.
<u>Authors:</u>	Nathan Bruning Medya Amidi Michelle Cheung
<u>Supervisor:</u>	Ir. H.J.A.M. Geers
<u>Coordinator</u>	Ir. B.R. Sodoyer

Table of Contents

1. Executive summary	113
1.1. System Overview	113
1.2. System Architecture and Software Components	113
2. Test Environment	113
2.1. Software Items Under Test.....	113
2.2. Components in the Software Test Environment.....	113
3. Test Results.....	114
3.1. Overall Assessment of the Software Tested.....	114
3.2. Detailed Test Results	114
3.2.1. Unit testing.....	115
3.2.2. System testing	116
3.2.2.1. <i>Errors</i>	117
3.2.2.2. <i>Test Cases</i>	118
3.2.3. Acceptance Test	121

1. Executive summary

1.1. System Overview

The mobile application under test is myGreeting mobile version 1.0. myGreeting mobile is developed to provide mobile phone users a visual voicemail and personal greeting service with a user friendly graphical interface. This software is written in the programming language Java ME programming language for Nokia S60 series mobile phones.

1.2. System Architecture and Software Components

To build the mobile application, the Model View Controller (MVC) architectural pattern is used. Using this pattern ensures a clear division between the presentation (view) and the application logic (controller). There are three main packages: view, mode and controller. The View package is responsible for the interface of the application. The Model package provides storage of data. And the Controller package facilitates data retrieval and sending, responds to user stimuli and handles error conditions.

2. Test Environment

2.1. Software Items Under Test

The software we tested is the myGreeting mobile application version 1.0.

2.2. Components in the Software Test Environment

The tests are done in two different environments:

- 1) Emulators:
 - a. DefaultColorPhone emulator of Sun Java Wireless Toolkit 2.5.2 for CLDC.
 - b. S60 Emulator of S60 3rd Edition FP1 SDK for MIDP
- 2) Mobile devices:

Nokia 6120 classic using S60 3rd edition

We used the Java compiler of the Java SE 1.3 edition.

Based on J2ME's Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP), Sun Java Wireless Toolkit's wireless applications are designed to run on cell phones, mainstream personal digital assistants, and other small mobile devices. The toolkit includes an emulator, performance optimization, and tuning tools.

By running the application with the Sun emulator we suggest that the application would run on the actual mobile devices as well. However it was different in reality. The results of the tests on the mobile and in the emulator were not always the same. If this differences exist we will discuss them in the error description (section 3.2.2.1).

Connecting to the server is done through the internet. The internet connection can be made by the following internet access media:

1. dial- up
2. broadband
3. Wi-Fi
4. GPRS / UMTS / HSDPA

The emulators use the broadband connection and the mobile phone under the test use the GPRS network.

For unit testing we used the testing tool JUnit version 1.0.1 for CLDC 1.0.

3. Test Results

3.1. Overall Assessment of the Software Tested

As it will be explained in section 3.2, the results of the test are promising for the developers. The main functions of the application work as they were defined in the requirement documents³⁷. A user can listen to the messages and is able to add a new Greeting and linked this to the contacts.

The bugs found are relatively small and mostly were quickly recognized in the code and debugged. Other errors were caused by the server side. Since we were only responsible for the mobile application and we had no access to the server, we could not correct the server side errors. Instead we had to report these server-side errors to the responsible department of the company and wait until they were fixed. In this case we could only ensure that the client side, hence the mobile application, was tested well. This has caused delays in our testing phase.

Another constraint for testing was the compatibility of the application with the mobile devices with different characteristics and platforms. This made it more complicated to test the mobile application for us. Restricting to S60 3rd edition mobile phones only ensures that the application will perform well in this environment, but does not tell anything about the performance on other mobile phones.

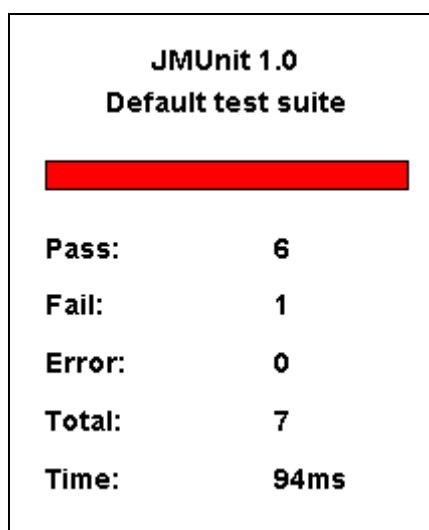
3.2. Detailed Test Results

In this section the results of the test are exposed. We have applied two test methods, unit testing and a system testing. Unit testing verifies that a specific module of source code is working. Unit testing is treated in section 3.2.1. In system testing, the logic of the whole system was tested to verify the functionality, usability and installation. We used both the emulator and a mobile device (Nokia 6120) for system testing. We combined our final results into a set of tables with test cases. Differences between results of emulators and results of mobile device are described in errors descriptions. This part is described in section 3.2.2.

³⁷ Requirements document v.6

3.2.1. Unit testing

Unit tests are designed to test a single class, component or module in isolation. We applied unit testing only for some classes. In this case, we applied unit testing on the model package. For other classes, it was not possible to test each of them as a single class because they could not be isolated. For each of the classes in the model package (`Account`, `Contact`, `Greeting`, `Mailbox`, `Message`, `Phone`, `Profile` and `Setting`) there is a `TestCase` class implemented which checks for the boundaries and functions. After running the test cases the results were displayed on the emulator screen. Figure 3.1 gives an example of such a unit test run. If a test method failed, an error message was displayed on the IDE's console. This error message told where the failed assertion was located.



- version of JUnit
- name of test case/suite
- bar can be red or green
 - o red indicates that failures/errors are detected
 - o green indicates that all tests passed

Figure 3.1: example of a test run on emulator

JUnit has all the basic assertions that JUnit has, but it does not allow us to insert assertions in the source code. Therefore we used primitive code to assert the pre- and post conditions. For example, for `assertNotNull()` we have used `if(something!=null)`.

During the whole test process, we detected three errors in the unit testing. After debugging these errors, we had the tests all 100% passed (see Figure 3.2). Unfortunately, we can not say anything about the coverage, because we could not find any coverage tools applicable for JUnit. To make it possible to do the unit test again after modifying source code, we made a test suite "TestAll" which runs all test cases together.

Error: testBrand
Location: AccountTest in test code model package.
Related class: Account.java in source code model package.
Fix: the setBrand() method in Account called setString() given the wrong name parameter. Changed the parameter in "AccBrand"

Error: testDefProfileId
 Location: MailboxTest in test code model package.
 Related class: Names.java³⁸ in source code package utils
 Fix: To ensure that methods in Serializable use the correct names for the elements of the class, a Vector of allowed names is stored in Names. The String for MaiDefaultProId did not meet the given name. Changed the String into “MaiDefaultProId”.

Error: testNumber
 Location: ContactTest in the test code model package.
 Related class: Serializable.java
 Fix: setNumber does not accept String with no number. Ensure that the given String contains a number and throw exception if it is not.

Class	Pass	Fail	Error	Total	Time
Account	11	0	0	11	31
Contact	5	0	0	5	12
Greeting	8	0	0	8	16
Mailbox	4	0	0	4	2
Message	9	0	0	9	41
Phone	3	0	0	3	1
Profile	6	0	0	6	18
Setting	3	0	0	3	1

Figure 3.2. Final results of test classes

3.2.2. System testing

System testing is needed to detect defects that will only occur when the complete system is assembled. Because of the complexity of our code, the testing is done manually. For the system test we first set up the possible situations for the test cases. We went through these situations and checked if the results corresponded with our expectations. Taking use cases and scenarios of the Requirement Documents³⁹ into account, we have the following testcases: Log In, Main Menu, Messages, Message Details, Greetingq, Greetingq Details, New Greetingq and Settings.

³⁸ Names.java is deleted after the test. This class is added for testing the correct names, not in use after test.

³⁹ Requirement Documents v.6

3.2.2.1. Errors

The errors we found during the test process are described in this section. For each error we give a name and the location where it has occurred. The Steps shows how the error was invoked. Then a short description is given how the application behaves because of the error. And at last a solution for this error is described.

Error:	Log in
Location:	LoginHandler.java in source code controller package
Steps:	1) Login with wrong username and/or wrong password 2) Login with wrong/correct login data.
Problem:	After the Warning Alert the system returns to the LoginScreen, but it does not clean the input data. The user cannot login again.
Solution:	Call LoginScreen again, with clean input textfields

Error:	Delete Message
Location:	Server
Steps:	1) Go to Messages 2) Choose a Message and delete 3) Confirm the deletion with “Yes”
Problem:	Cannot delete message, NullPointerException thrown.
Solution:	The cause of this error is not in our code but in the Server. We turned this to the responsible person of the Server.

Error:	Delete Greeting
Location:	Server
Steps:	1) Go to Greetings 2) Choose a Greeting and delete. 3) Confirm the deletion with “Yes”.
Problem:	Cannot delete greeting, NullPointerException thrown.
Solution:	The cause of this error is not in our code but in the Server. We turned this to the responsible person of the Server.

Error:	Delete Default Greeting
Location:	Server
Steps:	1) Go to Greetings 2) Choose the default Greeting and delete. 3) Confirm the deletion with “Yes”.
Problem:	Cannot delete, failedHideAndDefault error returned.
Solution:	The Server developer has a new constraint for deleting a default Greeting. It is not possible to delete a default Greeting. Added a Warning.

Error:	TimeZone
Location:	TimeZone.java in package Java.util
Steps:	-
Problem:	The emulator only knows UTC as time zone, so in Holland all times are one or two hours behind (depending on whether summer time is in use). Mobile devices do support our local time zones.

Solution:	It works on mobile devices.
-----------	-----------------------------

Error:	Greeting Standard Type
Location:	GreetingRecorderHandler.java
Steps:	1) New Greeting 2) Choose a standard type 3) Record 4) Go back to previous screen
Problem:	A list of contacts was shown and not the list of types. The RecordScreen calls an incorrect previous screen.
Solution:	Let the handler ask the type of the new Greeting and return GreetingNewTypeScreen if it is standard else return GreetingNewContactsScreen if it is personal.

Error:	Language
Location:	LanguageSetting.java in package control.util
Steps:	1) Go to Settings 2) Change language “Nederlands” to “English” and Save the change
Problem:	Title was not translated.
Solution:	Using the method LanguageManager.get() was redundant, it automatically translates the word.

Error:	Yes/No
Location:	BooleanSelector.java
Steps:	1) Go to Settings 2) Change “Nederlands” to “English” and Save the change
Problem:	“Yes” and “No” were not translated.
Solution:	Added LanguageManager.get().

Error:	Maximum message duration
Location:	SettingsScreen.java
Steps:	1) Go to Settings 2) Fill nothing in the field of maximum message duration.
Problem:	NumberFormatException returned.
Solution:	It does not accept an empty String. Check that the Field is not empty. Otherwise set Maximum message duration sec = 0.

3.2.2.2. Test Cases

Each test case has a Pass/Fail column in the tables below. The green colour indicates that the test has passed. After debugging, all tests have passed. Therefore the tests which failed once have their row in the table coloured orange. Situations that will never happen are coloured grey and not tested. The test cases we used are the following:

Log In

Username	Password	Expected	Result	Pass/Fail
correct	correct	Login Succeed, IF messages = 0 MainMenu Screen ELSE Messages Screen	Login Succeed IF messages = 0 MainMenu Screen ELSE Messages Screen	P
correct	incorrect	Login Failed, Warning	Login Failed, Warning	P
incorrect	incorrect	Login Failed, Warning	Login Failed, Warning	P
incorrect	correct	Login Failed, Warning	Login Failed, Warning	P
Null	null	Login Failed, Warning	Login Failed, Warning	P

Main Menu

	Option	Expected	Result	Pass/Fail
Messages = null	Messages	Alert	Alert	P
Messages ≠ null	Messages	Messages screen	Messages screen	P
Greetings = null	Greetings	Alert	Alert	P
Greetings ≠ null	Greetings	Greetings screen	Greetings screen	P
	New Greeting	GreetingNewType screen	GreetingNewType screen	P
	About	About screen	About screen	P
	Settings	Settings screen	Settings screen	P
	Exit	Application closed	Application closed	P

Messages

Messages	Expected	Result	Pass/Fail
0	Alert	Alert	P
1	Messages Screen	Messages Screen	P
>1	Messages Screen	Messages Screen	P

Message	Option		Real result	Pass/Fail
Null	Details	-	-	-
True && New	Details	Message details Screen , Old	Message details Screen, Old	P
True && Old	Details	Message details Screen, Old	Message details Screen, Old	P
Null	Delete	-	-	-
True	Delete, Confirmation = YES	Messages - Message Messages Screen	Messages – Message Messages Screen	P
True	Delete, Confirmation = NO	Messages Screen	Messages Screen	P
Null	Mark	-	-	-
True && New	Mark	Messages Screen , Old	Messages Screen , Old	P
True && Old	Mark	Messages Screen , New	Messages Screen , New	P
	Main Menu	MainMenu Screen	MainMenu Screen	P

Message Details

	Option	Expect	Result	Pass/Fail
AudioState = Stopped	Play	AudioState = Playing	AudioState = Playing	P
AudioState = Playing	Pause	AudioState= Paused	AudioState= Paused	P
AudioState = Playing	Stop	AudioState=Stopped	AudioState=Stopped	P
-	Back	Messages Screen	Messages Screen	P

Greetingqs

Greetings	Expected	Result	Pass/Fail
0	Alert	Alert	P
1	Greetingqs	Greetingqs	P
>1	Greetingqs	Greetingqs	P

Greetingq	Menu	Expected	Result	Pass/Fail
Null	Details	-		
True	Details	Greetingq Details	Greetingq Details	P
Null	Delete	-		
True	Delete Confirmation=NO	Greetingqs Screen	Greetingqs Screen	P
True	Delete Confirmation= YES	Greetingqs - Greetingq Greetingqs Screen	Greetingqs - Greetingq Greetingqs Screen	P
	Main Menu	MainMenu Screen	MainMenu Screen	P

Greetingq Details

Greetingq Details	Menu	Expected	Result	Pass/ Fail
	Play	AudioState = Playing	AudioState = Playing	P
AudioState = Playing	Pause	AudioState = Paused	AudioState = Paused	P
AudioState = Playing	Stop	AudioState = Stopped	AudioState = Stopped	P
	Edit	Edit Greetingq Screen	Edit Greetingq Screen	P
	Add Contact	GreetingAddContact Screen	GreetingAddContact Screen	P
Contact = 0	Delete Contact	GreetingRemoveContact Screen, Contact = 0	GreetingRemoveContact Screen, Contact = 0	P
Contact >= 1	Delete Contact	GreetingRemoveContact Screen, Contact >=1	GreetingRemoveContact Screen, Contact >=1	P
	Back	Greetingqs Screen	Greetingqs Screen	P

Screen	Option	Expected	Result	Pass/Fail
Edit Greetingq	Cancel	GreetingDetail Screen	GreetingDetail Screen	P
Edit Greetingq	Save	GreetingDetail Screen + updated details	GreetingDetail Screen + updated details	P
GreetingAddContact	Cancel	GreetingDetail Screen	GreetingDetail Screen	P
GreetingAddContact	Add	GreetingDetailScreen + added Contacts	GreetingDetailScreen + added Contacts	P
GreetingRemoveContact	Cancel	GreetingDetail Screen	GreetingDetail Screen	P
GreetingRemoveContact	Save	GreetingDetail Screen – deleted Contact	GreetingDetail Screen – deleted Contact	P

Settings

	Option	Expected	Result	Pass/Fail
Maximum messageduration = null	Save	Maximum messageduration = 0	New Maximum messageduration = 0	
Maximum messageduration = t>0	Save	New Maximum messageduration = t>0	New Maximum messageduration = t>0	P
Voicemail Activation = No	Yes	Voicemail Activation = Yes	Voicemail Activation = Yes	P
Voicemail Activation = Yes	No	Voicemail Activation = No	Voicemail Activation = No	P
Default Greeting = null	Save	Default Greeting = null	Default Greeting = null	P
Default Greeting = Greeting	Save	Default Greeting = Greeting	Default Greeting = Greeting	P
Language = Dutch	Modify, English, Save	Language = English	Language = English	P
Language = English	Modify, Dutch, Save	Language = Dutch	Language = Dutch	P
	Main Menu	MainMenu Screen	MainMenu Screen	P

3.2.3. Acceptance Test

The acceptance test will be conducted to enable a user/customer to determine whether to accept a software product. Normally it is performed to validate whether a software product meets a set of agreed acceptance criteria. Greetingq has made an appointment on July 10th to test the final product. Therefore we cannot include the acceptance testing in this report. We will discuss the results during the presentation⁴⁰.

The following requirements and constraints should be satisfied:

Work with voicemail messages:

1. Browse through a list of received voicemail messages
2. Listen or remove a message
3. View details⁴¹ of a message
4. Mark a message as listened or new

Work with the Greetingq service:

5. Browse through a list of greetings
6. Listen or remove a greeting
7. View and edit the details⁴² of a greeting
8. Change the assigned contacts of a greeting
9. Record a new greeting

⁴⁰ Presentation at July 15th, 2008 at Mekelweg 4, Delft

⁴¹ Details screen of messages include: phone number of the caller, the date the message is sent and the Greetingq (standard/personal) made for the caller

⁴² Detail screen of greeting include: name of the greeting, date in which the greeting is recorded, list of contacts associated with the greeting, and description.

Settings:

10. Turn voicemail service on and off
11. Change maximum time limit for new voicemail messages
12. Remember password
13. Choose the default greeting
14. Language settings⁴³

View “About screen”:

15. Greetingq Logo
16. Application Version
17. Platform Information
18. Website address
19. Greetingq service e-mail address
20. Greetingq hotline

Constraints

21. The application must communicate with the user in Dutch.
22. The application graphics must be consistent with the house style of Greetingq.
23. The user interface must be designed to provide easy changing language or graphic style services.

⁴³ Currently: Dutch and English