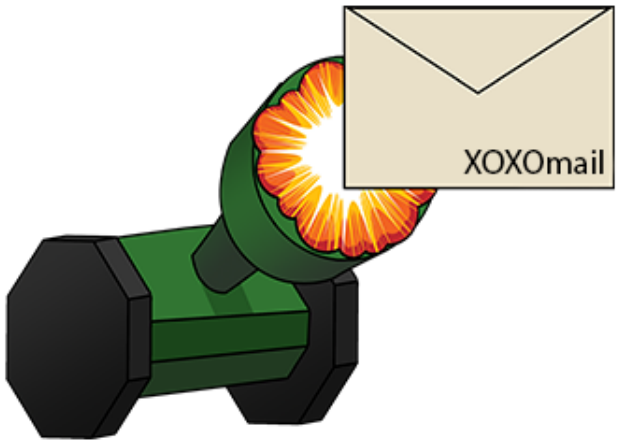
XOXOmail

Formal and secure messaging on a mobile platform

Lars Smørås Høysæter
Aleksander Aas Sjåfjell
Ida Katrine Børstad Thoresen
Kristin Tønnesen
Magnus Hertzberg Ulstein
Nicklas Sørlie Utgaard

November 20, 2012



NTNU
Norwegian University of
Science and Technology

THALES

ABSTRACT

This report describes the problem of communicating with the XOmail system from a mobile device and how this problem was solved. XOmail is an existing information handling and transfer system developed by Thales that focuses on security and on delivering messages to the correct destination in a short time. During the project we created an Android application for cell phones that was supposed to provide a similar level of security and stability as XOmail does on computers. The primary challenges of the project were how to make an application for handheld devices that was secure enough to send important and classified messages and to develop a user interface that was customized for small screens and easy to use.

The motivation behind working on this task was that even if there exist several email applications for handheld devices today, none of them support the amount of security we needed. The most interesting part of the assignment was to figure out how to implement the required level of security and to design an application suited for small handheld devices. None of us had done extensive work with the Android framework before, so we were eager to learn more about the technology. Android development is getting increasingly more attention in the development departments today, so getting the opportunity to learn about the newest technology on the market was very interesting.

The demands of our software project were, at a first glance, quite simple. By implementing a mobile version of XOmail, the XOXOmail client, we wanted to show how a functioning Android client might look like. It is important to note that we focused on creating a proof-of-concept messaging client that used the standards given by Thales. The primary focus was not on creating a fancy user interface, but rather on fulfilling the needs of the users.

The result of this project was first of all a working application that contained realizations of the better part of the requirements of the project. Secondly, the project resulted in this report, which has the purpose of documenting the problem, the process and the final product.

PREFACE

This report was written for a project in the course TDT4290 Customer Driven Project at Norwegian University of Technology and Science (NTNU). The project was executed on behalf of Thales Norway AS between the 21th of August and the 22th of November.

The project team consisted of six students from the Department of Computer and Information Science at NTNU. Our task was to develop a mail application for handheld devices that could communicate with Thales' existing XOmail server.

The team would like thank our advisor Mohsen Anvaari for his advice and help throughout the project. We would also like to thank Reidar Conradi for his feedback.

In addition we would like to thank our customer Thales and their contact persons Christian Tellefsen and Stig Bjørlykke. Their input and feedback have been invaluable and we would like to thank them for the cooperation.

We would also like to thank the persons that set aside the time to participate in our usability tests and give us their valuable feedback about the design and usability of the XOXOmail application.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

DETAILED CONTENTS

Goal of the report

This report is meant to document the entire process behind developing the XOXOmail, from planning and pre-studies through the different sprints and ending with a conclusion and reflection part. This report should describe what was done, why it was done and when it was done. The pre-studies and the decisions made from these studies are documented. It will describe the process of making the application the customer wanted. It should describe and justify the decisions and actions made, as well as sum up what went well and what we could have done better if we were to start over.

What this report covers

Part I - Planning & requirements gives an overview of the planning we did based on the requirements of the project, as well as the pre-studies, test plan and architecture.

Part II - Scrum process describes all of the sprints very carefully. It also includes the product backlog and the changelog.

Part III - Conclusion & reflection describes the results from the project and gives reflections on how the team worked together and what we have learned from this project.

Part I contains the following chapters:

Chapter 1 - Introduction describes the existing product, the wanted solution and gives an overview of the military standards we had to conform to in our XOXOmail.

Chapter 2 - Project directive describes all the involved parties in this project.

Chapter 3 - Project planning gives an overview of how this task was planned, how the team was organized and the quality assurance measures of the project.

Chapter 4 - Requirements contains the functional and the non-functional requirements that describes how the application should look and behave. In addition the stakeholders of the project, the use cases and the requirements draft from the customer are listed in the chapter.

Chapter 5 - Project study gives summaries of all the preliminary studies done in this project, as well as the decisions that were made from these studies.

Chapter 6 - Test plan explains how the code and application were tested in this project.

Chapter 7 - Architectural description describes the architecture of the application, both backend and frontend.

How this report is structured

The report is divided into parts as well as chapters. This gives a much better overview of the report, and makes it a lot easier to find your way around it. It is divided into three parts: a planning and requirements part, a part about the scrum process and the conclusion and reflection part.

Part I

Planning & Requirements

CHAPTER 1

INTRODUCTION

This chapter is a technical introduction to the project. The purpose of this chapter is to give an overview of the problem that we solved and the most important terms used in the report.

The existing product

Thales has developed an information handling and transfer system called XOmail [1], which has been in operational use for more than 20 years. In many ways, it could be compared to the regular, well-known email system. The first and biggest difference between regular email systems and XOmail is that the latter is developed with large formal organizations and military messaging procedures in mind. This is the reason why Thales has been delivering XOmail as a complete product to several European NATO member countries.

The messages transferred in the system are defined to have certain special attributes, where the most important is a security label. A security label is a declaration of the required clearance level the receivers of a message need to have. The security label specifies that only people or groups with the required or higher security clearance can see the message. Security is an important feature of XOmail, and it has never lost a message in 20 years of operation.

What Thales wanted us to make

Thales' existing product is a complete messaging platform for messaging in large organizations, but the current system is primarily made for use on computers with relatively large screens. What Thales wanted to know was if it was possible to create a mobile messaging platform building on the existing infrastructure already in use by XOmail. It is important to note that this mobile platform would only have a subset of the functions that the original system has and the main focus was on security, efficiency and ease of use.

Since Thales had not tried to implement XOmail on mobile platforms, there was no existing framework to start with. Thales wanted a prototype that demonstrated a possible solution to their problem. Security was of great importance, but the main focus was placed on implementing the functionality that was critical for showing the concept software.

Even though XOmail primarily targets the military and other large organizations where security and security declarations are important, we had to keep in mind that this new software also should be well suited for other groups or organizations that have a need for a user-friendly portable system for sending urgent messages. A few of the possible candidates include paramedics, security firms and joint civilan-military operations or exercises.

For a summary of what Thales already had and what they wanted, see the table below.

What Thales has	**What Thales wants**
An information handling and transfer system developed for large formal organizations and military messaging procedures	A system for handheld devices with a subset of the existing functions
A large number of ways of giving a message attributes	Focus on security label, priority and type
Extensive focus on security and reliability	Focus on security and usability
Support for a broad range of attachments	Support for images, text and video

Table 1.2: Thales' current and wanted situation

A short overview of the standards of XOmail

XOmail is based on the NATO military messaging standard Stanag 4406. This standard is used for both Strategic and Tactical messaging [2]. It has a number of special protocols used to support tactical messaging to support links with very low bandwidth.

The standard is based on binary encoded data, and there are limited libraries freely available for these protocols. Special attributes like information sensitivity and priority are defined by extensions to the Internet Mail RFCs. Our connection to Thales' system goes through a XOmail SMTP gateway. SMTP, which is shorthand for Simple Mail Transfer Protocol [3], is an Internet standard for electronic mail transmission over IP networks. Messages to other users are sent by sending an email to the server with an address, and then the server handles the message headers and other important attributes. The messages are then pushed to the correct recipients via IMAP [4], which is an abbreviation for Internet Message Access Protocol, and is one of the most prevalent Internet standard protocols for email retrieval.

CHAPTER 2
PROJECT DIRECTIVE

This chapter is about how we planned our project. The purpose of this chapter is to explain how our team was organized, who we are, why we conducted this project and how we did it.

2.1 Overall project plan

2.1.1 Project mandate

The purpose of this project was to create a handheld version of the already existing XOmail system, as this system only works on computers. The task we were assigned was to make a simplified version of XOmail optimized for handheld devices such as mobile phones.

We eventually called the project and the application XOXOmail as we thought this was a fun name. It is a combination of the name of the existing system and a common way of ending letters and mails, namely "XOXO". The term is an abbreviation for the term hugs and kisses and is used to express affection or friendship and was made worldwide famous by the TV series "Gossip Girl".

2.1.2 The client

The customer for this project was Thales Norway AS, a subsidiary of the leading international electronics and systems group Thales. Thales' focus areas are aerospace, defence and security, and the company has positioned themself as one of the world leaders in mission-critical information systems. Thales Norway has had over 50 years of industrial activity in Norway and is one of the principal suppliers of a military communication system to the Norwegian Armed Forces. [5].

For the contact info of the contact persons at Thales see table 2.1 below.

Name	**Office**	**Phone nr**	**Email**
Sølve Conradi Olsen	Thales Norway	907 80 179	solve.olsen@thalesgroup.com
Christian Tellefsen	Thales Norway	959 98 765	christian.tellefsen@thalesgroup.com
Stig Bjørlykke	Thales Norway	982 29 806	stig.bjorlykke@thalesgroup.com

Table 2.1: The customer representatives

2.1.3 Involved parties

In this project there were three involved parties: a) the customer b) the project team and c) the advisor. The customer, Thales, described in the section above was represented by Christian Tellefsen and Stig Bjørlykke. The project team consisted of six students from the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU).The advisor was Mohsen Anvaari who was assigned to our team to guide and help us during the project period.

For contact info of the group, see table 2.3 below.

Name	Address	Phone nr	Email
Ida Thoresen	Klæbuveien 143, 7031	936 68 688	idakatt@stud.ntnu.no
Kristin Tønnesen	Lars Onsagersvei 12, 7030	986 28 958	kristonn@stud.ntnu.no
Lars Høysæter	Innherredsveien 2a, 7014	900 31 814	larssmor@stud.ntnu.no
Nicklas Utgaard	Odd Brochmanns veg 47, 7051	976 87 790	nicklau@stud.ntnu.no
Magnus Ulstein		472 31 418	magnuul@stud.ntnu.no
Aleksander Sjåfjell	Odd Brochmanns veg 60, 7051	456 01 212	aleksasj@stud.ntnu.no

Table 2.3: The project group

For contact info of the advisor, see table 2.4 below.

Name	Work	Phone nr	Email
Mohsen Anvaari	PhD-stud. NTNU	405 70 403	mohsena@idi.ntnu.no

Table 2.4: Our advisor

2.1.4 Background for the project

This project was given as an assignment in the course TDT4290 Customer Driven Project. Since the society is constantly changing, the technology must follow along. Because of these changes Thales found that they wanted a prototype of an Android version of their existing XOmail system. This new handheld system could help the users of XOmail so that they could also use it on mobile devices and not only at desktop computers. When able to use this mailing system not only in contact with a computer, the regular working day of Thales customers could be a lot easier, especially for their primary customer, the Norwegian Armed Forces.

2.1.5 Project objective

The objective from the customer was twofold: The first part was to develop a functioning prototype to demonstrate a possible way of implementing formal and secure messaging on a mobile platform. The second part was to explore possibilities for further development, e.g. what solutions already exist and what is required to implement the level of security that must be present in military systems. The application should include functionality much like existing mail applications, but the security aspects are quite different and must be investigated and documented, as there probably will not be time enough to implement all the desired features.

The objective from NTNU's point of view was to give the students practical experience with being involved in all the phases of a large IT project and also experience in teamwork.

The goal of the team was to become more experienced with working in a real development project and get a glance into how development with real customers works. It was also important for us to develop an application that the customer could be satisfied with.

2.1.6 Duration

The course staff suggested an estimated 25 hour week for each student. This gave us a total of 1950 working hours since we were six students in our team and the project was distributed over 13 weeks.

- Project start: 21.08.2012
- Project finish: 22.11.2012

CHAPTER 3

PROJECT PLANNING

This chapter is about the planning of the project. Section 3.1 covers the project plan, section 3.2 the project organization, section 3.3 the quality assurance efforts we planned and section 3.4 the risk management.

3.1 Project plan

This section describes the organization of the group and all the tools we used.

3.1.1 Measurement of project effects

The measurements of project effects state what the result of the project should be measured in. The documentation required by the Customer Driven Project itself is one project effect. An equally important effect is what we accomplish with the development of the application. Thales has stated what results they want from this project, and nothing is more important than a happy customer.

Thales proposed the following project effects:

- Test a possible user interface for message-based communication on a handheld device.
- Make a prototype that can give inspiration for further product development and be used as a demonstration internally and to customers.

3.1.2 Limitations

In this section it will be described both the technical and non-technical limitations the group had to deal with in the project.

Technical limitations

The technical limitations is limitations for our project according to technical difficulties.

Android

Android is a quite new platform and none of us have been doing much programming with this framework prior to this project. The task of learning how Android worked and to find technical tools that Android supports was going to be a time consuming task. We wanted to develop our project in Android version 2.2 with API level 8. The reason behind choosing this version of Android was that we did not want to depend on features that only existed in newer versions of Android, and 2.2 is a version that is very common.

Android security limitations

Using Android as our platform gives us a number of limitations when it came to security. Firstly, the device is small and easy to lose compared to a fixed installation such as a computer. Secondly, we need to rely on existing civilian network rather than purpose-built military communication channels.

A major issue for our security considerations is the mobility of XOXOmail. The small device could easily be lost or stolen and fall into the wrong hands, so we have to take this into account when developing for security. We therefore plan to encrypt the entire phone and require a login on system resume, with a short timer.

An Android-based application is problematic in that it communicates over the Internet, thus it is conceivable that someone could listen in to the transmissions from the application. We plan to resolve this through the use of network layer encryption through SSL.

Screen

The mail client is going be used on devices with relatively small touch screens, which means that the application must be carefully designed to accommodate the limited size. In addition we need to put a lot of thought into the design so that it becomes easy to use. The application must be designed to scale well with different screen resolutions.

Non-technical limitations

Language

The entire project is to be delivered in English, so we all have to write in our second language. This means that the report work would take more time than if we could write it in Norwegian.

Duration

We have a set timeframe of 13 weeks with a delivery deadline that can not be changed. This gives us a set amount of effort and we probably do not have the time to implement all the desired parts of the application. We have to focus on the parts we find most important. The specified timeframe will likely give us challenges when it comes to planning and priorities of the different tasks.

3.1.3 Tool selection

This section will describe the tools we chosed to use during the project.

Git & GitHub

Git is an extremely fast, efficient and distributed version control system ideal for the collaborative development of software. It can be used to manage all your public and private repositories, and makes it easy to share and work with the same source code and documents [6]. GitHub is a hosting system that implements git and is free to use if you choose to share your source code with others.

NetBeans IDE

NetBeans IDE is an integrated development environment for developing with Java. It is written in Java and runs on Windows, Linux and OS X. NetBeans IDE is one of the most popular IDEs. Using a common IDE helps speed up the development process.

Google Docs

Google Docs is a free web based service that offers users the ability to create and edit documents online. Google Docs lets us share documents easily and collborate on the same document simultaneously.

Apache Maven

Maven is a build automation tool which is typically used with Java projects. It uses an XML file to describe the project, the project dependencies on other external modules and components, the build order, directories, and required plug-ins.

JIRA with GreenHopper extension

JIRA is an issue tracking system developed by Atlassian that is used for bug tracking, issue tracking and project management. When used along with the GreenHopper extension it adds support for agile development.[36]

Wireshark

Wireshark is a tool used for network troubleshooting and analysis. It captures network traffic and displays it to you through a graphical user interface. Wireshark is great for understanding how your application acts on a lower network layer.

LaTeX

LaTeX is a free high-quality typesetting system designed for writing technical and scientific documentation. It is most often used for medium-to-large documents, but can be used for almost any form of publishing.

LaTeX is based on the idea that it is better to leave document design to document designers and to let authors get on with writing documents. So it encourages authors not to worry too much about the appearance of their documents, but to concentrate on getting the content right [35].

Fluid UI

Fluid UI [111] is an online mobile application prototyping tool for Android and iOS application. Fluid UI was chosen because it gave us a quick and easy way of creating interactive prototypes to share with the customer.

Violet

Violet is a UML editor which is good at drawing diagrams. It is basically intended for developers and students that needs to produce UML diagrams [105]. UML is a standarized modeling language for developing of a software program [106]. We have used this program to make our use cases.

Software Ideas Modeler

Software Ides Modeler is also a UML tool. But it is a more advanced tool than Violet, and it supports all the different diagram types specified in UML. This tool does even support ERD diagrams, BPMN, CRC, flowcharts and data flow diagrams [107].

3.1.4 Organizational demands

There are no organizational demands from Thales, but we have agreed on a scrum based approach. We decided that scrum was a good lifecycle-model to use since it could give us a runnable product early that could be revised during the project. This is, after all, a pilot project and the specifications are not fully completed, so many revisions are anticipated.

3.1.5 Resources

The group consists of six members with different personalities and interests. We are all students working on a master's degree in computer science, but the programming experience of each member varies. Two of the group members have programmed since they were about 13 years old, while the rest started when they came to NTNU. All of us have at some point had a programming job, and hence have some experience with working in teams.

Fortunately we all have different interests regarding what we want to contribute with in the project. We have all agreed that everybody has to participate in both planning, implementation and report work, but we have also tried to make use of our special interests. As some of us enjoy report work and others have had much experience in the setup of programming related tasks, we have allocated the main roles accordingly.

Having a group consisting of people with different personalities could always be a challenge with regards to the group dynamics. By conducting a daily standup where everyone shares what they have done, what they will do the current day and what they need to get it done, we plan to avoid communication problems and misunderstandings. We want to share a common understanding of what needsto be done and get everyone in sync.

3.1.6 Schedule of results

We have chosen to use scrum because of the flexibility it offers. It is a lightweight solution with very little extra overhead for us to worry about. This will allow us to focus on the actual work rather than having to waste our limited resources on superfluous coordination.

Milestones

See table 3.1 below for an overview of the milestones in our project.

Milestone	**Date**
Project start	21.08.2012
Pre-delivery of project report	14.10.2012
Final delivery of project report	22.11.2012
Project presentation and demonstration	22.11.2012

Table 3.1: Milestones

Sprints

See table 3.2 below for an overview of the sprints that this project consisted of.

Sprint	**Duration**
Sprint 1	27.08.2012 - 16.09.2012
Sprint 2	17.09.2012 - 07.10.2012
Sprint 3	08.10.2012 - 28.10.2012
Sprint 4	29.10.2012 - 18.11.2012

Table 3.2: Sprints

3.1.7 Concrete project work plan

Sprints

See table 3.3 on page 12 for an overview of the planned effort for all the sprints in this project.

Phase	Norm	S1	S2	S3	S4	Total
Project management	10	41h	35h	2h	-	78h
Lectures	10	4h	8h	15h	9h	36h
Planning	7	26h	99h	93h	48h	266h
Pre study	15	34h	59h	16h	-	109h
Requirements specification	20	18h	-	-	-	18h
Design	15	87h	57h	21h	8h	173h
Programming and documentation	13	243h	102h	213h	215h	773h
Project evaluation	5	-	-	-	20h	20h
Presentation and demonstration	5	-	-	-	60h	60h

Table 3.3: Plan for all the sprints

3.2 Project organization

We decided to use the scrum model for organizing our group. This means that we have a flat organizational structure where everyone contributes with what they are able to. Even in a flat structure, it is important to share responsibilities, so each person of the group got responsibility of a main focus area. If it becomes too much work for one person, we have decided that it is important that the work is delegated to the others.

3.2.1 Organizational diagram

See figure 3.1 on page 13 for an organizational diagram that shows the internal structure of our group, as well as our connection to our advisor and Thales. Notice that even though we have a project leader, he is at the same level as all the others. This is to indicate that the project leader is not a big boss, but that we are all equally responsible for this project being a success.

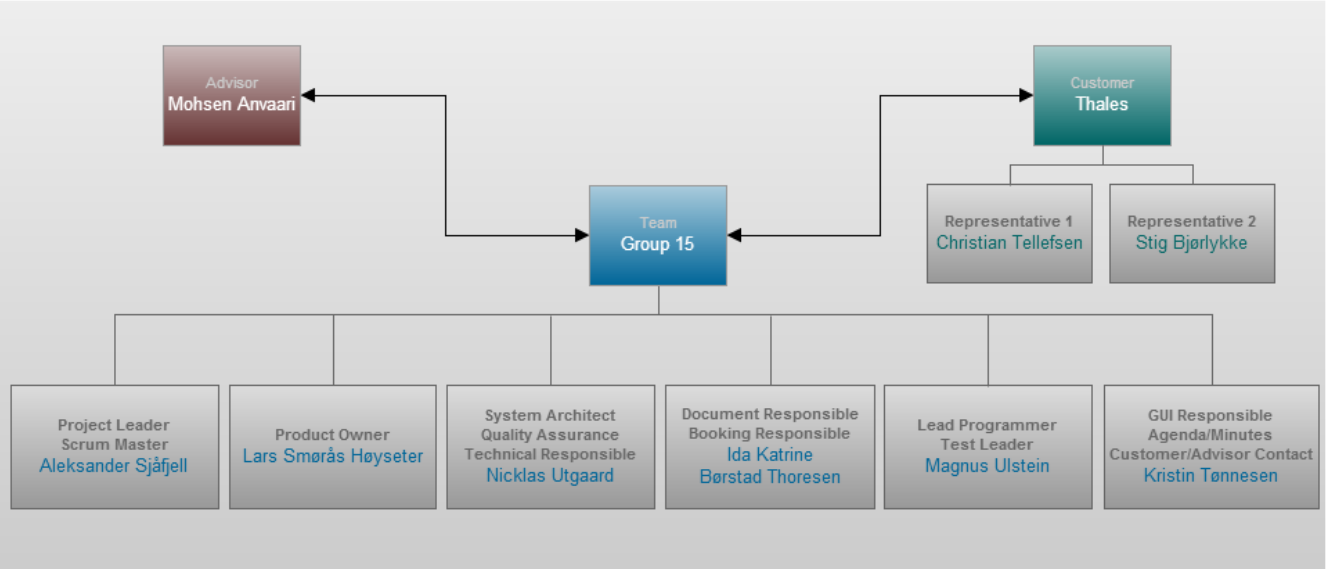


Figure 3.1: Organizational chart

3.2.2 Role allocation

See table 3.5 on page 14 for a role allocation table that shows what roles each person were assigned, as well as a short description of the role. The roles are merely a guideline for who is responsible for the depicted area. If the person should need more help, then we have all agreed to contribute to get the result we want. We are all in this together.

Role	Person	Responsibilities
Project leader	Aleksander	Make sure everybody does what they are supposed to and peacefully resolve disputes between participants
Scrum master	Aleksander	Make sure the team's work conform to Scum standards and help the team do the best work possible
Booking of rooms	Ida	Booking rooms for the meetings and other activities
Document responsible	Ida	Keep track of what is to be contained in the project report, what has been written and what remains
Project report layout responsible	Ida	Find software to make tables and graphs, and check that all diagrams are similar in style
Responsible for graphical user interface	Kristin	Design the views and the interactions and setting up the MVC structure
Agendas, minutes of meetings	Kristin	Write agendas and minutes of meeting and send these out within the specified time limits
Customer/advisor contact	Kristin	Main contact person for customer and advisor
Product owner	Lars	Represent the stakeholders and ensure that the team delivers value to business
Test leader	Magnus	Lead the testing team
Lead programmer	Magnus	Responsible for having a general overview of the code. This means knowing what is to be implemented next, and seeing that this is done at the right time
System architect	Nicklas	Defining the system architecture
Technical responsible	Nicklas	Setup of Netbeans, Git and other tools that the team uses in the development
Quality assurance responsible	Nicklas	Make sure that all routines, templates and standards are followed

Table 3.5: Role allocation

3.2.3 Weekly schedule

See table 3.6 below for a weekly schedule that shows how we have allocated time for the Customer Driven Project. As we all have said how much we are able to contribute to the project, we have agreed to ensure that work outside of group work-hours must be done to reach the goal.

	Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
08-09		Group work			
09-10		Group work			
10-11		Group work			
11-12		Advisor meeting			
12-13	Group work	Group work	Customer meeting		
13-14	Group work	Group work	Group work		
14-15	Group work	Group work	Group work		
15-16	Group work		Group work		
16-17	Group work		Group work		
17-18	Group work		Group work		

Table 3.6: Weekly schedule

3.2.4 Work breakdown structure

See table 3.7 on page 16 for a work breakdown structure of the project.

Task	From date	To date	Effort	
			Est.	Act.
Misc	**21.08.2012**	**21.11.2012**	**291**	**204.75**
Project Management	21.08.2012	29.10.2012	86	64.75
Lectures	03.09.2012	01.11.2012	56	38.5
Planning	21.08.2012	26.08.2012	40	39
Pre- Study	27.08.2012	24.10.2012	109	62.5
Sprint 1	**27.08.2012**	**16.09.2012**	**271**	**186.5**
Planning	27.08.2012	16.09.2012	26	95.5
Design	27.08.2012	16.09.2012	87	42.5
Implementation/Testing	03.09.2012	16.09.2012	158	48.5
Sprint 2	**17.09.2012**	**07.10.2012**	**156**	**167**
Planning	17.09.2012	07.10.2012	99	66.5
Design	17.09.2012	07.10.2012	57	100.5
Implementation/Testing	-	-	-	-
Sprint 3	**08.10.2012**	**28.10.2012**	**241**	**250.5**
Planning	08.10.2012	28.10.2012	93	58.5
Design	08.10.2012	28.10.2012	21	102
Implementation/Testing	08.10.2012	28.10.2012	127	90
Sprint 4	**29.10.2012**	**18.11.2012**	**58**	**119.5**
Planning	29.10.2012	14.11.2012	48	48.5
Design	29.10.2012	12.11.2012	8	46.5
Implementation/Testing	29.10.2012	05.11.2012	2	24.5
Report & Completion	**27.08.2012**	**21.11.2012**	**640**	**696.25**
Report work	27.08.2012	21.11.2012	560	638.25
Evaluation	06.11.2012	20.11.2012	20	13.5
Presentation	12.11.2012	14.11.2012	60	44.5
Total	**21.08.2012**	**21.11.2012**	**1657**	**1624.5**

Table 3.7: Work breakdown structure

3.3 Quality assurance

This section explains what we have planned to do to ensure quality in the project. Quality assurance is important to establish routines, not duplicate work or never lose any of the work. We want to ensure that all group members know how to do things, use the correct format and keep the deadlines.

3.3.1 Time of response

Agenda and questions for the meetings must be sent to the customer no later than 24 hours before the customer meeting is scheduled. Minutes of meeting should be sent to customer within 48 hours after the meeting, but preferably as soon as possible. Approval of minutes of customer meeting should be done by the rest of the group before sending it to the customer. The customer should comment and approve of the minutes within 48 hours after receiving it from the group. Agenda and weekly documents should be sent to the advisor before Mondays at 14:00. We should get approval and feedback of the documents sent to the advisor within 72 hours of having sent him the documents.

For a table of the agreed response times, see 3.8 on page 17.

What	Time of response
Approval of minutes of customer meeting	48 hours
Feedback on weekly documents	72 hours
Approval of weekly documents	72 hours
Answer to a question	48 hours
Producing requested documents	48 hours after approving request

Table 3.8: Time of response table

3.3.2 Routines for producing high quality internally

To ensure high quality internally we will make sure every item produced is reviewed by a team member who did not work on the given item. We have planned to do this to increase the quality of the items and give the group members a better feel of the entire project and not just the parts they work on themselves.

Most of the working hours are planned together with other group members, either the entire team or in groups of two or three people. We want to ensure that there should always be someone to ask if any questions comes up, either in the 18 hours we have scheduled to meet physically each week or by communicating electronically.

3.3.3 Routines for approval of phase documents

Making sure that the different documents throughout the phases have the desired quality is a high priority for us. Therefore we have allocated one group member with the overall responsibility of putting together the different document parts and simultaneously reviewing of the documents.

We also want to make sure that the customer and advisor get opportunities to view the important documents that would affect the direction of our project. This way we will receive vital feedback and make the necessary changes according to the response of the advisor or customer.

The visibility of the documents and progress is a part of the scrum method, meaning that the customer always has insight into what the group is doing and how far the group has progressed.

3.3.4 Calling for a meeting with the customer

In advance of all meetings with the customer we will send a call for the meeting, specifying time, place, intention, agenda, and background documents. It is important to specify preparations that have to be made by both the customer and the group prior to the meeting.

A calling for a meeting should be sent before Monday at 14:00 the week the meeting took place.

3.3.5 Minutes of a customer meeting

The group should write a summary of each meeting with the customer. The most vital points are decisions, actions (what, who and deadline) and clarifications that are important for further work on the project. The customer should approve the minutes of meeting to make sure there are no misunderstandings of decisions made. The minutes of meetings are part of the "contract" with the customer, not uncommon in normal settings.

The minutes of meeting should be sent to the customer as soon as possible and always before Friday the week the meeting took place. If the minutes are not approved by the customer, they should be rewritten and resent for approval.

3.3.6 Calling for the weekly advisor meeting

Calling for meetings should be sent before Monday at 14:00 the week the meeting was scheduled. The meeting with the advisor will normally takes place every Tuesday at 11:15 if no cancellation or exceptions are made.

3.3.7 Agenda for the weekly meeting with the advisor

We were given a template from the course administrative that was to be followed. This template can be found in appendix G.1.

3.3.8 Minutes of the weekly meeting with the advisor

Minutes from the last meeting will be attached to the next meeting call and is a fixed subject on the agenda.

3.3.9 Templates and standards

The group have made templates and standards for the most relevant document types. Even though it took some time to create these in the beginning, we believe it will benefit us over time.

Templates

We have made templates for agendas and minutes of meetings, as well as for the weekly documents. These are located in appendix F. The template will hopefully make the work for our agenda and minutes responsible a lot easier.

Standards

We have used standards so that everybody in the group do the work in the same way, so that we all can have an idea about how to do things.

Coding style
See table 3.9 on page 19 for an overview of the naming conventions for the UI elements.

Element	Convention	Example
Button	btn	btnSubject
ImageButtonr	imb	imbSubject
TextView	lbl	txtSubject
ImageView	imv	imvSubject
EditText	txt	txtSubject
Spinner	spr	sprSubject
ListView	lst	lstSubject
CheckBox	chb	chbSubject
RadioButton	rbt	rbtSubject
ToggleButton	tbt	tbtSubject
Layout	lay	laySubject

Table 3.9: Naming conventions - UI elements

Organizing files
We have chosen to use different ways to organize files so that we don't loose any work, and so that it will be easier to find all our files.

Google Docs
We will use Google Docs to organize our files so that our entire team can view and collaborate on the documents simultaneously. We have one folder that contains our entire project and in that folder we have subfolders to make the best overview possible. We will try not to have too many files in each folder so that finding a specific document becomes as easy as possible.

Git

We will also use Git so that it is easy to share all the code and documents as well as to keep a version control of the files. When using Git all the team members will have access to change, read and use all the code and document files whenever they feel like it.

Saving files on gitHub

All documents and Java files must compile before they are pushed to the repository.

Code repository structure

```
kpro-app/
  res/ -- application specific resources -- resources used in the application
    drawable/ -- images for the application
    layout/ -- layouts for the the activies
    values/ -- common values for the application
  src/ -- java source folder
    main/java/no/ntnu/kpro/app -- the differente activites
  target/ -- build folder
kpro-instrumentation/
  src/ -- java source folder
    main/java/no/ntnu/kpro
    app/ -- gui tests folder
  core/service/ -- service tests folder
  target/ -- build folder
kpro-lib/
  res/ -- application specific resources
  src/ -- java source folder
    main/
      java/no/ntnu/kpro/core
        model/ -- application specific models
        service/ -- service related package
          factories/ -- factory classes
          implementations/  -- implementating classes
          interfaces/ -- all relevant interfaces
        utilities -- common javabased utilities
    test/
      java/no/ntnu/kpro/core/
        service/
          implementation/
  target/ -- build folder
```

LaTeX repository structure

```
Backlog/ -- contains the teams backlog
Meetings/ -- agendas, feedback and minutes from every meeting
  Internal/
    Agendas/
    Feedback/
    Minutes/
  Mohsen Anvaari/
    Agendas/
    Feedback/
    Minutes/
  Thales/
    Agendas/
    Feedback/
    Minutes/
Project report/ -- the tentative complete report
Templates and standards/ -- templates in use
Weekly status reports/ -- weekly timesheets
```

Naming of files
All the files in our project shall be given a name that starts with the date they are created so that they will be easier to find. The name should also contain one or two words that explains what the documents contain.

Agendas
"YYYY-MM-DD-Agenda-[Customer/Advisor/Internal]"

Minutes of meeting
"YYYY-MM-DD-Minutes of Meeting-[Customer/Advisor/Internal]"

3.3.10 Version control procedures

The group have created a systematic procedure for version control for all textual and code files, which is outlined below.

Version control for code

For source code management there are several viable options on the market, the prevalent being Git and SVN. Although they do pretty much the same thing, there are some key differences in how they work which makes Git favorable over SVN. One key feature is performance, where Git greatly outperforms SVN. Git gives you the possibility of a hierarchy of repositories, which offers great flexibility. Another advantage is the local repository every user has on their computer. This enables the developer to work even though he does not have access to the remote repository. Lastly, and perhaps the most important difference comes with the branch/merge differences between the two tools. Git does this incredible well, while the general consensus is that SVN's merge tool is tedious to use.

In addition to technical differences between Git and SVN, it was the experiences of the developer team that finally made the decision to use Git for this project. To learn how to use Git check out the Git Book at [78].

Version control procedure

All procedures assume the use of a terminal similar to Bash, and that the current directory is the workspace of your IDE.

[]: optional arguments & <>: argument

Setup

Description: Setup of project (usually just once)

Command:

git clone <git remote repository address>

On programming session start

Description: Whenever a developer starts programming

Command:

git pull [<remote repository> <branch>]

On change

Description: Whenever a developer complete a change in functionality.

Command:

git add <file> #Repeat for all changed files
git commit -m "<commit message describing the changes made>"
[git pull && git push] #should only be done for if project builds

On complete

Description: Whenever a developer finishes a change in functionality.

Command:

git pull [<remote repository> <branch>]
#Fix any merge conflicts
git push [<remote repository> <branch>]

Version control for report documents

Version control for documents is planned to be divided into two bulks. In the development stage of a document it should reside within Google Docs. This will give us a rapid way of writing, editing and sharing all documents related to the project. Google Docs also provides the necessary version control needed at this point. When a document or part of a document is finished it should then be typeset in LaTeX and pushed to Git. Once on Git, the documents work in the same way as the code.

3.3.11 Design guidelines

External resources

- Create all strings in res/values/strings.xml for easy access and modifiability
- Create all colors in res/values/colors.xml
- Use the same style for similar components throughout the application, and preferrably create styles in the res/values folder
- Provide assets for all screen densities, e.g. images for low, medium and high screen density
- Consider using separate XML files for portrait and landscape mode to optimize for both views
- Font sizes: Use 12 sp (scale independent pixels) for micro sized text, 14 sp for small, 18 sp for medium and 22 sp for large
- Be careful with the use of colors

General from Android [7]

- If it looks the same, it should act the same
- Only interrupt the user if it is important
- Keep texts brief
- Try to use pictures instead of long texts
- Make important things fast
- Be gentle if the user does something wrong and give clear recovery instructions

Pure Android [8]

- Do not mimic UI elements from other platforms
- Do not carry over platform-specific icons
- Do not use bottom tab bars
- Do not use labeled "Back" buttons on actions bars
- Do not use right-pointing cares on line items

3.4 Risk management

The defined risks in the project are listed below, and described in table 3.10 on page 26.

R1. Illness/Unavailability: A member or members of the group become ill or unavailable.
R2. Internal conflicts/disputes: A conflict/dispute between two or more group members.
R3. Problems with internal or external communication: Misunderstandings between group members or with the customer/advisor.
R4. Bad technical solution: A bad technical solution/implementation is selected for the problem.
R5. Too much priority given to a certain task: Too much effort is spent working on a task.
R6. Technical incompetence: The group does not possess the technical experience to solve a task.
R7. Inexperienced with software development method: The group does not have the required experience with the software development method.
R8. Hardware failure: A hardware failure halts the projects progress.
R9. Chosen tools do not deliver: Development tools cause the project to slow down.
R10. Misplacement of code or documents: Code or written documents gets misplaced and can not be recovered.

Explanation of abbreviations used in table

- Probabilitiy/Consequence:
  - L = low
  - M = medium
  - H = high
- Strategy and actions:
  - Reduce = How to reduce the consequence
  - Accept = Accept that the risk has occurred
  - Avoid = How to avoid that the risk happens
  - Transfer = How to transfer the consequence from a risk that has happened, so that the consequence gets minimized

Risk ID	**R1**
Risk factor	Illness/Unavailability
Consequences	**M:** Absence may halt the project progress
Probability	M
Strategy & actions	**Reduce:** Involve group members in different parts of the project to avoid complete halts for certain tasks
Deadline	Continuous
Responsible	Aleksander and Kristin

Risk ID	**R2**
Risk factor	Internal conflicts/disputes
Consequences	**M:** Can result in problems with cooperation and thus cause the project progress to slow down
Probability	M
Strategy & actions	**Accept:** The conflict can be a constructive discusion **Reduce:** Stop the conflict by going between the parts and try to come to a common solution fast
Deadline	Continuous
Responsible	Aleksander and Magnus
Risk ID	**R3**
Risk factor	Problems with internal or external communication
Consequences	**M:** Group members can end up doing things the wrong way or the group misunderstands the customer's wishes or the advisor
Probability	M
Strategy & actions	**Avoid:** Have frequent internal and external meetings to discuss what has been done and what needs to be done **Accept:** If something is done the wrong way we have to do it again
Deadline	Continuous
Responsible	Everyone
Risk ID	**R4**
Risk factor	Bad technical solution
Consequences	**H:** The project will not be completed before deadline, or result in a poor quality
Probability	M
Strategy & actions	**Avoid:** Plan well and conduct a thorough pre-study
Deadline	Sprint 1
Responsible	Nicklas and Kristin
Risk ID	**R5**
Risk factor	Too much priority given to a certain task
Consequences	**M:** May affect the quality of other important tasks
Probability	M
Strategy & actions	**Avoid:** Plan the time distribution carefully **Reduce:** If we find that a task is given to much priority, let everybody on the team know about your concern, and then change the priority
Deadline	Continuous
Responsible	All

Risk ID	**R6**
Risk factor	Technical incompetence
Consequences	**H:** The group will be unable to finish the project
Probability	H
Strategy & actions	**Avoid:** Choose familiar solutions for the problems that occur and do a thorough investigation into unfamiliar technologies that are to be used **Transfer:** If one or more of the members get a problem they cannot solve, ask someone else
Deadline	Continuous
Responsible	Everyone
Risk ID	**R7**
Risk factor	Inexperience with software development methods
Consequences	**M:** Lack of documentation and too much time spent on administration
Probability	M
Strategy & actions	**Reduce:** Get familiar with the software development method early and be thorough with the documentation
Deadline	Continuous
Responsible	Everyone
Risk ID	**R8**
Risk factor	Hardware failure
Consequences	**M:** May cause unexpected delays
Probability	M
Strategy & actions	**Avoid:** Have version control and make information available from different locations
Deadline	Continuous
Responsible	Everyone
Risk ID	**R9**
Risk factor	Chosen tools do not deliver
Consequences	**M:** May halt the project progress
Probability	M
Strategy & actions	**Reduce:** Find out what the most suited tools on the market is and choose one that is best suited.
Deadline	Preliminary study
Responsible	Nicklas and Ida
Risk ID	**R10**
Risk factor	Misplacement of code or documents
Consequences	**M:** May result in work having to be redone
Probability	M
Strategy & actions	**Reduce:** Use version control to continuously update and store the code and documents
Deadline	Continuous
Responsible	Everyone

Table 3.10: Table for handling of risks

CHAPTER 4
REQUIREMENTS

This chapter will describe the requirements stated in the project. Section 4.1 outlines the functional requirements, section 4.2 the non-functional requirements, section 4.3 the stakeholders and their concerns in the project, section 4.4 the use case diagrams and textual use cases and 4.5 the draft of the requirements given by Thales.

4.1 Functional requirements

The functional requirements of the project are listed in table 4.1 on the subsequent pages.

Req. ID	Description	Priority
FR1	**Start application and log in:** The user should be able to start the application and authorize himself against an authorizing mechanism.	High
FR2	**Send and receive a message:** The user should be able to send and receive a simple message via regular email protocols to a recipient of his own choice.	High
FR3	**Browse received messages:** The user should be able to browse all messages he has received.	High
FR4	**Browse sent messages:** The user should be able to browse all messages he has sent.	High
FR5	**View address book:** The user should be able to view the address book, so that he is able to choose a recipient from the address book when sending a message.	High
FR6	**Mark messages with military attributes:** The user should be able to set the security label, priority and type attributes on a message, so that the recipient of the message knows who the message is intended for, how important it is and in what environment the message is of interest.	High

FR7	**Send and receive message with attachments:** The user should be able to add an attachment to the message. The attachments should be viewable when opening the message.	Medium
FR8	**Answer, delete and forward messages:** The user should be able to, by clicking on a message, choose to answer to, forward or delete the message, and be brought to the correct screen for doing the selected action.	Medium
FR9	**Send instant message:** The user should be able to, via maximum three screen interactions, send an instant message with predefined message attributes to a predfined recipient.	Medium
FR10	**Settings menu:** The user should be able to view and edit the following settings: • Change between push and pull strategy • Choose which security labels should be visible in the view for sending a message, if the user does not want all the choices visible • Settings for instant message standard attributes (security label, priority, type, recipient)	Low
FR11	**Request and see message status:** The user should be able to request a delivery report and receipt notification for a message before sending it and to see the status of these fields of a message where it was requested.	Medium
FR12	**View high priority message** The user should, when a message with priority "Flash" or "Override" is received, be notified directly and be able to open the message directly.	High
FR13	**Sort the messages:** The user should be able to sort the messages according to different sorting criterias, e.g. date ascending and descending, priority from highest to lowest or from lowest to highest.	Medium
FR14	**Search in the message folders:** The user should be able to search through the messages based on different search conditions to quickly find messages in the message folders.	Low
FR15	**Add signing to messages:** Signing of messages will increase security in the application.	Medium

Table 4.1: Functional requirements

4.2 Non-functional requirements

This section will explain the requirements for the application that do not add specific functionality to the product.

4.2.1 Quality requirements

The requirements for quality attributes will be listed and explained below.

Usability
Usability explains how we made our application effective to learn and use.

U1 Ease of use
The user should be able to use the application with very few mistakes, such as pressing the wrong button because button labels are ambiguous or too small. The scenario is described in table 4.3.

Portion of scenario	Values
Source	End user
Stimulus	Use system without problems
Artifact	System
Environment	At run time
Response	Provide clean and understandable interface
Response measure	Maximum 5% of user operations should be mistakes (another action than the user wanted)

Table 4.3: U1 Ease of use

U2 Efficent use
The user should be able to use the main features of the application with as few screen interactions as possible. Streamlined design is therefore of great importance. The scenario is described in table 4.5.

Portion of scenario	Values
Source	End user
Stimulus	Use system efficiently
Artifact	System
Environment	At run time
Response	Organize application so that important functions are easy to reach from the main menu
Response measure	A user with only a quick (5 min) training course should never spend more than one minute finding what he looks for

Table 4.5: U2 Efficent use

Performance ***P1 Latency***
The user should be able to read the message quickly after receival of a new message. The scenario is described in table 4.7.

Portion of scenario	**Values**
Source	End user
Stimulus	Wants to open message
Artifact	System
Environment	Under normal operations
Response	The operation is performed
Response measure	With a latency of maximum 3 seconds, the user should be able to read the message after it is received. This is the latency when we subtract the download time of message, which is dependent of the network connection

Table 4.7: P1 Latency

Security ***S1 Data access***
Data should not be exposed to unauthorized users or other application. The scenario is described in table 4.9.

Portion of scenario	**Values**
Source	Unauthorized user or application
Stimulus	Wants to access data saved by XOXOmail
Artifact	System
Environment	Under normal operations
Response	User is blocked from accessing data by the Android OS
Response measure	No data should be exposed to the unauthorized user or application

Table 4.9: S1 Accessing locally stored data outside of application

S2 Unprivileged use
Users that try to use the application without the correct privileges should be denied. The scenario is described in table 4.11.

Portion of scenario	**Values**
Source	Unauthorized user
Stimulus	Wants to use application features
Artifact	System
Environment	Under normal operations
Response	User is blocked from using functions
Response measure	No features should be exposed to the user, the user should be stopped by the login screen

Table 4.11: S2 Trying to use application with wrong privileges

S3 External data traffic
External data traffic should not be exposed to unauthorized users or application or someone else sniffing network packages. The scenario is described in table 4.13.

Portion of scenario	**Values**
Source	Unauthorized user or application, or someone sniffing network packages
Stimulus	Wants to access data stream
Artifact	System
Environment	Under normal operations
Response	Is unable to get useful information because of SSL encryption
Response measure	No useful data exposed to the user

Table 4.13: S3 Trying to access the application's external data traffic

4.2.2 Component and technology related constraints

Vanilla Android is the platform of choice, compatible down to Android API 8.

Programming language
Developing for Android will in this project means that code will be written in Java for the Dalvik Virtual Machine. This means that there is no support for multiple inheritance, and this limits design choices to some degree. It prevents us from playing with some more complicated architectures and inheritance schemes.

Mobile devices
Android is a mobile OS, so the application will run on cell phones and potentially tablets.

Input methods
Android phones have small screens and usually no keyboard. This means that the user interface must be designed to work with touch interactions as the only input method.

Memory
Phones have limited memory, typically 256-512 MB. The application will probably never be close to breaking this limit, as the most memory consuming components are string based messages and potentially an attachment or two; very little of which has to remain in memory at the same time.

Screen resolution/aspect ratio
As the application will support different Android devices, it also has to support different resolutions as well as aspect ratios. The GUI must be programmed to scale to different formats without degrading quality or proportions.

Bandwidth
Since the application is network based, it is important to pay attention to bandwidth usage. We want to be able to wait for incoming messages and receive them as soon as the server does without having to waste bandwidth with constant polling, especially as the application is expected to see some use outside of 3G net areas.

Commercial off-the-shelf (COTS)
This project has security requirements that require special consideration with regards to COTS. We have to be able to trust the underlying software if it ever touches unencrypted data or has access to a memory partition (either legitimately or not) that contains unencrypted data. This means some care has to be taken in regards to what COTS solutions we can use and where we can use them.

4.3 Stakeholders and their concerns

For an overview of the stakeholders and what concerns these have in the project, see table 4.14.

Stakeholder	Concern
User	• Easy to understand the flow of the application • Installation should be easy • User friendly user interface
Developers	• Easy to understand the goal of the application • Easy to extend and change the application • Want to use technology they are familiar with • Easy to understand the requirements.
Course advisor	• Effective and healthy communication with the group • Easy to read documentation • Gets all deliveries on schedule
Customer	• Effective and healthy communication with the group • A working prototype that gives a possible solution to the requirements • Wants to see what is possible on the Android platform • Understandable requirements and architectural description documents.
Graphic designers	• Understanding of how the GUI should be • Understand the limitation of graphics due to screen size

Table 4.14: Stakeholders and their concerns

4.4 Use Cases

This section starts by discussing the different actors that will use our system. Then it continues to show a full use case diagram derived from the functional requirements. Finally, it lists the textual use cases to detail the use case diagram.

4.4.1 Actors

An actor specifies a role to be played by internal or external persons interacting with the application. In our case, we have only one actor, namely the end-user. The user is a person who does not necessarily have much education in using XOXOmail, but will use it regularly in his workday.

4.4.2 Use case diagrams

For a combined figure of all the use cases, see figure 4.1 below.

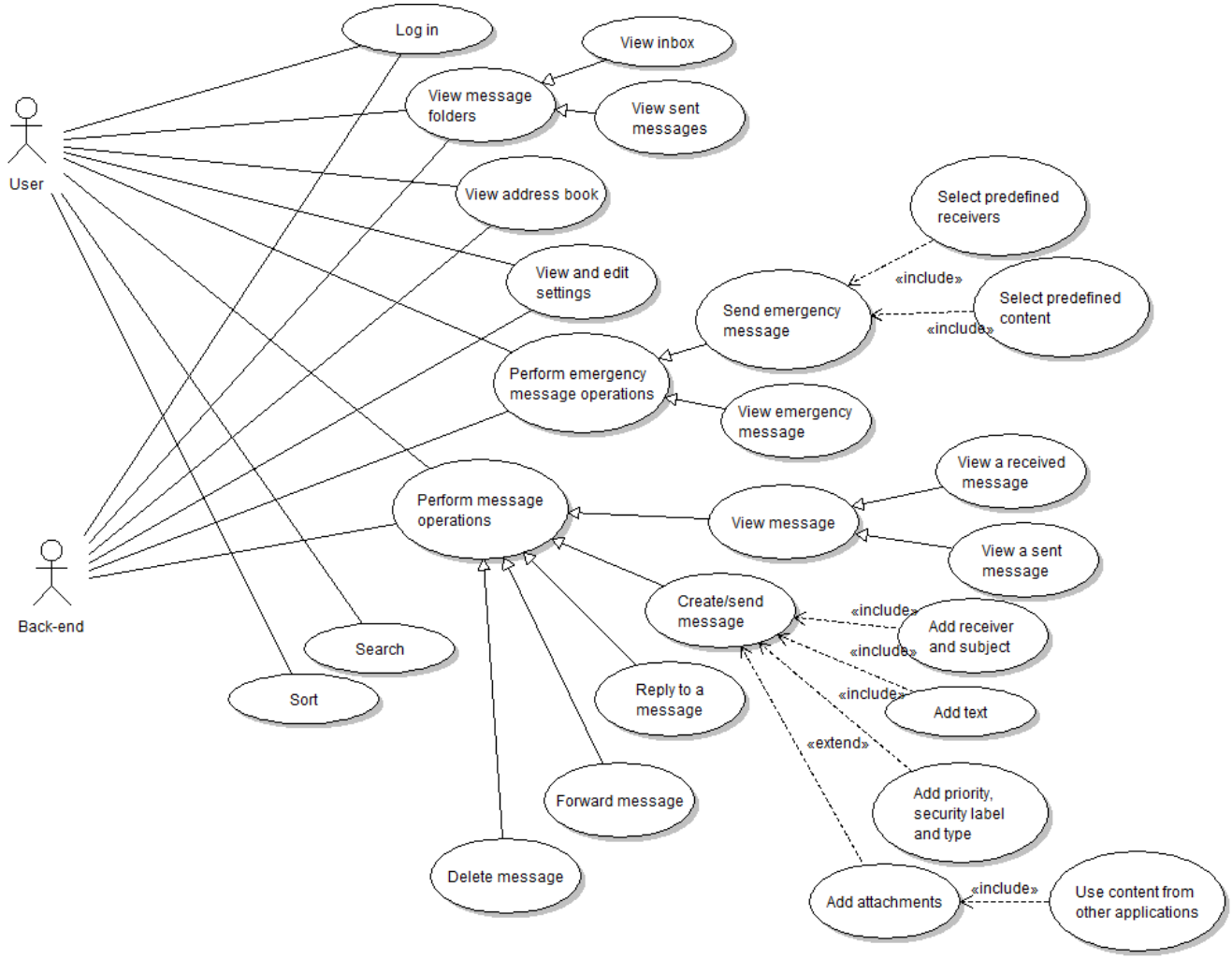
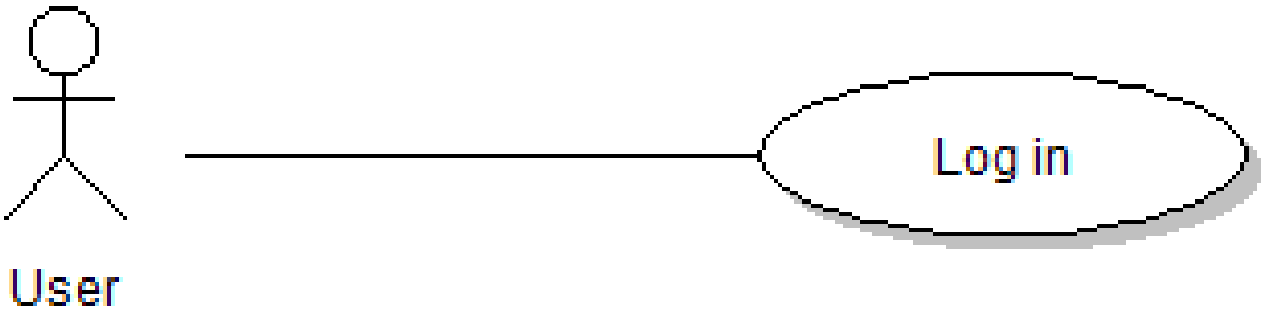


Figure 4.1: Use case diagram

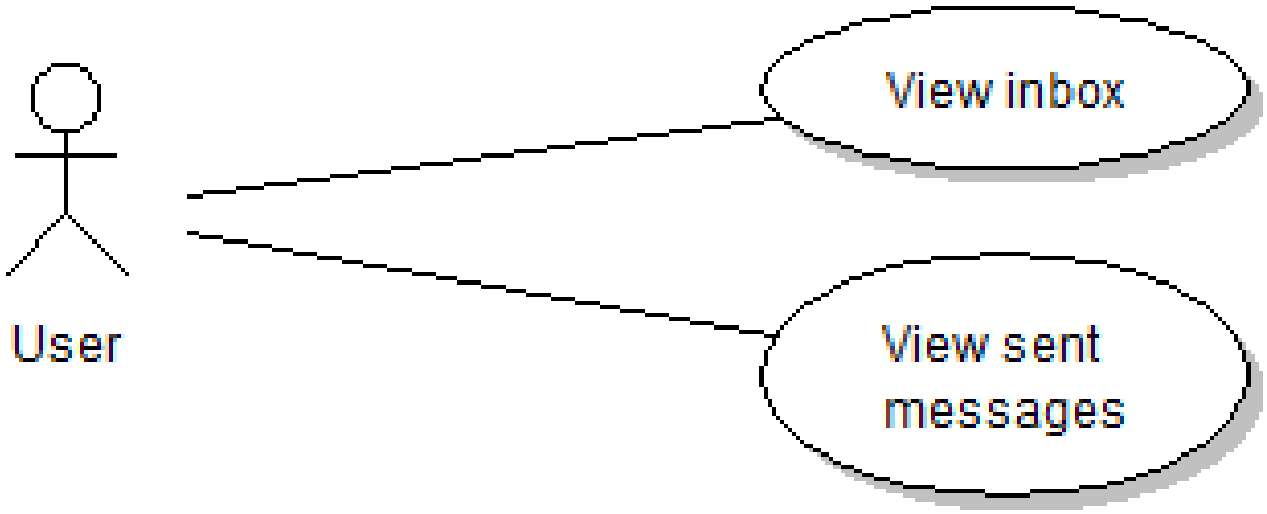
4.4.3 Textual use cases

Each of the use cases is described below, so that the use case diagram will be easier to understand. See table 4.15 - 4.25 starting below to page 45 for a more detailed explanation of the use cases.



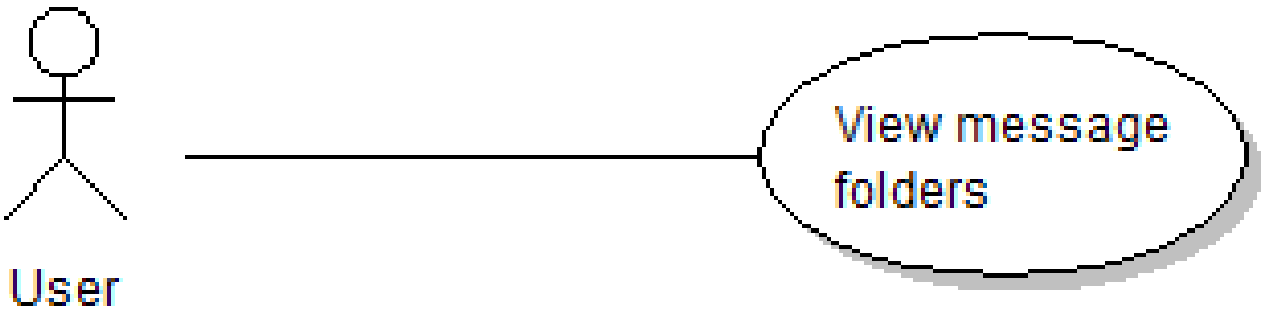
Element	Description
Use case name Requirement Goal	Log in FR1 User logs in with a username and password
Summary hline Preconditions	The user is prompted with a login screen and must type in his username and password 1. The application is running
Flow of Events	1. User starts application 2. User is prompted with username and password 3. User types in username and password and presses the login button 4. User gains access to the application data and functionality
Exceptions	User types wrong username and password and is denied access

Table 4.15: Textual use case - Log in



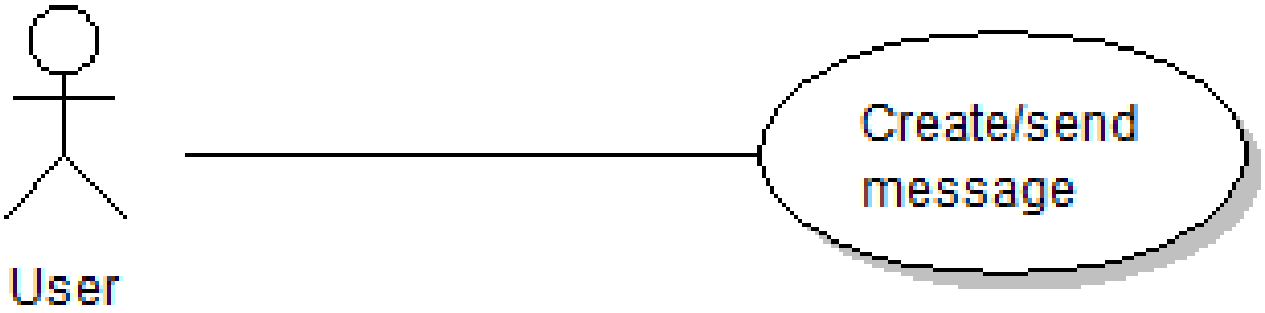
Element	Description
Use case name	View messages
Requirement	FR3, FR4
Goal	View received and sent messages
Summary	The user would like to view received and sent messages
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. The user selects a message from either the inbox or sent messages 2. Message is showed to user
Exceptions	There are no existing messages

Table 4.16: Textual use case - View messages



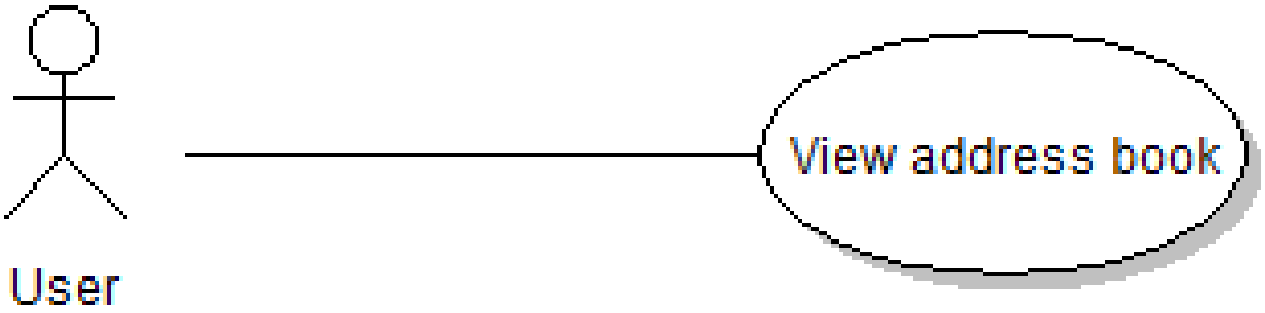
Element	Description
Use case name	View message folders
Requirement	FR3, FR4
Goal	View inbox and sent messages
Summary	The user would like to view the inbox and a list of sent messages
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. The user enters the inbox or sent messages 2. The list of messages in the inbox or sent messages is presented to the user
Exceptions	There are no existing messages

Table 4.17: Textual use case - View message



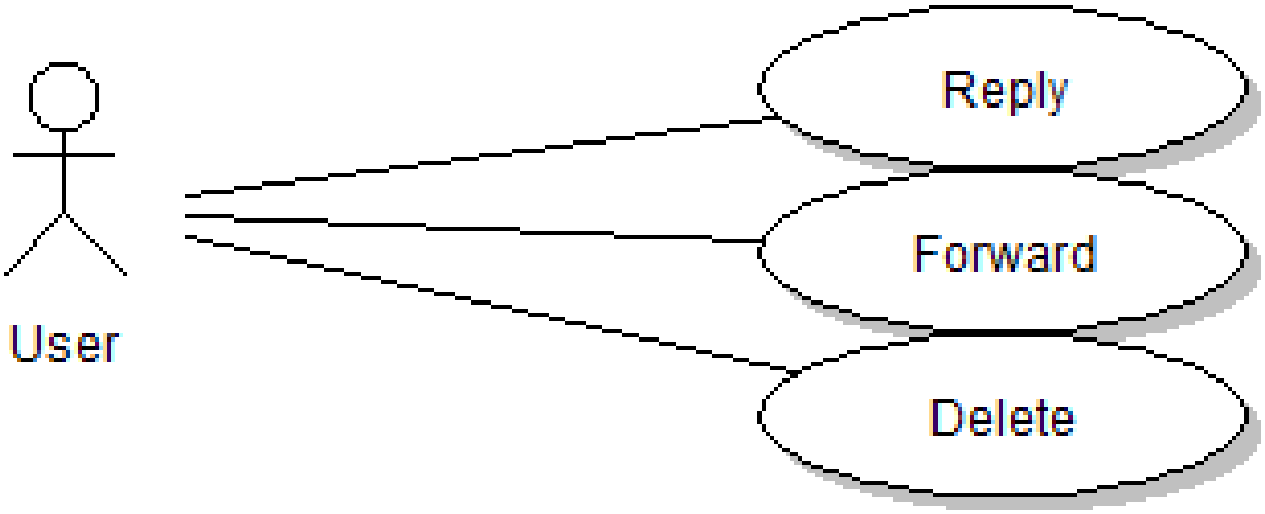
Element	Description
Use case name Requirement Goal	Create message FR2, FR6 User creates and sends a complete message
Summary	The user would like to create/send a message to a receiver with a subject and text. The user would also like to set the priority, security label and type of the message as well as being able to add an attachment from other applications.
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. User selects new message 2. The user adds the recipient(s) and subject to the message 3. The user sets the priority, security label and type 4. The user adds attachments if needed
Exceptions	The user does not set priority, security label and type and default values are set

Table 4.18: Textual use case - Create message



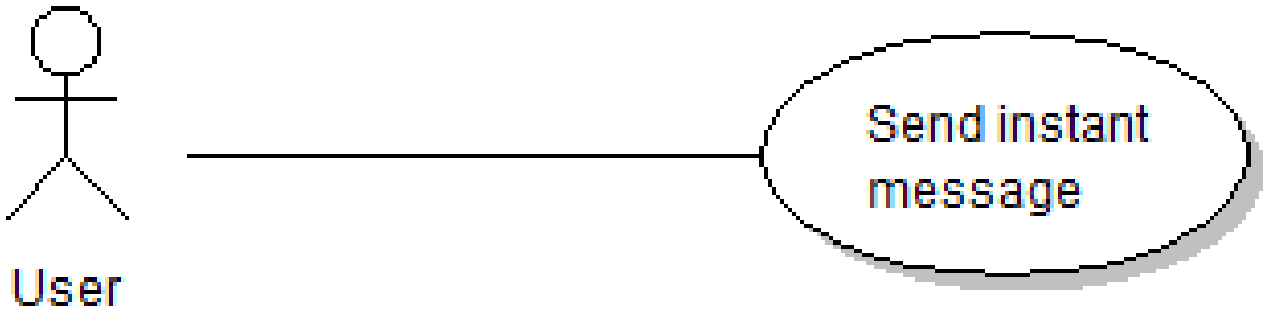
Element	Description
Use case name	View the address book
Requirement	FR5
Goal	User can view and interact with the address book
Summary	The user enters the address book and is able to view contacts
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. User selects address book 2. User selects a contact to send a message to
Exceptions	-

Table 4.19: Textual use case - View the address book



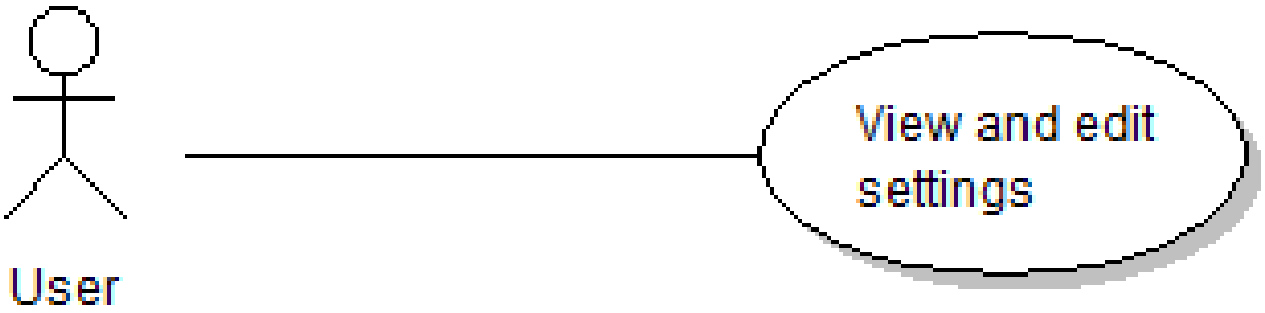
Element	Description
Use case name	Reply, forward and delete
Requirement	FR8
Goal	Reply, forward and delete messages
Summary	The user would like to reply to a message, forward it or delete it
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. The user selects a message from either the sent or inbox messages 2. Message is showed to user 3. User chooses to reply, forward or delete the message
Exceptions	There are no existing messages

Table 4.20: Textual use case - Reply, forward and delete message



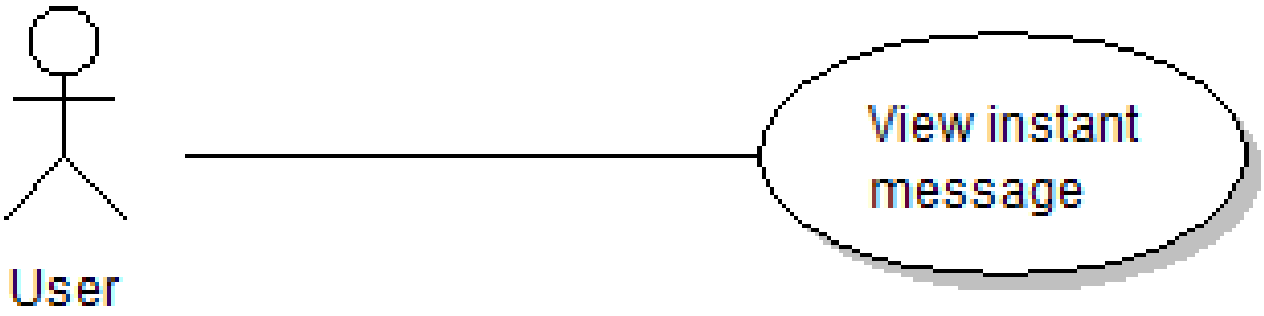
Element	Description
Use case name Requirement Goal	Send instant message FR9 User sends an instant message
Summary	The user sends an instant message with predefined content to predefined receivers
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. User selects new instant message 2. The user sends the message
Exceptions	The predefined values have not been set

Table 4.21: Textual use case - Send instant message



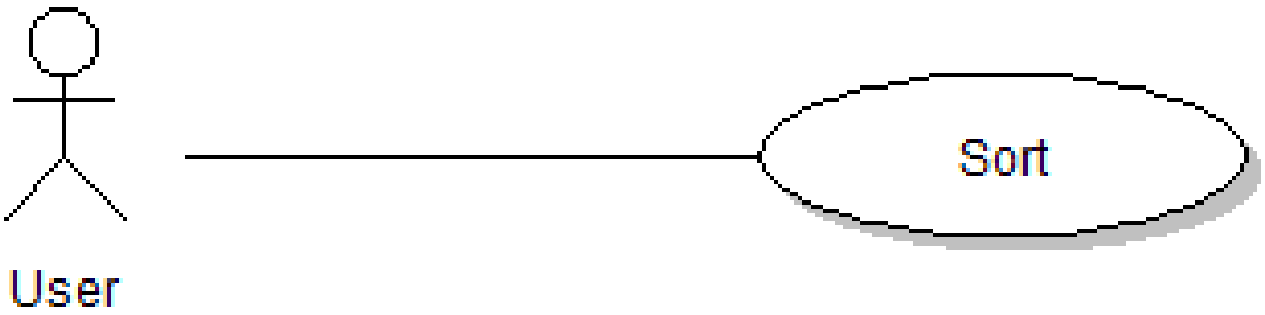
Element	Description
Use case name Requirement Goal	Settings FR10 The user can alter settings
Summary	The user enters the settings menu and alter the settings to suit his own preferences
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. User presses the settings button 2. User alters the settings he wishes to 3. Application saves the alterations
Exceptions	-

Table 4.22: Textual use case - View and edit settings



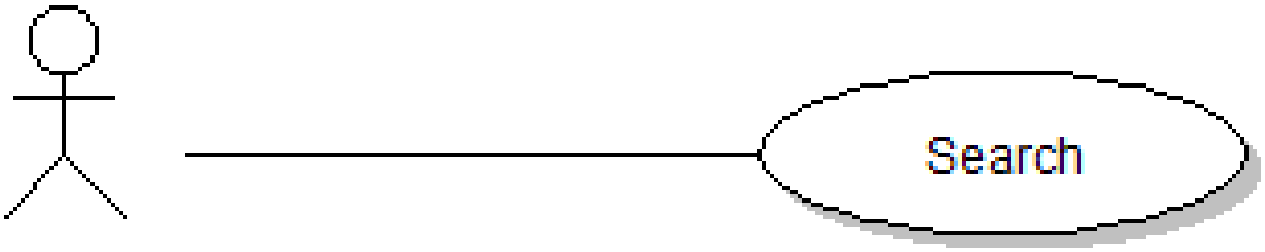
Element	Description
Use case name	View high priority message
Requirement	FR12
Goal	User views a high priority message
Summary	The user receives and views a high priority message
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. User receives a high priority message 2. A popup appear on the users screen 3. User presses the popup and views the high priority message
Exceptions	-

Table 4.23: Textual use case - View instant message



Element	Description
Use case name Requirement Goal	Sort FR13 User sorts the messages in his inbox
Summary	The user chooses a value that he wishes the messages to be sorted by
Preconditions	1. The application is running 2. The user is logged in 3. There are messages to sort
Flow of Events	1. The user selects the value he wishes the messages to be sorted by 2. The application sorts the messages and displays them to the user
Exceptions	-

Table 4.24: Textual use case - Sort



Element	Description
Use case name Requirement Goal	Search FR14 User receives search results for a given search text
Summary	The user search in the application for messages, contacts etc through a search field at the top of the screen
Preconditions	1. The application is running 2. The user is logged in
Flow of Events	1. User highlights search field and inserts his the text he wants to search for 2. User presses the search button 3. Application shows search results to the user
Exceptions	-

Table 4.25: Textual use case - Search

4.5 Requirements from the customer

1. **General functional requirements**

   (a) The application should offer a message service adapted for use on mobile phones. The application must have a user interface that requires little or no training and is efficient to use.

   (b) Intended users are people in need of a reliable and secure communication channel to receive and send more or less time-critical information. Users are assumed to have access only to a table or phone.

   (c) The application's functions are similar to a regular email client, but with a user interface that is suited for small devices, sending of short messages (maxmimum 500 characters), support for domain-specific attributes, as well as reliability and safety modifications.

   (d) The application should use common standards for Internet Mail to communicate with a XOmail SMTP Gateway. The gateway offers email services based on SMTP (RFC 5321) and SMTP Submit (RFC 4409), Internet Message Format (RFC 5322), X.400-SMTP MIXER (RFC 2156) and MMHS in Internet Mail (RFC 6477).

2. **Platform**

   (a) The application should work on an Android-based smartphone or tablet. The user interface should be adapted to the specific screen sizes of these devices, as well as for touch screens in general.

   (b) The application should be able to work over any network with bandwidth 64kbps or more.

   (c) The application should be able to communicate via IP networks towards a XOmail SMTP Gateway (IMAP/POP3 and SMTP).

   (d) The application should work with reduced functionality towards a common mail server with IMAP/POP3/SMTP. Functionality that only works against XOmail must be documented.

3. **Message operations**

   (a) The application should make it possible to communicate by messages. Messages are to consist of a combination of text, pictures and video.

   (b) Attachments should be retrievable from files stored on the device, or through other applications. For example, it should be possible to take a picture and add it to the message.

   (c) It should be possible to create, edit, send, reply to, forward and delete messages. It should also be possible to browse and open received messages.

   (d) The application should support the following military attributes from RFC 6477: MMHS-Primary-Precedence, MMHS-Message-Type.

(e) The application should support security labeling using the SIO-Label header. The application must support the security labels listed below. All messages must have a security label, but it should be possible to configure a default value.

- Norwegian: UGRADERT (ug), BEGRENSET (b), KONFIDENSIELT (k)
- English(generic): UNCLASSIFIED (u), RESTRICTED (r), CONFIDENTIAL (c)
- NATO: NATO UNCLASSIFIED (nu), NATO RESTRICTED (nr), NATO CONFIDENTIAL (nc)

(f) It should be possible to ask for a delivery report and/or a receipt notification from the message reciever.

(g) It should be possible to see status of the messages where a delivery report or a receipt notification was requested.

(h) It should be possible to send instant messages with a predefined classification and priority to a predefined list of recipients. These may consist of a predefined text, or information (i.e. a picture or current location GPS) from another application. To send an instant message, it should only be necessary to perform three or less GUI operations, for example: Open application with seperate icon for instant message, select the text, and select the recipient.

(i) It should be possible to send messages with content created by other applications on the same device.

4. **Sending and receiving of messages**

(a) If message sending fails, the user should be notified. The application should automatically attempt to resend the message, and the user should receive a warning.

(b) Upon receipt of a mesage with priority FLASH or OVERRIDE the user should be notified and the application given focus. The user should be able to see what the message is about (title and perhaps the top text of the attachments). The user should be able to open the message directly.

(c) The application should support push messages from the server.

(d) There should not be used bandwidth when the application is not sending or receiving new messages.

(e) Messages to be sent should be sorted by priority.

(f) If the user sends a message with priority OVERRIDE, it should take precedence over all other messages. If a message is about to be sent, the transfer should be canceled and the high-priority message should be sent first.

5. **Security**

(a) Communication with the server should be encrypted with SSL.

(b) Messages should be signed with S/MIME when sending.

(c) S/MIME-signed messages should be verified upon reception.

(d) Private keys for email signing should not be stored in clear text.

6. **Adress book**

   (a) The application should retrieve any updated address books from the server.

7. **Non-functional requirements**

   (a) The application must describe the design and implementation of security features, and demonstrate that they are sufficient. These include login, SSL, signing/verification, priority handling and safety labeling.

CHAPTER 5

PROJECT STUDY

This chapter will cover all the preliminary studies carried out in this project and will provide an understanding of the problem to be solved, what solutions already exist and what solutions we have chosen. Section 5.1 describes the problem and the domain-specific features the application is supposed to have. Section 5.2 describes the current system, XOmail, and its functions. Section 5.3 outlines the planned solution. Section 5.4 investigates solutions similar to what is wanted from this project. Section 5.5 discuss different software development models and our choice. Section 5.6 lists the different programming langauge options as well as a reason behind our choice. Section 5.7 describes the parsers libraries. Section 5.8 describes the choice of an IDE and the reason behind the choice. Section 5.9 discuss the issue of creating a service locally or remotely. Section 5.10 describes the security assumptions we have to make about the hardware, environment and users of the application, and the studies into secure communication and storage. Section 5.11 give an analysis of results of trying to monitor the bandwidth usage from the application with Whireshark. Section 5.12 gives details about the theory of compressing data as the customer requested a study of how this might be done in our application. Section 5.13 lists the different intellectual property rights and licenses we have used in the project. Finally, section 5.14 gives a summary of all the conclusions made in the different sections.

5.1 Problem description

This section will describe what the problem is and the domain-specific features that must be implemented in the application. In this section it refers to the requirements from the customer in section 4.5.

5.1.1 What Thales has requested

Thales have requested us to develop a message service that is custom made for use on mobile phones, with a user interface that has good affordance and is easy and effective to use.

The intended users of this application are people who need a reliable and secure way of communicating more or less time-critical information. Users are expected to only have access to either tablets or smartphones, not computers.

5.1.2 Functionality

The application's functionality should be similar to a regular email client, but with a user interface that is custom made for small screens, short messages, military domain-specific attributes, reliability adjustments for field usage and security adjustments for handling classified information. The application should use the SMTP and IMAP to communicate with Thales' XOmail gateway.

The application should not use bandwidth when not sending or receiving messages. The application should also be able to retrieve updated address book information from the server.

5.1.3 The solution

As Thales has requested, the solution should be implemented on Android-based smartphones or tablets (Req. 1a and 1b) and the user interface should be optimized to the screen size of the given device (Req. 2a). The application should support bandwidths as small as 64 kbps (Req. 2b) and communicate with a XOmail server throught IMAP/POP3 and SMTP(Req. 1d and 2c). The application should also work towards normal email servers, but then with reduced functionality (Req. 2d).

5.1.4 Messages and their content

The messages sent with XOXOmail should support text, pictures and video attachments (Req. 3a). The media should be accessed from the local storage on the phone or from other applications (Req. 3b). A user should be able to create, edit, send, answer to, forward and delete messages as well as browse and open received and sent messages (Req. 3c).

5.1.5 Military attributes

The messages sent with our application should support certain military attributes (MMHS), including priority (Req. 3d), type (Req. 3d) and security label (Req. 3e). Priority indicates the level of urgency, whereas type indicates the type of the message. Security label says something about the receiver of the message, as it states what security clearance the receiver need to have to be able to open the message.

5.1.6 Message status

It should also be possible to request a delivery report and a receipt notification from the recipient of the message (Req. 3f). The one who sends the message should be able to monitor the status of the delivery report and receipt notification (Req. 3g).

5.1.7 Instant messages

A special feature of the application should be the ability to send an instant message, with predefined military attributes to a predefined receiver with at most three screen interactions (Req. 3h). The receiver can be a single person or a group. The instant messages should have the ability to include predefined text or information from another application, such as GPS coordinates. It should also be possible to send messages with content that is created by other applications on the device.

5.1.8 Sending

When a message fails to be sent by the application, the user should get a notification and the message should be sent again automatically (Req. 4a). The messages being sent should be sorted on priority, where the message with the highest priority should be sent first (Req. 4e).

5.1.9 High priority messages

If a message with priority "FLASH" or "OVERRIDE" arrives, the application should alert the user in an undeniable way and make it easy to access the newly arrived message (Req. 4b). The application should halt the sending of all other messages when the user sends a message with high priority, and instead give the resources to the high priority message (Req. 4f).

5.1.10 Security

Security is an important issue in the application. Communication with the server should be encrypted with SSL (Req. 5a), messages should be signed with S/MIME when being sent (Req. 5b), S/MIME signed messages should be verified on reception (Req. 5c) and private keys for signing mail should not be stored in clear text (Req. 5d).

5.2 Current system

At the moment, there are no mobile solutions to the already existing XOmail system on computers.

5.2.1 XOmail - Secure message based information handling

XOmail is a message system tailored for messaging within military organizations. The messaging functions within the system are suited for the needs of large organizations with the ability to differentiate between social messages to the entire organization and personal messages to individual users. XOmail has the ability to send messages to predefined distribution lists where the messages gets distributed based on a given criteria like the subject of the message. [9]

The system also has several APIs that can be accessed by other applications. XOmail supports message coordination for use in the drafting process, as well as the ability to have a release authority for a drafted message. Another feature of XOmail is the ability to add precedence to messages, giving high priority messages access to resources and communication channels before messages with lower precedence. This is very important in a military setting. [9]

XOmail has also a high focus on security, and offers security for the messaging platform, servers and gateways. The system also has an extensive administration tool that can be accessed both locally and remotely. [9]

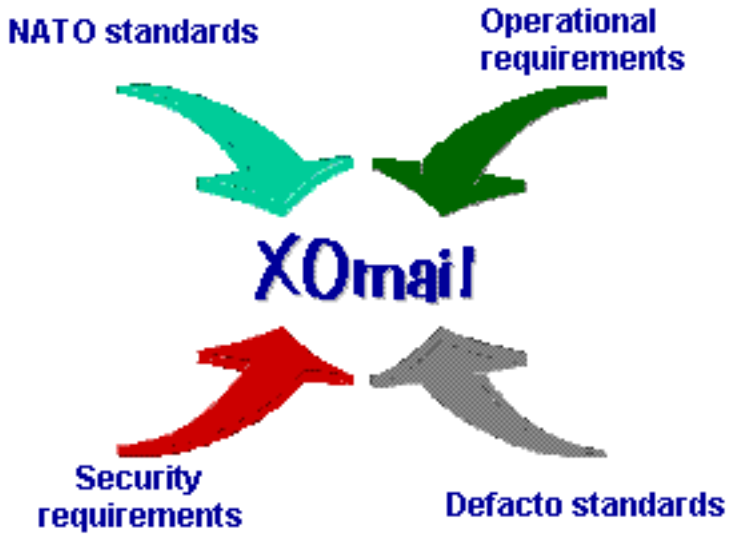


Figure 5.1: XOmail logo

5.3 Planned solution

The planned solution should, in a secure and reliable manner, be able to send and receive messages to and from Thales' XOmail servers from an Android phone. Naturally, this is easier to plan than to evaluate, but this section will describe how we plan to accomplish most of what Thales has requested.

When it comes to sending and receiving messages, these features are easy to test. Ensuring security is more complex. When can XOXOmail be called "secure"? When is our system "reliable"? We imagine that the best way we can proceed is to document all our choices and their implications, as well as suggestions for further improvement if Thales should want to extend our simple prototype.

In the sections above we listed a lot of functional requirements that we will try to implement in XOXOmail. Most of these were given to us directly by Thales through the customer requirements from chapter 4.5, with a relatively short period of discussion and clarification.

The project has also stated some non-functional requirements, mostly related to usability and security. These are more difficult to quantify, and we will try to describe and justify the measures taken do ensure these qualities by extensive documentation.

5.4 Similar solutions

This section will investigate the similar solutions to the application we are to implement and find out if we could get any help from these solutions. The result of the study was the realization that there are few, if any, solutions that give the capabilities that are expected of XOXOmail. However, we will give an overview of existing applications that do fulfill some of the requirements of XOXOmail, starting with discussing of signing and verification and available solutions that implement these features and continuing with a discussion of different email clients that are available and their capabilities.

5.4.1 Signing and verification

A very important aspect of XOXOmail is the signing and verification part. This is an application that will be used to exchange important and classified information, and a message should therefore be verified. This section will start with a short theoretical study of signing and verification and then continue to investigate the available solutions. The study will start with an introduction to public key encryption as well as a short overview of S/MIME and the available applications that use S/MIME.

Public key encryption

Public key encryption is used to secure data in such a way that only the intended receiver can open the message. It involves using two separate keys; one private key and one public key [10], as seen in the example below.

If Bob wants to send a message to Alice, he first have to get Alice's public key. Her public key is available to everyone who wants it, hence the name public key. Bob can use Alice's public key to encrypt the message he want to send her. Given by the mathematical relation between the private and public key, only Alice can decrypt and read the message, and it is nearly impossible or atleast very computational expensive to calculate the private key, and we can therefore within a reasonable doubt guarantee that only Alice can see the message.

Digital signing and verification

Alice might not be sure that Bob was the sender of the message, and this is not possible by just using the private-public key scheme as explained above. But by introducting digital signing, this becomes possible. Digital signing is a way of signing data so that the receiver of the document knows that the sender is exactly who he says he is, and that the content has not been tampered with [11]. By continuing our example, Alice wants a way of knowing for sure that Bob was the sender of the message. This is ensured by digital signing, which first is done by computing a message digest, which is just a hash of the message content. This hash is then encrypted with Bob‘s private key. When Alice receives the message she will use Bob‘s public key to decrypt the signature, and thereby receiving the hash value. If Alice creates a hash of the received document and it matches the hash of the signature, she knows that the message was from Bob and that the message has not been tampered with.

Digital certificates

But what if a third person wants to deceive Alice, and sends her a message claiming to be Bob and giving him his public key? Alice knows that the message was sent by someone with Bob‘s name, but how can she be sure that he was the sender? The solution is a Certificate Authority, also known as a CA. The CA creates a digital certificate for Bob‘s public key, which ensures that it really is from Bob. To check that it really was the CA that created the certificate, Alice can use the CA's public key to check the signature on Bob‘s certificate. If everything turns out correct, she can now trust that the message was from him.

S/MIME

S/MIME is an abbreviation for Secure/Multipurpose Internet Mail Extensions. It is a standard for public key encryption and signing of MIME data [12]. MIME content is text in character sets that differ from ASCII, non-text attachments, message bodies with multiple parts and non ASCII headers. It is just an extension of the well known mail format so that it is possible to send content other than plain text.

Available application with signing functionality

An already existing application for sending signed messages is X509Tools.

X509Tools
X509Tools is an application that can send mail with S/MIME capabilities, but its main purpose is not sending mails. The main purpose is to give these capabilities to external mail clients via an interface. It also has a Certificate Store which can be used to check certificates already present on the phone.

5.4.2 Available email applications

There are many applications available for sending mail from an Android phone. This section will give a superficial overview of the applications as well as a short discussion of their advantages and disadvantages.

Built-in client

The first and most obvious client for sending email is the built-in Android email client which is what one can call abandonware. It is often outdated and lacks a lot of basic functions.

Gmail

One of the most used email clients for phones is the Gmail application. It is developed by Google, and is therefore seen as secure application with a streamlined design. Unfortunately, it stops there. Google has been neglecting all problems that it has, and new features are not emerging from complaints stated by existing users. Functions like sorting of mail is lacking and the application is not very reliable. Fortunately, there exists a fork of GMail, called K-9 Mail. GMail does not have S/MIME implemented on the phone version, but an extension for Firefox integrates reading and sending of signed mail directly from Gmail's web interface [13].

K-9 Mail

K-9 Mail has all the features the users wanted in Gmail, and some extra. Most features are configurable. It is possible to edit application polling settings, show a short excerpt of the message text, and message selection checkboxes can be temporarily hidden. So can the starring of the messages and it is even possible to adjust font sizes and date formats. The clue here is configurability. K-9 Mail does not currently have an official S/MIME implementation, but there exist some hacks around that partially or fully manages to implement it. This is therefore not seen as a fully secure alternative. An upside is that K-9 Mail is free.

MailDroid

MailDroid has many of the same functions as in K-9 mail, but also incorporates spell checking, changing font sizes and colors on text and coloring on email text. It is a solid email client that will meet the needs of most users. MailDroid currently costs 111,86 NOK, which is in the high-end price range of apps. A free ad version is available. As far as we can see, MailDroid does not come with S/MIME.

R2Mail2

R2Mail2 is the result of further development of X509Tools. Here, the focus is more on taking the good parts of X509Tools and integrating it into a fully functional email client. It uses the same security libraries as used in X509Tools, and does also include digital signatures and digital encryption and decryption based on personal Soft-Token keys [14]. It has the ability to store private keys and passwords in an encrypted database known as the Key-Store. This is encrypted by a master password. The certificated can be validated by OCSP and LDAP. As it is built on top of X509Tools, it has full S/MIME support. This is a no nonsense email client which provides the ability to send encrypted and signed mails.

5.4.3 Similar solutions conclusion

It became very clear through these studies that a solution similar to what we want from XOXOmail is not available on the market, but there exists solutions that partially try to implement S/MIME, like K-9 Mail. There also exists solutions that fully implement S/MIME for external clients, like X509Tools. A problem with using X509Tools, is that it gives poor performance to applications that utilize this S/MIME capability, as the email message will have to be stored to disk between the mail client and X509Tools. An alternative is therefore to use R2Mail2, which gives all one can want regarding security.

As an aside, it is of interest to note that we have been in direct contact with on of the developers of X509Tools and R2Mail2. His name was Stefan Selbitschka, and he was very interested in this project. If necessary, he could build us a separate version of the R2Mail2 library that incorporates e.g. header specific data that is unique to XOXOmail. This was mentioned at a meeting with Thales, but no discussion was made further on this topic. It seems that it is more important to investigate possible solutions than to brute force a fast solution to the prototype of XOXOmail.

Even though there exist email clients on the market where security is important, none of these have the special needs that XOXOmail will have to fulfill. This is mainly because XOXOmail is a specialized software created for a specific user group. A general purpose application will never be able to fulfill these requirements. It is therefore clear that XOXOmail has to be implemented ground up to get the capabilities we are looking for. We can still use a lot of the knowledge regarding what works well and not so well in other mail applications in our product, as the basic functionality is quite similar.

5.5 Software development model

This section will describe the two different options we had for a software development model; waterfall and agile.

5.5.1 Waterfall

The waterfall model is a sequential design process that is often used in software development processes, in which progress is seen as a flow of water through the phases of conception, initiation analysis, construction, testing, production/implementation and at last maintenance [15].

This is a model that was originally used in hardware industry, but in lack of a better model, was adapted for software development. In the waterfall model, each of the stages is sequential, and another phase starts where the previous ends. There is no room for different approaches here. Before implementations of a software product, the documentation has to be carved in stone. If, for example, the documentation is only half done, it will ruin the whole process. Therefore, one has to be entirely sure that a phase is over before a new one begins.

5.5.2 Agile

Agile development is based on interative and incremental development, and promotes a workflow process that embraces change [16]. The Agile Manifesto states that we get better software by concentrating on the software itself, the customers receiving the software and focusing on the team dynamic of the development team as well as responding to changes as they arrive. A lot of software development processes has emerged from the agile development method. We will now discuss two of them, which is the two most relevant to consider for this project.

Kanban

Kanban Development is a fully transparent process that has an emphasis on just-in-time delivery where the main focus is not to overload the developers [17]. It consists of Kanban, which is a process board, which is a overview of what to produce, when to produce it, and how much to produce; and the Kanban method itself. It is an agile software development method that uses a pull system to find problems and encourage change. What is important to notice about the Kanban development process, is that it is not a process in the way that we have a series of steps from start to finish. We just focus on what is good in the current development environment and stimulate further change. In order for this to work, we must respect the process of development, the roles of the team, their respective responsibilities and their title.

Scrum

Scrum is used in agile software development. Rather than being a full description of the process of software development, it is a framework setting the boundaries for the software development team [18]. The reason this is done is because the team knows best how to solve the task they are presented with.

Scrum relies on a self-organizing, cross functional team. This means that there is no team leader who decides who will do what. This also sets a boundary on the maximum size of team, which is about eight to ten persons.

The development cycle is created by basic units, called a sprint. These sprints last between one week and one month [19]. In the start of each sprint, there is a planning meeting, where tasks are defined and goals are made. The tasks are taken from the backlog (in scrum terminology: "defined from"), and are refined into a task specification which can be performed by a programmer. During each sprint, a part of the completed product is made. It is not unusual to create a basic version of the complete software during the first sprint, and then add more functions as we go, during the later sprints.

Each day during a sprint starts with a scrum meeting. The main purpose of this meeting is to give everyone a status report of what is going on. Each member of the group summarizes what he has done, what he is about to do, and what stands in his way of doing his tasks. These meetings have a maximum duration of 15 minutes, and should be done standing, as this keeps the talks short and effective.

The organization of the groups tasks is done by using the scrum task board. Here, one can see which tasks are unassigned, in progress, in testing and done. As the group consists of few persons, most of organization can be done via direct communication from person to person or during the meeting.

5.5.3 Software development model discussion

Software development projects are either agile, like scrum, or based on a more rigid model like the waterfall model. There are reasons that they both coexist today. There are both positive and negative aspects of both of them.

Supporting arguments for using the waterfall model are many. By using as much time as possible on the early stages of software one hopes to eliminate problems later on. If we see that something is impossible to implement, we can find a new solution, and still use no extra time on the coding phase of the software.

As the waterfall model is a rigid development method, a lot of documentation is done. This may be seen as bureaucracy, but it has one huge advantage; all code is well documented, which means that if all the developers of a project decides to quit, a new team of developers can read up on the documentation and continue where the others left. This makes the waterfall model good for projects that do not change over time and where documentation is of outmost importance. It is, for example, very clear that a software system that keeps an airplane in the air, cannot be developed using an agile process where there exists little or no documentation. Life critical systems like these must go through bureaucracy to maintain the levels of reliability that is needed.

On the other hand, there are also many supporting arguments of an agile process. One of the most central ones are customer awareness. In an agile process, the developers know that the customers might change their minds. Or said in another way: They know the customers will change their mind. An agile process will not see this as a problem, as changes can easily be incorporated into the next build of the software. In a waterfall model, there is no way of accomodating changes when coding has started. If one sees that a part of the software is difficult to implement in an agile process, it is just a matter of finding a new way of getting the task done and then implement it. This makes agile processes perfect for fast changing environments where the software is in constant change or when the developing firm is unsure about certain aspects of the software, but know approximately which direction to take it.

Another supporting argument for agile processes is that these are low cost processes. Almost no time is used on time consuming documentation that will never be read. Almost all time is used on developing the software, and is therefore more efficient.

5.5.4 Software development model conclusion

As we can see, there are many pros and cons of both agile processes and waterfall models. In the context of this Customer Driven Project, it seems most benefitial to use scrum, as the specifications are unclear. We are not sure how long the documentation or the development process will take, so both activites will be done continuously. Going for a waterfall model will give us little or no room for error. With the experience of this developing team, it would be ridicolous to think that we would go through the entire project without making mistakes.

We could go for a Kanban process, but it is a bit hard to rely on the existing structure of the group when there is no such thing. In a group where we do not know how we are going to organize the group, it is best to go for a model where we have some rigidity in the process itself to help us find the right direction.

We have chosen scrum as our group organization model, as it fits this projects size and timeframe. There is a lot of uncertainty in our project regarding how we should organize the project from start to finish, as well as a limited time frame. The limited time frame forces us to make choices on which features we are able to implement. But, as we are able to see what needs to be done in the near future (three weeks to one month), we can divide the project into sprints of this size and create a more detailed description of each task as we go.

5.6 Programming languages

We have decided to write the program in Java, and used XML for creating parts of the supporting structure of our project. A summary of these choices are presented at the end of this section.

5.6.1 Java

Java is a class based, object oriented language that incorporates concurrency and is a general purpose language. Writing in Java, means that we do not have to recompile our code for every kind of Android device we want the application to run on. Our program will run on top of the Dalvik Virtual Machine, which encapsulates the Android application. This makes programming a lot easier, as we do not have to worry about where our program runs - the Dalvik VM takes care of this.

One of the major benefits from writing a program in Java, is that it is compiled into bytecode that can run on any Java Virtual Macine. This means, that the program can run on any architecture that Java can run on.

5.6.2 Native code

It is possible to implement parts of the application using native-code languages such as C and C++ by using the Java Native Development Kit (JNDK) [21]. This may be beneficial if there are good libraries written in these languages, or if we need the extra performance you get by not going through the Dalvik VM. We have chosen not to do this as as it is stated on the Android developer page for NDK: "... you should understand that the NDK will not benefit most apps."

5.6.3 Scripting

By using the Scripting Layer for Android (SL4A), it is possible to edit and execute scripts and interactive interpreters directly on the Android device [22]. By using the APIs available to the full-fledged Android application, it is indeed possible to run software using scripting. Currently available programming languages are Locale, Python, Pers, JRuby, Lua, BeanShell, JavaScript, Tcl and shell. We are not using this method, as stated by Google: "SL4A is designed for developers and is alpha quality software." We will probably be running in to a lot of problems that is not well enough documented.

5.6.4 PHP

Programming can be done in PHP by installing the PHP for Android application and PHP library. This includes a lot of scripting to do common Android tasks, but it is certainly feasible. Unfortunately, documentation is a bit tricky to find, and the user community is small [23].

5.6.5 C#

It is possible to write Android software using C#. By installing MonoDroid created by Xamarin [24], you can develop C# software for Android. The advantages of using C# may be many, depending on the developer. If you are familiar with Visual Studio and the .NET framework, you will certainly feel at home here. The framework is well documented, so there should not be any problem developing XOXOmail using MonoDroid.

The drawbacks of programming with MonoDroid are a few. The first drawback is the pricing, which of current date (10. september 2012) is 2305 NOK for a professional licence. While MonoDroid does support a vast majority of the .NET framework, there are still some features missing, so one should be aware of this before deciding on using MonoDroid.

5.6.6 Programming languages conclusion

As we have seen, there are many ways to create software for Android. One can rely on only one language for programming, or use a combination of languages. We have chosen to develop purely on a Java platform. Why is this relevant for us, as we are creating an Android application?

First, it is the most used developing language for Android, which means we are using a well tested and documented development framework. It is unlikely that we will run into big problems. Second, we are all familiar with using Java, so the learning curve will only be a bit steep for those in the group that are new at Android development. Another benefit of using Java is that, if we decide on implementing a server part, we can create the server first and then decide if we want it to run on the phone or on a computer, be that a Linux or Windows computer or something else. This flexibility is something we only can get from Java.

We chose Java as our main programming language as this is the most widespread language and has most support online. It is the language which has been pushed forward the most from Google, and is the recommended language to use when developing Android applications.

5.6.7 Markup languages

XML

Extensible Markup Language, also known as XML, is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable [25]. Its purpose is to have a simple way of representing data in a textual format which is easy to learn and use. It is often used in programming as a medium for persisting data from objects into a file format which then can the be used later on. Many libraries have been written to make it easier to generate XML data.

JSON

JavaScript Object Notation, also known as JSON, is a text-based open standard designed for human-readable data interchange. This makes it quite like XML. The main difference is that its structure is much simpler than XML. The grammar is smaller and maps directly onto the data structures used in modern programmming languages [26].

5.6.8 Markup languages conclusion

XML is a widespread exchange format which has a lot of libraries that can be used to generate XML data. Even though, it is known that XML is primarily a document exchange format, where as JSON is better suited for data exchange. This means that JSON often is much more readable, as the mapping between objects in programming and JSON representation is much more alike. As a result, the JSON code is easier for machines to read and write. JSON is becoming more and more common, and is now widely adopted by the computer industry. Still, the format is irrelevant for us, as we will be using a serializer and deserializer that ensures that syntax is correct, so we will use a XML-serializer.

5.7 Parsers libraries & tools

5.7.1 XStream

XStream is a simple Java library to serialize objects to XML and back. Using XStream, you can serialize most Java objects without any mapping. Object names become element names in the XML produced, and the strings within classes form the element content of the XML.

The classes that you serialize with XStream do not need to implement the Serializable interface, thanks to XStream handling all serializations. XStream is a serialization tool and not a data binding tool, which means that it does not perform class generation from an XML Schema Definition (XSD) file [27] [28].

Another feature of XStream is that it has the capability to serialize to and from JSON as well.

5.8 Integrated development environment

5.8.1 NetBeans

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, and others. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform (including the NetBeans IDE) can easily be extended by third party developers. [39]

NetBeans IDE is an open-source integrated development environment. The NetBeans IDE supports development of all Java application types. Among other features are an Ant-based project system, Maven support, refactorings, and version control. [41]

All the functions of the IDE are provided by modules. Each module provides a well defined function, such as support for the Java language, editing, or support for the CVS versioning system, and SVN. NetBeans contains all the modules needed for Java development in a single download, allowing the user to start working immediately. Modules also allow NetBeans to be extended. New features, such as support for other programming languages, can be added by installing additional modules.
A choice on what development environment had to be made. The choice was between NetBeans and Eclipse.

5.8.2 Eclipse

Eclipse is a software development environment that can be used to develop in Java, amongst others. Even though it is meant for Java developers, users can customize Eclipse by using some of the various plugins that exist. [40].

One of the benefits of using Eclipse is that it has good plugins for Android development. The aim of the plugins is to make it easier to start programming for Android.

5.8.3 Integrated development environment conclusion

We made an early choice to use Netbeans. The reason for this is that some of us have had a lot of problems with Eclipse. We decided that we all should use the same IDE to be sure that we would not get compatibility problems on the code generated. One of the features that is missing in NetBeans is the ability to design and preview the Android GUI design directly in the IDE, but we did not find this feature important enough to choose Eclipse.

5.9 Remote vs. local service

As the requirements document states the application should be able to receive messages even when the application is not running in the foreground. Fortunately, the Android platform facilitates this functionality in the form of a 'Service' [42]. But in order to utilize this feature effectively, some key architectural decisions must be made. The biggest decision is if the Service should run in the same process as the application itself (local service), or if it should have its own process (remote service). Although the choice of can be made transparent to the rest of the application, there are some key differences that must be considered.

5.9.1 Resource footprint

As stated above, a remote service will run in its own process. This means that the application will allocate two processes worth of memory in order to run. It will also require more from the CPU since any call to the service will require the CPU to switch to another process. These two drawbacks are mitigated by the use of a local service, where the service runs in the same process as the application.

5.9.2 Communication with service

For a remote service you will have to use some sort of interprocess communication(IPC). On the Android platform this is solved by the use of 'Android Interface Definition Language'(AIDL) [43], whereas for a local service a simple IBinder [44] is needed. With respect to modifiability the latter is the best solution since the developer does not update AIDL files whenever an interface changes, nor does he/she have to manage the intricate process of breaking down object used in the application down to primary types for sending over IPC.

5.9.3 Remote vs. local service conclusion

There was little that supported the use of a remote service for this application, so the natural choice is the local service, both for its ease of use and performance.

5.10 Security studies

This section will describe the security studies carried out in the project. The section will start by stating what assumptions we needed to establish in order to make the XOXOmail application. The section will then continue with describing how secure communication can be done. Then it concludes by looking at how to store information securely. Each of these topics will have a conclusion to sum up what were the most important points learned from the study.

5.10.1 Security requirements

In order to make the XOXOmail application, we need to establish some assumptions about the underlying hardware and the environment the application is going to be used in. One can never be sure that a program is completely safe from attacks, so one has to assume that the surroundings are at least a bit safe. In this context, surroundings might mean a number of different things. It can mean the Android phone itself and other software installed on it, it can mean the user and his system administrator, and it can mean the network the messages travel over. This section will go through each of these situations in turn.

The phone

All modern operating systems focus at least somewhat on security, and the Android OS especially. Building on a Linux core, Android counts each application as a separate user. This means that, in principle, each application has a separate hard disk region, separate memory area, and no way of stealing data from the other applications' areas. However, there are ways to circumvent this.

The most known way of circumventing the Android security is by rooting the device. A user with root acces can do whatever he wants with the phone.

In order to mitigate the possible security breach of a rooted device we want the persistent storage to be encrypted. This ensures that even though the device is rooted, data confidentiality is maintained. The Android OS provides such a feature in newer version, using AES with ESSIV. This is however not related to the implementation of XOXOmail, as it is controlled by Android itself, not the application. In addition to not being under the application's control, it still leaves the backdoor partially open due to the fact that a unlocked device will automatically decrypt data on demand from the application.

In addition to the encryption, we want a timeout lock on the phone, locking it down and encrypting whatever is currently not encrypted if the phone has not been used in a while. Exactly how often the encryption should be performed would depend on the sensitivity of the data that could conceivably be stored on the phone, and would be up to the system administrator. Last, we have to assume that the phone arrived to the user in an uncompromised state. This means that the phone was not compromised by the manufacturer, or during transport out to the user. Tampering during transport is generally easy to check, as most if not all manufacturers deliver their goods in a sealed state. Tampering by the manufacturer is harder to detect. The best one could do is to carefully choose manufacturers that the user can trust or, for more high security applications, manufacturers that the user can maintain careful oversight over.

The users

We have to make some assumptions about the users of the application. User error and social engineering remain the most common reasons for data leaks, so we need to know to what level we can trust the user.

We assume that the user has at least some knowledge of basic digital security. This means that we assume that the user password is not susceptible to to a dictionary attack, that the user will not give his password to anyone, and that he is aware that his phone contains restricted information with all the implications this has.

In case the user should lose his phone, we assume the system administrator has taken advantage of Android's ability to have a phone reset to factory settings remotely. Resetting the phone would destroy all application data, mitigating the possibility of sensitive data getting into the wrong hands.

The system administrator can also help by limiting the user's ability to install new apps. The most effective data mining attacks on Android phones have historically been trojans disguised as games, wallpapers or other inconspicuous software. Sometimes a potential user could be warned by the downloaded application requiring unusual permissions, but in practice most users lack the experience to notice this. In other words, if the administrator enforces locks on some or all installations it would help to keep the system safe.

Having user courses in digital security could be beneficial, but we are hesitant to add it as a safety requirement as it would contradict our goal of producing a simple, intuitive system that can be quickly deployed without the need for a training course.

Security requirements conclusion

In order to be able to make assumptions on what is secure and insecure on a phone, we had to make assumptions about the underlying hardware and environment. This was something we had to do before we could continue to assesments about what is secure and not.

5.10.2 Secure communication

Drivers

Communication channels are probably the easiest way to gain unauthorized access or internal information from the system since it does not require any physical interaction with the device. For example a packet sniffer would easily pick up on unprotected data sent through a wireless network. In an effort to mitigate this seccurity issue, network encryption is widely in use today, which is the topic of discussion in this section.

Requirements

In order to discuss and evaluate the different possible opportunities regarding end-to-end encryption of the communication channels it is needed to have some features to compare.

Transparency

In order to provide flexibility and modifiability one should prefer a transparent implementation of the security protocol from the application's point of view. By using a protocol that is transparent to the application it allows the developers to focus on product development, and opens up for the possibility of changing the protocol at a later stage.

Level of security vs. performance

Different security protocols have different levels of security and performance. It is therefore important to recognize the level of security needed and the performance penalty associated with the chosen protocols. In general a higher level of security would always be used when performance is not an issue.

Implementation complexity

The implementation complexity of today's secure protocols is very high and not feasible during this project assignment. There exists, however, implementations of several secure protocols on the Android platform as it is.

Secure communication discussion

There exist several methods which can be used to secure a communication channel, some more prominent than others. Secure communication could in theory be implemented at every level of the TCP/IP 5-layer reference model [108]/"Internet model" [109]. The first option was to create our own secure protocol at the application layer that would provide a transparent wraparound of the transport layer, but this is rarely an optimal solution and will probably result in a poor and insecure imitation of an existing solution.

Second, we choose to implement the encryption at a lower level, as the network or link layer [45]. For network layer encryption it would be possible to use IPSec, this is however not possible through Java or Android [46] as it would require interfering with the operating system beneath. The same can be said about link layer solutions; it would require interference with the operating system. But the link layer encryption relies on the security of each link host, something that cannot be guaranteed when sending over the Internet.

Third we have the option of using a pre-existing application/transport layer protocol like SSL/TLS. In order to maximize security gain it is recommended to use TLS 1.2 [47] which has improved security relative to earlier version of TLS and SSL.

Secure communication conclusion

After this study we found that the only reasonable and feasible solution is to use SSL/TLS. This was mainly due to the transparency of the implementation and the lack of complexity since it is already implemented. This notion was only strengthened by the requirements from the customer.

5.10.3 Secure storage

The application we are developing will need to store some information that is private and should not be easy to access. In order to do this we need to make the users log in with a username and a password. The password is used to encrypt the private information and make it unavailable to unwanted eyes.

No retyping of username and password

A feature that would be nice for XOXOmail to have is to not be required to retype your username and password every time the application is started.

A way to do this is to store the username and password locally on the phone. In Android applications this is usually done through SharedPreferences. Though SharedPreferences is sandboxed and thus prevents other applications from accessing the data. If you also encrypt the credentials instead of just storing them as clear text it would add even more security.

The problem with this solution is that if an attacker should in some way gain physical access to your phone he may still be able to get the data. If you are already logged in, the perpetrator could just open up the application and view the data he wants. Even if you were not logged in the attacker could root the phone and gain access to the values saved in Shared Preferences. The values could be encrypted but it would not help, because an attacker that has access to the values in SharedPreferences is also likely to have access to the application's binary, and thus the keys to decrypt the password.

Creating a custom account type and server side authentication

Another solution to avoid retyping of username and password is to create an online user account and add this to the centralized AccountManager registry. The user can then type in his username and password once, and gain access to the various resources online. The credentials are here authenticated on the server and we can therefore store the credentials as a cryptographically secure token on the device. This would make sure that an attacker would not get access to your actual username and password.

This solution also suffers from the fact that if the user is already logged in, an attacker could view all the information. One could solve this by implementing a timeout on the application (If the user does not perform an action after x minutes, the application is terminated and the user gets disconnected).

Symmetric encryption with retyping of username and password

Symmetric encryption is also a possible solution for securing the data. If we are going to use this type of encryption we can't store the keys because of the lacking general purpose, system-level secure storage.

A way to solve this is to derive the keys from a user-entered password. To make sure the keys derived is random and hard to brute-force we need to use a standard PBE key derivation method. A common way of doing this is with PBKDF2.

This could be a suitable solution, at least for the parts of the application that does not need to happen instantaneously because the password never should be stored in plain text.

Secure storage conclusion

If we chose to create custom account types it would involve a lot of server side implementation that would be time consuming and problematic. So we should try to avoid this solution. The two other possibilities are both valid ones if we make certain assumptions.

If we are going to implement the version where username and password is saved in the local storage we would need to assume that no unwanted physical access to the phone will occur. Given this assumption this would be the best and easiest solution to implement.

To implement the secure storage with derived keys we need to assume there will be no problem for the user to log in every time he starts the application. This will make sure the information is secure even though the phone should get misplaced or stolen, but this is more complicated to implement than saving the credentials locally.

5.11 Wireshark analysis

This section will describe a Whireshark analysis of the bandwidth usage from the application, as requested by Thales.

5.11.1 Wireshark

One of the most useful features of Wireshark is the ability to filter different packages on various parameters and sort them. This allowed us to view only the packages that went to and from XOXOmail and not all the non-related messages coming from the device. Wireshark also has the ability to give a summary of the packages filtered, giving us vital information in a clear and concise visual manner. [79]

5.11.2 Analysis

In order to get an overview of how XOXOmail sends and receives messages in the form of packages, we needed to perform an analysis of the data. Important areas that we were interested in more information about:

- How many bytes it takes to send and receive a message
- How long it takes to send and receive a message
- If there are any inconsistencies with regards to the way it should work

To monitor the traffic in XOXOmail we used an Android virtual device with the network speed set to GSM (14.4 kbits/s in upload and 14.4 kbits/s in download). This device then generated a capture file that could be viewed and analyzed in Wireshark.

We use TLS (Transport Layer Security) when sending and receiving messages and this is visible in the Wireshark capture.

TLS uses a handshake to establish a connection between the client and the server [80]. A simple TLS handshake looks like this:

- The client sends a **ClientHello**
- Server responds with **ServerHello**
- Server sends its **Certificate**
- Server sends **ServerHelloDone**
- Client responds with **ClientKeyExchange**
- Client sends a **ChangeCipherSpec**
- Client sends **Finished**
- Server sends **ChangeCipherSpec**
- Server sends **Finished**

After the client sends his **ChangeCipherSpec** message, he starts to send the encrypted packages. The same goes for the server.

5.11.3 Receiving a message

See figures A.1 and A.2 on page 186 for two screenshots from Wireshark containing all the traffic going to and from XOXOmail when receiving a single message from a Gmail account.
See table 5.1 below for a summary of the event of recieving a message.

What	Measurements
Packets	69
Time between first and last package	5.746 seconds
Average packets per second	12.008
Average packet size	131 000 bytes
Bytes	9039
Average bytes per second	1573.069
Average Mbit per second	0.013

Table 5.1: Summary of the event of receiving a message

The figure A.3 on page 188 is a Wireshark screenshot that only includes the TLS packets sent to and from XOXOmail.

After investigating the Wireshark capture we could see that receiving a message acts according to the TLS protocol with one exception. The handshake between the client and the server is performed according to the rules of the protocol, except the **Finished** message is never sent from either part, instead there appears to be an **Encrypted Alert**. This does not seem to be a problem as the application ends the IMAP connection shortly after and thus closes the connection. The encrypted alert could be a normal "close notify" message.

When considering the amount of data sent and the time it takes to send it, we can see that the data exchanged between the client and server is 8992 Bytes and that it takes 5.384 seconds to finish the session. This is not bad considering that the information was transferred over GSM bandwidth. The actual message itself was 990 bytes but the application data sent from the server was 2953 bytes. This increased the data transferred by 66.47% During reception of this message there was only one duplicate packet.

5.11.4 Sending a message

For a screenshots from Wireshark containing all the traffic going to and from XOXOmail when sending a single message from XOXOmail see figure A.4 on page 189.
See table 5.2 below for a summary of the event of sending a message.

What	Measurements
Packets	648
Time between first and last package	4.056 seconds
Average packets per second	11.835
Average packet size	131 667 bytes
Bytes	6320
Average bytes per second	1558.314
Average Mbit per second	0.012

Table 5.2: Summary of the event of sending a message

For a Wireshark screenshot including the TLS packets sent to and from XOXOmail see figure A.5 on page 190.

When sending a message from the application it acts in a similar manner to the receiving part. It follows the TLS protocol with the same **Encrypted Alert** message at the end instead of the server and client **Finished** message.

Sending a single message takes 4.056 seconds and transfers 6320 bytes between the server and client. This is not bad for a GSM connection. The message sent from the client to the server had a size of 3150 bytes. The data sent from the client to the server was 1348 bytes. This is an increase of data by 42.7 %. During this transmission there was only one duplicate packet sent from the client to the server.

5.11.5 Wireshark conclusion

Sending and receiving messages with XOXOmail over GSM seems to work in a respectable manner. The time it takes to receive and send messages is pretty good considering this was done over 14.4 kbits/s. The TLS protocol also seems to be working as intended for both sending and receiving messages. The initial handshake is completed and the following data is encrypted as it should be.

The only problem we could find is that the amount of data sent when considering both the sending and receiving functionality appears to be much larger than the actual size of the message when you inspect it in clear text. The receiving part of the application had an increase of 66.47 % whilst the sending part had an increase of 42.7 %.

At first we thought this could be a result of the TLS encryption of the data, but if the encryption is working as intended the only part of the encryption that could create some overflow is the initial handshake. This was not included in the calculation, only the actual application data sent or received. The actual encrypted TLS packages are based on relatively efficient symmetric ciphers and should not take up a lot more space than unencrypted packages.

It is difficult to pinpoint the exact reason for this large increase in data transferred at this point, but it was sent in a reasonable time and according to protocol, and those are the most important attributes for our client.

5.12 Compression of data

This section will investigate the theory behind data compression, as well as trying to find compression techniques that could be used in the application. This study was requested by Thales.

Source coding, better known as data compression, is a way of reducing resource usage by compressing data. It performed on the data source before storing or transmitting the data. By encoding the data using fewer bits than the original representation the size of the data file decreases.

Figure 5.2: Compression in practice [81]

5.12.1 Benefits

Compression is mainly used to reduce data storage space and increase transmission capacity. For the XOXOmail application both these features are important reasons to use data compression, but the most important is the one involving transmission capacity. Since the users of the XOXOmail should be able to send pictures and videos as well as text, it is crucial to make the attachments as small as possible so that sending of a message with such attachments is possible, also in networks with poor bandwidth.

5.12.2 Disadvantages

Because that the data is compressed when receiving it, it must be decoded before use. This decoding is often resource demanding. Decoding of text and images are not very expensive and is done pretty fast. When it comes to compressing and decompressing of video it is a quite different story. To encode a video without losing data is a time consuming task which requires expensive hardware. If the process is to slow, it will not be possible to watch the movie while it is being decoded. It is important to note that the result of decoding a compressed video may or may not give a different result thant the original data. This is dependent of the compression method used.

5.12.3 Data compression approaches

There are two main ways to perform data compression. The two types are lossy and lossless data compression [82]. First we will describe lossy data compression as well as techniques for this method, then continue to do the same for lossless data compression.

Lossy data compression

Lossy compression means that data is lost during the compression procedure. When encoding the image, you do not get the original image out, because data is lost during the process. The purpose is to decrease the amount of data that must be sent or handled. Often the data, especially multimedia data, can be useful even though some of the data is lost, e.g. an image that will become coarser but still be very much like the full version. The compression algorithms often use well known properties of the human eye to remove details we cannot see. This makes it very hard to see the difference between a compressed and and uncompressed image. A well known usage of lossy compression is in streaming. [83]

Lossy compression techniques for images

Lossy copression of images is done by using various algorithms that removes information from the original file. A range of different alternatives exist here, and the results are varying. Some of the algorithms do compression with different ratios, and still uses the same time to compress. Examples of lossy compression standards for images are JPEG and PGF [84] [85].

Lossy compression techniques for video

Compression of video can be divided into two main groups: Inter and intra frame. The difference is in how we find out how we can compress. In inter frame encoding we look at a given set of frames in a special order and compress based on similarities between them, while we in intra frame encoding only look at one image at the time, which will be the same as image compression.[86] [86] [88] [89]

Lossy compression techniques for audio

Lossless compression of audio is done by removing frequencies not audible for the human ear. This reduces the data size a lot. We often compress audio more than this to make the audio files so small that they can fit without problems on portable devices like mobile phones and laptops. The most common file formats here are MP3, AAC and WMA.[90] [91] [92]

Lossless data compression

There will come a point where lossy compression will have made the original data useless. Although the human eyes and ears may not catch small degradations of images and sounds, the decreased quality eventually will be apparent.

To have lossless compression means that the reconstruction of the compressed data is identical to the original data. A well-known example is the ZIP file format. Lossless compression is used when it is important that the original and decompressed data is identical, e.g. text or source code. The common way of doing lossless compression is to first generate a statistical model for the input data and secondly use this model to map input data to bit sequences. There are two ways of constructing statistical models. The first is to use static models, where data is analyzed and a model constructed, then the model is stored with the compressed data. This method is simple and modular, but the model can be expensive to store and the method may not perform well on files with heterogeneous data, as all data will have the same mode. The second method is to use an adaptive model. Here the model is dynamically updated as the data is compressed. The model is trivial at first but improves rapidly.

The most used encoding algorithms for producing bit sequences are Huffman coding and arithmetic coding. There are many special-purpose compression algorithms for e.g. images, text and video. In addition, there are many general-purpose lossless compression algorithms that may be used on any type of data. The problem arises when the input data is on another form than the algorithms were originally designed to compress. A list of some common techniques and formats for lossless encodings will follow. [93]

Lossless compression tecniques for text

Lossless compression of text is tricky. It has to have good performance and guarantee that the result of converting it back to the original data gives the correct result. Context tree weighting and LZ77 are examples of compression algorithms often used [95]. It is often advantageous to transform data to a format that compression algorithms will run better on. An example of such an algorithm is the Burrows-Wheeler transform [96].

Lossless compression tecniques for images

When we are doing lossless compression of images, we must find data in the image that is redundant and find a better way of representing it. This is why lossless compression of images works best on images with large areas of identical color. Examples of image formats often used are GIF and PNG. [98] [99] [100]

Lossless compression tecniques for video

The same principle holds for lossless video compression as for image compression; a lot of area with similar coloring is necessary for good performance. If a inter frame compression algorithm is used, we will get much better compression ratio compared to intra frame, but it requires a lot of processing power. Examples of algorithms are CorePNG and Animation Codec [101] [102]

Lossless compression techniques for sound The most common formats used for lossless compression of sound is FLAC and MPEG-4 SLS. They both are able to compress the data with 50-60% . The biggest problem of compressing using these standards, is that they are often not supported on portable devices. [103]

5.12.4 Discussion

As we can see, there are two main types of compression. What we can see from the previous sections, is that there exist a lot of methods for doing both lossy and lossless compression. Some of these methods are not so computational heavy, while others require a lot of processing power. The heaviest compression algorithms, processing wise, are those that operate on video streams. It is not trivial to compress images and text, but it is often very fast.

Choice of algorithms
When choosing a compression algorithm, it is important to use a compression algorithm that gives good enough compression rate within the acceptable time frame. It is always good to compress data if the compression time is close to zero. This is however seldom the case.

Often we do not know how good XOXOmail's connection to the Internet is. This can be calculated by taking the time it takes for data to be sent to / received from a server. If we then have some statistics on the compression time for, e.g. 30 seconds of video, we can do some number crunching. If we know that with the current connection speed, the transfer of the compressed video will take 2 minutes to the mail server and 1 minute to get downloaded to the receiver (if assuming download rate is twice of the upload rate, and that the receiver has the same connection as the sender) and it takes 20 seconds to check the connection speed, the total time used is then 3 minutes and 50 seconds. For the compression to be beneficial, the time used on compression and bandwith checking should be less than the time difference between sending and receiving the uncompressed and the compressed video.

Compressing text
It is quite obvious that if we are to compress text sent via XOXO-mail, it has to be lossless. If we loose data on the way, the message received will look completely different than the one sent. This is not a result that we are interested in. One important thing to notice, is that the text size on the message sent via XOXOmail is often trivial. The sender often wants to convey a few sentences, together with the current coordinates and maybe an image. The text size compared to the image is negligble, so the effort of compressing the text is often not worth it.

Compressing images
Compression of images is in this setting more resource demanding than compression of text, and can with good algorithms be compressed to 20 percent of original size, without too much degradation of the original content. Implementation of image compression methods are often compact and not so hard to implement. It is just a matter of finding the compression method which has the wanted end result.

Compressing video
Compression of video is one of the most resource demanding tasks that a phone can do. There exists a lot of algorithms for this, but unfortunately not so many for Android. Fortunately, it is a possiblity in Android to choose the video quality before starting to record the video. This can be done based on connection speed, but this does require that bandwidth is tested. This will result in creating a video with lower quality, so no compression has to be done. One must then do a judgement on how good video is good enough for showing what you intend to show via video. If the low settings of Android video is not good enough, then an algorithm needs to be used, either by a third party vendor or be implemented from scratch for Android. If a compression is chosen for getting wanted quality, then a choice on whether to use lossless or lossy compression also has to be made.

Compressing sound
Compression of sound can be done with varying results. If you want to keep the original quality, lossless is the only alternative. There are a lot of alternatives for both lossy and lossless audio encoding. Some of them do not have licence fees, while others have. There exist some libraries that do what you want, but this of course depends on the format of the sound you are trying to send.

5.12.5 Compression of data conclusion

We have now gone through a lot of methods for both lossy and lossless compression and it is evident that there are many choices. The intention of this compression study was merely to give an overview of what exists. We have not listed all options, but just peeked into some of the most interesting methods. In order to give a qualified answer to what is the best algorithms, a more thorough study has to be done in order to find the requirements on compression time in relation with connection speed, what Android devices will be using the algorithms and what algorithms that have a working implementation for Android.

5.13 IP rights and licenses

We would like to use free-licenses that allows us to make a product where we do not have to give away our source code for free, but can sell our product to different customers. There are no problems in having to refer to the licenses we have used along the way, but we would like to decide the license for our product independent of the licenses we have used.

Thales has not said anything about what licenses we should use. They have however said that we must make it clear which parts of the code we have not done ourselves, and where we found the specific code. We should also specify what licence agreemets the code was under.

The following list shows all licences used and under what part of our solution it was used.

IDE

- Netbeans[48]
  - GPL-2.0 /w Classpath Exception
  - CDDL-1.0

Android

- Android[49]
  - Apache 2.0

Building

- android-maven-plugin[50]
  - Apache 2.0
- Maven[51]
  - Apache 2.0

Testing

- JUnit[52]
  - Common Public License - v 1.0
- GreenMail[53]
  - Apache 2.0

Libraries

- JavaMail-android[54]
  - CDDL-1.0
  - GPL-2.0
  - BDS
- XStream[55]
  - BSD
- BouncyCastle[56]
  - adaptation of the MIT X11

Latex

- TeXnicCenter/MiKTeX[58]
  - GPL

Version control

- Git[59]
  - GPL-2.0

Terminal

- Cygwin[60]
  - GPL

Licenses used

- Apache 2.0 [61]
- GPL-1.0 [62]
- GPL-2.0 [63]
- GPL-2.0 w/ Classpath Exception [64]
- MIT X11 [65]
- CDDL-1.0 [66]
- CPL-1.0 [67]
- BSD [68]

See table 5.4 below for a comparison of the different licenses used.

License	Distribution	Distribute under another license	Sell the new product	Refer to this license	Available source code
Apache 2.0	YES	YES	YES	YES	NO
GPL-1.0	YES	YES	YES	YES	YES
GPL-2.0	YES	YES	YES	YES	YES
GPL-2.0 w/CE	YES	YES	YES	YES	NO
MIT X11	YES	YES	YES	YES	NO
CDDL-1.0	YES	NO	YES	YES	YES
CPL-1.0	YES	YES	YES	YES	NO
BSD	YES	YES	YES	YES	NO

Table 5.4: License comparison

5.14 Evaluation and conclusion

Through the project study we had looked at the desired functionality of the application we were requested to make. In many ways, it was to function as a regular mail client, but with additional specific functionality related to miltary attributes and, of course, security. We conducted a study to see if any similar solutions existed. This kind of application is targeted at a very specific group, and we realized early on that we could not find any existing applications that could solve our problem. We found out that there exist a lot of other mail clients as well as applications and libraries that support signing and verification, but not the full package. Some of the ideas regarding design and solution could be used in XOXOmail, but not directly.

This project had a lot of uncertainties, and choosing our development model was important. The choice fell on agile development in the form of scrum. This method fits this kind of project where uncertainty and regular revisions are central parts of development. Using a waterfall model would be too rigid for a project like this.

One of the most important decisions we did at the beginning of the project was to decide on what tools we wanted to use during the development process and what programming language(s) we should use. The first and obvious choice was to use Java, as this is the de facto standard for Android programming. Even though we did an investigation to see if other programming languages could do the work better for us in uncertain settings, this turned out to not be the case. All other languages would give us more problems than advantages, so the natural choice was Java. We also had to choose a development enviroment, where the choice was between Eclipse and Netbeans, and the latter won. This was in some ways an unfortunate choice regarding Android development, as we were sometimes missing some tools in Netbeans, but in the long run it was not a big problem.

We also made some important decisions about the application. The first choice was to have the service running in the same process as the application instead of in a separate process.

One of the most important aspects of the application was to find out what security features that were facilitated by Android and what needed to be implemented by us. In order to do this we had to make some assumptions about the phone, the users of the application and the network. After this was done we could discuss what is secure and what is not secure under these assumptions. We ended up with different solutions on how a secure communication channel could be made and that local storage of the current user is sufficiently protected under the assumption that no one will get unwanted access to the phone. We also assumed that the user would be able to log in to the application everytime it started without any problems.

Since the application should do encryption before sending messages, we wanted to check that the application performed well under conditiions where the network connection is limited. We saw that this worked well without any problems.

When it comes to compression, we saw that this was a very complicated field of study. What we ended up with was a listing of a lot of current techniques that is possible. Deciding on a compression technique for the different data types is beyind the scope of this project.

CHAPTER 6

TEST PLAN

This chapter will describe how we plan to test the application, both continuously and at the end of the project. Section 6.1 describes what methods we plan to use to test our code. Section 6.2 describes the different test tools we will use in the project and why these were used. Section 6.3 details the design verification to be carried out. Section 6.4 details the plan for testing functionality and usability of the project.

6.1 Methods and goals for testing

The main part of project testing will be done as we go along. Using unit tests, each part of the code will be thoroughly tested before it is put together. Larger scale integration and instrumentation tests are done largely the same way, compounded by informal testing by group members. The sending and receiving of messages will be tested using Greenmail, while Mockito is used to handle not yet implemented classes as well as emulator free testing for Android specific classes. All of these tools are described in the section below. Towards the end of the project period, including large parts of the fourth sprint, the focus will shift from producing additional code to finding and repairing bugs. While our plan is vaguely based upon, and in large part inspired by, IEEE829 [69] we are too informal about it to be able to use such grand claims. In short, our testing plan is to test each public method both for valid and invalid input, and then run larger scale integration tests between modules. By using dependency injection and Mockito, we will also be able to test top level modules without testing the building blocks of that module.

The overall goal for the test plan is that every part of our code related to the backend service is tested thoroughly, both for valid and invalid input. Though the same would be good for the frontend, it is harder to do this automatically, and therefore it will only be tested whether or the functionality works or not. The optimal goal would be 100% code coverage, but this is rarely feasible, hence we will settle for anything between 60-80%.

In order to ensure the greatest possibility of correctness for our code, each member of the team will be responsible for writing test for their code before it is submitted to our common repository. Changes in the user interface should also be tested, though if there is no automatic way of doing it, one should tests all functionality related to the changes one have made in the code.

6.2 Test tools

Unit testing is a method where individual units of source code are tested to determine if they are fit for use. These units consist of sets of one or more computer program modules together with their associated control data, usage procedures, and operating procedures [29]. This section will describe the different unit testing frameworks we will use.

6.2.1 JUnit

JUnit is the industry standrad for test driven development. It works by creating a separate test method for each method in the class that is to be tested. In this way one can test larger portions of the code independent of the other code sections. [30]

6.2.2 Cobertura

Cobertura is a tool for calculating the code covered by our tests, and generate a report based on this. While all other testing tools are automatically run at each build cycle, this is not the case with Cobertura, which needs to by invoked by a separate command.[117]

6.2.3 Mockito

Mockito is an open source testing framework for Java that allows creation of Test Double objects called "Mock Objects". Mock Objects are simulated objects that mimic the behavior of real objects in a controlled way [31].

Mockito distinguishes itself from other mocking frameworks by allowing developers to verify the behavior of the system under test (SUT), without establishing any expectations beforehand [32]. One of the criticisms of mock objects are that there is a tight coupling of the test code to the system under test [33].

6.2.4 Greenmail

Green is an open source mail server package for testing in Java. It provides all the server side software needed to test sending and receiving of mail without having an actual server. It does, however not support the IMAP IDLE command.[34]

6.2.5 Why we selected these test tools

The choice of tools was mainly based on previous experience inside the group, the framework provided by and the demands of the platform.

Junit was chosen because of our previous experience and the fact that the Android instrumentation framework is based on Junit. Bundled with Mockito it also gives us the possibility to test otherwise Android dependent modules in a Java environment with the Android specifics mocked up. This can greatly reduce the build cycle time, since the tests no longer need to be pushed onto a device in order to run. Mockito also provides another benefit, in that you can test a component without testing its dependencies. This allows for highly specific testing of just one class, component or module.

Greenmail was chosen based on the need to test the message module in our project, ease of use and fast response time. If the code passes the Greenmail tests, the code is then tested against gmail's mail server to ensure that functions with external servers as well.

6.3 Design Verification

In order to accommodate the agile development plan used by the team, our early design verification is necessarily somewhat vague. In the early stages of the project we had limited knowledge of how the final product would be structured and which features it would involve.

Early in the first sprint we decided upon a set of basic interfaces, which neatly split up the project into manageable parts. We began early by writing up tests for future classes that implemented these interfaces, allowing low level testing of individual methods. At the time of writing these tests we had no idea how each method was going to be implemented. This enabled us to write generic black box tests.

Later in the project this method level testing will be augmented by white box and stress testing; essentially us deliberately trying to crash the system with strange inputs or action combinations.

6.4 Black-box testing

Black-box testing [112] [113] is a software testing method to test the functionality of the program. The tests focus on how the software is supposed to behave, but not how the software does it behind the scenes. The internal of the software is not known to the tester, hence the software can be called a black box. The tester provides input (touch, keystroke etc.) and compare the output to the expected result. Black-box testing is useful in all levels of software testing, from unit testing to acceptance testing, but it becomes increasingly more useful with higher levels.

We have chosen to use two forms of functional testing, namely functional testing and usability testing.

6.4.1 Functional testing

Functional testing [114] is a type of black-box testing that is based on test cases derived from the specification of the software. One tests by providing input, as touch or keyboard gestures, and compare the actual to the expected result.

Template for functional testing
The template for functional testing can be found in table 6.1 on page 86.

Item	Description
Test case ID	id
Name	Shortname for the test
Requirement	Requirement reference
Description	Description of test
Preconditions	The conditions of the application before the tests are started
Flow of events	The list of steps necessary to complete the test
Expected results	The end result if everything works as expected
Actual results	The end result
Comments	What went wrong, what could be the cause
Status	OK (no errors), OK- (minor issues) and Failed (not working)

Table 6.1: Test case template

The tests
The test cases 1-25 are meant to verify that the application works correctly to normal use, whereas the test cases 26-28 are meant to that the application behaves well with incorrect input. The test cases can be found in appendix B. The specific test should be carried out whenever a feature is finished, and all the tests should be carried out at the end of the project.

6.4.2 Usability testing

Usability testing is testing of the application on possible end-users. This is also a kind of black-box testing meant to observe how users use the application to find unknown errors and get an evaluation of the design and usability of the application. We created a test with six cases that mirrored the most important actions in the application. The test person tried to solve the tasks while the test leaders measured the time spent on each task and noted problems/comments/thoughts. After the test the persons filled out a SUS form.

SUS
SUS is an abbreviation for System Usability Scale [115] and is a simple Likert scale of ten questions where the test persons shall give a score from 1 to 5 where 1 is strongly disagree and 5 is strongly agree. The purpose is to obtain a global view of subjective evaluation of the usability of the software.
The SUS score is calculated in the following way:

1. Take the odd numbered questions and subtract 1 from these scores.
2. Take the even numbered questions and subtract the score from 5.
3. Add these scores together.
4. Take this sum and multiply by 2.5.
5. Then you have a SUS score between 0 and 100.

Usability goals
Before conducting the usability tests, we have set some usability goals, to have something to assess the usability against.

1. The users should not spend more than 5 minutes on a task
2. The application should not crash during the usability tests
3. The users should not make more than 1 error during the tasks
4. The users should solve task 6 faster than task 1 (memorability)
5. The average SUS score should be above 70 [1]

Templates
SUS form can be found in appendix C.

Observation form can be found in appendix C.

[1] Studies have shown that 68 is average [118]

The tests
We will have five people perform the tests. The reason behind this number is the theory of Jakob Nielsen [116], who states that a test with five persons should discover about 85 % of the usability problems. This approach suggested that we should test and redesign three times, but due to time constraints we probably will not conduct more than one iteration.

The tests can be found in table 6.2 - 6.7 on page 88 - 89.

Task no.	1
Task name	Log in
Description	Log in to the application with username and password
Input	Username="kprotesting" Password="kprotest"

Table 6.2: Usability test - task 1

Task no.	2
Task name	Send message
Description	Send a regular message
Input	To="kprothales@gmail.com" Subject="Usability test" Security label="UNCLASSIFIED" Priority="Routine" Type="Operation" Message text=Enter your own text here

Table 6.3: Usability test - task 2

Task no.	3
Task name	Settings
Description	Change the settings for predefined attributes for instant message
Input	Standard receiver="kprotesting@gmail.com" (yourself) Security label="RESTRICTED", Priority="Flash" Type="Drill"

Table 6.4: Usability test - task 3

Task no.	4
Task name	Send instant message with attachments
Description	Send instant message with an image
Input	Image=Take one with the camera Message text=Enter you own text here

Table 6.5: Usability test - task 4

Task no.	5
Task name	Read high-priority message with attachments
Description	Open a received high-priority message and view the attachments
Input	

Table 6.6: Usability test - task 5

Task no.	6
Task name	Send message to a contact from the address book
Description	Send message to a contact by choosing from from the address book
Input	To="kprothales@gmail.com" Security label="UNCLASSIFIED"

Table 6.7: Usability test - task 6

CHAPTER 7
ARCHITECTURAL DESCRIPTION

This chapter will revolve around the final architecture of the prototype created during this project. In our definition of architecture, we have choosen to follow the definition carved out by Len Bass, Paul Clements and Rick Kazman.

"The software architecture of a program or computing system is the stucture of structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships between them"[71, p. 3]

Section 7.1 will describe the architectural drivers of the project. Section 7.2 will outline the architectural viewpoints used and the different views. Section 7.3 will describe how the frontend part of the application was developed, how the Android platform behaves and how we have tried to utilize these Android features and how the activity flow of XOXOmail is set up. Section 7.4 will outline the differetn architectural tactics and patterns implemented in XOXOmail. Section 7.6 gives an overview of the sequence of operations in XOXOmail. Section 7.7 describes how the requirements stated in chapter 4 are fullfilled (or stated if they are not fulfilled), both the functional and non-functional requirements.

7.1 Architectural Drivers

From the nature of the assignment and the requirements received from Thales, we decided that SECURITY had to be one of our quality attributes. After the first couple of meetings with the Thales, it became clear that usability also was something of interest to them, and we decided to add USABILITY to our quality attributes as well. It should however be noted that all of the quality attributes specified in Len Bass's book 'Software Architecture in Practice'[71] had their place within this project. Especially MODIFIABILITY, though not one of our main attributes, was given a fairly large amount of consideration in order to ease development at later stages of the project. The reason as to why Modifiability was not one of our main attributes comes down to the expected lifespan of the prototype, which in this case is largely dependent on the success of the project, and what prospects Thales sees when presented with the final prototype and documentation. However, the project revolves around making a prototype and documenting the steps taken to complete that process, and it is therefore safe to assume that the lifespan of the prototype will be relatively short.

7.2 Architectural viewpoints

The "4+1 view model" [70] by Philippe Kruchten suggest four different views, logical, process, development and physical. In the case of XOXOmail we are going to use the logical, process and development view. The rationale behind removing the physical view is that XOXOmail will run on a single physical device and only utilize one process. A security view, describing the layers of security, will be added instead.

7.2.1 Development view (Subsystem decomposition)

The development view focuses on software modules and subsystems. These subsystems/modules are organized in a hierarchy of layers where each layer provides a well-defined interface to layers above.

"The complete development architecture can only be described when all the elements of the software have been identified. It is, however, possible to list the rules that govern the development architecture: partitioning, grouping, visibility" [70].

Figure 7.1 on page 91 shows the Backend Service Connects to all parts of our application. The main focus of this view is to show that it is also the connector to the external parts of the application, which in our case is just a mail gateway. The final goal will be to connect to Thales' gateway, but as long as the gateway comply with the standards of SMTP and IMAP there should be no problem.

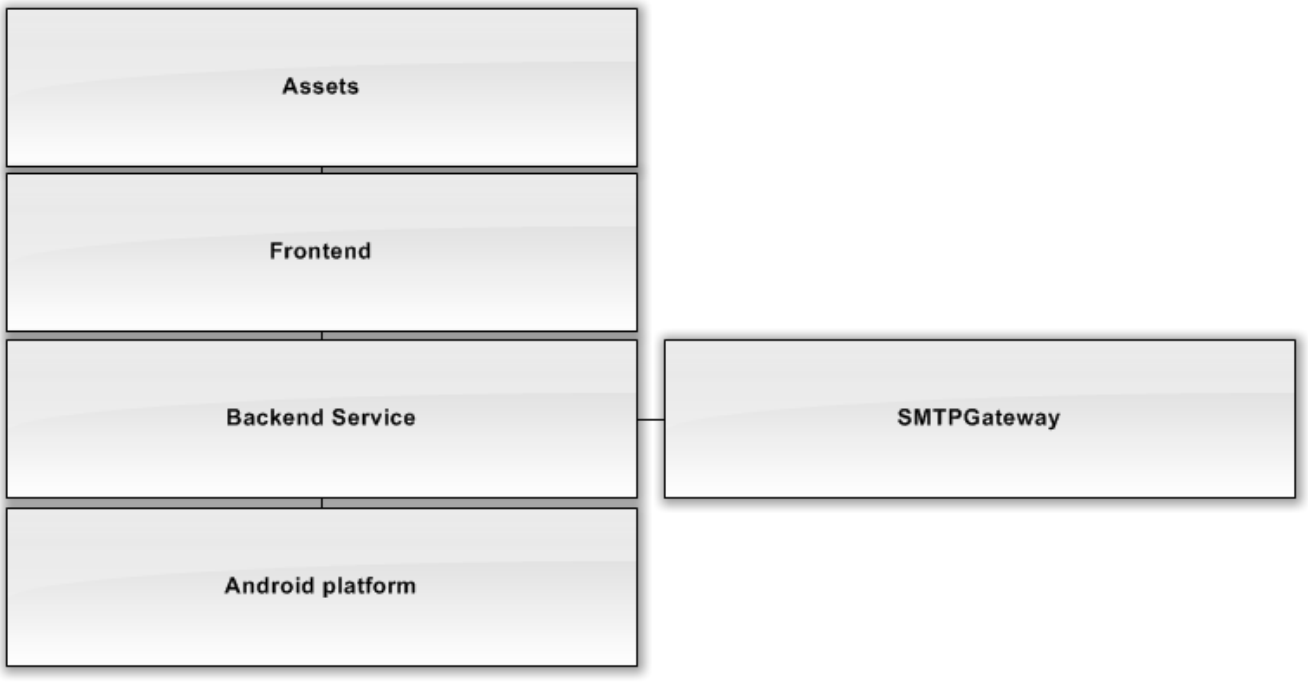


Figure 7.1: Development view

7.2.2 Logical view (Object-oriented decomposition)

"The logical architecture primarily supports the functional requirements — what the system should provide in terms of services to its users. The system is decomposed into a set of key abstractions, taken from the problem domain, in the form of objects or object classes"[70].

A common way to represent this view is with a class diagram that shows a set of classes and their relationships. We have however used a slightly modified variant, in order to make all the figures more readable.

Backend

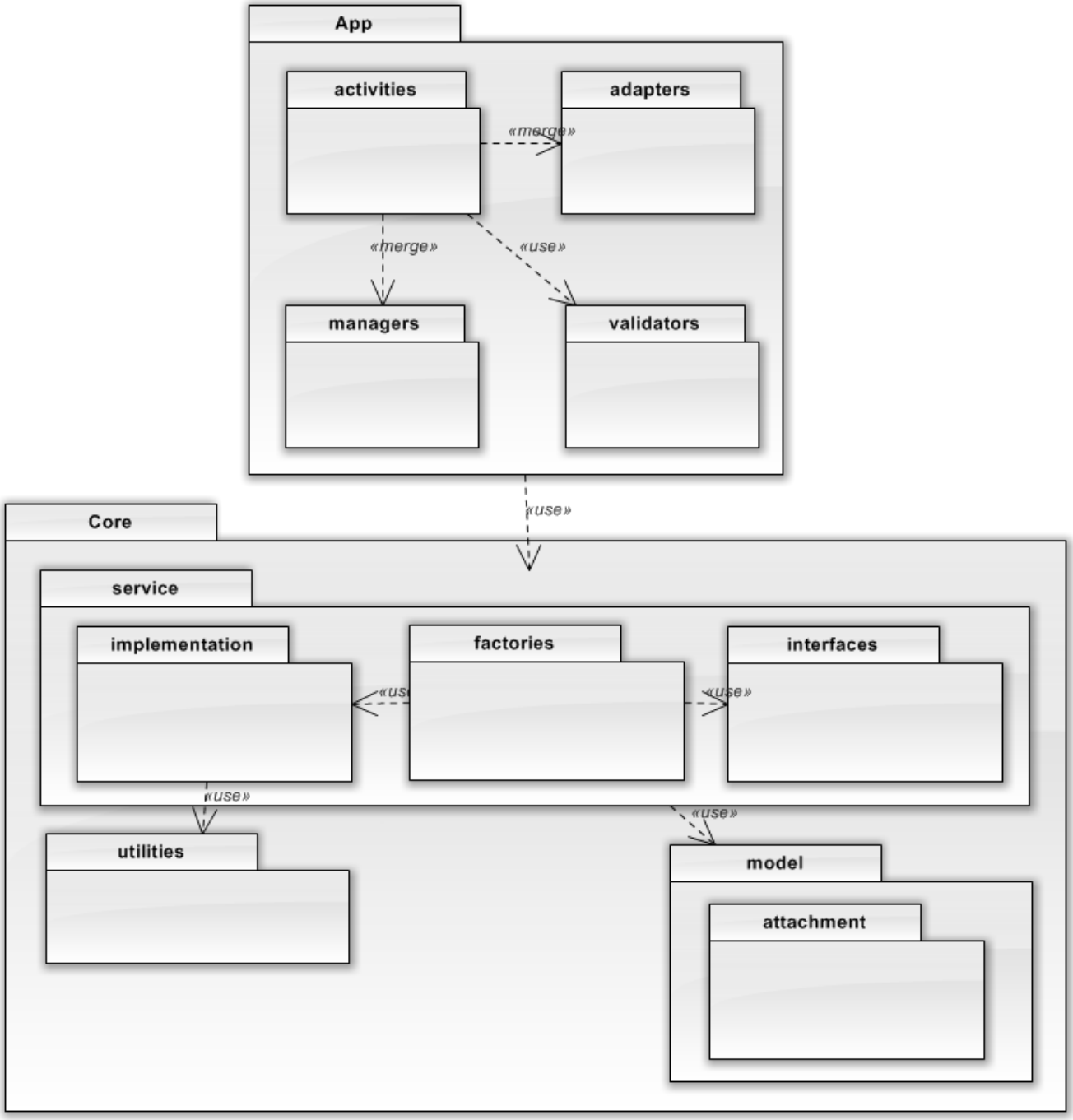


Figure 7.2: The logical package view of the architecture

Figure 7.2, shown on page 93, shows a package overview over the project backend. Though rather rudimentary, it shows the basic structure of the architecture and the most common communication paths between packages. On thing to note, is that in order to add a new service to the architecture, one would need to implement three different objects, a factory for creating the service, an interface describing the service, and the service implementation itself.

In the case of XOXOmail, we planned out four different services, the NetworkService (Fig. 7.3 p. 94), PersistenceService (Fig. 7.4 p. 95), HALService and SecurityService. The two latter was though never used, and their diagrams are therefore omitted here, alongside the diagram for our Utilities package. The structure of our models can be seen in figure 7.5 on page 96

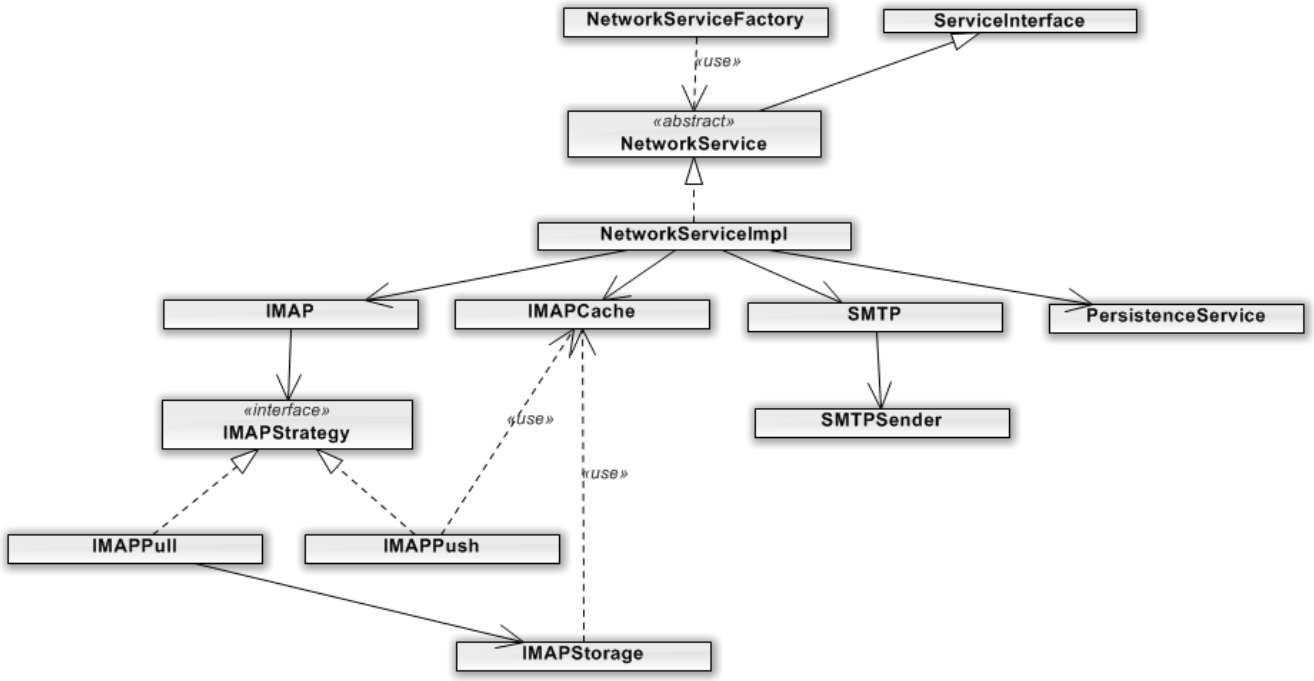


Figure 7.3: The logical view of the NetworkServicePackage

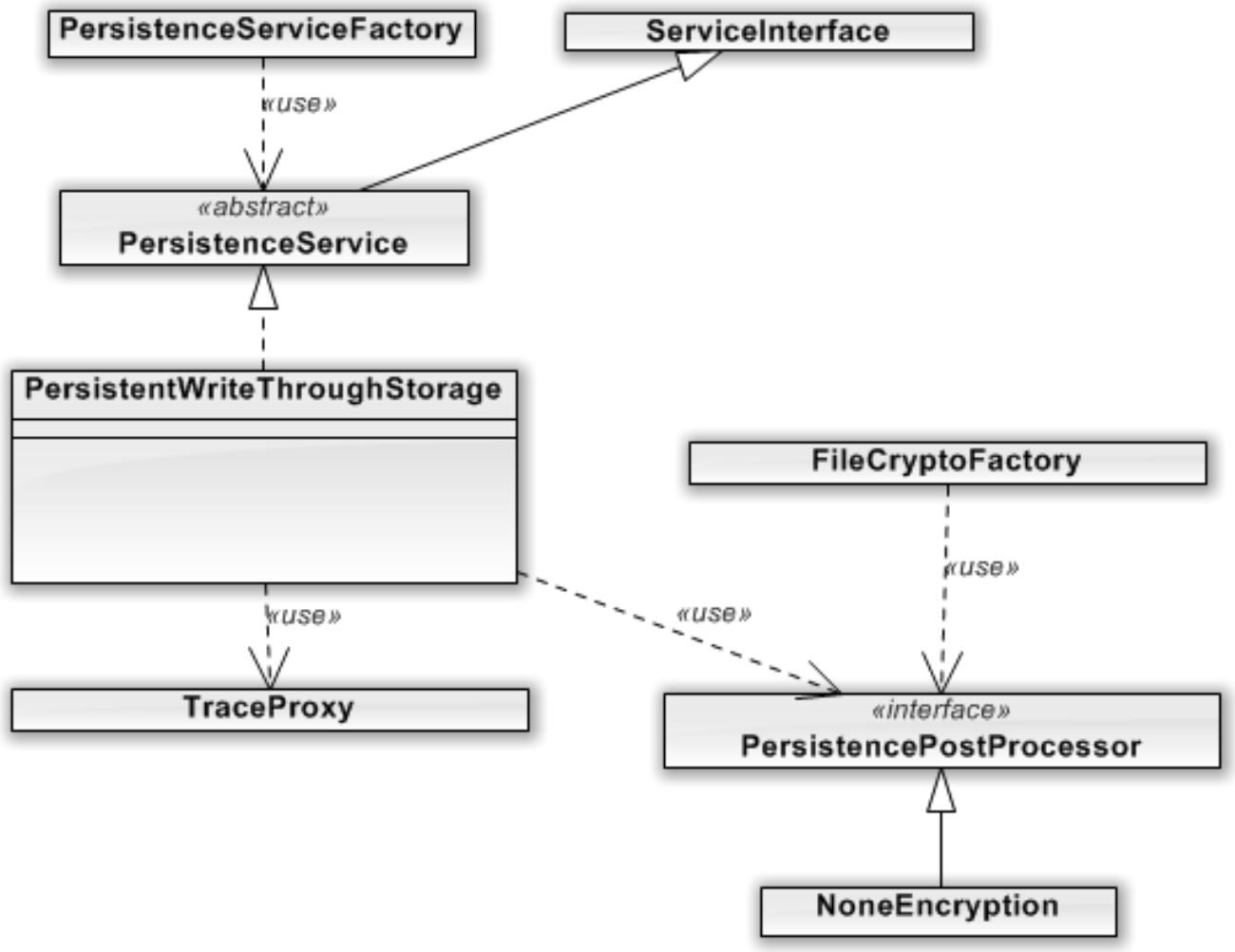


Figure 7.4: The logical view of the PersistenceService

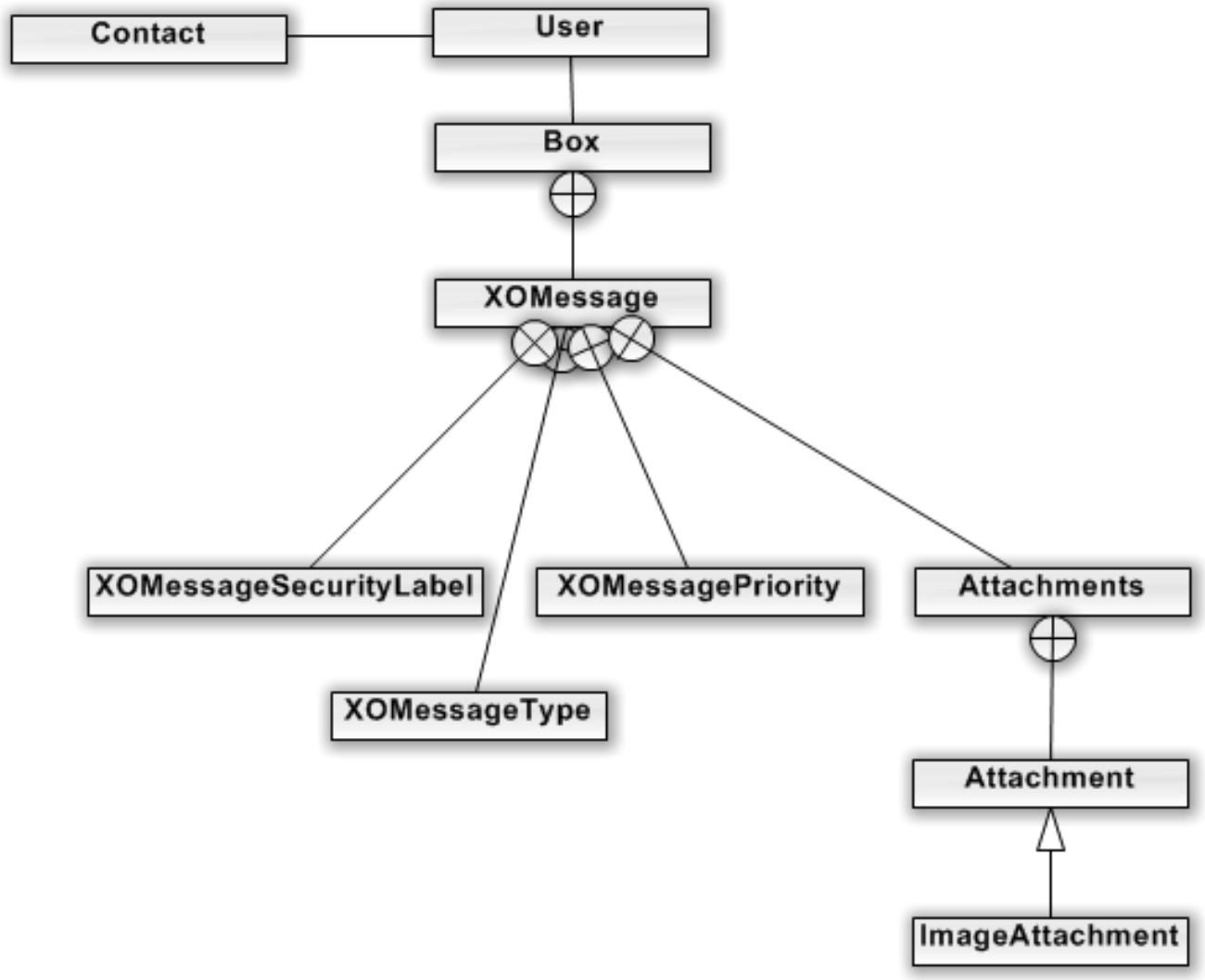


Figure 7.5: The logical view of the model package

The intended purpose of each of the different services is somewhat given by their respective names, but in order to avoid any ambiguity there will be a short recap of their purpose.

NetworkService The NETWORKSERVICE's reponsebility is to provide the core functionality of sending and receiving messages.

PersistenceService The PERSISTENCESERVICE's responsebility is to keep track of objects being changed, and reflect these changes in a persistent way in case of system failure, or if the application is shut down.

Frontend

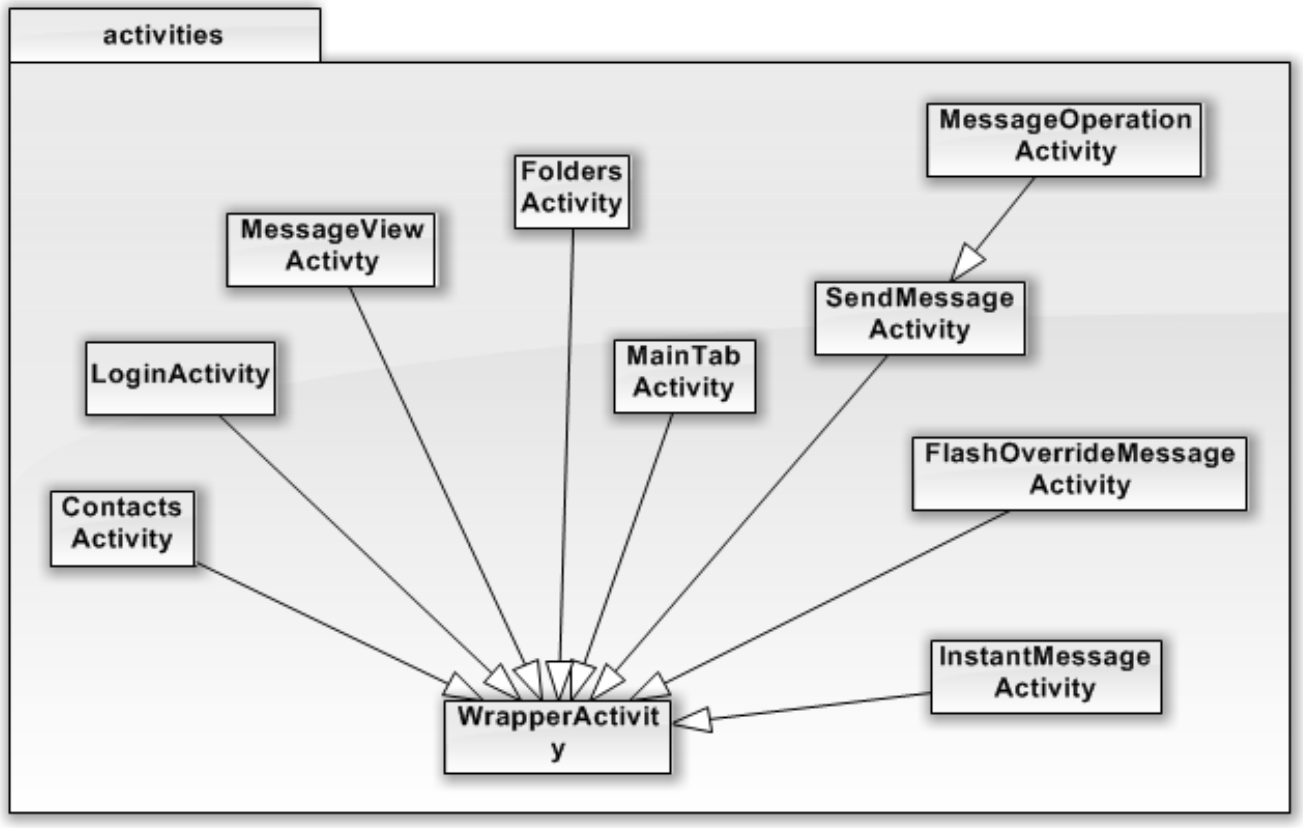


Figure 7.6: The logical view of the frontend activities package

7.2.3 Process view

The process view is concerned with how different tasks bind together to form one executable unit. More specifically, the communication between threads/processes and on which thread/process a task is executed. We will differentiate between major and minor tasks, where major tasks are architectural elements open through a public interface, and minor tasks being tasks introduced in order to implement some functionality in one class or module. In figure 7.7 on page 98, you can differentiate between minor and major tasks by looking at the SMTP branch of NetworkService, here SMTPSender is a minor task, while both SMTP and NetworkService is considered major tasks.

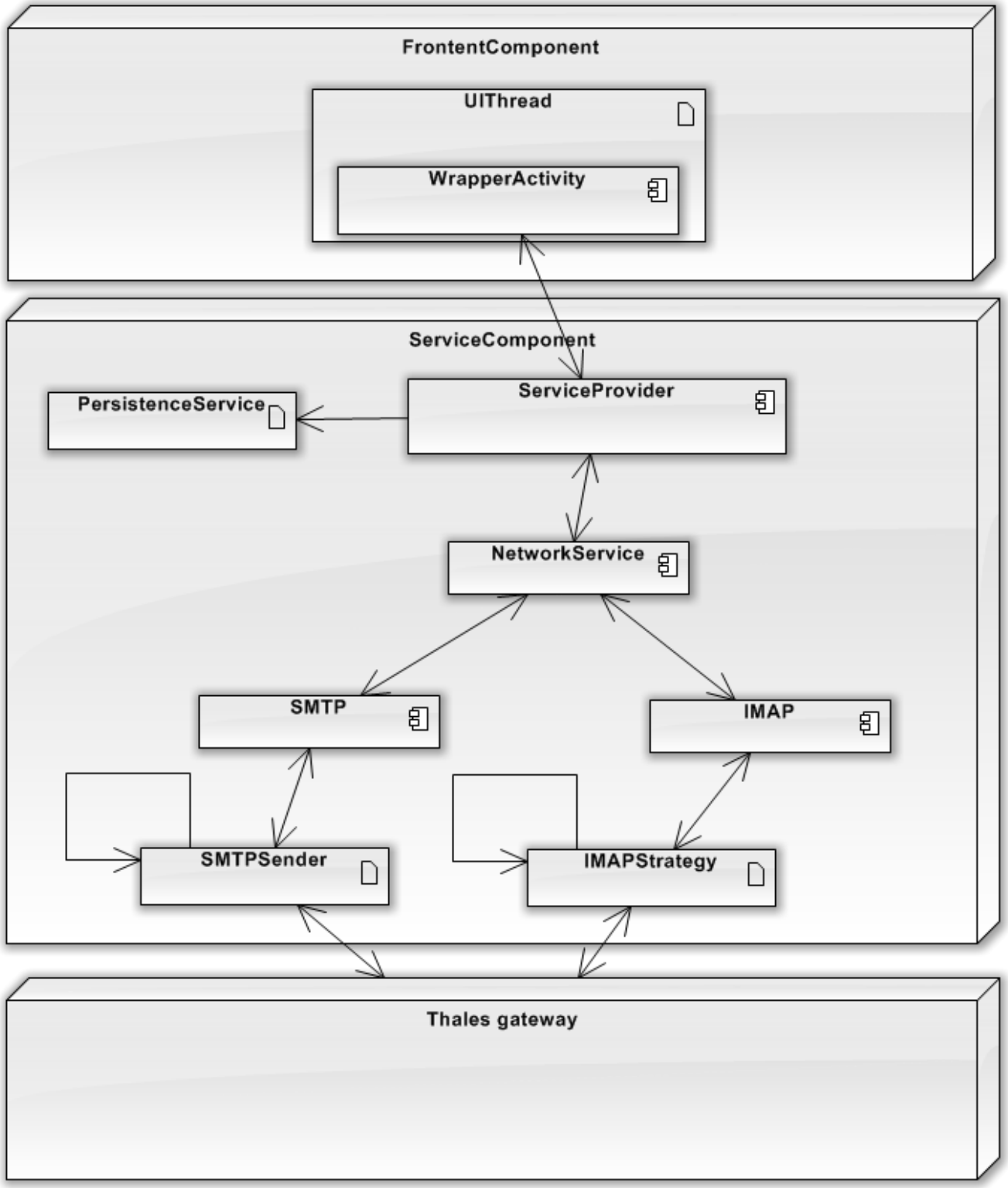


Figure 7.7: Process view

One thing to note about the communications paths is that almost all communication between the FRONTENT COMPONENT and SERVICE COMPONENT is between two wrapper classes, WRAPPERACTIVITY at the frontend, and SERVICEPROVIDER at the backend. The services under the SERVICEPROVIDER does however implement the OBSERVABLE pattern, which opens up some alternative communications paths for callbacks.

7.2.4 Security view

The security view is concerned with how the system as a whole is secured. This involves intercommunication, storage and communication with external sources. The view is not however concerned about implementation, just the layers of security. See figure 7.8 on page 99.

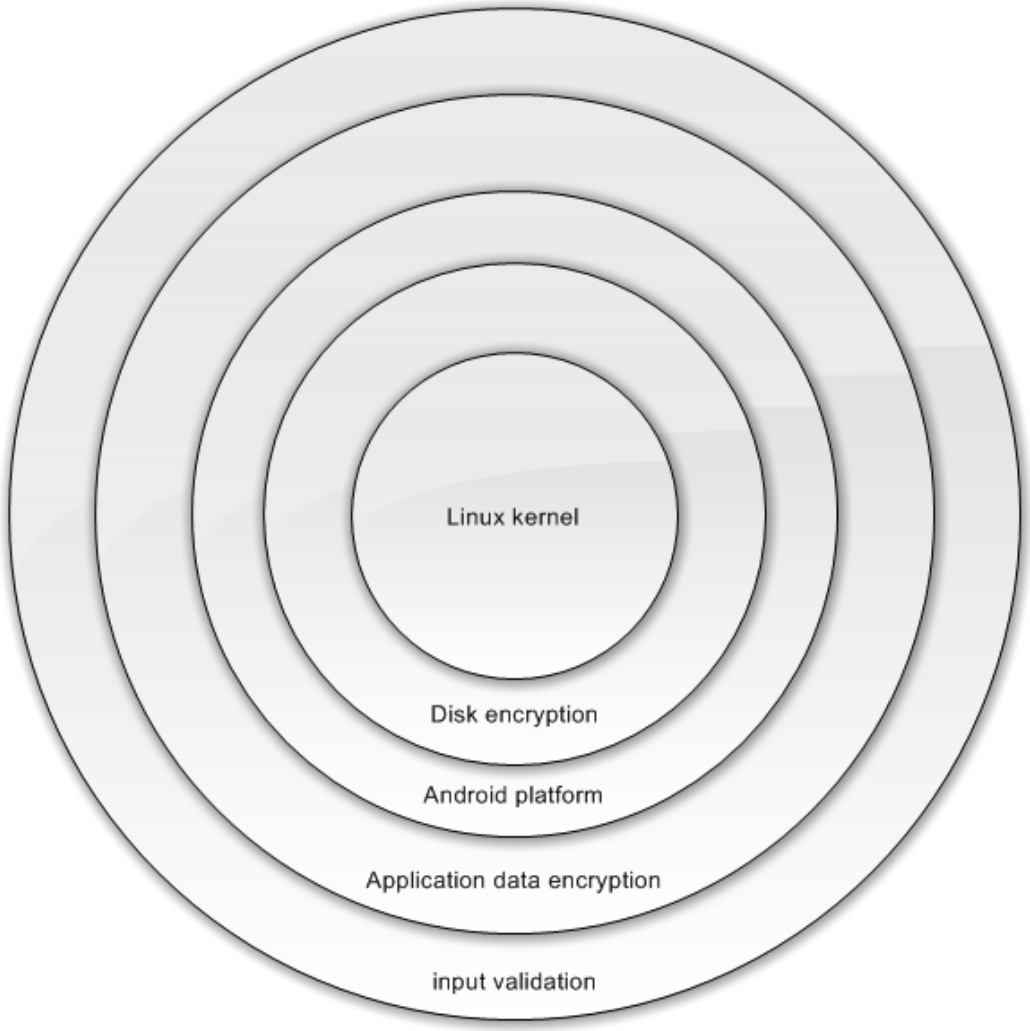


Figure 7.8: Security view

The Linux kernel provides us with a user-based permissions model and process isolation, which ensures that another process cannot access the memory of XOXOMail during runtime.

Disk encryption, is feature provided by the Android platform, which encrypts the whole Android device using AES with CBC and ESSIV:SHA256.[72]

Android platform, most of the security features provided by the Android platform utilize the Linux kernel. The application sandbox feature is one of these, and since it is located at the kernel level, it is hard to circumvent.

Application data encryption is a response to the possibility of rooted devices. Normally an Android application will not run with root access on the device, this however is not the case on rooted devices. In which case the application and user will have full access to all applications and all application data. By adding our own encryption layer with the key stored off-device we can ensure data security even with root access to the phone[73]. The encryption we opted for is an AES based encryption with a user-password derived key by PBKDF2 HMAC-SHA1. Regarding communication with external resources as a mail server, it will be done over a secure communication channel providing SSL or TLS.

Input validation is always necessary in order to provide a secure service. All input to the XOXOMail application should be validated, this includes received mail, user-input etc. For message validation we planned on using S/MIME signing and verification provided by an Android ported version of BouncyCastle called SpongyCastle, this was never implemented due to time restrictions.

7.3 Frontend

The section will describe the process of creating the user interface, the Android guidelines for creating an application, out take on these guidelines, and how the user interface turned out including the rationale behind the decision made.

7.3.1 The design process

This section will outline the design process from requirements and prototyping to implementation and testing.

Planning from the requirements

At the start, we looked at the requirements to get an overview of what views we needed and the basic functionality each view should have.

Prototyping

Then, we started to create simple sketches of each view, what components each view should consist of and the layout of each view. We also tried to outline the flow between the views. In this phase we had several suggestions as to the layout and main form of navigation. After some consideration we decided on the tab navigation pattern.

After the inital sketches we created an interactive prototype using the online prototyping tool [111]. This enabled us to share our thoughts with Thales and give them an early insight into what direction we were headed with the design.

Continuous feedback and implementation

We continued to keep an open dialog with Thales about what they wanted regarding functionality and feedback from the user interface. The implementation was split up in iterations according to the sprints where we implemented functionaliy that we had planned and Thales gave us feedback at the end of each sprint and demonstration.

Even though we had tried to dig into the Android framework prior to the implementation, as outlined in section 7.3.2, we did not have a full overview over what was possible in the framework. Implementing each view as we wanted became a process of trial and error and we learnt a lot during this process. We tried to stick to Android's specified way of doing things and using the built-in architecture and components, to avoid the effort of creating custom components from scratch, as we did not have that much time available.

Testing

We tested the application continously throughout the sprints, as well as conducting a final black-box functional testing as described in section 6.4.1. We also conducted a usability study to investigate whether we had succeeded in creating a usable user interface. These tests, in addition to the results, are described in section 6.4.2.

7.3.2 The Android framework

This section will cover the Android frameworks's guidelines regarding the organization of an application, and our take on these guidelines.

External resources

The user interface elements, ranging from whole views to the structure and look of a single list element, are defined in XML and stored external to the application. An Android project contains a folder named "res" where all external resources are placed. User interface elements are stored in a sub folder called "layouts". Other resources, e.g. colors or images, are stored inside separate sub folders. The advantage of declaring the user interface and other resources in XML is that it enables decoupling of the presentation of the application and the code that controls the application.

All resources must be assigned an ID if they are to be referenced to in the application code. All resource IDs are defined as public constants in the project's R.class, which is a class that is automatically generated during build and contains subclasses for the types of resources where we have defined at least one resource, e.g. R.layout (for user interface elements) or R.color (for definition of colors). The resources can be referenced both inside other resources and in the application code.

We also wanted to use styles, which are collections of properties specifying the look of a layout or a component. It can be compared to the mindset of CSS. The reason behind using styles and fixed color definitions would be to ensure consistent appearance across the application.

Activities

Activities are components that provide a view that the user can interact with, and are classes written in Java. Each activity has a window where the user interface is drawn. One activity is specified as the main activity and is the start screen of the application. Each activity can start another activity to perform different actions. The activities inflate the XML layouts to form the user interface and controls the behaviour of the elements. It is also possible to create layout elements programmatically, but we have chosen not to do this as we want the decoupling mentioned above.

An activity is always in one of four possible states [74]:

- Active/running: When the activity is in the foreground of the screen
- Paused: When the activity has lost focus but is still visible
- Stopped: When the activity is completely obscured by another activity
- If an activity is paused or stopped, it is likely that the system asks the activity to finish or just kill its process, hence drops it from memory. It then has to be restarted completely.

The life cycle of an activity can be seen in figure 7.13 on page 110.
All the methods starting with 'on' (onCreate etc.) can be overwritten to administrate what the application should do in the changes of state.

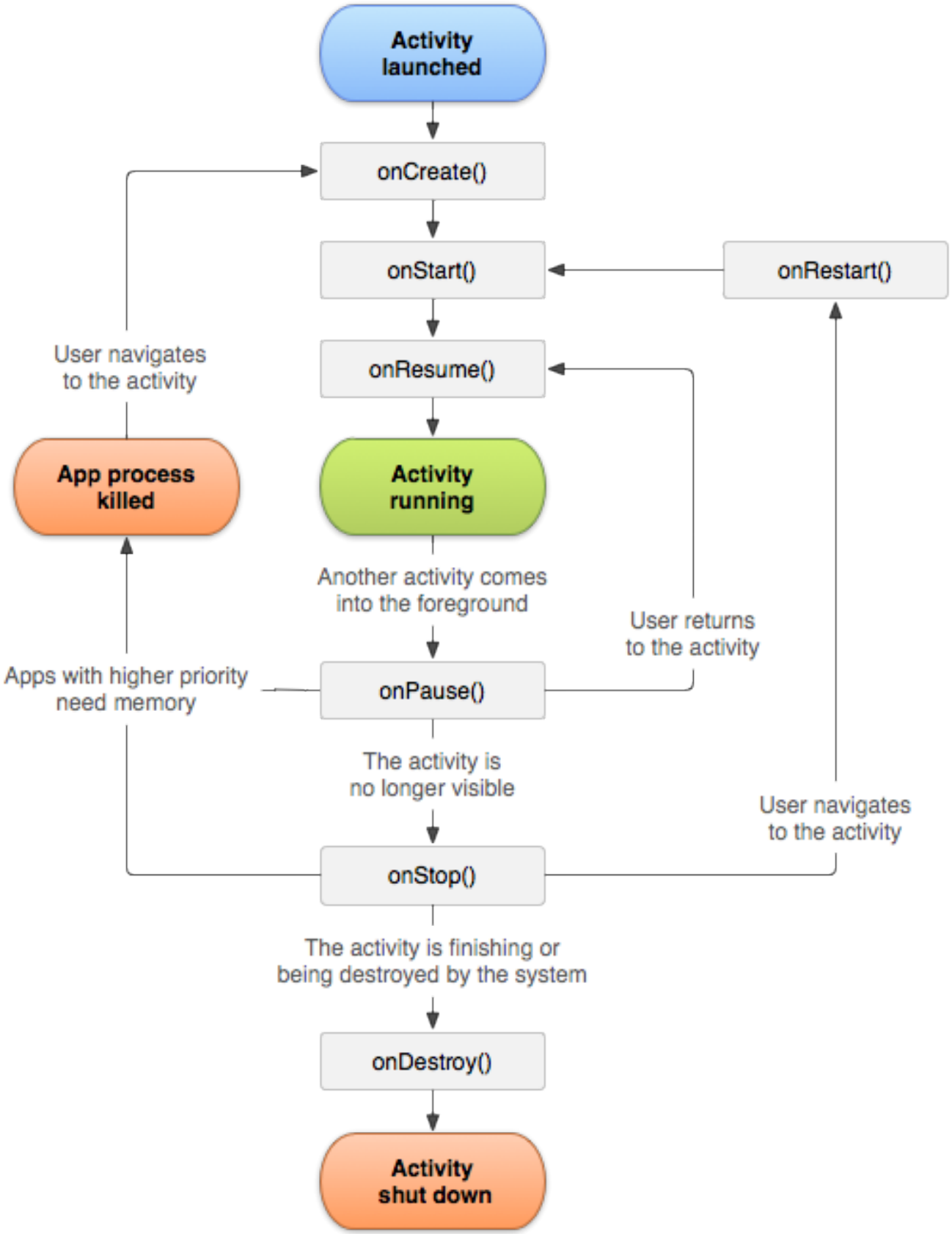


Figure 7.9: Activity life cycle [75]

The Android manifest

The Android manifest [76] is what binds the application together. It is defined in XML and details the structure and metadata of the application, its components and requirements. The manifest needs to have nodes for each of the components, which in our case are activities, services and broadcast receivers. Relevant metadata is the application name, icon and theme. The manifest also declares the permissions the application needs to access protected parts of the API and interact with other applications.

Supporting different hardware and internationalization

By conforming to using external resources on can easily build applications that support differences such as varying screen sizes and languages. For example, it is possible to create a low, medium and high dpi version of an image and Android will select the correct version based on the screen size. Android can pick layouts based on the screen orientation, where the possibilities are that the phone is an portrait or landscape mode. It is also possible to have Android decide what language to use based on location, but this is not relevant to this project.

Activities and flow between them

Figure 7.10 on page 105 shows the flow in XOXOmail. The entry point for XOXOmail is shown at the upper left corner at "Start application". The first activity you see is the Login Activity. The user is granted access if a correct username and matching password is entered. He is then sent directly to the main menu, where he can choose to either see messages, send a new message or alter settings. Which messages are shown are based on if you choose Sent or Inbox. By pressing Sort he can choose to alter the way the messages are shown. Choosing to send a message can either be done by going into the regular send message view, or the instant message view. Either way, for a message to be sent, it has to be validated. When all fields are valid, the message is sent to the correct recipient. By pressing return button when inside application we are minimizing the application, it will however still be running in the background.

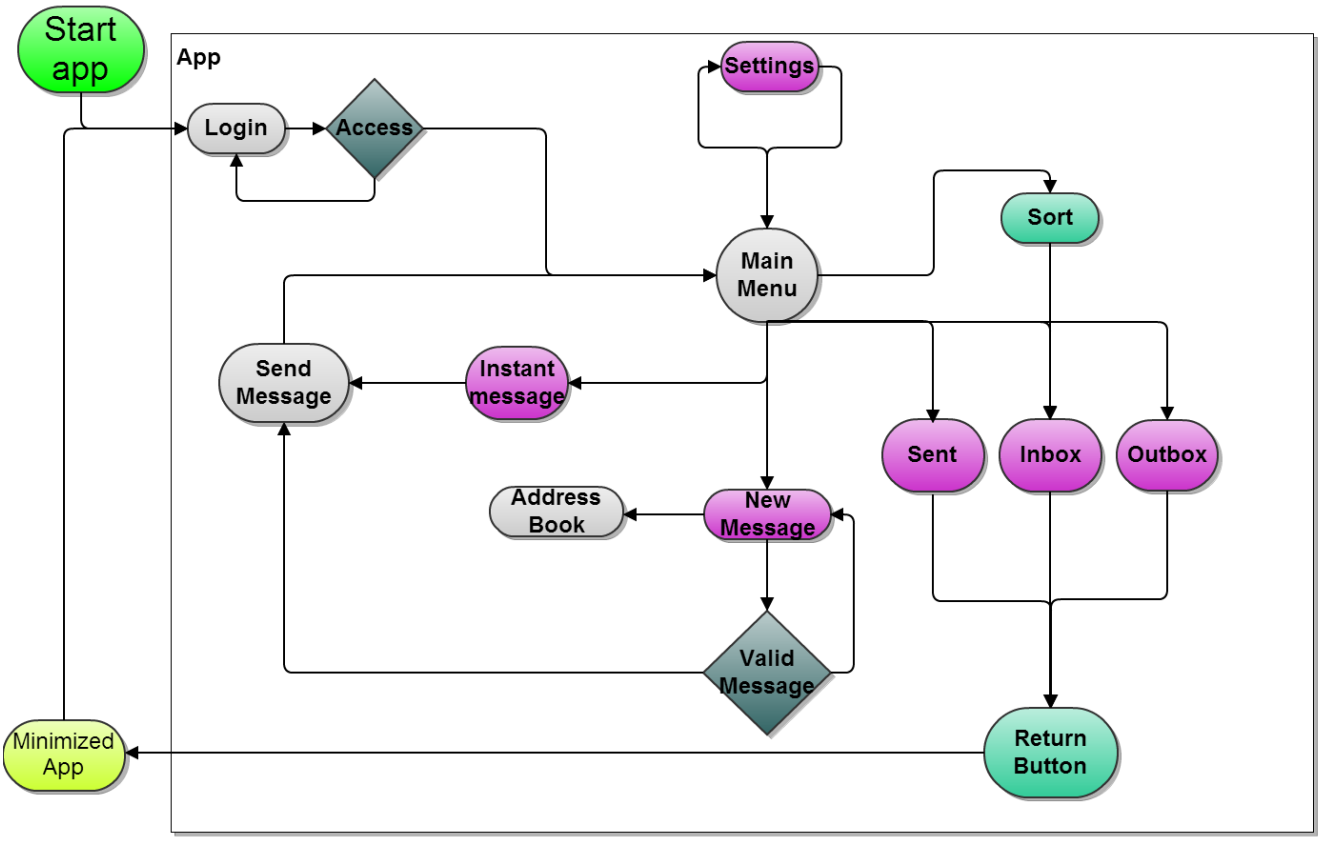


Figure 7.10: The activity flow for XOXOmail

7.4 Architectural tactics

This section will in general be a discussion about and around tactics found in 'Software Architecture in Practice'[71].

7.4.1 Security

Tactics for ensuring the security of an application and its data is often divided into three seperate ways of operating, resisting attacks, detecting attacks and recovering from an attack. In the case of XOXOmail, resisting attacks will be the main focus. As seen in figure 7.8 on page 99, we have opted for a layered security architecture in order to best secure the application data. In terms of what is listed in 'Software Architecture in Practice'[71, p. 119], XOXOmail will in one way or another touch upon all of the six tactics listed.

Authenticate Users XOXOmail will keep a users credential stored after the first time login. The first time login authentication procedure consists of trying to login to mail gateway.

Authorize Users After authenication, a folder is allocated for the user. All data within this folder will be encrypted using a derivative of the users password.

Maintain data confidentiality The local data storage will be secured by encryption, as described in the previous point. Network is encrypted using SSL/TLS.

Maintain Integrity Data integrity is a concern at three points: when updating of a model, storing data and network communication. Data integrity when updating a model will be assured by encapsulation, network communication will be secured by SSL/TLS running on top of TCP, the message content will in addition be signed using S/MIME. For the storing of data the PersistenceService should ensure the data integrity by conforming to the ACID properties.

Limit Exposure The exposure of services within XOXOmail is limited by the authenication of user, since no services are available before the user is successfully authenticated.

Limit access Access can be gained to data in two separate ways, and XOXOmail tries to prevent both. In the case of physical access to the device, an authentication process must be completed. Attempting to gain access through network communication will be prevented by a trust manager, only accepting connections to gateways with a known certificate.

7.4.2 Usability

"Usability is concerned with how easy it is for the user to accomplish a desired task and the kind of user support the system provides."[71, p. 90] The paragraph goes on to explain areas of interest like; learnability, efficiency, error prevention, system adaptations and user confidence and satisfaction. Though usability itself has few tactics that can be implemented directly, there are some general stategies as to how a designer can increase each of these areas quantitative value.

Consistency of data display Terminology, abbreviations, formats, colors and so on should be standardized.

Efficient information assimilation by the user The format should be familiar to the user.

Minimal memory load on the user Users should not be required to remember information from one screen for use on another screen.

Compatibility of data display with data entry The format of displayed information should be linked clearly to the format of the data entry.

Flexibility for user control of data display Users should be able to get the information from the display in the form most convenient for the task on which they are working.

In most cases making a good user interfaces requires several iterations with designing and testing. Hence a easily modifiable user interface is preferred, this can be achived by separating the user interface from the rest of the application. This separation can be achived by numerous tactics, most known is perhaps the Model-View-Controller pattern. Though not directly applicable to the Android platform, it carries some general principles which can be used in order to achive the general notion of a MVC architecture.

7.4.3 Modifiability

Even though modifiability was not one of the choosen quality attributes, it has had a significant impact on the overall architecture and is therefore included here. In 'Software Architecture in Practice'[71, p. 111] there are three main areas for modifiability tactics listed, which each has several tactics. The ones used by XOXOmail are listed below.

Localize change In an attempt to maximize the systems functional cohesion several tactics where utilized. The services in XOXOmail utilizes a tactic known as GENERALIZE MODULE as it uses dependency injection instead of multiple fairly similar functions. This allows for a more general use of the module by changing the input parameters to it, instead of the module itself. Another tactic used is ABSTRACT COMMON SERVICES, which can be seen on both the frontend and backend part of the application. Frontend, all activities inherit from a wrapper class which manages the connection to the backendservice. Backend, all services inherit from an abstract service which manages all commonalities between each service.

Prevention of ripple effect The prevention is done by the use of interfaces, supplemented by a factory pattern for easy creating of services.

Defer binding time The concept of configuration files is utilized by XOXOmail when the application is first starting up. The user also has the possibility to change the configuration files through the user interface, and thus making the application fit the users needs.

7.4.4 Testability

One of the problems with testing when creating an Android application is the Android dependent code and long build-test-cycle time, which in many cases discourages testing. As a biproduct of the tactics mentioned in the previous sections, both these problems were handled. By keeping the modules as close to pure Java as possible it is possible to run tests locally on your computer, without the need of compiling the complete application and deploying it to the phone, which in turn greatly reduces the build-test-cycle time. This was of course made possible through very generalized modules, were most dependency were injected into the module.

7.5 Architectural patterns

XOXOmail is built up of several architectural pattern working together to create the best possible experience for every stakeholder involved in this project. The most prominent being our own derivation of the Model-View-Controller pattern, where the whole frontend has taken the part of being a view-controller, and the backend service a pure controller. A great amount of modifiability is achived with this pattern, and by using interfaces to describe modules and their respective callbacks.

7.6 The sequence of operations

The sequence diagrams below show the general sequence of executed methods when the application is running normally. It is however somewhat simplified in order to best convey the general notion of the sequence, this is due to the fact that the implementation uses multiple concurrent threads, and callbacks instead of return values to methods.

7.6.1 The login sequence

The key element in the login sequence is the query to the UserManager object, which allows for offline login to the application. This is however only possible if the user has been logged in on that particular device before. In the case that a user has never logged in before, the gateway server is used to verify the user credentials, they are then stored if they are correct.

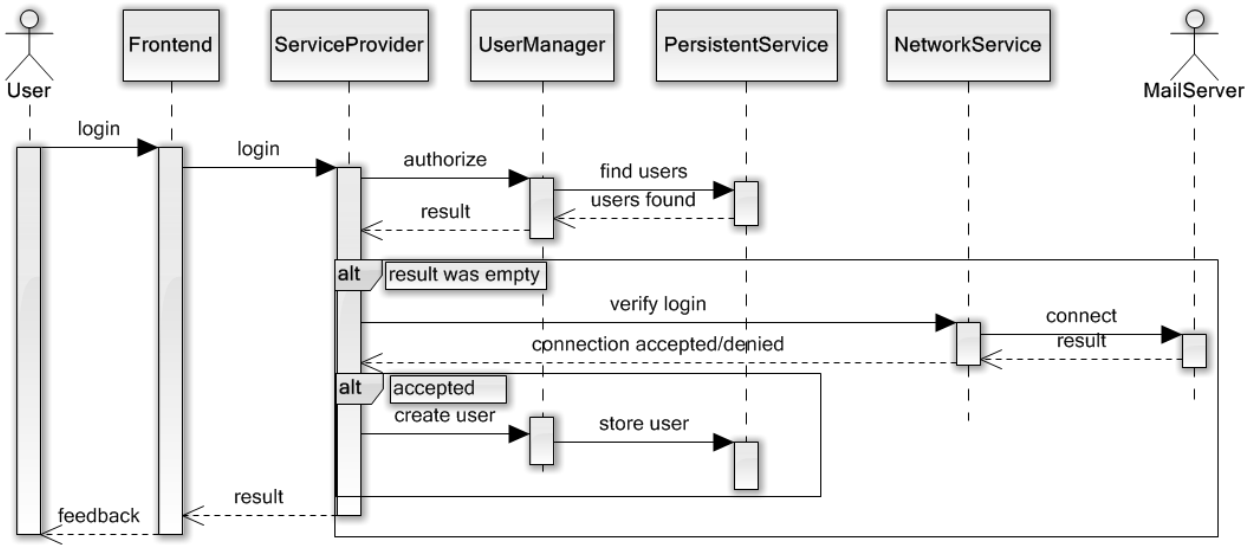


Figure 7.11: The login sequence

7.6.2 The send message sequence

In this sequence diagram you see the general flow when sending a message. One thing to note is the REPORTMESSAGEs in the diagram, these are implemented as callback functions in XOXOmail. The reasoning for this is that every time a consuming task is run in a separate thread, there is a need for asynchronous callbacks. These callbacks follows the OBSERVABLE PATTERN[119], which allows for multiple observers.

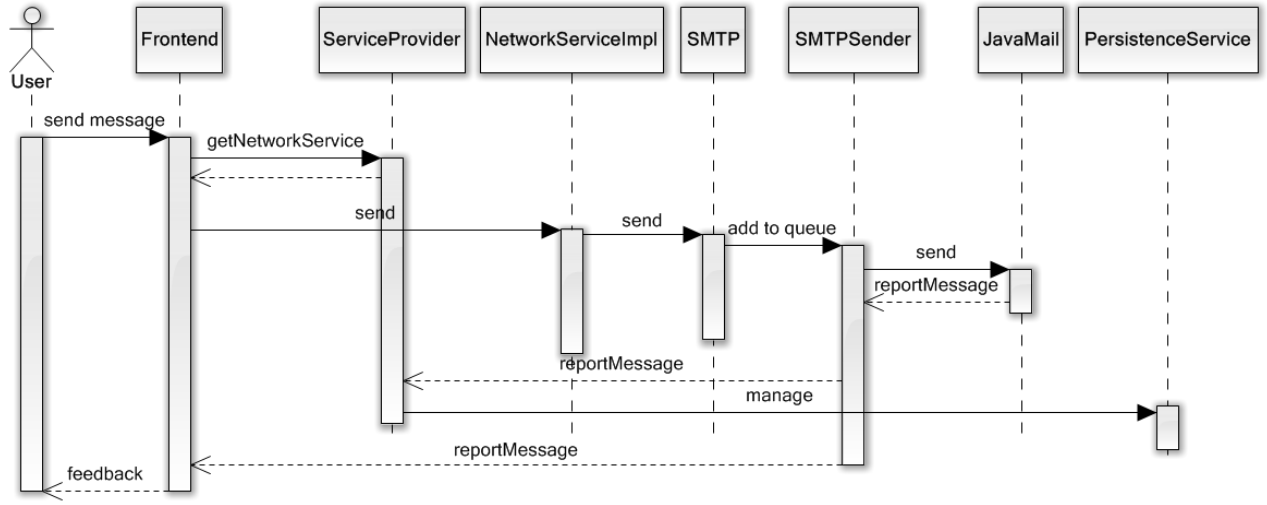


Figure 7.12: The send message sequence

7.6.3 The retrieve message sequence

The sequence diagram for retrieving messages shows the general setup of the IMAPIdle strategy. The thing to notice here is the loop structure within the sequence diagram, this is to illustrate the continuality of the idle command issued to the server.

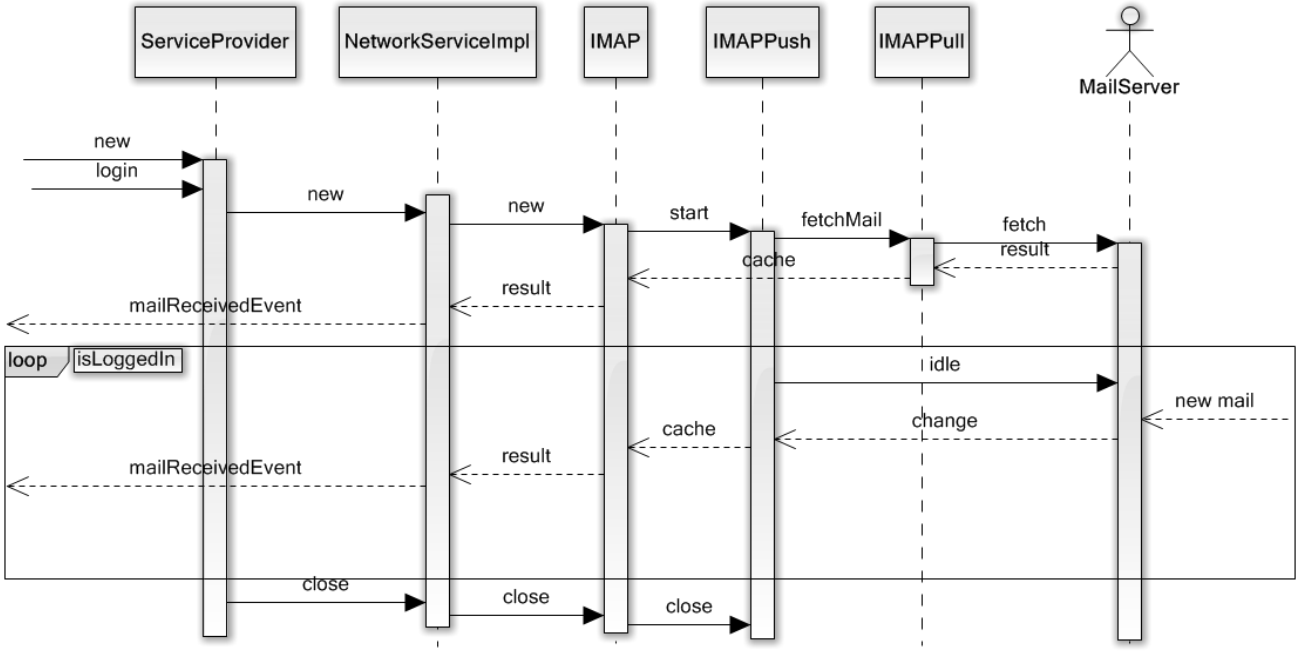


Figure 7.13: The imap push sequence

Part II

Scrum process

CHAPTER 8

PRODUCT BACKLOG

The product backlog is an ordered list of "requirements" for the project. Aspects taken into consideration when ordering the backlog include business value and dependencies.

Each entry in the backlog is on the form:
Priority.**Title(Complexity in storypoints)** Description

1. **Starting the application(5 SP)** As a user, I should be able to start the program so that i can begin browsing all features of the program (FR1)
2. **Create interface between core and gui(13 SP)** As a programmer, I will make the interfaces that is needed so that the service and gui can communicate in a simple way, making it easy for GUI programmers to start using code that has not been written yet (U1).
3. **Persisting data to phone(17 SP)** As a programmer I will have to implement persistence, so that the application is able to save data and retrieve it whenever it wants (FR3,FR4,FR5,FR9,FR10).
4. **Sending a message(13 SP)** As a user, I should be able to click the "New message" button, so that I am brought to the new message page, making me able to create a message and pressing "Send" to send it to a user (FR2).
5. **Browse previously sent messages(13 SP)** As a user, I want to see all messages that I have previously sent, so that I can check their status (FR4).
6. **Browse inbox(13 SP)** As a user, I should be able to review all previous received messages (FR3).
7. **Browse outbox(13 SP)** As a user, I should be able to review all previous sent messages (FR4).
8. **Read message(11 SP)** As a user I should be able to show the list of all messages received, both read and unread (FR3).
9. **Top menubar(5 SP)** As a user I should be able to use the bottom menubar to switch between pages. This should include *inbox*, *outbox*, *emergency message*, *new message* and *settings* (FR3,FR4,FR9,FR10).

10. **Architecture documentatio(40 SP)** As a programmer I would like to have an architectural description document, so that I know that my code fullfills all requiremetns.

11. **Security prestudy(40 SP)** As a developer I should research the security aspects of Android, in order to make well informed decisions.

12. **2. iteration, Networkservice(13 SP)** As a programmer I should extend the communication implementation to support more features, so that we can have message priorities, message types, message grading, message status and notification of failed deliveries (FR6).

13. **Receiving messages with attachments(15 SP)** As a user I should be able to receive a message with an attachment and open the attachment (FR7)

14. **Sending messages with attachments(13 SP)** As a user I should be able to add an attachment to the message I want to send , so that the recipient gets my attachment as well as the message (FR7).

15. **Log in to application(13 SP)** As a user, I should be able to log in via the login screen so that after this process I am an authorized user inside the program (FR1).

16. **Answer, delete and forward a message(8 SP)** As a user I want to be able to utilize the "Answer", "Delete" and "Forward" features that is associated with each message so that i am brought to the correct screen for each of these operations (FR8).

17. **Add signing to messages(20 SP)** As a programmer I should be able to create or use an existing library for digital signing and verification of messages (Sercurity).

18. **Compression algorithm(13 SP)** As a programmer I want to be able to create or find a library that minimizes data traffic that is needed to send a message so that the messages are sent faster over low speed internet connections (Preformance).

19. **Settings menu(8 SP)** As a user, I should be able to utilize the settings menu to alter different settings of the application, so that I have the settings set to what I prefer (FR10).

20. **Viewing address book(8 SP)** As a user I should be able to view my address book with all of my contacts, so that I am able to choose a recipient from a list when I want to send a message (FR5).

21. **Secure communications channel(5 SP)** As a programmer I should implement a wrapper that makes the communication with the server secure (Security).

22. **Implement hardware abstration layer(20 SP)** As a programmer I will implement the hardware abstraction layer, so that I can use the phones physical inputoutput devices, as GPS, camera, video and sound (FR7).

23. **Distribution of addressbook(13 SP)** As a programmer, I should be able to distribute address books between each of the users via LDAP or Control message so that all users have an address book with all users (FR5).

24. **Message templates(17 SP)** As a user, I should be able to choose a template for a message with all relevant information is predefined, as well as creating new ones, so that I can send a message without writing anything at all (Efficient Use).

CHAPTER 9

SPRINT 1

This chapter is meant to give an overview of sprint 1. Section 9.1 gives an overview of the sprint. Section 9.2 gives the duration of the sprint. Section 9.3 describes the concrete project work plan of the sprint. Section 9.4 describes the goal of the sprint. Section 9.5 details the ordered sprint backlog. Section 9.6 details the system design at the end of the sprint. Section 9.7 describes the feedback given by Thales on the sprint results. Section 9.8 gives an overview of how the hours were spent in the sprint. Section 9.9 closes the chapter off with a final conclusion on the sprint. The other sprint chapters are structured in a similar way.

9.1 Sprint 1 - Planning

The time leading up to Sprint 1 was spent looking into different solutions and agreeing on details such as programming language, development tools and top level architecture. We also spent a large part of the week setting up various tools on our personal computers; especially accounting for the wide variety of hardware and operating systems.

Sprint 1 was our first major coding spree. We set aside some time to set up all the required software and communications on the last few computers, but most of the time was to be spent putting together a rough prototype. Our goal was to be able to send and receive messages from the phone by the end of the sprint (fulfilling most of customer requirement 3.3), providing us with a functional framework we could bolt our later expansions on to. Tasks like security and intuitive GUI were saved for later sprints.

As well as the coding, which was to be largely handled by three of us in this first sprint, a lot of time was scheduled for theoretical studies. Things to be implemented later had to be researched and documented, and lot of basic things like agenda templates and documentation structure had to be set up.

9.2 Sprint 1 - Duration

The first scrum sprint officially began with a planning session August 27th and ended 22 days later on the September 17th. We choose to divide our time into four three week sprints, as this allowed us to divide up the 13 week project simply and evenly, with 1 week to plan and organize in the beginning.

9.3 Sprint 1 - Goal

Our goal was a to have a demo ready that we could show the client, that could send and receive messages through an SSL channel. This would basically serve as a proof of concept; a demonstration of the technical possibilities. This would require three major components. First we would need a rudimentary user interface in order to display a received message and send a given message to an SMTP server. Secondly we would require some kind of listener that could keep in contact with the server and be informed when a new message is received, and retrieve that message from the server as soon as possible (preferably instantly). Finally we would need some kind of sender that could interface with a SSL protocol.

9.4 Sprint 1 - Ordered sprint backlog

- **Setup of Jira:** We decided to use Jira to get an overview over all the work that we need to do. Jira also gives us an overview over what tasks are supposed to be done in which sprint. We also use Jira to assign the tasks to each of the team members.
- **Report work:** The report that is to be delivered at the end of the project needs a lot of work and we have decided that we will use time on the report in every sprint.
- **Set up programming environment:** Working with setup of software in an administrative sense.
- **Group Administration:** To administrate on behalf of the entire group, with booking group rooms and sending emails to Thales, the advisor and the team to call in for meetings.
- **Meetings:** Arranging meetings and the effort we use on meetings.
  - **Meetings with Thales:** We have weekly meetings with Thales so that we can get rapid feedback on what we do. To know if they agree with our decisions and that we haven't misunderstood the task thay have given us.
  - **Meetings with Mohnsen Anvaari:** We have regular meetings with our advisor so that he can give us feedback on how we are doing our work, and make sure that we do what are expected of us in the course.
  - **Internal meetings:** Every time we meet we first have a status meeting were we share what we have done and how far we have come with our tasks.
  - **Lectures:** There are some lectures during the semester, and we are adviced to participate in these. In this sprint there are 3 lectures and courses that we have decided to attend.

- **Create interfaces between Core and GUI:** Code interfaces that let GUI and core communicate in the simplest way. We need this code to be able meet our Sprint 1 milestone. We also need these interfaces to make a simple model.
  - **Persistence Service Interface:** Preliminary design of interfaces for the Persistence Service module.
  - **Hardware abstraction layer interface:** Preliminary design of interfaces for the HAL Service module.
  - **Network Service Interface:** Preliminary design of interfaces for the Network Service module.
  - **Security Service Interface:** Preliminary design of interfaces for the Service Service module.
- **General setup of tools:** General setup of different tools we use.
- **Meeting and agenda document writing:** To write all the meeting agendas and minutes using the right template.
- **Starting Application:** Making our program good enough so that it is possible to start the program and begin browsing all features of the program.
  - **Create Main menu:** Make a simple main menu with a header showing the name of the program and a list containing all views it is possible reach from the menu.
  - **Create a basic Android application skeleton:** Make an Android application project so that we have a running application with nothing in it.
  - **Learn how Android MVC works:** Find out how Android MVC works and get a general feeling of how the layout of the Android project will be.
- **Sending a message:** The user should be able to click the "New message" button, be brought to the new message page, create a message and send it by pressing the "Send" button.
  - **Adding subject and text to a message:** Create a simple GUI to make the user able to create a message with a title and a text and sending it to a recipient. At first it is enough to send to a predefined receiver until the address book is made.
  - **Make connection between "Send Message" button and backend:** Make the GUI communicate with the backend responsible for sending the actual message.
  - **Implement a Network class for sending email through gmail's smtp-server:** Implement an instance of the NetworkService interface which sends mail via gmail's mail servers.
  - **Create the new message view:** Make it possible for the user to get a view showing all fields relevant to creating a message by clicking "New message".
  - **Implement receiving mail from gmail's imap-service:** Make the application able to receive the mail automatically from Gmail's IMAP, as soon as a message is received at the account. This must be done via push to client, not constant pulling.
  - **Create core bridge:** Make the connection from GUI to core and implement return value from interface on core side.

- **Persisting data to phone:** Implement persistence, so that the application is able to save data and retrieve it whenever it wants.
  - **Research on structure for saving and reading data**: Find a structure for saving data to the phone. What is the best way of organizing data with respect to files created by application, file settings and other files?
  - **Save data to phone storage:** Ensure proper saving of data to phone.
  - **Read data from phone:** Ensure proper saving of data to phone.

9.5 Sprint 1 - System design

At the end of the first sprint, the the system worked thusly: The user opens an Application (XOX-Omail) on his phone. He is taken directly to a menu, though the final product would begin with a login screen. In the first sprint the menu was rudimentary; designed more for ease of demonstration than for actual application use. An updated version was under development but had not yet been integrated with the application at the end of the first sprint. The menu, in this iteration, had three options: Inbox, Sent, and SendMail.

Clicking inbox would bring the user to a simple list of the emails received while the application was active (persistent storage had yet to be connected to the rest of the system at the end of the first sprint). Each email was click able, leading to a detailed view with the message text as well as further clearance and sender info. The GUI here sat on top of a Activity which held references to the models as well as keeping in contact with the network adapter which listened to a server for new emails. Clicking Sent would similarly bring the user to a list of sent emails, which sat on its own Activity.

SendMail would bring the user to a simple form for sending messages to the server. Details like subject, receiver, security level and priority could be specified and a text message written. The demo was capable of sending to an arbitrary email address.

The system consisted of four main components. The XML coded GUI was displayed and controlled by a number of activities; together forming the main GUI layer. The GUI layer communicated with a service core (which at the time only had the network system), with in turn created instances of methods in the modeling layer to refer around the system.

9.6 Sprint 1 - Customer feedback

Thales was largely positive about our progress in Sprint 1. They were extremely pleased that we had managed to build a working prototype. They noted that we had gotten far in already being able to send and receive messages, and that boded well for our ability to implement some "interesting" components later in the process.

There was, however, a number of points they felt needed work. They had a number of comments on the GUI; both the rudimentary version used in the demo and the mock up "paper"

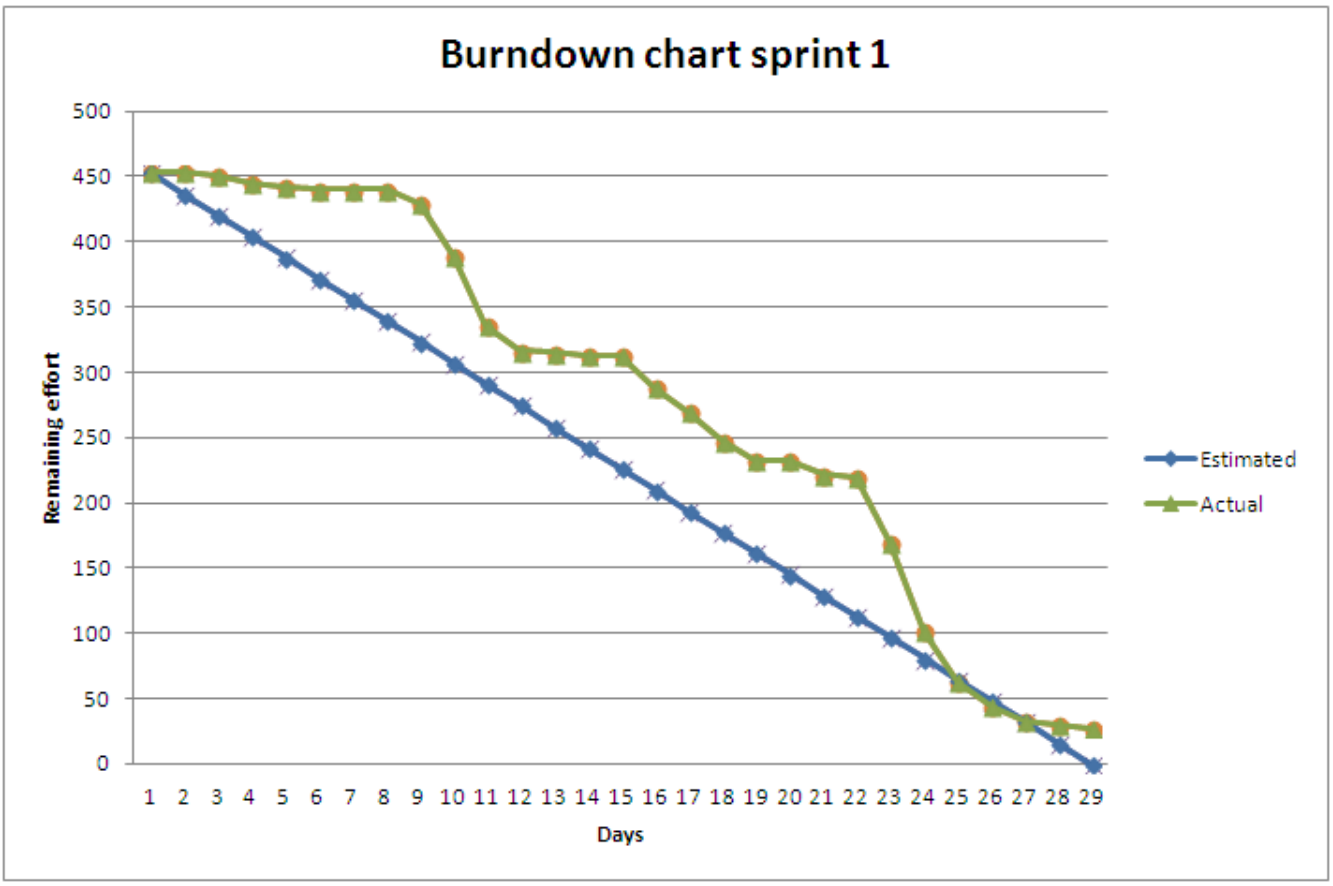


Figure 9.1: Burndown chart sprint 1

prototype that we had sent them earlier. They wanted label (drill, exercise, etc) and the sent time/date included in the inbox display of emails, as well as security clearance. They also wanted security clearance visible in the top right corner whenever classified information was visible on the page.

They also had a number of comments about our project planning and use of scrum. Most importantly, we had fallen into the practice of using too general posts, leading ambiguities about what parts of that post was included in the current Sprint and what was to be implemented later. We agreed to avoid this in the future, and resolved to include this in our discussions of Sprint 2. There was also some minor comments on various sub-tasks and small ambiguities to clear up, but nothing major.

9.7 Sprint 1 - Effort

See table 9.2 below.

Group no: 15 Date: 27/08-16/09					
Activity	**Start**	**W2** 27/08-02/09	**W3** 03/09-09/09	**W4** 10/09-16/09	**Activity sums**
Management	**E:15** A:0	**E:15/15** A:15.5/15.5	**E:15/30** A:12.5/28	**E:11/41** A:7.25/35.25	**E:41** A:35.25
Lectures	**E:0** A:0	**E:0/0** A:0/0	**E:15/15** A:16/16	**E:9/24** A:0/16	**E:24** A:16
Planning	**E:10** A:0	**E:10/10** A:33/33	**E:11/21** A:28/61	**E:5/26** A:34.5/95.5	**E:26** A:95.5
Pre study	**E:10** A:0	**E:10/10** A:8/8	**E:10/20** A:0/8	**E:14/34** A:4/12	**E:34** A:12
Requirement	**E:5** A:0	**E:5/5** A:1/1	**E:5/10** A:2/3	**E:8/18** A:11/14	**E:18** A:14
Design	**E:40** A:0	**E:40/40** A:31.5/31.5	**E:25/65** A:4/35.5	**E:22/87** A:12/47.5	**E:87** A:42.5
Implementation	**E:0** A:0	**E:0/0** A:0/0	**E:58/58** A:21.5/21.5	**E:100/158** A:27/48.5	**E:158** A:48.5
Documentation	**E:10** A:0	**E:10/10** A:8.5/8.5	**E:20/30** A:11/19.5	**E:55/85** A:67.25/86.75	**E:85** A:86.75
Evaluation	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Demonstration	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Period sums	E:90 **A:0**	E:90/90 **A:97.5/97.5**	E:159/249 **A:95/192.5**	E:224/473 **A:160/355.5**	E:473 **A:355.5**

Table 9.2: Table for effort registrations in sprint 1

Period comments:

- **W2:**

  - **Requirement:** The one who started on the task found out that he did not understand how to solve the task so the task was moved to W3.

- **W4:**

  - **Lectures:** The lecture that was announced was moved to next week.

Activity comments:

- **Planning:** We experienced quickly that to have many internal meetings was wise, but still there were more than first anticipated, and some took longer than expected.
- **Pre study:** We decided to do some other parts of the documentation instead, and moved this task to sprint 2.
- **Design:** the design part turned out be simpler than expected, and the tasks was smaller than we tought.
- **Implementation:** We haven't had the time to test the code that was written, so the testing have been delayed to sprint 2.

9.8 Sprint 1 - Conclusion

Despite a number of setbacks, some late planning and technical problems, the first sprint was a success. We managed to create a working prototype, as well as a number of (as of yet) unconnected components. We hammered out a functional architecture, which should be more than robust enough to handle any changes forced on it in the next sprint. And we are well on schedule when it comes to the documentation; including a number of templates and designs for the type of files we produce a lot of (agendas and minutes especially).

In the sprint 2 we want to better at planning and more structured in our work. We want to create more concrete tasks for people to work on instead of big ones that no one knows what are really about, that is, conform better to the Scrum process. We also hope that the tools will give us less problems and more time to focus on the real work. We have also realized that we need to utilize every hour of the sprint and work hard continuously.

CHAPTER 10

SPRINT 2

10.1 Sprint 2 - Planning

Finishing the first sprint left us with a working demo that could send and receive emails. After meeting with Thales and showing them the demo, we came to the conclusion that we needed to perform a thorough study regarding the security aspect of the application. The security issues that were most vital to the functionality of the application were local storage of program data and secure sending with SSL (fulfilling customer requirement 5.1). We also realized that we needed more concise documentation of the architecture, which was a bit vague at this point. This was something which we did intentionally, as we weren't sure what we could implement and what would be left as a pre- and poststudy.

We knew that the second sprint would be hectic. We did not have much room for extra workloads, as the regular documentation work takes a lot of time. Thales' request for a security prestudy did not make it easier, but we agreed to continue at a higher pace with both the frontend and backend part of the application. The GUI of the frontend was almost non existing, so it had to be developed from scratch within just one sprint. One of the biggest goals was to implement the ability to add attributes to outgoing messages, so that they could be prioritized on these attributes, as well as getting a GUI with a more complete look and feel.

Sprint two had a lot of time set aside for further documentation. This was crucial for being able to get better progress in the later sprints regarding programming. Most members of the project group were set to work on the documentation while two of the developers continued to work with the application.

10.2 Sprint 2 - Duration

The second sprint started with a planning session on September 18th and ended on October 7th.

10.3 Sprint 2 - Concrete project work plan

Milestone: Have a working demo to show Thales and the advisor. In addition to the milestone of sprint 1, it should incorporate an inbox and sent box in a much more comprehensive way where all GUI functionality is coarsly completed. At completion of the sprint, substantial documentation regarding architecture and security prestudy should also be completed. See table 10.2 on page 124.

10.4 Sprint 2 - Goal

The goal of the second sprint was to increase the depth of the documentation, especially the architecture documentation, and get further insight into the security issues that arise with several parts of the application.

At the end of the sprint we were also going to show a demo to the Thales that would have the functionality of adding attributes to messages.

In order to achieve this sprint's goal, we needed one group member to document our architecture properly, two group members to continue the work with the application, while the rest of the group members looked into the security problems we were facing, and how we best could solve them.

10.5 Sprint 2 - Ordered sprint backlog

- **Report work:** The report that is to be delivered at the end of the project needs a lot of work, and we have decided that we will use time on the report in every sprint.
- **Group Administration:** To administrate on behalf of the entire group, to book group rooms and send emails to Thales, the advisor and the team to call in for meetings.
- **Meetings:** Arranging meetings and the time we use on meetings.
  - **Meetings with Thales:** We have weekly meetings with Thales so that we can get rapid feedback on what we do. To know if they agree with our decisions and that we haven't misunderstood the task thay have given us.
  - **Meetings with Mohnsen Anvaari:** We have regular meetings with our advisor so that he can give us feedback on how we are doing our work, and make sure that we do what are expected of us in the course.
  - **Internal meetings:** Every time we meet we first have a status meeting were we share what we have done and how far we have come with our tasks.
  - **Lectures:** There are some lectures during the semester, and we are adviced to paticipate in these. In this sprint there are 2 lectures and courses that we have decided to attend.
- **Browse previously sent messages:** Make it possible to see all messages that have been previously sent, so that it is possible for the user to check that they actually were sent.

Explanations of the short names in the table:
KP-10: Meetings
KP-21: Browse previously sent messages
KP-76: Sending a message
KP-78: Browse inbox
KP-82: Read message
KP-89: Browse outbox
KP-90: Bottom menubar
KP-91: Security prestudy
KP-96: Architecture documentation

Main activity	**Sub-task**	**Estimate**
KP-1	Setup of Jira	5h
KP-2	Report work	70h
KP-3	Set up programming environment	5h
KP-9	Group administration	25h
KP-10	Meetings with Thales	15h
KP-10	Meetings with Mohnsen Anvaari	6h
KP-10	Internal meetings	60h
KP-10	Lectures	8h
KP-21	Show basic listing of all elements in list	3h
KP-21	Style elements of list	8h
KP-60	Meeting and document writing	18h
KP-76	Implement metadata structure and show it	6h
KP-78	Show basic listing of all elements in list	3h
KP-78	Style elements of list	8h
KP-82	Showing all metadata	4h
KP-82	Showing all message subject and text	3h
KP-82	Reply/Delete/Forward	8h
KP-89	Show basic listing of all elements of list	3h
KP-89	Style list for pretty viewing	8h
KP-90	Create bottom menu bar	3h
KP-91	Secure sending	8h
KP-91	Secure storage on phone	10h
KP-91	Signing and verification	13h
KP-91	Limitations and facilities of Android	15h
KP-91	Security requirements by CT	13h
KP-96	Fullfillment of requirements	4h
KP-96	Graphical view of architecture	18h
KP-96	Frontend	5h
KP-96	Backend	5h
		360h

Table 10.2: Sprint 2 tasks

  - **Show basic listing of all elements in list:** A basic listing of all elements should be implemented, and each data element in the list should have all necessary fields that are required.
  - **Style elements in list:** Use design theory to make all the elements look presentable.

- **Meeting and agenda document writing:** Write all the meetings agendas and minutes using the right template.

- **Sending a message:** The user should be able to click the NEW MESSAGE button, be brought to the new message page, create a message and send it pressing the SEND button.

  - **Implement metadata structure and show it:** Implement all the fields that are required to send a message; from, to, classification, label, priority, and make them look presentable.

- **Browse inbox:** The user should be able to show a list of all received messages, both read and unread.

  - **Show basic listings of all messages received:** Implement a basic list of all elements, and make sure that all necessary fields that are required are displayed.
  - **Style elements in list:** Style each element of the list, so that it is pleasing to look at.

- **Read message:** Making the user able to read a message.
  - **Showing all metadata:** Implement all fields that are related to a message, and display them.
  - **Showing message subject and text:** Show all fields that are related to a message; date received, from.
  - **Reply/Delete/Forward:** Implement the buttons so that the user can reply to a message, delete a message and forward a message.
- **Browse outbox:** The user should be able to show a list of all messages that are waiting to be sent, or haven't been sent yet.
  - **Show basic listing of all elements in list:** Implement a basic listing of all elements, and that each data element in the list has all necessary fields that are required.
  - **Style list for pretty viewing:** Make the list pleasing to look at.
- **Bottom menu bar:** Implement the bottom menu bar so that a user can switch between pages using the menu bar.
  - **Create bottom menu bar:** Implement a menu bar and make it presentable and useful by applying design theory.
- **Security Pre-study:** Dig deeper into the security aspects of Android, in order to make well informed decisions.
  - **Secure sending:** Write security documentation on secure sending.
  - **Secure storage on phone:** Write documentation regarding secure storage of data and password.
  - **Signing and verification:** Find out how secure signing works, and write documentation on it.
  - **Limitations and facilities of Android:** What are the limitations regarding Android security, and what does Android facilitate when it comes to encapsulation? Is data in memory secure? Find the answers to the questions and document it.
  - **Security requirements:** Write a summary of the documentation linked to by Tellefsen.
- **Architecture documentation:** Make an architectural description document, so that we know that the code fulfills all the requirements.
  - **Fulfillment of requirements:** Describe how our architecture fulfills the security, usability and latency requirements.
  - **Graphical view of architecture:** Create a graphical view of the architecture, both backend and frontend.
  - **Frontend:** Document the frontend properly.
  - **Backend:** Document the backend properly.

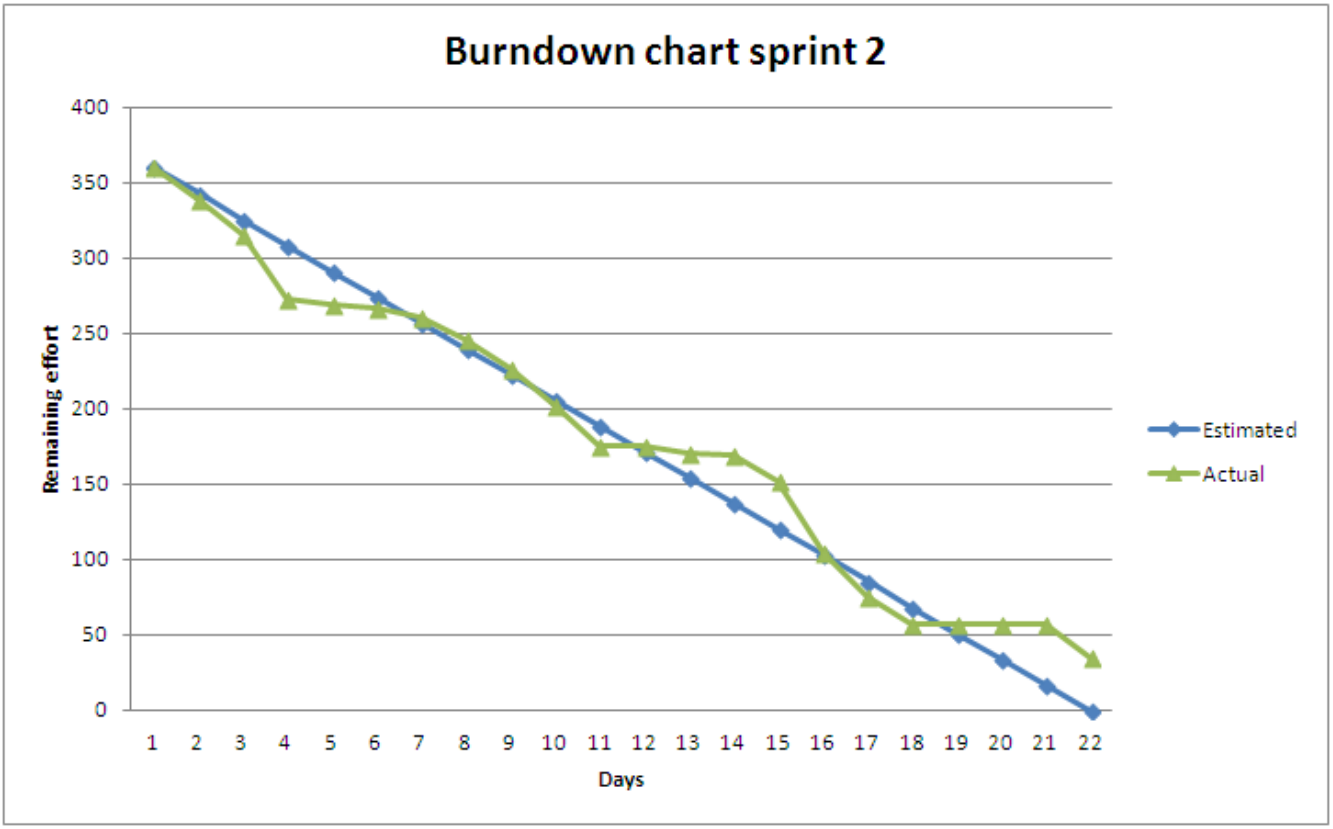


Figure 10.1: Burndown chart sprint 2

10.6 Sprint 2 - System design

After the second sprint, the system worked in the following way: A user opens XOXOmail on his phone. He is then brought directly to a tab called "Folders" which at first displays all the inbox messages of the application.

In the FOLDERS TAB, the user gets the opportunity to choose either INBOX, OUTBOX or SENT from a dropdown menu. INBOX shows a list of received messages, which the user can read one by one by clicking them. SENT takes the user to a list of sent messages which works in the same way as INBOX. OUTBOX is in this iteration not yet implemented. There is also a button that is called sort, which is also not implemented.The backend of this feature works in the same way as it did after the first sprint.

If the user selects the NEW MESSAGE tab, he is taken to the message creation screen where he can add recipients and subject to the message. In this iteration we also included the functionality of adding attributes to the messages sent (Security Label, Message Priority and Message Type). The attributes are sent along with the message in the message header.

The changes to XOXOmail with respect to functionality in sprint one, is the addition of attributes to the messages and the menu bar at the top. It is all now packed into a better user experience where the number of errors a user can do is a lot fewer.

10.7 Sprint 2 - Customer feedback

Thales was mainly satisfied with the work done in sprint 2. The most important changes had been made in the appearance of the application, and Thales approved of many of the choices made in the user interface. Thales had no comments with regards to the back-end functionality, as this part had not been significantly extended.

Thales requested certain improvements and changes in the user interface, but most of them were minor issues. Some of the points were to change the validation of fields in the New Message-view, reduce the size of the menu and using capital letters and non-breaking space in the security labels to conform to the SIO-label standard.

10.8 Sprint 2 - Effort

See table 10.4 on page 130

10.9 Sprint 2 - Conclusion

The second sprint was well planned and had tasks that was doable in the given time-span we had available. We ended up with a lot more thorough documentation, especially on the parts regarding security and system architecture. We also ended up with a new demo with additional functionality of adding attributes to the messages. During this sprint there was also less time wasted on group bureaucracy since routines and group dynamics had become more integrated in the daily routines. We managed to complete all the goals we set for this sprint and can therefore conclude that the sprint was a success.

Group no: 15 Date: 17/0-07/10					
Activity	**Start**	**W5** 17/09-23/09	**W6** 24/09-30/09	**W7** 01/10-07/10	**Activity sums**
Management	**E:11** A:0	**E:11/11** A:5.5/5.5	**E:12/23** A:2.5/8	**E:12/35** A:1.5/9.5	**E:35** A:9.5
Lectures	**E:5** A:0	**E:5/5** A:6/6	**E:3/8** A:3.5/9.5	**E:0/8** A:0/9.5	**E:8** A:9.5
Planning	**E:40** A:0	**E:40/40** A:38.5/38.5	**E:29/69** A:11.5/50	**E:30/99** A:16.5/66.5	**E:99** A:66.5
Pre study	**E:11** A:0	**E:11/11** A:10/10	**E:23/34** A:10.5/20.5	**E:25/59** A:8.5/29	**E:59** A:29
Requirement	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Design	**E:19** A:0	**E:19/19** A:18.5/18.5	**E:19/38** A:40/58.5	**E:19/57** A:42/100.5	**E:57** A:100.5
Implementation	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Documentation	**E:34** A:0	**E:34/34** A:40/40	**E:34/68** A:59.5/99.5	**E:34/102** A:55.5/155	**E:102** A:155
Evaluation	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Demonstration	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Period sums	E:120 **A:0**	E:120/120 **A:118.5/118.5**	E:120/240 **A:127.5/246**	E:120/360 **A:124/370**	E:360 **A:370**

Table 10.4: Table for effort registrations in sprint 2

Period comments:

- **W5:**
  - **Management:** We thought that it would be almost as much time spent on management this week as W4, but that was not the case.
- **W6 & W7:** We estimated almost every activity wrong this week, but the weekly effort was still pretty good since we had both under and over estimated our tasks.

Activity comments:

- **Planning:**Some of the planning tasks was done last sprint for this sprint also.
- **Pre study:**The topics that we were studying this week turned iut be be prettey easy to understand.
- **Design:**We thought that the GUI-issues in this task would be pretty simple. But there were some challenges that resulted in a lot more hours than expected.
- **Documentation:**We did not expect that we would spend so much time on documentation, but a lot could be written this early in the project.

CHAPTER 11

SPRINT 3

11.1 Sprint 3 - Planning

After finishing the second sprint we ended up with a lot more documentation and as a result of this a more general understanding of how XOXOmail should function when considering both the frontend and backend part. As we had become *a jour* with most of the documents we could start planning a sprint that had more focus on the actual application and the features that it should include.

After meeting with Thales we ended up with two areas that we agreed to make the main focus in the third sprint. One of the two was the ability to connect XOXOmail with a test-server provided by Thales (as per customer requirement 2.3), the other was to implement the feature of receiving and sending messages with an attachment

In this sprint we also made room for several application features in the customer requirements that had not yet been implemented. This included:

- Logging in to the application (FR1)
- Answering and forwarding messages (FR8)
- Adding signing to messages (FR14)
- Adding a settings menu (FR10)
- Sending instant messages (FR9)
- Deleting received messages (FR8)
- Miscellaneous graphical user interface design issues

We also wanted to see how the application worked on a lower network layer, so we planned to perform a Wireshark analysis of the application regarding the sending and receiving part of the system.

Most of the tasks to be solved in the third sprint involved a lot of programming. This resulted in many tasks that had an uncertain time estimate. Sometimes a solution comes to life with little bugs and issues, other times a seemingly easy task can take up a lot of time due to unforeseen problems. We did however acknowledge that this sprint was going to have a heavy focus on the programming and that there were many tasks to complete. We therefore put most of the group on the various programmatic problems whilst having one group member continuing the report work. We also had one person that concluded the Wireshark analysis in between the other programming tasks.

With the completion of sprint 3 we hoped to be more or less done with developing the application so that we could focus on the documentation in the fourth and final sprint. This was important because much of the added functionality needed to be documented along with many of the choices taken.

11.2 Sprint 3 - Duration

The third sprint started with a planning session on October 8th and ended on the 28th of October.

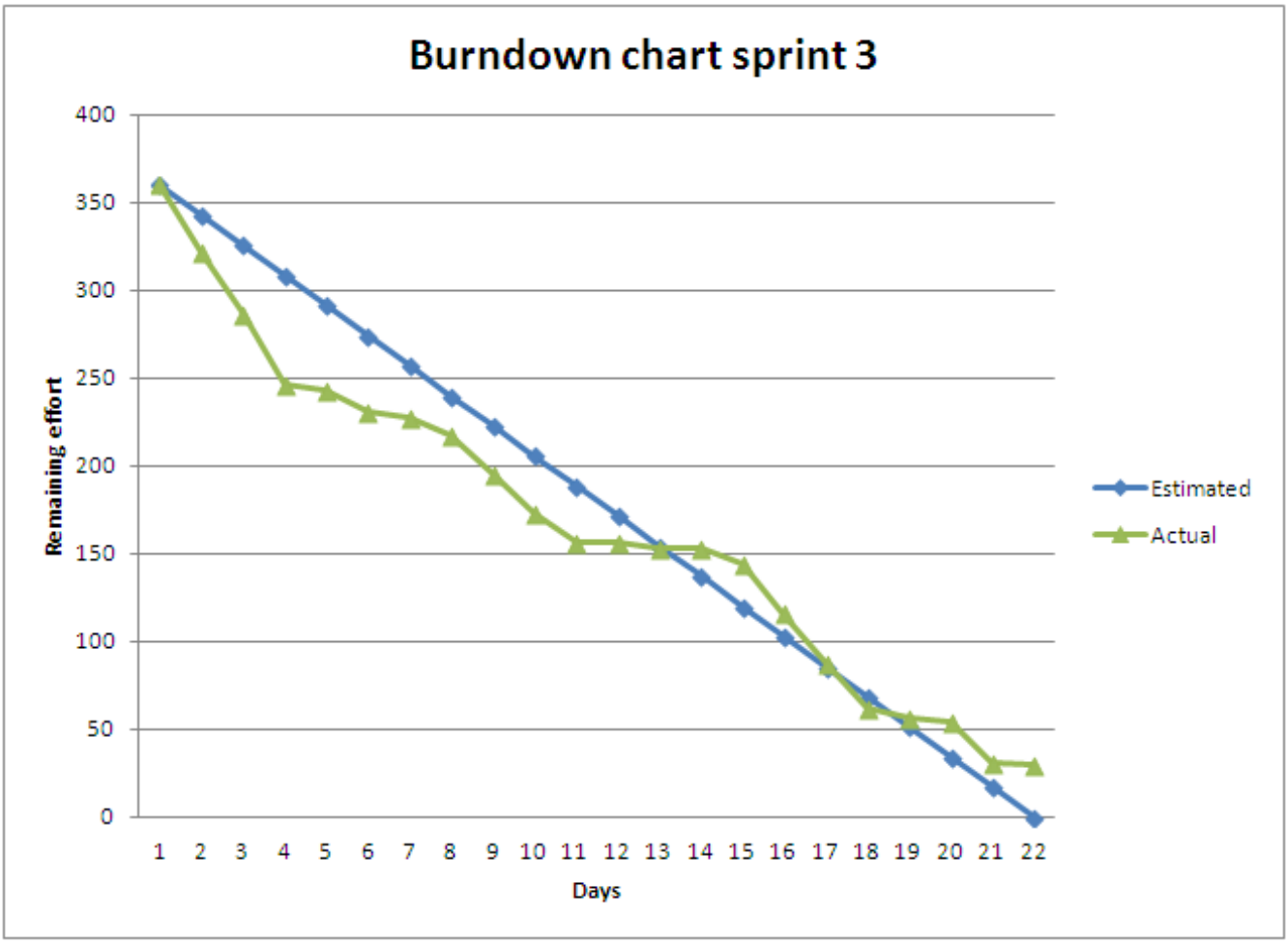


Figure 11.1: Burndown chart sprint 3

Main activity	Sub-task	Estimate
KP-2	Report work	68h
KP-9	Group administration	2h
KP-10	Meetings with Thales	18h
KP-10	Meetings with Mohnsen Anvaari	15h
KP-10	Internal meetings	60h
KP-10	Lectures	15h
KP-19	Create basic GUI	1h
KP-19	Find solution for encrypting the login information	1h
KP-19	Persist and fetch login data	5h
KP-26	Implement threaded SMTP queue	15h
KP-26	Implement stripping of message based on network connection	8h
KP-26	Implement usage IMAP specific attributes	6h
KP-26	Save messages downloaded from server	4h
KP-26	Update message IMAP message status	6h
KP-26	Implement pull strategy	6h
KP-26	Implement push strategy	10h
KP-26	Implement handling for pre-send processing of messages	8h
KP-26	Implement handling for control-messages	8h
KP-28	Study	2h
KP-28	Find what attachments we should support	1h
KP-28	Documentation of choice for attachments	1h
KP-28	Implement	6h
KP-28	Implement GUI	8h
KP-29	Locking security label on reply and forward	1h
KP-32	Implement a keystore for saving and loading trusted keys	8h
KP-32	SPIKE: bouncycastle for Android	4h
KP-32	Implement S/MIME signing of messages	10h
KP-32	Implement verification of signing messages	10h
KP-34	Set update interval for checking of mail to avoid pulling	1h
KP-34	Set security labels available	1h
KP-34	Standard receiver of instant message	1h
KP-35	Compression study	6h
KP-79	Study	2h
KP-79	Implement	6h
KP-79	Document	2h
KP-109	Learn Wireshark	4h
KP-109	Document Wireshark	2h
KP-109	Analyse traffic	10h
KP-109	Discussion	3h
KP-119	Find design solution	1h
KP-119	Incorporate instant message button into GUI	1h
KP-119	Create send instant message view	2h
KP-127	Delete from local strage	2h

KP-127	Delete from mail server	4h
KP-127	Update GUI	2h
KP-131	Menu	1h
KP-131	Header	1h
KP-131	Change security labels to upper case	1h
		360h

Table 11.1: Sprint 3 tasks

11.3 Sprint 3 - Goal

The goal of this sprint was divided into two parts. The first part consisted of being able to send a message to a testserver that Thales had arranged. We wanted to see if our messages could be processed through their internal systems and that the flags included in the header were picked up correctly. The second part was to send and receive messages with attachments. This was an important feature in the requirements specified by Thales, especially being able to send pictures and GPS coordinates as they can be a vital part of information in a military setting.

When this sprint ended we also hoped to be done with the development of the application so that we could focus on the remaining documentation that needed to be completed. As with the first and second sprint we were to show a demo of XOXOmail at the end of the sprint. This time it is most likely the final product that will be demonstrated.

To complete the goal of this sprint we sat aside many hours for implementation of new functionality. Most of the group worked full time on the application with two people working on the backend while three people worked on the various frontend issues.

11.4 Sprint 3 - Ordered sprint backlog

- **Report work:** The report that is to be delivered at the end of the project needs a lot of work, and we have decided that we will use time on the report in every sprint.
- **Group Administration:** Administrate on behalf of the entire group, to book group rooms and send emails to Thales, the advisor and the team to call in for meetings.
- **Meetings:** Arranging meetings and the time we use on meetings.
  - **Meetings with Thales:** We have weekly meetings with Thales so that we can get rapid feedback on what we do. To know if they agree with our decisions and that we haven't misunderstood the task thay have given us.
  - **Meetings with Mohnsen Anvaari:** We have regular meetings with our advisor so that he can give us feedback on how we are doing our work, and make sure that we do what are expected of us in the course.
  - **Internal meetings:** Every time we meet we first have a status meeting were we share what we have done and how far we have come with our tasks.
  - **Lectures:** There are some lectures during the semester, and we are adviced to paticipate in these. In this sprint there are 2 lectures and courses that we have decided to attend.

- **Log in to application:** As a user I should be able to log in via the LOGIN screen so that after this process, I am an authorized user inside the program.
  - **Create basic GUI:** Create a basic GUI for login. This means a GUI that has a username and password field, and a LOG IN button
  - **Find solution for encrypting the login information:** Figure out what solution we are going to use for save the login information. This means that we need to decide for an implementation. The best is if we can make a solution that is so good that we don't need to revise it.
  - **Persist and fetch login data:** Implement the saving and fetching of the login data, making the login functionality complete.
- **Iteration network service:** As a programmer I should extend the communication implementation to support more features, so that we can have message priorities, message types, message classification, message status and notification of failed deliveries.
  - **Implement threaded SMTP queue:** Implement all SMTP related method to run in a separate thread other then the GUI-thread, in order to ensure responsive user interface.
  - **Implement stripping of message based on network connection:** Implement a pre-sending processing step for XOMessage which removes possible attachment based on the network connection at the current time. This should be done through the use of interfaces to ensure modifiability.
  - **Implement usage IMAP specific attributes:** Implement the usage of IMAP states into NetworkService so that the phone and server is in a consistent state. E.g SEEN flag ect.
  - **Save messages downloaded from server:** Persist a number of the latest received messages locally on the device. The number of messages that are to be persisted should be determined by an option in the settings menu.
  - **Update message IMAP message status:** Update flags on messages whenever the state changes, e.g. the messages is opened locally on the device and this change should be reflected on the server as well.
  - **Implement pull strategy:** Implement a pull strategy for periodically pulling the mail server for new messages.
  - **Implement push strategy:** Implement a push strategy using the IMAP-Idle command in order to get the server to push messages to the device.
  - **Implement handling for pre-send processing of messages:** Implement a pre-send processing of messages that handles template codes. e.g. #GPS : will fetch the gps data and inject them into the message before sending.
  - **Implement handling for control-messages:** Implement a onReceive handler the stops control-messages of getting through to the user, and in addition responds to the control-message in the correct way.

- **Sending message with attachments:** The user should be able to send a message containing different attachments.
  - **Study:** Figure out how to fetch images, i.e. from the phone, or another application on the phone. How does it work on Android?
  - **Find what attachments we should support:** Based on how difficult it is to get images, GPS coordinates etc, make a decision on what kind of attachments we should support. Maybe there are some things we should be careful about using, maybe it isn't. Find out!
  - **Documentation of choice for attachments:** Document what attachments we chose and why.
  - **Implement:** Imlement sending of attachments based on what found out in the study. What this tasks means, depends on which attachments we are to send. A picture will be sent differently than GPS coordinates. Maybe the coordinates should be implemented into the message body, whilst the image will be shown by a button. All this should be reveiled during the Study of this main task.
  - **Implement GUI:** Implement the GUI-side of sending messages with attachments based on the conclusion of the study. How to send GPS vs binary data.
- **Answer and forward a message:** As a user I want to be able to utilize the REPLY and FORWARD buttons that is associated with each message so that I am brought to the correct screen for each of these operations.
  - **Locking security label on reply and forward:** The user should not be able to set the security label of a message when he/she replies or forwards.
- **Add signing to messages:** As a programmer I should be able to create or use an existing library for digital signing of messages, so that only signed messages are identified by the recipient as a valid message.
  - **Implement a keystore for saving and loading trusted keys:** Implement a way of safely storing keys and key derivatives locally on the device.
  - **SPIKE bouncycastle for Android :** Research the use of bouncycastle on an Android device in order to find out if this is a possible solution for encryption, signing and verification of messages.
  - **Implement S/MIME signing of messages:** Bases on the bouncycastle Spike, implement the signing of messages.
  - **Implement verification of signing messages:** Based on the bouncycastle Spike, implement the verification of signing messages.

- **Settings menu:** As a user, I should be able to utilize the settings menu to alter different settings of the application, so that the settings are set to what I prefer.
  - **Set update interval for checking of messages to avoid pulling:** This is a setting making it possible for the user to choose to have push or pull solution for messages. If they chose to use push, the application always have a connection up to the server. If the user don't have an Internet connection, the application will try to set up the connection all the time and use a lot of battery. If the user chose pull, the application uses a predefined interval for how often the application should check for messages. So one solution could be a radio button list: PUSH, PULL.
  - **Set security labels available:** The user should be able to choose which security labels he will find available in the dropdown when sending a new message.
  - **Standard receiver of instant message:** The user should be able to set a standard receiver to use when sending an instant message.
- **Compression study:** As a programmer I want to be able to create or find a library that minimize data traffic that is needed to send a message, so that the messages are sent faster over low speed Internet connections.
- **Receiving message with attachment:** As a user I should be able to receive a message with an attachment and show it.
  - **Study:** How should we show the different attachments? Use some embedded features of Android or use our own? Are the attachments shown immediately or do we click a button to show it? Do some studies so that you are able to answer all the questions above.
  - **Implement:** Implement showing attachments based on what was found out in the study. What this task involves, depends on which attachments we receive. A picture will be shown differently than GPS coordinates. Maybe the coordinates should be implemented into the message body, while the image will be shown by a button, as figured in the above study task.
  - **Document:** Document the different options that are found relevant for the solution of the task, but was excluded due to complexity or because it was a bad alternative.
- **Wireshark study:** Do a study with Wireshark and network traffic of our application.
  - **Learn Wireshark:** Figure out how Wireshark works, and learn to use it.
  - **Document Wireshark:** An important part of the Wireshark study is to document how Wireshark works, the reason behind using Wireshark in this project and what results we get from it. Create statistics on the gathered results.
  - **Analyse traffic:** Find out how the data from all the traffic from our application should be analyzed. How much is regular package data that always will flow when sending a message, how much of it is the content, and how much is data that we don't need to send, e.g. what data is unnecessary polling, if any?
  - **Discussion:** Do a discussion on the findings of the data gathering. We will not be able to do a conclusion yet, as we have not implemented sending of messages with pictures, videos etc. Document your thoughts.

- **Instant message:** The user should be able to send an instant message using only three touches.
  - **Find design solution:** Find out how to implement the instant message feature. Where should the instant message button be placed? What is the fastest solution? How should the send instant message window look like?
  - **Incorporate instant message button into GUI:** Get a working button in the GUI that takes the user to the instant message view.
  - **Create send instant message view:** Create a view that is to be used for sending an instant message, based on what is found to be the best design solution.
- **Delete message:** The user should be able to delete a message that is received.
  - **Delete from local storage:** Implement the response of deleting a message locally whenever a user wants to delete the message.
  - **Delete from mail server:** Implement the response of deleting a message from the mail server whenever a user wants to delete the message.
  - **Update GUI:** Implement the response of removing a message from the gui whenever a delete operation is completed.
- **GUI-Issues:** Revisions of the GUI based on input from Thales.
  - **Menu:** The top menu bar must be made smaller by removing the text, and only use pictures.
  - **Header:** Remove the header saying XO-mail. It is not necessary.
  - **Change security labels to upper case:** All the security labels should be of the format CAPS_LOCK Upper case and underscore for spaces.

11.5 Sprint 3 - System design

After the completion of the third sprint we had arrived at a place where XOXOmail had most of the initial requirements implemented, or partially implemented. It presented itself in the following manner:

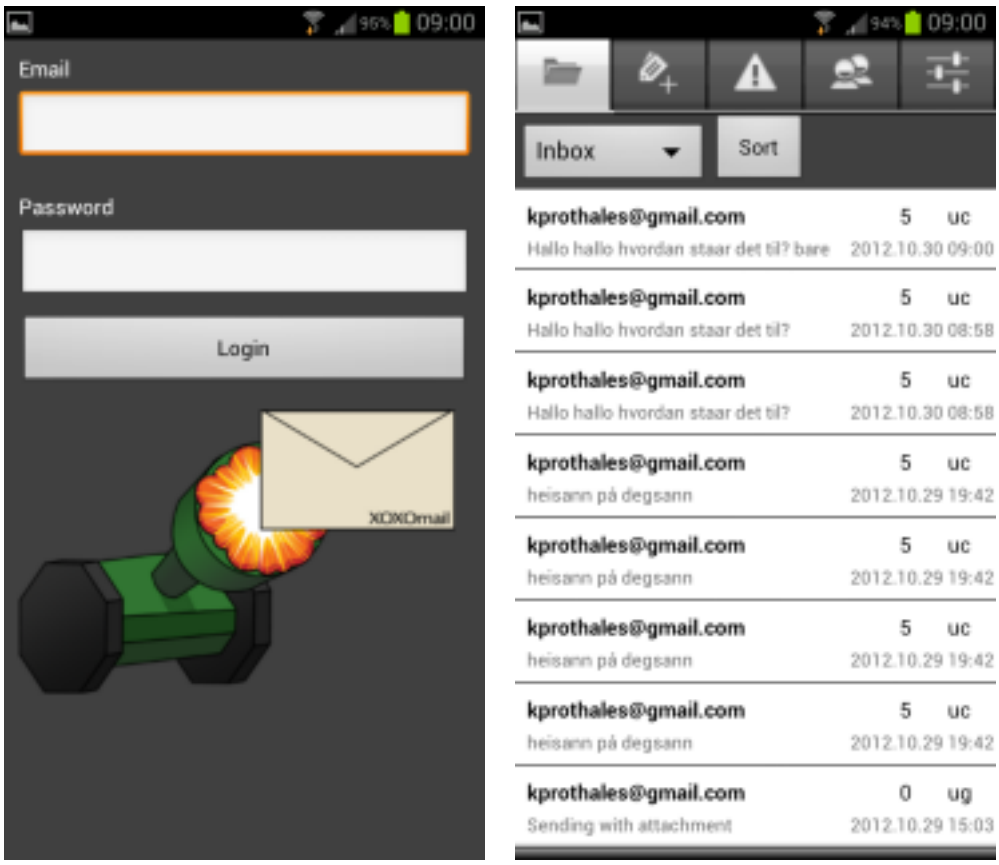


Table 11.2: Login and inbox GUI

At first the user is prompted with a login screen where he has to type in his or her username and password in order to get access to the application features. If the username and password is correct he/she is brought to the tabbed section of XOXOmail, more specifically the inbox tab. From the inbox tab the user is presented with a list of messages which he can sort after different criterias. He also has the ability to change between the inbox view and the sent messages view. If the user touches one of the messages displayed he is brought to the message view. See figure 11.2 on page 141.

Figure 11.2: Message screen view

From the MESSAGE view the user can see the message he selected. The title, message text, label, priority and type is shown to the user. From this view he can reply to the message, forward it and switch between received messages. See Figures in table 11.3 on page 142.

If the user presses the second tab icon he is taken to the send message view. From here he can add a recipient either by entering the address manually or pressing the contacts icon and selecting a contact from a list that appears. He then types in a subject, selects label, priority and type. He also gets the ability to add an attachment to the message he is sending.

When having pressed the third tab icon the user is brought to the instant message view. He he has the ability to send a message with predefined receiver, label, priority and type in a convenient way.

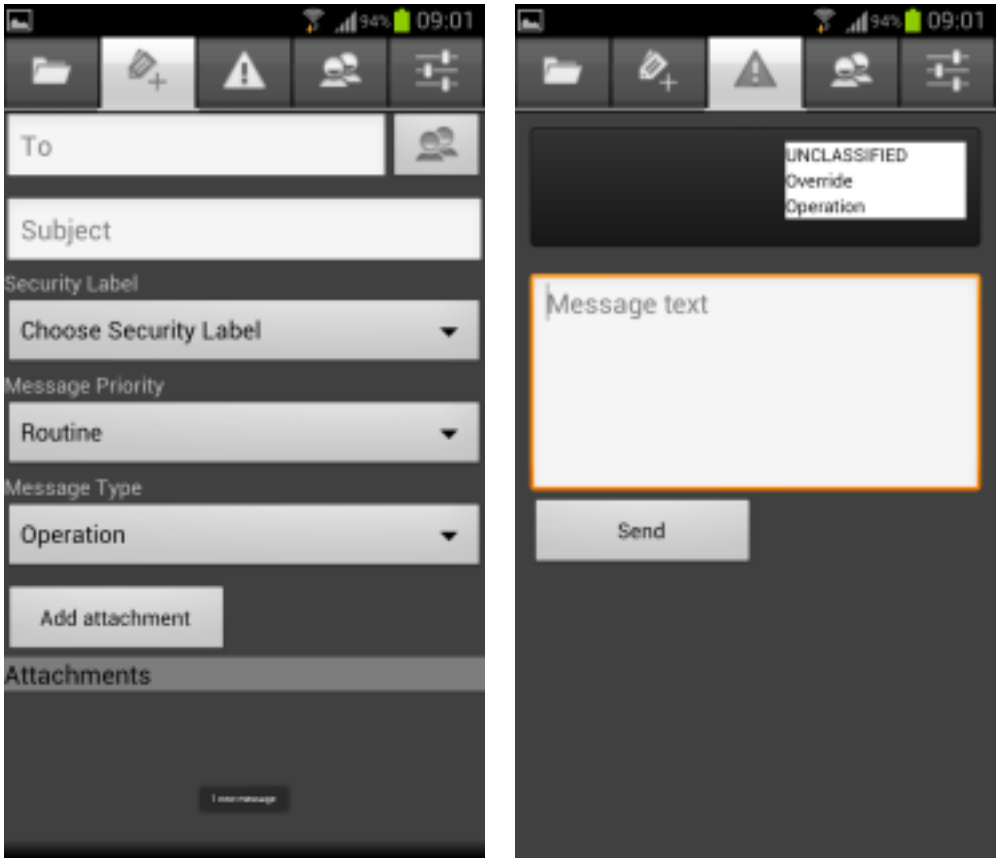
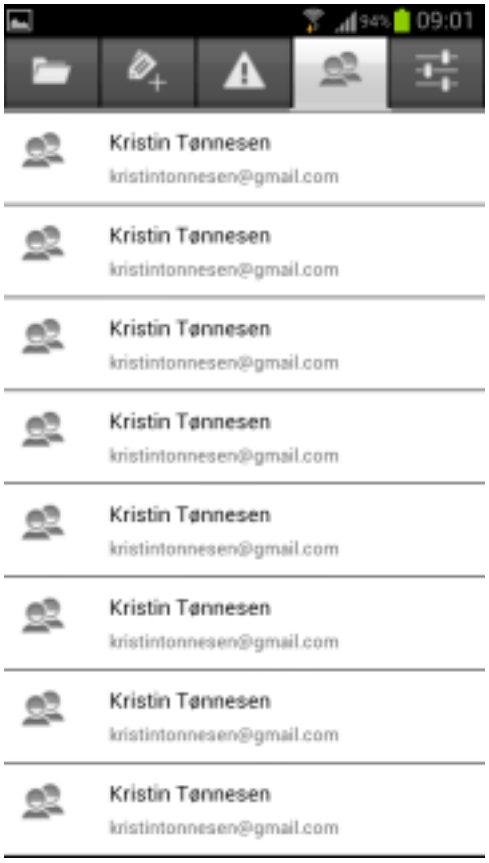


Table 11.3: Send message and Instant message GUI

If the fourth tab is selected the user is now viewing the contacts. This is the only functionality of this tab.

When the fifth and final tab icon is selected the settings screen is showed to the user. From here the user can select the message retrieval strategy (either push or pull) as well as the poll interval for the pull implementation. He can also select the security labels that are presented to when sending a new message. The instant message standard fields are also selected from the settings menu along with some location data settings.

The application has changed quite a bit since sprint two, it now has a more complete feel to it and new vital functionality has been added. See the firgures in table 11.4 on page 143.



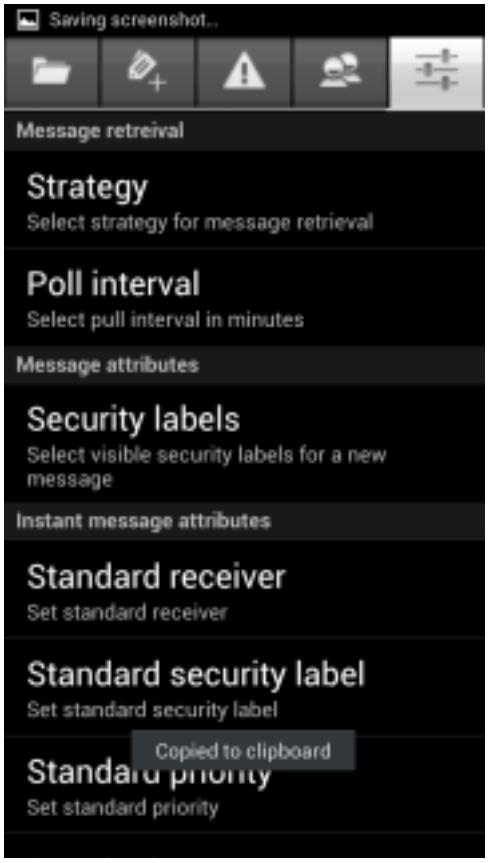


Table 11.4: Contacts and Settings GUI

11.6 Sprint 3 - Customer feedback

The customer was contented with the result of sprint 3. The most important features were to send and receive messages to and from Thales' system, and the customer was very happy that this worked seamlessly. They were also satisfied with the sending of an attachment to their system.

Thales was a little disappointed with the fact that receiving a message with an attachment from Thales' server failed, and wanted this to be fixed before the end of the project. Fortunately, this turned out to be a small issue as it was just to include support for another content type of the messages. Thales also wanted us to change the mapping of the priorities to the correct abbreviations that conform to Thales' standard.

Thales' representatives promised to test the application and report back about bugs, so there might be issues to come here.

11.7 Sprint 3 - Effort

See table 11.6 below.

Group no: 15 Date: 8/10-28/10					
Activity	**Start**	**W8** 8/10-14/10	**W9** 15/10-21/10	**W10** 22/10-28/10	**Activity sums**
Management	**E:0** A:0	**E:0/0** A:9.5/9.5	**E:2/2** A:2/11.5	**E:0/2** A:0/11.5	**E:2** A:11.5
Lectures	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:15/15** A:6/6	**E:15** A:6
Planning	**E:34** A:0	**E:34/34** A:31.5/31.5	**E:34/68** A:10.5/42	**E:25/93** A:16.5/58.5	**E:93** A:58.5
Pre study	**E:2** A:0	**E:2/2** A:1.5/1.5	**E:12/14** A:18/19.5	**E:2/16** A:2/21.5	**E:16** A:21.5
Requirement	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Design	**E:7** A:0	**E:7/7** A:15.5/15.5	**E:7/14** A:40.5/56	**E:7/21** A:46/102	**E:21** A:102
Implementation	**E:37** A:0	**E:37/37** A:23/23	**E:57/94** A:36.5/59.5	**E:33/127** A:30.5/90	**E:127** A:90
Documentation	**E:40** A:0	**E:40/40** A:75/75	**E:8/48** A:8/83	**E:38/86** A:39.5/122.5	**E:86** A:122.5
Evaluation	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Demonstration	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Period sums	E:120 **A:0**	E:120/120 **A:156/156**	E:120/240 **A:115.5/271.5**	E:120/360 **A:140.5/412**	E:360 **A:412**

Table 11.6: Table for effort registrations in sprint 3

Period comments:

- **W8:**
  - **Management:** We forgot to estimate some of the weekly work that had to be done, so therfore there were used some more effort then planned.
  - **Documentation:** Some of the group members had a lot to do in another class, so they postponed some of the documentation tasks for later.
- **W9:**
  - **Pre study:**Some of the topics that we were studying this week was a lot harder to learn than expected.
- **W10:**
  - **Lectures:**The lecture was a bit shorter than announced, and not all group members participated.

Activity comments:

- **Planning:** The sprint planning did not take as much time as in the other sprints. This time we agreed on the tasks in shorter amout of time.
- **Design:** It was some more design work than we thought, and a lot of the tasks that we thought was implementation turned out to be design instead.
- **Implementation:** Some of the implementation tasked turned out to be design tasks instead.

11.8 Sprint 3 - Conclusion

After finishing the third sprint we ended up with an application that satisfied most of the initial requirements presented by Thales. We were reasonably pleased with the way the application worked both at the backend and frontend. We considered XOXOmail user friendly and with a solid backend that supported modifiability.

During the sprint we had many programmatic problems to solve, some of them were solvable in a convenient way, but some ended up being time consuming and difficult to solve. Even though a tasks could be simple in theory there, was a chance that bugs and unforeseen problems arised. An example is when a group member tried to fetch gps coordinates from the Android framework. This was initially functioning without any noticeable problems, but after testing it on different phones it was revealed that it did not function on all the Android devices we had available. This and other similar issues caused us to exceed the time planned on some of the tasks.

There was a lot of work to be done in this sprint and due to the fact that programming tasks can be hard to give a time-span we ended up spending more time than we planned. The end result however, was more or less what we pictured when we started the sprint and we were pleased by the way it performed both visually and functionally. We ended up in the place we wanted to be before the final sprint.

CHAPTER 12

SPRINT 4

12.1 Sprint 4 - Planning

Completing the third sprint left us with a functional application that we could use to demonstrate the potential of a future product. Since we now had completed most of the coding planned for this project we could switch the focus towards completing the report and start preparing for the presentation to be held at the end of the project span.

After a meeting with Thales where we gave them a quick demonstration of the application, we received some feedback on 3 areas that needed minor additions. These included the ability to receive messages properly from Thales' system, taking a picture and sending it directly from outside the application and changing the priority listing of messages in the inbox from numbers to the equivalent letter code. We also agreed that after completing these 3 additions the application prototype was finished and that there was no room for further development.

The tasks to be done in the fourth sprint evolved highly around the report and the final presentation. The report contained a lot of areas that needed additional information and/or figures and some sections were still to be written. Since the presentation is an important part of the this project we also needed to dedicate a reasonable amount of time to the planning of the presentation. There was also room to perform some product testing outside of the project group consisting of user test-cases that needed a run-through. We realized that this sprint consisted of many tasks that had a great variation in size and duration. We therefore assigned the tasks on the go to whoever was available and most suited at that time.

After this sprint; the report, application and presentation needed to be more or less finished since we only had 3 days after the end of sprint to make the final adjustments. We realized that this final sprint would be tough and require some more hours put in than the prior sprints. This was needed in order to end up with a complete solution.

12.2 Sprint 4 - Duration

The fourth sprint started with a planning session on October 29th and ended on the 18th of October.

12.3 Sprint 4 - Goal

The fourth sprint had a pretty clear goal: complete the project. This involved many tasks of various size and span:

- Some programming to make the application meet Thales' requirements.
- Some additional documents needed to be written.
- Various small alterations through the entire report needed to be done.
- Prepare for the final presentation.

Some of the tasks were pretty clear and could be done in a given time span, but mosts of the tasks were pretty unclear and could basically never be completed in the available time. The report could always be improved and same goes for the presentation. To resolve this we agreed to put in as much work as we could, and be satisfied with the result we ended up with.

12.4 Sprint 4 - Ordered sprint backlog

- **Report work:** The report that is to be delivered at the end of the project needs a lot of work, and we have decided that we will use time on the report in every sprint.
- **Meetings:** Arranging meetings and the time we use on meetings.
  - **Meetings with Thales:** We have weekly meetings with Thales so that we can get rapid feedback on what we do. To know if they agree with our decisions and that we haven't misunderstood the task thay have given us.
  - **Meetings with Mohnsen Anvaari:** We have regular meetings with our advisor so that he can give us feedback on how we are doing our work, and make sure that we do what are expected of us in the course.
  - **Internal meetings:** Every time we meet we first have a status meeting were we share what we have done and how far we have come with our tasks.
  - **Lectures:** There are some lectures during the semester, and we are adviced to paticipate in these. In this sprint there are 2 lectures and courses that we have decided to attend.
- **Meeting and agenda document writing:** Write all the meetings agendas and minutes using the right template.
- **Receiving message with attachment:** As a user I should be able to receive a message with an attachment and to see the attachment.
  - **Implement:** Implement showing attachments based on what was found out in the study. What this task involves, depends on which attachments we receive. A picture will be shown differently than GPS coordinates. Maybe the coordinates should be implemented into the message body, while the image will be shown by a button, as figured in the above study task.

- **Instant message:** The user should be able to send an instant message using only three touches.
  - **Incorporate instant message button into GUI:** Get a working button in the GUI that takes the user to the instant message view.
- **Delete message:** The user should be able to delete a message that is received.
  - **Update GUI:** Implement the response of removing a message from the gui whenever a delete operation is completed.
- **Evaluation:** Write an evaluation of the project.
- **Presentation:** Make a presentation to show for the examinator.

12.5 Sprint 4 - System design

After the completion of the fourth sprint XOXOmail was fully functional with most of the initial requirements from Thales implemented. To see how the final product presented itself please see the conclusion in chapter 14.

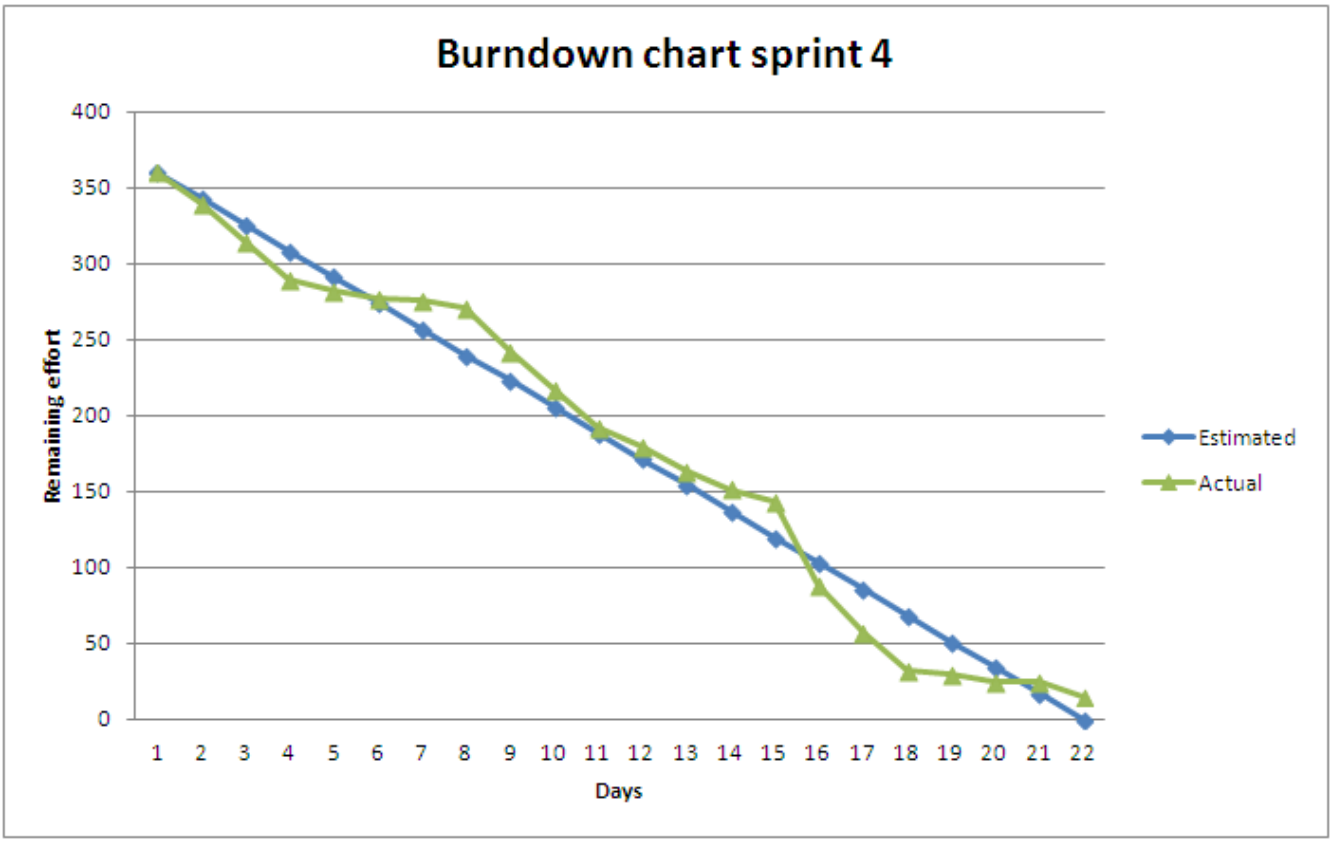


Figure 12.1: Burndown chart sprint 4

12.6 Sprint 4 - Customer feedback

At the end of sprint 4 we had completed a preliminary presentation. We thought it was beneficial to perform this presentation for Thales to get some feedback on it. Thales had gathered not only the representatives from Thales for this presentation, but also a group of other employees that was interested in seeing the result of our project. The presentation went well and it was clear that the representatives from Thales was pleased with the result. A demonstration of XOXOmail connecting to Thales' systems was also performed, and was a welcoming sight to the additional employees from Thales that were present.

The overall feedback from this meeting was positive and we got a few pointers on the presentation. There were no other inquiries from Thales' representatives.

12.7 Sprint 4 - Effort

See table 12.2 on page 150

Group no: 15 Date: 29/10-18/11					
Activity	**Start**	**W11** 29/10-04/11	**W12** 05/11-11/11	**W13** 12/11-18/11	**Activity sums**
Management	**E:0** A:0	**E:0/0** A:1/1	**E:0/0** A:0/1	**E:0/0** A:0/1	**E:0** A:1
Lectures	**E:9** A:0	**E:9/9** A:7/7	**E:0/9** A:0/7	**E:0/9** A:0/7	**E:9** A:7
Planning	**E:** A:0	**E:15/15** A:15.5/15.5	**E:17/32** A:15/30.5	**E:16/48** A:18/48.5	**E:48** A:48.5
Pre study	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Requirement	**E:0** A:0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0/0** A:0/0	**E:0** A:0
Design	**E:3** A:0	**E:3/3** A:28/28	**E:5/8** A:16.5/44.5	**E:0/8** A:2/46.5	**E:8** A:46.5
Implementation	**E:2** A:0	**E:2/2** A:16.5/16.5	**E:0/2** A:6/22.5	**E:0/2** A:2/24.5	**E:2** A:24.5
Documentation	**E:81** A:0	**E:81/81** A:72/72	**E:71/152** A:124.5/196.5	**E:61/213** A:85.5/282	**E:213** A:282
Evaluation	**E:10** A:0	**E:10/10** A:0/0	**E:7/17** A:6/6	**E:3/20** A:1/7	**E:20** A:7
Demonstration	**E:0** A:0	**E:0/0** A:0/0	**E:20/20** A:0/0	**E:40/60** A:40.5/40.5	**E:60** A:40.5
Period sums	E:120 **A:0**	E:120/ **A:140/140**	E:120/120 **A:168/308**	E:120/240 **A:149/457**	E:360 **A:457**

Table 12.2: Table for effort registrations in sprint 4

Period comments:

- **W11:**
  - **Lectures:** Not all the people in the group attended the lecture.
  - **Evaluation:** We did not have the time to start the evaluation work yet, so we postponed it to next week.
- **W12:**
  - **Documentation:** We decided to to some more implementation this week, so there were less time for the documentation.
  - **Demonstration:** We postponed the demonstration work to next week.

- **W13:**
  - **Documentation:** It became a lot documentation work this week because of the implementation choise last week.
  - **Evaluation:** Evaluation was easier to write than expected.

Activity comments:

- **Management:** We did not think it would be necessary to do any management this week, so we did not plan for it.
- **Design:** The design tasks we were finishing this sprint turned out be be more time consuming than planned.
- **Implementation:** There became some more implementation work than expected.

12.8 Sprint 4 - Conclusion

After the completion of the fourth and final sprint we had a product that met the requirements from Thales and presented itself in a manner that we could be proud of. We also reached a point where the report had a complete feel to it with the elements we thought important were included.

While performing the fourth sprint we worked rather sporadically with a lot of different tasks resulting in a sort of "chaotic" progression. This was more or less the only way to get things done because of the variation of the tasks. Even though we had to work in a somewhat different manner with regards to prior sprints, we managed to keep the workload under control and complete most of the tasks set for the sprint. The work distribution in this sprint was based on who that was most suited to perform the tasks as we needed tasks to be completed fast and with quality. Some of the group members were given tasks that were larger and more demanding than other tasks, but there were a lot of smaller tasks as well that could be distributed to remaining group members. This approach worked rather well and we completed most of the goals we had set.

This sprint had a bigger workload then the sprints performed in the past and and required us to put in some extra effort in order to finish it in a respectable way. We had some problems estimating the tasks because they varied so much. This resulted in some extra time being spent on tasks that were underestimated. We managed to finish all of the larger tasks included in this sprint and most of the minor ones. This left us in a solid position at the end of the sprint. We had finished the report and application as well as being ready to present our project. We were pleased with how this sprint ended up.

CHAPTER 13

CHANGELOG

2012.10.31 Revised product backlog, removing elements no longer relevant to us.

2012.10.08 Renamed BL18 'Extends communication interface' to 'Networkservice, 2.iteration'

2012.09.24 Inserted new backlog elements BL6 to BL11, to incorporate a shift in focus for sprint 2. Updated goal for sprint 2, removed goals for sprint 3 and 4. Sprint planning for sprint 2 added.

2012.09.12 Rewrite of document. Changed from taskdescriptions to stories in backlog, added storypoints for each entry.

2012.09.06 Goals added to each sprint. BL5 (Message template) moved down in backlog as requested from customer.

2012.09.04 Sprint planning for sprint 1 added

2012.08.28 Backlog pdf created

Part III

Conclusion & reflection

CHAPTER 14

CONCLUSION

This chapter describes the final state of the XOXOmail application, as well as suggestions for further development of the application. The chapter starts off with section 14.1 that describes how the XOXOmail looked and behaved when the implementation was finished. Section 14.2 lists and discusses the final tests results, as well as describes what improvements we made from the usability testing results. Section 14.3 discusses the possible ways one could improve and extend the application. Section 14.4 details how we have fullfilled the different functional and non-functional requirements. Section 14.5 contains a conclusion of this chapter.

14.1 System overview

This section will give an overview of the XOXOmail application after the implementation ended. It starts of by discussing the architecture, continues to discussing the user interface of the application and concludes with discussion the combined functionality of the application.

14.1.1 Architecture

The final architecture is very much like the planned architecture and the full description of the architecture can be found in chapter 7.

14.1.2 Graphical user interface

The final version of the graphical user inteface is described here by showing the most important screens from the application, as well as the reason behind the most important choices made. The evolution of the user interface during the different sprints is documented in the sprint chapters.

The tab navigation system
The choice of a main form of navigation fell on a tabbed system, as seen in figure 14.1 on page 155. The initial thought behind using this form of navigation was that we got an easy way of navigating around the different views and a quick view of the available views. At a later point we figured out that the tab navigation is supposed to be used for switching between different views of the same type, which is not the case in our application. We also ran into some problems with the built-in Android tab component. Unfortunately this was at a point too late to conduct any drastical changes, and we think that the solution is well enough suited for this prototype.

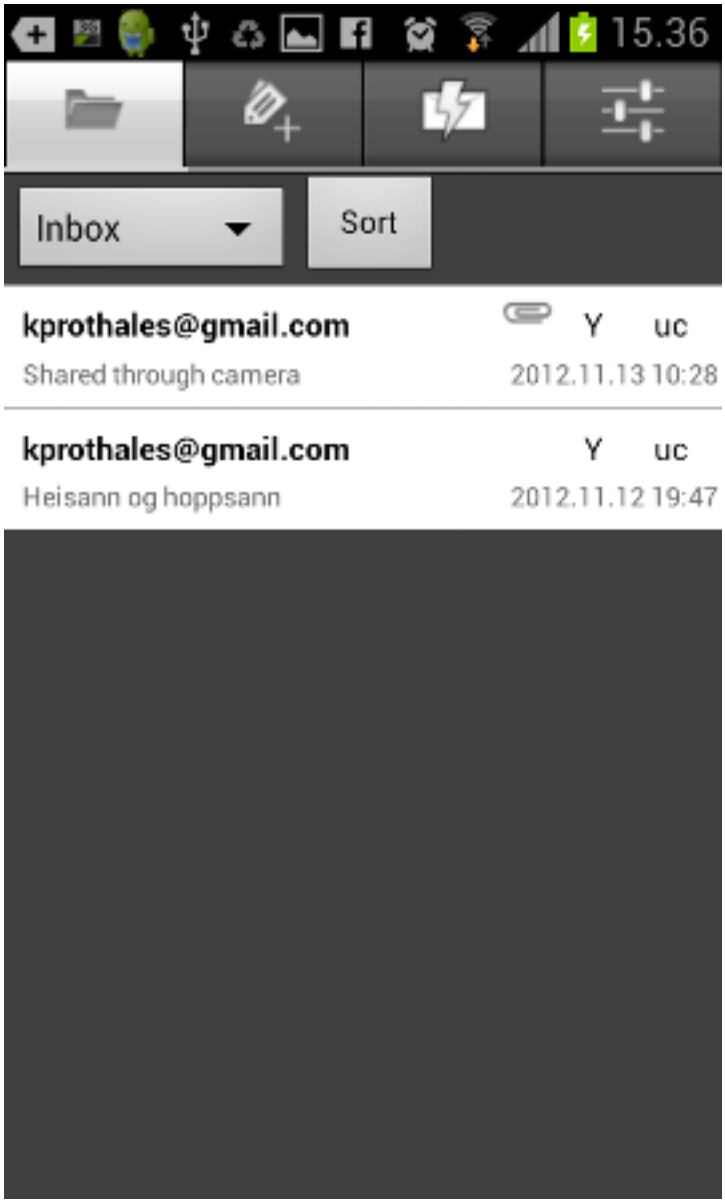


Figure 14.1: Inbox screen view

The inbox

The inbox folder and sent messages folder are composed in an identical way. We used a list where we specified a custom layout of each message item in the list, as seen in figure 14.1 on page 155. All the information on each item was requested by Thales, and we tried to make it look like a the standard way of listing mails in an inbox. We used abbreviations for the security labels (e.g. the Z and R) and priorities (e.g. r and uc) as specified by Thales because of the limited space of each message item. We also added the common clip icon to inform the user that the message has files attached.

Messages

If the user clicks an item in the message list he is sent to the view for showing the full information of the message, as seen in figure 14.2 on page 156. We chose to list the military messaging attributes in a separate box to easy give an overview of them. We originally had a more elaborate plan as to show the attachments, but ended up with a button the opens a dialog box as this was easier to implement, and we were short on time. The common choices of replying to and forwarding a message, as well as buttons with arrows to browse between the messages are placed at the bottom of the screen.

Figure 14.2: Message screen view

Sending a message

The view for sending messages is broken down into the logical part of a message conforming to the MMHS standards, as shown in figure 14.3 on page 157. The selection of military attributes are realized by the use of the Android component spinner, which from a behavioural aspect looks like a drop-down menu. As the diligent reader may have noticed, there is no button for sending the message. This is however just because view is bigger then the screen, and scrolling down to the bottom of the view would reveal the button.

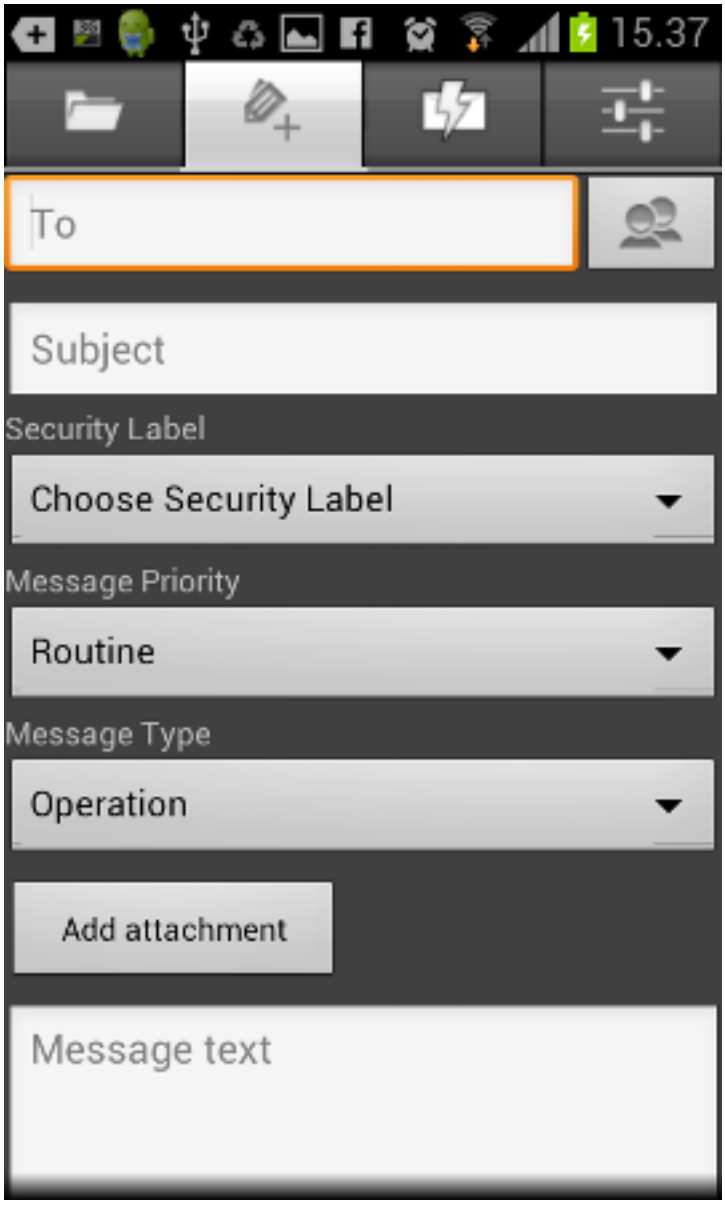


Figure 14.3: Sending message screen view

Instant message
The requirement for instant messages was that it should be possible to send an instant message by using three or less user actions. We thought about different ways of doing this, but ended up with the solution that is shown in figure 14.4 on page 158. One has to set the standard attributes for the instant message in the settings, thus requiring a couple of actions before being able to send an instant message, but after this step one can send instant messages with no more than three clicks (e.g. push the instant message tab, enter text and push send).

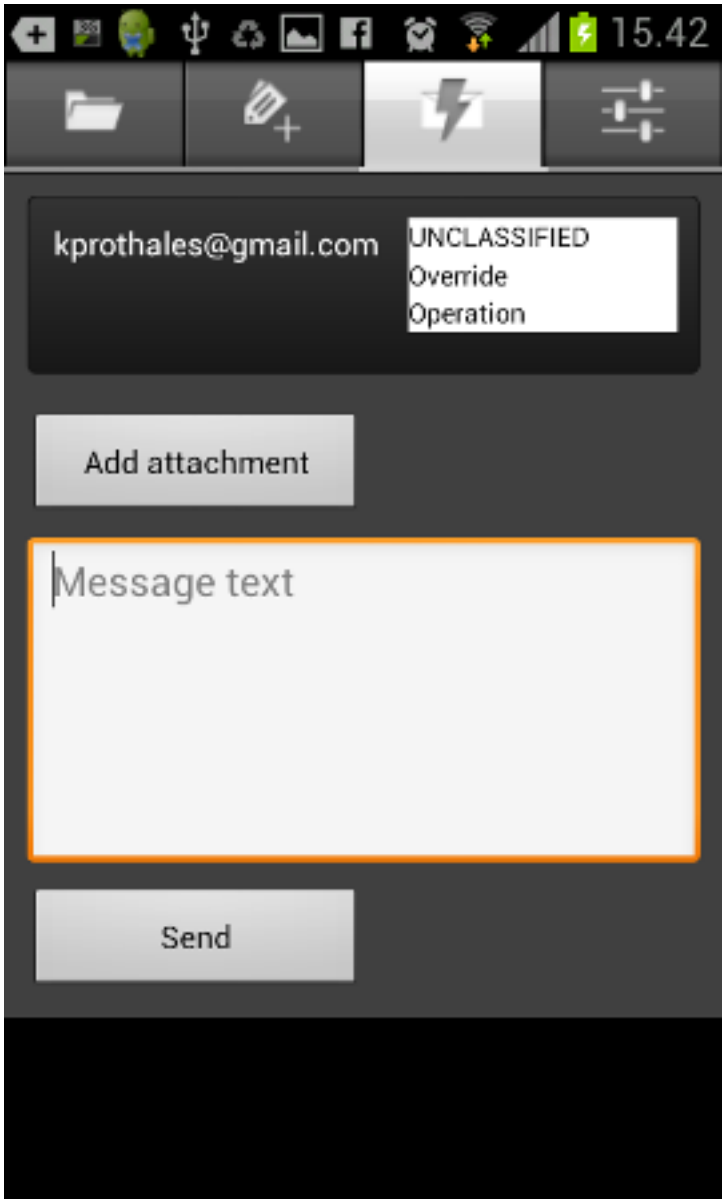


Figure 14.4: Sending instant message screen view

14.1.3 Functionality

The XOXOmail application has a quite extensive functionality, although we did not manage to implement all functional requirements. The functional requirements that were not implemented or finished are listed below:

- Address book - partly implemented (FR5)
- Video attachments - not implemented (parts of FR7)
- Delete message - partly implemented (parts of FR8)
- Message status - not implemented (FR11)
- Search - not implemented (FR14)
- Signing of messages - partly implemented (FR15)

14.2 Testing

This section will start by looking at the results from the functional testing and discuss these results, and then move on to discussing the results from the usability testing and the improvements we made from the feedback we received from the usability testing.

14.2.1 Functional testing

Results
As stated in chapter 6, we tested the application continuously after implementing each functionality. However, a summary of the test results from the end of the project can be found in table 14.1 on page 160.

Test case ID	Test name	Result
1	Login	OK
2	Send regular message	OK
3	Send message to contact from the address book	OK
4	Send full message	OK
5	Sent messages folder	OK
6	Read and browse messages	OK
7	Send attachments (camera)	OK
8	Send attachments (gallery)	OK
9	Send attachments (GPS)	OK
10	Attachments received	OK
11	Instant message settings	OK
12	Message retrieval strategy settings	Failed
13	Security labels settings	Failed
14	Send instant message	OK
15	Receive flash/override message	OK
16	Send instant message with attachments	GPS failed, images OK
17	Receive instant message outside the application	OK
18	Widget for instant message	OK
19	Reply	OK-
20	Forward	OK-
21	Delete	Failed
22	Delivery report and receipt notification	Failed
23	Status of delivery report and receipt notification	Failed
24	Sort messages	OK
25	Search in inbox	Failed
26	Login incorrect input	OK
27	Receiver incorrect input	OK
28	Security label incorrect input	OK

Table 14.1: Functional test result summary

Discussion
The results from the final functional testing showed that 19 out of the 28 test cases passed the test, six failed and three almost passed. Out of the six that failed, three were due to not having time to start with the implementation (test cases 22, 23 and 25) and three were known bugs that we did not get the time to fix (12, 13, 21). The three that almost passed were issues that we know how to fix but had not seen until the testing was done (16, 19, 20).

14.2.2 Usability testing

Results
The results from the tests can be found in table 14.2 on page 161. These show that three of the goals were OK and two failed. One of the goals (4) that was not fullfilled probably happened because we did not count seconds on the tests, just whole minutes. Overall, we were happy with these results, even though we did not expect the application to crash as often as it did, as this did not happen when we tested it ourselves.

Goal	Description	Status	Comment
1	The user should not spend more than 5 minutes on a task	OK	-
2	The application should not crash during the usability tests	Failed	The application crashed in the settings task on almost all tests
3	The users should not make more than 1 error during the tasks	OK	One user sent a regular message instead of an instant message
4	The users should solve task 6 faster than task 1	Failed	Two users spent the same amount of time, but this could also be due to the fact that the test leaders here just counted whole minutes
5	The average SUS score should be more than 70	OK	The average SUS score was 78.5

Table 14.2: Usability test - Test results

SUS scores
The SUS results are shown in table 14.3 on page 162. The average SUS score was 78.5.

Question/Test	**1**	**2**	**3**	**4**	**5**
1. I think that I would like to use this system frequently	3	3	4	4	3
2. I found the system unnecessarily complex	2	2	2	1	2
3. I thought the system was easy to use	4	4	4	5	4
4. I think that I would need the support of a technical person to be able to use this system	1	2	1	1	1
5. I found the various functions in this system were well integrated	3	3	3	5	5
6. I thought there was too much inconsistency in this system	2	2	2	3	1
7. I would imagine that most people would learn to use this system very quickly	4	3	4	4	4
8. I found the system very cumbersome to use	3	2	2	1	1
9. I felt very confident using the system	4	4	4	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	1	1	1
Score	**72.5**	**67.5**	**77.5**	**87.5**	**87.5**

Table 14.3: Usability test - SUS scores

Task/Time	**1**	**2**	**3**	**4**	**5**	**Average**
1	1 min	1 min	1 min	1 min	1 min	**1 min**
2	3 min	3 min	2 min	2 min	2 min	**2.4 min**
3	3 min	4 min	4 min	5 min	5 min	**4.2 min**
4	1 min	5 min	2 min	3 min	4 min	**3 min**
5	1 min	1 min	0 min	2 min	3 min	**1.75 min**
6	1 min	2 min	1 min	2 min	2 min	**1.6 min**
Sum	**10 min**	**16 min**	**10 min**	**15 min**	**17 min**	**14 min**

Table 14.4: Usability test - Task times

Time spent
A summary of the time spent on the different tasks is shown in table 14.4 on page 162.

Observation forms
The filled out observation forms can be found in appendix C.

Summary
There were many problems that reoccured in the test results. Below is a short summary of what were the problems:

- The testers were annoyed that the application does not remember login information
- Some were confused by odd choice of words
- The testers were annoyed that tilting of the phone sets you back to the inbox
- Almost all complained that the instant message and settings tab icons were not intuitive and spent some time finding the right tab
- Some complained about small fonts and small toasts
- Many were annoyed that the text boxes behaved weird, e.g. that the letters were not capitalized after a period and the keyboard did not always close.
- The settings were unstable and the application stopped during the task that tested the settings
- Many of the testers complained about the lack of confirmation after actions in the application
- Some testers commented that the user interface was not pretty enough

Discussion
Even though we did not have the time to make major changes and improvements to the user interface and functionality, we learned a lot from these usability tests about how to better do things in later projects.

Redesign
We were able to do some minor changes in the user interface. The changes are listed below, although some of the screenshots other places may not be updated accordingly.

- Hopefully more intuitive icon for instant message
- Implemented bigger difference between read and unread messages in the inbox folder
- Text fields give you capitalized first letter of sentences
- Rephrased some text, e.g. "username" instead of "email" and "Take picture" instead of "Image from camera"

14.3 Further development

Our XOXOmail application is, even though it contains a lot of the functional requirements stated in chapter 4, just a prototype and can hardly be called complete. Most of the advanced security features are lacking, or at best partly implemented. We managed to fix the most critical bugs, but the application can still be a bit unstable. There are of course a number of things we would have improved if we had more time with this project.

14.3.1 Functionality

The obvious way to go with the further development would be to start with the lacking features. Thales wanted the application to be able to fetch updated address books from their server, and here one would have to implement support for management messages and distinguish these from regular messages. Video attachment turned out to be more complex than image, but should in theory be doable. Deleting of messages was worked on, but due to some problems with the Android file system we did not have the time to complete this feature. Message status (delivery report and receipt notification) was not even started on, as we anticipated this to be a task too complex with the time resources we had available. This would include using IMAP attributes for synchronizing between our application and the server. Search functionality should in theory be straightforward to implement, but this was not prioritized.

14.3.2 Graphical user interface

We would have wished to improve the user interface further and make a more "fancy" and pretty application. In hinsight we realize that our expectations of what we would be able to create were a bit high. After these weeks with the Android framework we have just learned the basics and only in the last few weeks starting to realize how to customize components and layouts to make them more pretty than the Android default style. Functions that could highly improve the look and feel of the application, like being able to "swipe" between messages instead of pushing buttons, were planned, but due to the time constraints these were not implemented. If we could have started over, we might have chosen to use a higher Android version, as there are many components that exist only from version 3 or 4 that we could have needed.

14.3.3 Signing and verification

The biggest problem on the security side right now is that we have no support for signing or verification of the identity of the mail sender. The code for doing so has been written, but has not been integrated or tested. For a company that already has experience with security programming, this should not be significantly complicated, but ended up being a lot more time consuming than we anticipated.

14.3.4 Encryption and decryption

The current version of the application has encryption in the form of a hash function for storing of the passwords and SSL/TLS encryption on the network channel. We had originally planned to do disk encryption of the application's private storage as mentioned in chapter 7, but unfortunately we ran short on time here.

14.4 Fulfillment of requirements

14.4.1 Functional quality requirements

All of the functional requirements listed in chapter 4 on page 27, require two things from the application and architecture, a frontend graphical user interface for the user to interact with, and backend service providing the frontend with data and functionality. The table 14.5 shows these two components for every functional requirement.

Req. ID	Frontend components	Backend components
FR1	LoginActivity	PersistenceService
FR2	SendMessageActivity	NetworkService
FR3	FoldersActivity	PersistenceService
FR4	FoldersActivity	PersistenceService
FR5	ContactsActivity	PersistenceService (not implemented)
FR6	SendMessageActivity	NetworkService
FR7	SendMessageActivity, MessageViewActivity	NetworkService, PersistenceService
FR8	MessageViewActivity	PersistenceService
FR9	InstantMessageActivity	NetworkService
FR10	SettingsActivity	Android
FR11	Not implemented	Not implemented
FR12	FlashOverrideMessageActivity	NetworkService
FR13	FoldersActivity	PersistenceService
FR14	Not implemented	Not implemented
FR15	Not implemented	Not implemented

Table 14.5: Fulfillments of functional requirements

14.4.2 Non-functional quality requirements

Usability

In order to insure that our application was simple to use, we worked closely with Thales when designing our GUI. The design went through several iterations based on their comments and concerns. Towards the end of the development cycle, when most of the required back end components were in place, we tested the usability of the application by letting friends try their hands at it and noting down their comments. This gave us a lot of data on how we could make further adjustments to fit with our goals.

Security

Accessing locally stored data outside of application

"No data exposed to the user or application, as long as the user do not have root access." S1 - Quality requirements

This requirement is fulfilled by the service implementing the PersistenceService interface, see section 7.2.2 on page 92, which keeps track of all the models currently in use in the application, in the case of XOXOmail these models will mainly consist of messages sent and received in the application, but also the users themself have a model representation. If the malicious user or application does not have root access, then the data will be secure just by using Android provided features like sandboxing processes and private storage. We can not however guarantee that this will always be the case, and another layer of security is therefore needed.

There are three possible scenarios when looking at this requirement. First off, a malicious user /application finds a locked/encrypted device, where the owner is not logged in to the device. In this case, the android hardware encryption will successfully mitigate any malicious intent. Secondly, a malicious user/application find an unlocked device, but the owner is not logged into the XOXOmail. Under normal circumstances, without an additional layer of security, the malicious user could gain access to the application and application data, circumventing the Android private storage access control due to the users root privileges. In order to mitigate this, all XOXOmail specific data will be encrypted before stored into the private storage section. Thirdly, a malicious user/application find an unlocked device, where the owner is logged into the XOXOmail application and the application is open. In this case the malicious user is given some possible ways of exploiting the application by sending predefined instant messages. But it would still require the malicious user to type in the owner password in order to browse any other messages.

Trying to use application with wrong privileges

"No features exposed to the user - stopped by login screen" S2 - Quality requirements

This requirement is fulfilled by the ServiceProvider along with the graphical user interface. There are two possible scenarios when looking at this requirement. First off, the service backend is not started and a user starts the application. In this case no privileges will be given until the user types in the correct username and password. In fact, the application itself does not have the possibility to decrypt any data without this information. Second, the service is started, but not open. In this case a malicious user could send predefined flash messages, but not view any messages, or get access to the system as a whole.

Trying to access the apps external data trafic

"No useful data exposed to the user" S3 - Quality requirements

This requirement is fulfilled by the third party library, JavaMail, which ensures a secure communication channel to the mail server. JavaMail will setup a SSL/TLS connection, mitigating the possibility of looking into the data traffic generated by the application.

Performance

Latency

"With a latency of maximum 3 seconds, the user should be able to read the message after it is received. This is the latency when we subtract the download time of message, which is dependent of the network connection." P1 - Quality requirements

This requirement is fulfilled by the IMAPIdle module, figure 7.10, which keeps an open connection from the device to the IMAP server. The IMAP server will not respond to the IDLE request until a new message is received or a message is deleted. Once the client receives an answer from the server it can determine the correct course of action and issue a new IDLE request, causing it to go back into a waiting stage[77].

14.5 Summary

In this chapter we have looked at what features the XOXOmail application does and does not have, as well as describing options for further development. Finally, we looked at how the requirements in the project were fulfilled by the different parts of our code.

Although we did not manage to implement all the features we planned and wanted, we are quite satisfied with the application. There are obviously ways to improve and extend the application, but we think that we have implemented many of the most important features and the customer seemed quite satisfied with the final results.

CHAPTER 15

REFLECTION

This chapter is a reflection part that will describe what went well and what we could have done better in the project. Section 15.1 lists the different tools we have used as well as reflections on what these tools gave us of benefits and problems. Section 15.2 will give our thoughts on the teamwork and how we have worked to reach the goal. Section 15.3 will reflect on the relation and communication with Thales. Section 15.4 will describe our relation and communication with our advisor. Section 15.5 reflects on what lessons we have learned from the project. Section 15.6 gives suggestions for improvements of the course.

15.1 The tools we used

Tool	**Purpose**	**Way we used the tool**
Gmail	Group communication	We used Gmail for sending important messages to all the group members by using a Gmail mail-group.
Facebook	Group communication	We used Facebook for fast communication about less important issues, such as where to work and small questions regarding a task or a document written by someone else.
Google Docs	Cooperative editing of documents	We started using Google Docs to make it easy to simultaneously edit different report documents. It worked pretty well, and a great thing with it is that many people can change one document at the same time. After a short period of time, we discovered that it also has a built-in version control.
DropBox	File sharing	We decided to use Dropbox for picture sharing, since sharing pictures is not that easy in Google Docs. In Dropbox we made a folder that all our group members had access to, and then we synced it.

GitHub	Revision control	We used Git for revision control of all our code and Latex documents. This tool worked great throughout the entire project and enabled us to cooperate efficiently when writing code and Latex documents.
TeXWork	Document writing	We used TeXWork for writing the final report. We collected all our documents in a folder and included them all in a main file. It was pretty self explanatory, which was great since only one in our group had used it before.
NetBeans	IDE	We choose NetBeans as our IDE in this project and it worked without any significant problems during the entire project span. We were pleased with the services offered by this IDE.
Jira	Effort and tasks management	We decided to use Jira for making tasks, assigning tasks and follow the tasks lifecycle. We also used Jira for effort registration. Jira was a bit time consuming, and we unfortunately used more time at making tasks than necessary the first two sprints. But after using it for some time, Jira turned out to function pretty good. The effort management in Jira was very good all along.
Android Emulator	Application testing	Slow, but functional. Accompanied by testing on actual phones.

15.2 Internal Evaluation

The learning curve of the project has been steep for all of us. Fortunately, we became a lot more efficient in the last part of the project than we were in the beginning. The agreed work methodology was scrum, and only a few of us were familiar with this method. Especially the estimating of the effort needed to complete each task during the planning phases was hard, but we got increasingly better also at this point. One could say that we stumbled a bit before we found our stride, and this section is the record of our improvement. We have all contributed to this chapter, to better reflect our different experiences and contributions.

15.2.1 Teamwork

While the Norwegian school system provides its students with a near infinite supply of group projects, there is always more to learn about teamwork and group dynamics. Especially as few school projects are as long and comprehensive as this project has been. Clocking in at over three months and counting for half our grade this semester, this project is of respectable size. In addition, it covers the full process of a workplace project far closer than a typical project, providing valuable experience on the required paperwork.

We started off the project by talking through the background of each group member and their ambitions for the project. Some had their minds set on the best grade possible, whereas other would settle with a mediocre grade. We also discussed each person's experience with project work and technology. Based on these discussions we carved out the different roles needed and assigned these to the people that were most interested in the different roles. In retrospect, we realized that we could maybe have waited with delegating these roles and included more people in each work area. From this we might have distributed work loads better and given more persons a feel of what was going on in the different work areas.

There has been a lot of trial and error during the course of this project. None of us had any real experience with leading projects of this size. As the first week went by, the project leader was unofficially selected. We agreed that the project leader would not be the boss, but have the responsibility of keeping the group together and resolve disputes between members. At first the leader position had a practical manager role; arranging room reservations, making choices on important issues, being customer and advisor contact and delegating tasks. What we saw during the first weeks was that not all delegations of tasks were as good as they should be. The first Document responsible was not really interested in having this role, so a new one was chosen. After a few meetings we saw that it was more natural that Agenda and Minutes of meetings responsible had the direct customer contact, to avoid the flow of connection going through an extra person. A responsible for booking rooms was also appointed.

We have had a lot of discussions in the during the project, and a lot of these were constructive. However, some of the group members did talk about their frustration and irritation without the person in question present. We discussed this and agreed to always broach issues with the person in question first and then in plenum if needed. The group leader became increasingly better at discovering when people had problems and asking them what was bothering them. The approach was to try to discover and solve the problems before they got too big. We also struggled a little with the digital communication, as some of the group members were available on Facebook and mail, whereas others prefered SMS or a phone call. In urgent cases this could create communication problems.

We were a group consisting of six very different people, both regarding work ethics and personality. This probably made the group better, but the different ambitions and willingness to put effort into the project created a lot of tension and was frustrating for some of the group members. Some people easily took responsibilities and ended up with a lot of work in that responsibility area. Some of the group members were more passive and often needed others to tell them what to do. This trend contunied through the entire project, but people increased their efforts as we started the 3rd and 4th sprint. The differences in efforts spent has to a certain extent made some discord in the group.

The differences in personality and morale quickly became evident, and manifested themselves often when deadlines came closer and the pressure became higher. However, we feel that the group as a unit has managed these disputes in an acceptable fashion. The tools we chose turned out to cause certain problems, and there was only one group member that had experience with the tools. This caused everyone coming to this person with technical questions and probably strangling this person's productivity. In retrospect, we do however feel like the stress was worth it, and that the tools did in fact return with profit.

15.2.2 Reaching the project goal

The goal of this project was to provide Thales with a working prototype and, less importantly for them, a set of documents describing the prototype and why we built it as we did. We came a long way in reaching this goal. The prototype contains the most important requirements and the documentation is comprehensive. Even so, we had a number of difficulties getting to this point.

We started to work on the project shortly after the first introductiry lecture and quickly agreed to adopt the Scrum methodology as our work ethic. In the starting phase of the project we started to work on the application almost straight away. We quickly realized as the project went on that we should halt this part of the project for a while, and focus more on the planning part. This was beneficial in the later phases of the project and gave us a better overview of what had to be done.

At the start of the first sprint we used a lot of time on planning and setting up the programming environment. We saw that we were to finish the project we had to do programming and documentation tasks simultaneously and continuously. During the first sprint it was a lot of trial and error. No one knew exactly how the flow of management should be, so the sprint plan was as good as it could be at this point.

At the start of the second sprint we were able to create a much better sprint plan. We used the backlog and corresponding stories to create tasks and estimates. This made it more clear for all group members what needed to be done during this sprint.

What became clear during the next sprints was that a lot of the estimates we created were way too low. We always tried to create estimates as accurate as possible based on the story points, but due to the lack of experience these were bound to be wrong. In a real business environmen this would be a disaster for the company, but in our case we just had to spend more hours completing tasks and hence not having time to implement everything we wanted. We had to put a lot of extra effort into each sprint to be able to finish what we had planned. This extra effort has strongly contributed to finishing the project with a good result.

There was a noticeable progress in how the group cooperated and communicatd as the project went on. Routines and work methods became more or less second nature, and increased the productivity of the group. When the project came close to the end, we ended up with an application prototype that contained most of the features we had planned to implement, and a report that contained most of the elements we wanted to include.

15.3 Customer relations

We were lucky to have a customer that had participated in this kind of project before and so had a very good idea of what to expect. They were careful to communicate clearly with us about what they wanted out of the the project. They were helpful and understanding about the difficulties involved and offered possible solutions whenever we ran into an obstacle.

15.3.1 Relations

We worked well with our customer, agreeing on most issues. We generally had a good idea of where the project was headed and what they felt about our current progress. There were a few hiccups, but we generally functioned well. The group dynamic between us and Thales was good from the start.

15.3.2 Communication

Communicating efficiently can be one of the most difficult things in the world, and this project was no exception. We had weekly meetings planned with the customer, but in practice we ended up having meetings only every other week. This was mostly due to the fact that we did not have enough questions to require weekly meetings.

That aside, the communication with Thales has been good and fruitful. As the representatives of Thales have a programming background, we often got into very technical discussions. For many of our group members this was from time to time a bit tedious as they could not follow the discussion. Still, this gave all of us a technical insight and everyone got a bit more involved in all parts of the application. It has never been a problem understanding what Thales wanted, and they have been very understanding and good at restraining themselves on what tasks we should implement. They have been very clear on what features that they wanted for the demo and what we should just skip because it had no relevance to what they were curious about in a prototype.

15.3.3 The project

This was an interesting project, quite outside our areas of expertise. Though most of us had worked with the Android platform before, the worlds of security and communications were all new to us. It took a bit of effort to familiarize ourselves with the required protocols and standards.

A lot of us were surprised that the task would be so comprehensive, as it all seemed so big and insuperable at first. As we dug deeper into the task, we saw that this was an interesting task that was really hands on. We all felt relieved that it was a project in which it was easy to see the path ahead, as we all knew what an email client is. It was all about creating a specialized version of one.

15.4 Advisor relations

This section will elaborate about the relations we had with our advisor.

15.4.1 Relations

The relation with the advisors was not as extensive as the one we had with the customer. It was more about pointing us is the right direction with tips and general guidelines for performing such a project.

15.4.2 Communication

The communication with the advisor has not been without problems. The first few weeks we had just some short meetings of 10-15 minutes. This was all we needed because we were working with a general plan for the whole project.

Unfortunately, we did not hear from our advisor for about 1,5-2 weeks because he did not receive our mails because of a mail-forwarding problem. This resulted in one advisor meeting being canceled. We tried to call him, but were not able to reach him.

After some time we saw that we could have benefited from having more guidance from the advisor. We did not know what we could expect from him, so we sent a mail to course responsible, Reidar Conradi, and he said that he wanted to take part in the next two weeks sessions we had with our advisor.

These two weeks we got a lot out of the meetings. We got a fully reviewed version of our report with proposals on structure changes, and we were happy that we got thoughts and suggestions on how to improve the report. After these weeks we also got more out of the meetings with our initial advisor, as we had learned more about what we could expect from him.

15.4.3 Supervision

The supervision of our project was in the starting phases not as extensive as it could have been. This was mostly because of a misunderstanding involving how the meetings with the advisor were to be executed. The group was quite passive in the first few meetings and we had not prepared enough questions and things we were unsure of. After having a couple of meetings we realized that we needed to be more active and thus resulting in much more advice being given.

15.5 Lessons learned

This section will discuss what lessons we have learned from this project and what we might have done differently if we were to start a new project. These lessons come in addition to the experience we have learned from using the technical tools and programming experience.

15.5.1 Setting up our tools

In this project we have used an extensive array of tools, most of them software-based. Getting this set of tools to work on six different computers and three separate operating systems (not counting variations in version) created a lot of work for the one group member who had experience with this.

We lost a large number of man hours in this setup phase, and it might not have been worth the increased efficiency we got out of the toolset. In a longer project, with a more experienced team, it certainly would have been, but things are more ambiguous in our case. It will however probably help us in the long run, as many of us will probably use those tools again.

On the other hand, these tools were useful for us; they handle everything from importing outside classes, to testing and version control. There is a lot of later work that would continue to bog us down that we have been saved by having more trouble in the beginning. It was probably worth it, but the technical responsible who spent the first three weeks helping us all set up did probably not always feel that way.

15.5.2 Group communication and effort

At the end of the first sprint we had a discussion on group effort and time use, spurred on by an uneven distribution of labor in the group. Over the course of the three week sprint, we have managed to arrive in a situation where the top two workers in the group had worked an average of twice the hours of the bottom two workers. This clear mismatch of commitment was thoroughly discussed over a couple of days.

We learned a lot from those two days. We agreed on a revised work schedule, better taking into account our individual ability to invest into the project. By getting everyone to commit to an individual weekly minimum hours of work, the average work per week went up significantly, even as members stopped feeling pressured to work more than they were capable of.

This illustrated the need for communication within the group. When some people feel pressured into working more than they are comfortable with while others feel disheartened because they have a perceived higher workload than the rest, it is clear that resentment will start to grow on both sides. By discussing it early, before tensions had gotten too high, we saved ourselves from facing a far greater challenge later.

However, while the problem has been thoroughly discussed, the underlying cause is still there. As can be seen from the work log of the later sprints, there is still a large deviation in hours of work for various group members. It remains to be seen if this will affect group morale as crunch time approaches.

15.5.3 Project organization

The main thing we should have done differently in this project is that we should have performed the pre-study more thoroughly at an earlier state in the project span. We are not sure if this could have saved us any time, but it would make the final product much more clear at an earlier stage in the development and maybe some of the programmatic choices we made along the way could have been done in a better way. We also did the mistake of starting the sprints at Mondays and then not having the customer meetings until Wednesday. The hours in between were spent partly on fixing things from the previous sprint and partly on trying to plan ahead for the newly started sprint, although this was not always easy as we had discussed it with Thales.

15.6 Suggestions for course improvements

In any subject there are possibilties for improvement. There are always things that can be done better, clearer or more efficiently. In this section we will give our thoughts on this matter.

Firstly, we think that the course could give better guidelines as to what to expect from the advisor. We had no experience in having these kinds of meetings, and in the beginning the meetings often were not so helpful.

Secondly, we would have appreciated if the lectures had taken place in the original course time, as this would have made it easier to plan.

Thirdly, we have discovered that some of the deadlines mentioned in the compendium were not always correct and probably lagging from previous year.

We would also have liked to have a fixed meeting room for all our meetings. Having a room with a board that we could use as a scrum board would have helped us a lot.

GLOSSARY

ACID Atomicity, consistency, isolation, durability.

AES Advanced Encryption Standard.

AIDL Android Interface Definition Language.

Ant Java library to build packages.

API Application programming interface.

ASCII American Standard Code for Information Interchange.

BouncyCastle Java Cryptography Library.

CA Certificate Authority.

CBC Cipher-Block Chaining.

COTS Commercial off-the-shelf.

CPU Central Processing Unit.

CSS Cascading style sheets, RFC 2318.

CVS Concurrent Versions System.

dpi Dots per inc, a measure of screen density.

ESSIV Encrypted salt-sector initialization vector.

ESSIV:SHA256 Encrypted sector|salt initial vector where sha256 is a hash algorithm.

Git Distributed version control system.

GitHub Web-based hosting service.

GUI Graphical User Interface.

ID Identification/Identity/Identifier.

IDE Integrated development environment.

IDLE An integrated development environmentfor Python.

IEEE829 The 829 Standard for Software and System Test Documentation.

IMAP "Internet message access protocol" defined by RFC 3501.

IMAPIdle An optional expansion of the IMAP email accessing protocol that allows the server to send new message updates to the client in real time.

IP "Internet protocol" defined by RFC 791.

IPC Inter-process communication.

IPSec Internet Protocol Security.

IT Information Technology.

JNDK Java Native Development Kit.

JSON JavaScript Object Notation.

Junit A unit testing framework for the Java programming language.

JVM Java Virtual Machine.

K-9 Mail Evaluation criteria - open source email client for Android.

LDAP "Lightweight Directory Access Protocol", defined by RFC 4511.

Maven A build automation tool typically used for Java projects.

MB Megabyte.

MIME "Multipurpose Internet Mail Extensions", defined by RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 and RFC 2049.

MMHS Military Message Handling System, described by RFC6477.

NATO North Atlantic Treaty Organization.

NDK Native Development Kit.

NetBeans An open-source software development project with an active community of collaborating users and developers.

OCSP "Online Certificate Status Protocol", defined by RFC 2560.

OS X A series of Unix-based graphical interface operating systems.

PBE Password Based Encryption.

PBKDF2 HMAC-SHA1 Password-Based key Derivation Function 2.

POP3 "Post Office Protocol, version 3" defined by RFC 1081.

R2Mail2 An email client for Android OS.

S/MIME "Secure/Miltipurpose Internet Mail Extensions", defined by RFC 3369, RFC 3370, RFC 3850 and RFC 3851.

SL4A Scripting Layer for Android.

SMTP "Simple Mail Tranfer Protocol" defined by RFC 5321.

sp Scale independent pixels.

SpongyCastle Repackage of BouncyCastle for Android.

SSL "Secure sockets layer" defined by RFC 6101 and RFC 5246.

SSL/TLS "Secure sockets layer/transport layer security" defined by RFC 6101 and RFC 5246.

Stanag 4406 The NATO standard for Military Messaging based on X.400.

SVN "SubVersion", centralized version control.

TCP "Transmission Control Protocol" first specified by RFC 675.

TLS "Transport layer security" defined by RFC 6101 and RFC 5246.

UI User Interface.

UML Unified Modeling Language.

VM Virtual machine.

X509Tools ...

XML Extensible Markup Language.

XSD XML Schema Definition.

XStream Parsers libraries and tools - a simple library to serialize objects to XML and back again.

BIBLIOGRAPHY

[1] XOmail. Retrieved 2012-09-12 from <http://www.xomail.com>

[2] Stanag 4406. Retrieved 2012-09-12 from <http://www.isode.com/solutions/military-messaging.html>

[3] Simple Mail Transfer Protocol. Retrieved 2012-09-12 from <http://en.wikipedia.org/wiki/Smtp>

[4] Internet Message Access Protocol. Retrieved 2012-09-12 from <http://en.wikipedia.org/wiki/Imap>

[5] Thales Norway AS. Retrieved 2012-08-28 from <http://www.thales.no/pub/sites/index.php?siteID=4&m

[6] Git & GitHub. Retrieved 2012-09-10 from <https://github.com/>

[7] http://developer.android.com/design/get-started/principles.html

[8] http://developer.android.com/design/patterns/pure-android.html

[9] http://www.xomail.com/dl/xomail-brochure.pdf

[10] Public Key Encryption. Retrieved 2012-09-18 from <http://en.wikipedia.org/wiki/Public_key>

[11] Digital signing. Retrieved 2012-09-18 from <http://en.wikipedia.org/wiki/Digital_signature>

[12] S/MIME. Retrieved 2012-09-18 from <http://en.wikipedia.org/wiki/S/MIME>

[13] Gmail and S/Mime. Retrieved 2012-09-23 from <http://www.scientificcomputing.com/is-your-e-mail-secure.aspx>

[14] R2Mail2. Retrieved 2012-09-23 from <https://play.google.com/store/apps/details?id=at.rundquadrat.an

[15] Waterfall Model. Retrieved 2012-09-23 from <en.wikipedia.org/wiki/Waterfall_model>

[16] Agile Development. Retrieved 2012-10-13 from <http://en.wikipedia.org/wiki/Agile_development>

[17] Kanban. Retrieved 2012-10-13 from <http://en.wikipedia.org/wiki/Kanban_(development)>

[18] Agile Software Development And Scrum. Retrieved 2012-09-10 from <http://www.mountaingoatsoftware.com/topics/scrum>

[19] Scrum (Development). Retrieved 2012-09-10 from <en.wikipedia.org/wiki/Scrum_(development)>

[20] Java (Programming Language). Retrieved 2012-09-10 from <http://en.wikipedia.org/wiki/Java_programming_language>

[21] Android Native Development Kit (NDK). Retrieved 2012-09-10 from <http://developer.android.com/tools/sdk/ndk/index.html>

[22] Scripting Layer for Android (SL4A). Retrieved 2012-09-10 from <http://code.google.com/p/android-scripting/>

[23] PHP for Android (PFA). Retrieved 2012-09-10 from <http://phpforandroid.net/manual/en/index/faq>

[24] Monodroid by Xamarin. Retrieved 2012-09-10 from <http://xamarin.com/monoforandroid>

[25] Extensible Markup Language (XML). Retrieved 2012-09-10 from <http://en.wikipedia.org/wiki/Xml>

[26] JavaScript Object Notation (JSON). Retrieved 2012-09-10 from <http://www.json.org/xml.html>

[27] Xstream. Retrieved 2012-10-27 from http://xstream.codehaus.org/

[28] Xstream. Retrieved 2012-10-27 from http://www.ibm.com/developerworks/java/library/x-xstream/index.html

[29] Kolawa, Adam; Huizinga, Dorota (2007). Automated Defect Prevention: Best Practices in Software Management. Wiley-IEEE Computer Society Press. p. 426. ISBN 0-470-04212-5

[30] Junit. Retrieved 2012-10-01 from http://en.wikipedia.org/wiki/JUnit

[31] Mock objects. Retrieved 2012-10-01 from http://en.wikipedia.org/wiki/Mock_object

[32] "Features and Motivations". Retrieved 2010-12-29

[33] Fowler, Martin (2007). "Mocks Aren't Stubs". Retrieved 2010-12-29.

[34] Greenmail. Retrieved 2012-10-01 from http://www.icegreen.com/greenmail/

[35] LaTeX. Retrieved 2012-09-25 from http://www.latex-project.org/intro.html

[36] Jira overview. Retrieved 2012-09-25 from http://www.atlassian.com/software/jira/overview

[37] Jira. Retrieved 2012-09-25 from http://en.wikipedia.org/wiki/JIRA

[38] Greenhopper. Retrieved 2012-09-25 from http://www.atlassian.com/software/greenhopper/overview

[39] NetBeans. Retrieved 2012-09-25 from http://netbeans.org/features/index.html

[40] Eclipse. Retrieved 2012-11-19 from http://en.wikipedia.org/wiki/Eclipse(software)

[41] NetBeans IDE. Retrieved 2012-09-25 from http://en.wikipedia.org/wiki/NetBeans#NetBeans_IDE

[42] Service. Retrieved 2012-09-08 from http://developer.android.com/reference/android/app/Service.html

[43] AIDL. Retrieved 2012-09-08 from http://developer.android.com/guide/components/aidl.html

[44] IBinder. Retrieved 2012-09-08 from http://developer.android.com/reference/android/os/IBinder.html

[45] Techtarget. Retrieved 2012-09-25 from http://searchsecurity.techtarget.com/definition/link-encryption

[46] IPSec. Retrieved 2012-09-25 from http://stackoverflow.com/questions/3960802/implementing-ipsec-protocol-in-java

[47] TLS. Retrieved 2012-09-25 from http://en.wikipedia.org/wiki/Secure_Sockets_Layer

[48] NetBeans license. Retrieved 2012-10-08 from http://netbeans.org/about/legal/product-licences.html

[49] Android license. Retrieved 2012-10-08 from http://developer.android.com/license.html

[50] Andoid–plugin. Retrieved 2012-10-08 from http://code.google.com/p/maven-android-plugin/

[51] Maven license. Retrieved 2012-10-08 from http://maven.apache.org/license.html

[52] Junit license. Retrieved 2012-10-08 from http://www.junit.org/license

[53] GreenMail license. Retrieved 2012-10-08 from http://www.icegreen.com/greenmail/

[54] JavaMail-android. Retrieved 2012-10-08 from http://code.google.com/p/javamail-android/

[55] Xstream license. Retrieved 2012-10-08 from http://xstream.codehaus.org/license.html

[56] BouncyCastle license. Retrieved 2012-10-08 from http://www.bouncycastle.org/licence.html

[57] SpongyCastle, http://rtyley.github.com/spongycastle/

[58] MiKTeX license. Retrieved 2012-10-08 from http://miktex.org/copying

[59] Git license. Retrieved 2012-10-08 from http://git-scm.com/about/free-and-open-source

[60] Cygwin license. Retrieved 2012-10-08 from http://cygwin.com/licensing.html

[61] Apache 2.0 license. Retrieved 2012-10-08 from http://www.apache.org/licenses/LICENSE-2.0

[62] GPL-1.0 license. Retrieved 2012-10-08 from http://www.gnu.org/licenses/gpl-1.0.html

[63] GPL-2.0 license. Retrieved 2012-10-08 from http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

[64] GPL-2.0 w/Classpath Exception license. Retrieved 2012-10-08 from http://openjdk.java.net/legal/gplv2+ce.html

[65] MIT X11 license. Retrieved 2012-10-08 from http://opensource.org/licenses/mit-license.php

[66] CDDL-1.0 license. Retrieved 2012-10-08 from http://opensource.org/licenses/cddl-1.0

[67] CPL-1.0 license. Retrieved 2012-10-08 from http://www.junit.org/license

[68] BSD license. Retrieved 2012-10-08 from http://en.wikipedia.org/wiki/BSD_licenses

[69] IEEE829. Retrieved 2012-09-30 from http://gerrardconsulting.com/tkb/guidelines/ieee829/main.html

[70] Kruchten, Philippe (1995, November). Architectural Blueprints — The "4+1" View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50.

[71] Bass, Len; Clements, Paul; Kazman, Rick. Software Architecture in Practice, Second Edition.

[72] Android crypto implementation. Retrieved 2012-10-07 from http://source.android.com/tech/encryption/android_crypto_implementation.html

[73] Security. Retrieved 2012-10-07 from http://source.android.com/tech/security/index.html

[74] Android activity states. Retrieved 2012-10-13 from http://developer.android.com/reference/android/app/

[75] Activity life cycle. Retrieved 2012-10-13 from http://developer.android.com/guide/components/activities

[76] Android. Retrieved 2012-10-13 from http://developer.android.com/guide/topics/manifest/manifest-intro.html

[77] IMAP. Retrieved 2012-10-07 from http://www.isode.com/whitepapers/imap-idle.html

[78] Git Book. Retrieved 2012-11-05 from http://git-scm.com/book

[79] Wireshark. Retrieved 2012-11-19 from http://en.wikipedia.org/wiki/Wireshark

[80] Data Transport Layer Security. Retrieved 2012-11-19 from http://en.wikipedia.org/wiki/Transport_Layer_Security

[81] Compression Image. Retrieved 2012-10-15 from http://csunplugged.org/text-compression

[82] Data Compression. Retrieved 2012-10-16 from http://en.wikipedia.org/wiki/Data_compression

[83] Lossy data compression. Retrieved 2012-10-15 from http://en.wikipedia.org/wiki/Lossy_compression

[84] JPEG. Retrieved 2012-10-30 from http://no.wikipedia.org/wiki/JPEG

[85] PGF. Retrieved 2012-10-30 from http://en.wikipedia.org/wiki/Progressive_Graphics_File

[86] DV. Retrieved 2012-11-1 from http://en.wikipedia.org/wiki/DV

[87] MPEG. Retrieved 2012-11-1 from http://vsr.informatik.tu-chemnitz.de/~jan/MPEG/HTML/mpeg_tech.html

[88] Dirac. Retrieved 2012-11-2 from http://en.wikipedia.org/wiki/Dirac_(video_compression_format)

[89] VC-1. Retrieved 2012-11-2 from http://en.wikipedia.org/wiki/VC-1

[90] MP3. Retrieved 2012-10-30 from http://en.wikipedia.org/wiki/Mp3

[91] AAC. Retrieved 2012-10-30 from http://en.wikipedia.org/wiki/Aac

[92] MP3. Retrieved 2012-10-30 from http://en.wikipedia.org/wiki/Windows_Media_Audio

[93] Lossless data compression. Retrieved 2012-10-15 from http://en.wikipedia.org/wiki/Lossless_compressio

[94] Context Tree Weighting. Retrieved 2012-11-02 from http://en.wikipedia.org/wiki/Context_tree_weightin

[95] Context Tree Weighting Research. Retrieved 2012-11-02 from http://www.sps.ele.tue.nl/members/F.M.J.Willems/RESEARCH_files/CTW/ResearchCTW.htm

[96] Burrows Wheeler Transform. Retrieved 2012-11-02 from http://en.wikipedia.org/wiki/Burrows-Wheeler_transform

[97] LZ77. Retrieved 2012-11-02 from http://en.wikipedia.org/wiki/LZ77

[98] GIF. Retrieved 2012-11-02 from http://no.wikipedia.org/wiki/GIF

[99] GifSicle. Retreived 2012-11-02 from http://www.lcdf.org/gifsicle/

[100] PNG. Retreived 2012-11-03 from http://no.wikipedia.org/wiki/Portable_Network_Graphics

[101] corePNG. Retreived 2012-11-03 from http://de.wikipedia.org/wiki/CorePNG

[102] Animation Codec. Retreived 2012-11-19 from http://en.wikipedia.org/wiki/Animation_codec

[103] FLAC. Retrieved 2012-11-3 from 2012-11-03 http://en.wikipedia.org/wiki/MPEG-4_SLS

[104] MPEG-4 SLS. Retrieved 2012-11-03 from http://en.wikipedia.org/wiki/Free_Lossless_Audio_Codec

[105] Violet. Retrieved 2012-11-07 from http://alexdp.free.fr/violetumleditor/page.php

[106] UML. Retrieved 2012-11-07 from http://en.wikipedia.org/wiki/Unified_Modeling_Language

[107] Software Ideas Modeler. Retrieved 2012-11-07 from http://en.wikipedia.org/wiki/Software_Ideas_Mode

[108] Computer Networks written by Andrew S. Tanenbaum, 2003, Prentice Hall PTR

[109] RFC1122. Retrieved 2012-11-07 from http://tools.ietf.org/html/rfc1122

[110] Shneiderman, Ben; Plaisant, Catherine. Designing the User Interface, fifth edition.

[111] Fluid UI. Retrieved 2012-09-23 from https://www.fluidui.com/

[112] Black-box testing. Retrieved 2012-11-09 from http://en.wikipedia.org/wiki/Black-box_testing

[113] Black-box testing fundamentals. Retrieved 2012-11-09 from http://softwaretestingfundamentals.com/black-box-testing/

[114] Functional testing. Retrieved 2012-11-10 from http://en.wikipedia.org/wiki/Functional_testing

[115] System Usability Scale. Retrieved 2012-11-12 from http://en.wikipedia.org/wiki/System_usability_scal

[116] Jakob Nielsen. Retrieved 2012-11-12 from http://www.useit.com/alertbox/20000319.html

[117] Cobertura. Retrieved 2012-11-19 from http://cobertura.sourceforge.net/

[118] SUS avg. Retrieved 2012-11-19 from http://www.measuringusability.com/sus.php

[119] Observer pattern. Retrieved 2012-12-11 from http://en.wikipedia.org/wiki/Observer_pattern

Part IV

Appendices

APPENDIX A

WIRESHARK RESULTS

No.	Time	Source	Destination	Protocol	Length	Info
5270	117.397461	10.0.2.15	173.194.71.108	TCP	74	56311 > imaps [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=429949066 TSecr=0 WS=2
5271	117.425782	173.194.71.108	10.0.2.15	TCP	58	imaps > 56311 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
5272	117.426758	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=1 Ack=1 Win=5840 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5273	117.438477	10.0.2.15	173.194.71.108	TLSv1	142	Client Hello
5274	117.458985	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=1 Ack=89 Win=8760 Len=0
5275	117.493164	173.194.71.108	10.0.2.15	TLSv1	1514	Server Hello
5276	117.493164	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=89 Ack=1461 Win=8760 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5277	118.331055	173.194.71.108	10.0.2.15	TLSv1	299	Certificate, Server Hello Done
5278	118.331055	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=89 Ack=1706 Win=11680 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5279	118.379883	10.0.2.15	173.194.71.108	TLSv1	240	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
5280	118.500977	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=1706 Ack=275 Win=8760 Len=0
5281	118.531250	173.194.71.108	10.0.2.15	TLSv1	193	Change Cipher Spec, Encrypted Handshake Message, Application Data
5282	118.531250	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=275 Ack=1845 Win=14600 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5283	119.572266	10.0.2.15	173.194.71.108	TLSv1	94	Application Data
5284	119.572266	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=1845 Ack=315 Win=8760 Len=0
5285	119.603516	173.194.71.108	10.0.2.15	TLSv1	248	Application Data
5286	119.605469	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=315 Ack=2039 Win=17520 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5287	119.618164	10.0.2.15	173.194.71.108	TLSv1	116	Application Data
5288	119.780274	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=2039 Ack=376 Win=8760 Len=0
5289	119.859375	173.194.71.108	10.0.2.15	TLSv1	251	Application Data
5290	119.862305	10.0.2.15	173.194.71.108	TLSv1	94	Application Data
5291	120.028321	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=2236 Ack=416 Win=8760 Len=0
5292	120.095703	173.194.71.108	10.0.2.15	TLSv1	202	Application Data
5293	120.106446	10.0.2.15	173.194.71.108	TLSv1	97	Application Data
5294	120.208985	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=2384 Ack=459 Win=8760 Len=0
5295	120.276367	173.194.71.108	10.0.2.15	TLSv1	317	Application Data
5296	120.309571	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=459 Ack=2647 Win=26280 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5297	120.333985	10.0.2.15	173.194.71.108	TLSv1	111	Application Data
5298	120.454102	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=2647 Ack=516 Win=8760 Len=0
5299	120.780274	173.194.71.108	10.0.2.15	TLSv1	125	Application Data
5300	120.780274	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=516 Ack=2718 Win=26280 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5301	120.785157	10.0.2.15	173.194.71.108	TLSv1	127	Application Data
5302	120.850586	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=2718 Ack=589 Win=8760 Len=0
5303	120.894532	173.194.71.108	10.0.2.15	TLSv1	448	Application Data
5304	120.909180	10.0.2.15	173.194.71.108	TLSv1	108	Application Data
5305	121.146485	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=3112 Ack=642 Win=8760 Len=0
5306	121.214844	173.194.71.108	10.0.2.15	TLSv1	189	Application Data
5307	121.285157	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=642 Ack=3247 Win=29200 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5308	121.629883	10.0.2.15	173.194.71.108	TLSv1	109	Application Data
5309	121.629883	173.194.71.108	10.0.2.15	TCP	54	imaps > 56311 [ACK] Seq=3247 Ack=697 Win=8760 Len=0
5310	121.772461	173.194.71.108	10.0.2.15	TLSv1	134	Application Data
5311	121.773438	10.0.2.15	173.194.71.108	TCP	64	56311 > imaps [ACK] Seq=697 Ack=3327 Win=29200 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5312	121.778321	10.0.2.15	173.194.71.108	TLSv1	144	Application Data

Figure A.1: Traffic going to our application when recieving a single message

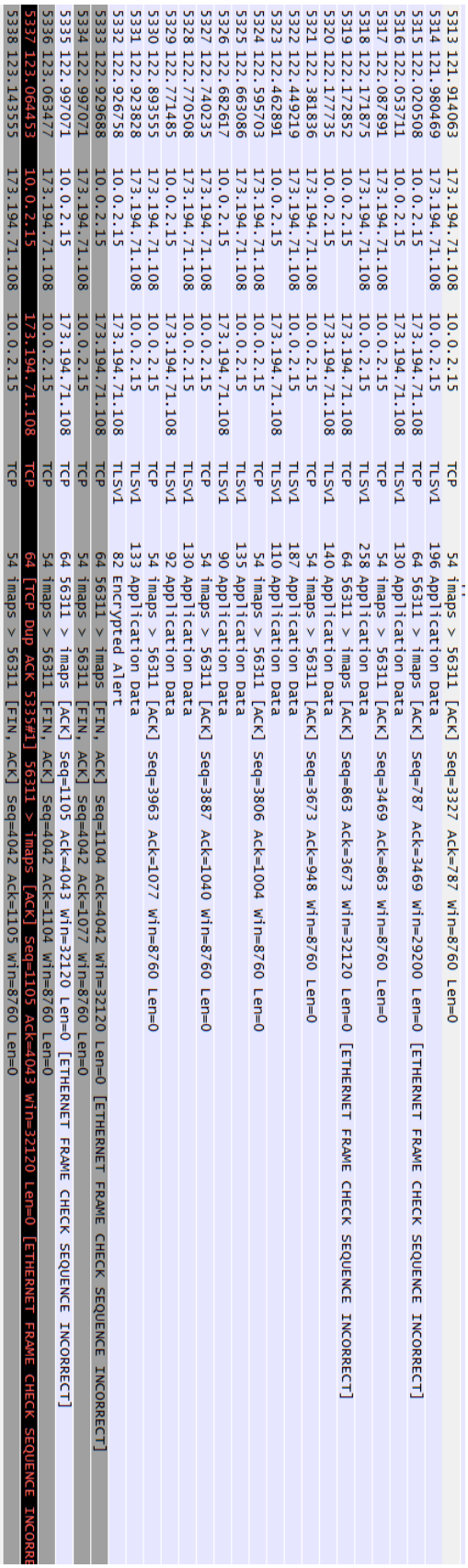
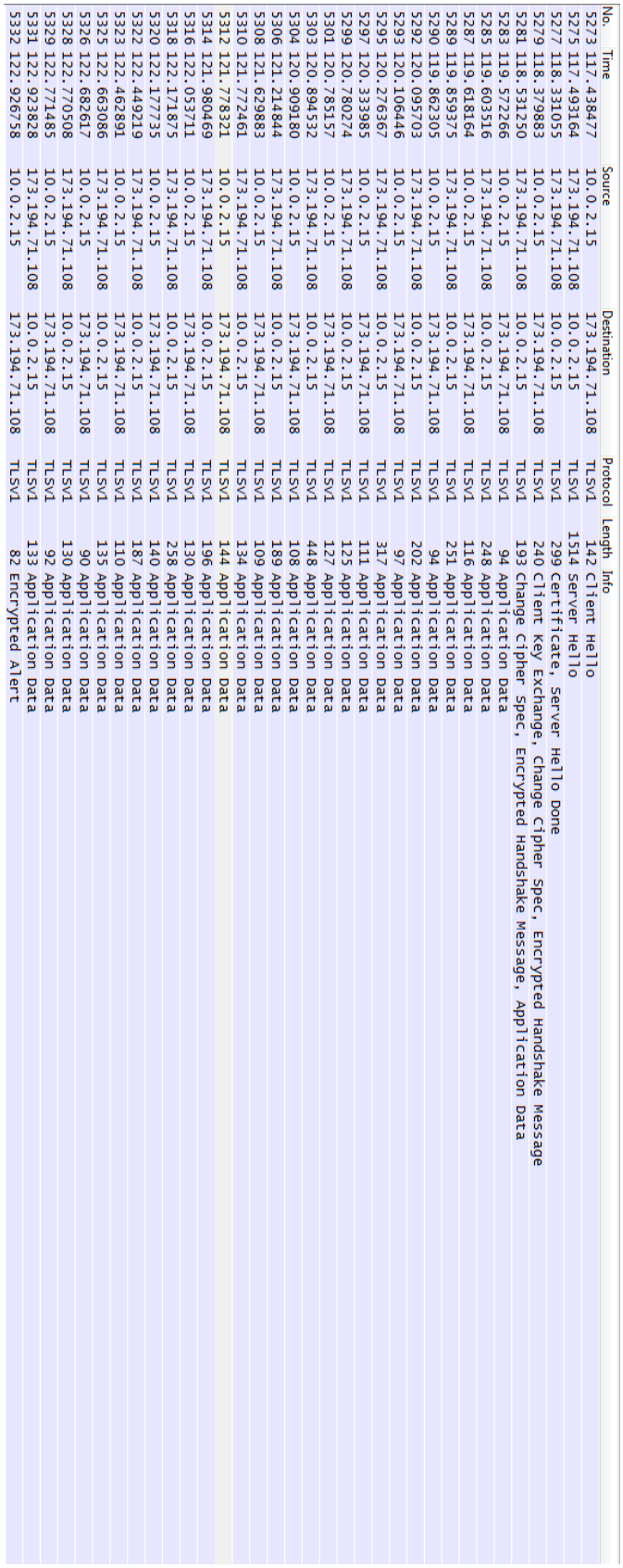


Figure A.2: Traffic going from our application when recieving a single message



No.	Time	Source	Destination	Protocol	Length	Info
5273	117.438477	10.0.2.15	173.194.71.108	TLSv1	142	Client Hello
5275	117.493164	173.194.71.108	10.0.2.15	TLSv1	1514	Server Hello
5277	118.331055	173.194.71.108	10.0.2.15	TLSv1	299	Certificate, Server Hello Done
5279	118.379883	10.0.2.15	173.194.71.108	TLSv1	240	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
5281	118.531250	173.194.71.108	10.0.2.15	TLSv1	193	Change Cipher Spec, Encrypted Handshake Message, Application Data
5283	119.572266	10.0.2.15	173.194.71.108	TLSv1	94	Application Data
5285	119.603516	173.194.71.108	10.0.2.15	TLSv1	248	Application Data
5287	119.618164	10.0.2.15	173.194.71.108	TLSv1	116	Application Data
5289	119.859375	173.194.71.108	10.0.2.15	TLSv1	251	Application Data
5290	119.862305	10.0.2.15	173.194.71.108	TLSv1	94	Application Data
5292	120.095703	173.194.71.108	10.0.2.15	TLSv1	202	Application Data
5293	120.106446	10.0.2.15	173.194.71.108	TLSv1	97	Application Data
5295	120.276367	173.194.71.108	10.0.2.15	TLSv1	317	Application Data
5297	120.333985	10.0.2.15	173.194.71.108	TLSv1	111	Application Data
5299	120.780274	173.194.71.108	10.0.2.15	TLSv1	125	Application Data
5301	120.785157	10.0.2.15	173.194.71.108	TLSv1	127	Application Data
5303	120.894532	173.194.71.108	10.0.2.15	TLSv1	448	Application Data
5304	120.909180	10.0.2.15	173.194.71.108	TLSv1	108	Application Data
5306	121.214844	173.194.71.108	10.0.2.15	TLSv1	189	Application Data
5308	121.629883	10.0.2.15	173.194.71.108	TLSv1	109	Application Data
5310	121.772461	173.194.71.108	10.0.2.15	TLSv1	134	Application Data
5312	121.778321	10.0.2.15	173.194.71.108	TLSv1	144	Application Data
5314	121.980469	173.194.71.108	10.0.2.15	TLSv1	196	Application Data
5316	122.053711	10.0.2.15	173.194.71.108	TLSv1	130	Application Data
5318	122.171875	173.194.71.108	10.0.2.15	TLSv1	258	Application Data
5320	122.177735	10.0.2.15	173.194.71.108	TLSv1	140	Application Data
5322	122.449219	173.194.71.108	10.0.2.15	TLSv1	187	Application Data
5323	122.462891	10.0.2.15	173.194.71.108	TLSv1	110	Application Data
5325	122.663086	173.194.71.108	10.0.2.15	TLSv1	135	Application Data
5326	122.682617	10.0.2.15	173.194.71.108	TLSv1	90	Application Data
5328	122.770508	173.194.71.108	10.0.2.15	TLSv1	130	Application Data
5329	122.771485	10.0.2.15	173.194.71.108	TLSv1	92	Application Data
5331	122.923828	173.194.71.108	10.0.2.15	TLSv1	133	Application Data
5332	122.926758	10.0.2.15	173.194.71.108	TLSv1	82	Encrypted Alert

Figure A.3: TLS packets sent to and from our application

No.	Time	Source	Destination	Protocol	Length	Info
5256	111.194336	10.0.2.15	173.194.71.108	TCP	74	56458 > urd [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=4294954020 TSecr=0 WS=2
5257	111.261719	173.194.71.108	10.0.2.15	TCP	58	urd > 56458 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
5258	111.263672	10.0.2.15	173.194.71.108	TCP	64	56458 > urd [ACK] Seq=1 Ack=1 Win=5840 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5259	111.281250	10.0.2.15	173.194.71.108	TLSv1	142	Client Hello
5260	111.349609	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=1 Ack=89 Win=8760 Len=0
5261	111.416992	173.194.71.108	10.0.2.15	TLSv1	1514	Server Hello
5262	111.417969	10.0.2.15	173.194.71.108	TCP	64	56458 > urd [ACK] Seq=89 Ack=1461 Win=8760 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5263	112.312500	173.194.71.108	10.0.2.15	TLSv1	299	Certificate, Server Hello Done
5264	112.312500	10.0.2.15	173.194.71.108	TCP	64	56458 > urd [ACK] Seq=89 Ack=1706 Win=11680 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5265	112.340820	10.0.2.15	173.194.71.108	TLSv1	240	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
5266	112.467773	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=1706 Ack=275 Win=8760 Len=0
5267	112.535156	173.194.71.108	10.0.2.15	TLSv1	170	Change Cipher Spec, Encrypted Handshake Message, Application Data
5268	112.545898	10.0.2.15	173.194.71.108	TLSv1	96	Application Data
5269	112.680664	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=1822 Ack=316 Win=8760 Len=0
5270	112.750000	173.194.71.108	10.0.2.15	TLSv1	224	Application Data
5271	112.753906	10.0.2.15	173.194.71.108	TLSv1	92	Application Data
5272	112.889648	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=1992 Ack=353 Win=8760 Len=0
5273	112.958008	173.194.71.108	10.0.2.15	TLSv1	97	Application Data
5274	112.960937	10.0.2.15	173.194.71.108	TLSv1	109	Application Data
5275	113.028320	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2035 Ack=408 Win=8760 Len=0
5276	113.095703	173.194.71.108	10.0.2.15	TLSv1	97	Application Data
5277	113.104492	10.0.2.15	173.194.71.108	TLSv1	102	Application Data
5278	113.171875	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2078 Ack=455 Win=8760 Len=0
5279	113.449219	173.194.71.108	10.0.2.15	TLSv1	99	Application Data
5280	113.456054	10.0.2.15	173.194.71.108	TLSv1	116	Application Data
5281	113.523437	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2123 Ack=517 Win=8760 Len=0
5282	113.589844	173.194.71.108	10.0.2.15	TLSv1	112	Application Data
5283	113.591797	10.0.2.15	173.194.71.108	TLSv1	115	Application Data
5284	113.658203	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2181 Ack=578 Win=8760 Len=0
5285	113.735351	173.194.71.108	10.0.2.15	TLSv1	112	Application Data
5286	113.737304	10.0.2.15	173.194.71.108	TLSv1	85	Application Data
5287	113.804687	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2239 Ack=609 Win=8760 Len=0
5288	114.007812	173.194.71.108	10.0.2.15	TLSv1	113	Application Data
5289	114.041992	10.0.2.15	173.194.71.108	TCP	64	56458 > urd [ACK] Seq=609 Ack=2298 Win=14600 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5290	114.166992	10.0.2.15	173.194.71.108	TLSv1	568	Application Data
5291	114.166992	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2298 Ack=1123 Win=8760 Len=0
5292	114.853515	173.194.71.108	10.0.2.15	TLSv1	123	Application Data
5293	114.853515	10.0.2.15	173.194.71.108	TCP	64	56458 > urd [ACK] Seq=1123 Ack=2367 Win=14600 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5294	114.860351	10.0.2.15	173.194.71.108	TLSv1	86	Application Data
5295	114.921875	173.194.71.108	10.0.2.15	TCP	54	urd > 56458 [ACK] Seq=2367 Ack=1154 Win=8760 Len=0
5296	114.952148	173.194.71.108	10.0.2.15	TLSv1	128	Application Data
5297	114.954101	10.0.2.15	173.194.71.108	TLSv1	81	Encrypted Alert
5298	114.955078	10.0.2.15	173.194.71.108	TCP	64	56458 > urd [FIN, ACK] Seq=1181 Ack=2441 Win=14600 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

Figure A.4: Traffic going to and from our application when sending a single message from our application

No.	Time	Source	Destination	Protocol	Length	Info
5259	111.281250	10.0.2.15	173.194.71.108	TLSv1	142	Client Hello
5261	111.416992	173.194.71.108	10.0.2.15	TLSv1	1514	Server Hello
5263	112.312500	173.194.71.108	10.0.2.15	TLSv1	299	Certificate, Server Hello Done
5265	112.340820	10.0.2.15	173.194.71.108	TLSv1	240	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
5267	112.535156	173.194.71.108	10.0.2.15	TLSv1	170	Change Cipher Spec, Encrypted Handshake Message, Application Data
5268	112.545898	10.0.2.15	173.194.71.108	TLSv1	96	Application Data
5270	112.750000	173.194.71.108	10.0.2.15	TLSv1	224	Application Data
5271	112.753906	10.0.2.15	173.194.71.108	TLSv1	92	Application Data
5273	112.958008	173.194.71.108	10.0.2.15	TLSv1	97	Application Data
5274	112.960937	10.0.2.15	173.194.71.108	TLSv1	109	Application Data
5276	113.095703	173.194.71.108	10.0.2.15	TLSv1	97	Application Data
5277	113.104492	10.0.2.15	173.194.71.108	TLSv1	102	Application Data
5279	113.449219	173.194.71.108	10.0.2.15	TLSv1	99	Application Data
5280	113.456054	10.0.2.15	173.194.71.108	TLSv1	116	Application Data
5282	113.589844	173.194.71.108	10.0.2.15	TLSv1	112	Application Data
5283	113.591797	10.0.2.15	173.194.71.108	TLSv1	115	Application Data
5285	113.735351	173.194.71.108	10.0.2.15	TLSv1	112	Application Data
5286	113.737304	10.0.2.15	173.194.71.108	TLSv1	85	Application Data
5288	114.007812	173.194.71.108	10.0.2.15	TLSv1	113	Application Data
5290	114.166992	10.0.2.15	173.194.71.108	TLSv1	568	Application Data
5292	114.853515	173.194.71.108	10.0.2.15	TLSv1	123	Application Data
5294	114.860351	10.0.2.15	173.194.71.108	TLSv1	86	Application Data
5296	114.952148	173.194.71.108	10.0.2.15	TLSv1	128	Application Data
5297	114.954101	10.0.2.15	173.194.71.108	TLSv1	81	Encrypted Alert

Figure A.5: TLS packets sent to and from our application

APPENDIX B

FUNCTIONAL TESTS

See table B.1 - B.28 on page 191 - 209 for test cases.

Test case ID	1
Name	Login
Requirement	FR1
Description	Test the login functionality
Preconditions	The application has started
Flow of events	1. Enter username "kprotesting" 2. Enter password "kprotest" 3. Press login button
Expected results	The login screen disappear and the inbox appear
Actual results	Correct
Comments	-
Status	OK

Table B.1: Test case 1

Test case ID	2
Name	Send regular message
Requirement	FR2
Description	Test sending of message with default values
Preconditions	The user is logged in
Flow of events	1. Navigate to the tab for sending a new message 2. Enter email address "kprothales@gmail.com" 3. Enter subject "Hello" 4. Select security label "Unclassified" from the list 5. Enter the text "Hello world!" in the message body area 6. Press send button
Expected results	A toast notification that says "Message sent" and sent back to the inbox. The message should now be in the "Sent" folder
Actual results	Correct
Comments	-
Status	OK

Table B.2: Test case 2

Test case ID	3
Name	Send message to contact from the address book
Requirement	FR2, FR5
Description	Test sending of message to a contact from the address book
Preconditions	The user is logged in
Flow of events	1. Navigate to the view for sending a new message 2. Press the contacts button 3. Select "kprothales@gmail.com" from the contacts list 4. Enter subject "Hello" 5. Select security label "Unclassified" from the list 6. Press send button
Expected results	A toast notification that says "Message sent" and sent back to the inbox. The message should now be in the "Sent" folder
Actual results	Correct
Comments	-
Status	OK

Table B.3: Test case 3

Test case ID	4
Name	Send full message
Requirement	FR2, FR6
Description	Test sending of message with attributes set
Preconditions	The user is logged in
Flow of events	1. Navigate to the view for sending a new message 2. Press the contacts button 3. Select "kprothales@gmail.com" from the contacts list 4. Enter subject "Hello" 5. Select security label "Unclassified" from the list 6. Select priority "Deferred" 7. Select type "Exercise" 8. Enter the text "Hello world!" in the message body area 9. Press send button
Expected results	A toast notification that says "Message sent" and sent back to inbox. The message should now be in the "Sent" folder
Actual results	Correct
Comments	-
Status	OK

Table B.4: Test case 4

Test case ID	5
Name	Sent messages folder
Requirement	FR4
Description	Test sent messages folder
Preconditions	The user is logged in and at least 3 message has been sent
Flow of events	1. Navigate to the view for the folders 2. Change the folder from "Inbox" to "Sent" 3. Press one of the message items
Expected results	There should be 3 messages in the folder. The message opens with all the correct attributes and text
Actual results	Correct
Comments	-
Status	OK

Table B.5: Test case 5

Test case ID	6
Name	Read and browse messages
Requirement	FR3
Description	Read and browse messages
Preconditions	The user is logged in and there are at least two message in the inbox folder
Flow of events	1. Navigate to the folders view and look in the inbox folder 2. Press a message in the message list 3. Browse between the messages (to previous and/or next message)
Expected results	The message view appears and shows the sender, subject, security label, priority, type, attachments if any and the message body. The message view changes the fields according to what message the user is viewing
Actual results	Correct
Comments	-
Status	OK

Table B.6: Test case 6

Test case ID	7
Name	Send attachment (camera)
Requirement	FR7
Description	Test sending of image from camera
Preconditions	The user is logged in
Flow of events	1. Navigate to the view for sending a new message 2. Press the contacts button 3. Select "kprotesting@gmail.com" (yourself) from the contacts list 4. Enter subject "Hello" 5. Select security label "Unclassified" from the list 6. Press the button for adding an attachment 7. Take a picture with the camera 8. Press the send button
Expected results	A toast notification that says "Message sent" and sent back to inbox
Actual results	Correct
Comments	-
Status	OK

Table B.7: Test case 7

Test case ID	8
Name	Send attachment (gallery)
Requirement	FR7
Description	Test sending of image from gallery
Preconditions	The user is logged in and has at least one image in the gallery
Flow of events	1. Navigate to the view for sending a new message 2. Press the contacts button 3. Select "kprotesting@gmail.com" (yourself) from the contacts list 4. Enter subject "Hello" 5. Select security label "Unclassified" from the list 6. Press the button for adding an attachment 7. Select an image from the gallery 8. Press the send button
Expected results	A toast notification that says "Message sent" and sent back to inbox
Actual results	Correct
Comments	-
Status	OK

Table B.8: Test case 8

Test case ID	9
Name	Send attachment (GPS)
Requirement	FR7
Description	Test sending of GPS coordinates
Preconditions	The user is logged in
Flow of events	1. Navigate to the view for sending a new message 2. Press the contacts button 3. Select "kprotest@gmail.com" (yourself) from the contacts list 4. Enter subject "Hello" 5. Select security label "Unclassified" from the list 6. Press the button for adding an attachment 7. Add location (GPS-coordinates) 8. Press the send button
Expected results	The message text should be filled out with the current location. A toast notification that says "Message sent" and sent back to inbox
Actual results	Correct
Comments	-
Status	OK

Table B.9: Test case 9

Test case ID	10
Name	Attachments received
Requirement	FR7
Description	Test reading and opening of a message with attachments
Preconditions	The user is logged in and there is at least one message with attachements
Flow of events	1. Navigate to the folders and make sure you view inbox messages 2. Press a message that has an attachment (e.g. one of the messages you sent to yourself in previous tests) 3. Press the button to show the attachments 4. Open the image that is attached
Expected results	The correct program for opening an image is started and the image is shown
Actual results	Correct
Comments	-
Status	OK

Table B.10: Test case 10

Test case ID	11
Name	Instant message settings
Requirement	FR9, FR10
Description	Test instant message settings
Preconditions	The user is logged in
Flow of events	1. Navigate to the settings view and view the settings for instant message attributes 2. Set "kprothales@gmail.com" as standard receiver 3. Set "Restricted" as standard security label 4. Set "Flash" as standard priority 5. Set "Drill" as standard type 6. Navigate back to the instant message view
Expected results	The instant message view should have fields corresponding to the settings chosen
Actual results	Correct
Comments	-
Status	OK

Table B.11: Test case 11

Test case ID	12
Name	Message retrival strategy settings
Requirement	FR10
Description	Test settings for push/pull strategy
Preconditions	The user is logged in
Flow of events	1. Navigate to the settings view and view the settings for message retrieval 2. Change the strategy to "pull" 3. Set the poll interval to 1 minute
Expected results	The application should now change strategy from push (default) to pull
Actual results	The application still uses push strategy
Comments	Unknown, has tried to debug
Status	Failed

Table B.12: Test case 12

Test case ID	13
Name	Security labels settings
Requirement	FR10
Description	Test settings for available security labels
Preconditions	The user is logged in
Flow of events	1. Navigate to the settings view and view the settings for security labels 2. Check the checkboxes for "RESTRICTED" and "NATO UNCLASSIFIED" 3. Uncheck the rest 4. Navigate to the send message view
Expected results	The dropdown list of security labels should be filled out according to the choices in the settings
Actual results	All the security labels are still shown
Comments	Known deficiency, has not been implemented
Status	Failed

Table B.13: Test case 13

Test case ID	14
Name	Send instant message
Requirement	FR9
Description	Test sending of an instant message
Preconditions	The user is logged in
Flow of events	1. Navigate to the instant message view 2. Enter the text "Hello World" in the message body text area 3. Press the send button
Expected results	The fields in the view are filled out according to the settings. The message is sent instantly. Toast notification that says "Message sent" and sent back to the inbox
Actual results	Correct
Comments	-
Status	OK

Table B.14: Test case 14

Test case ID	15
Name	Receive of an flash/override message
Requirement	FR12
Description	Test receiving of an high priority message
Preconditions	The user is logged in and someone has sent a high-priority message to the user
Flow of events	1. A red box takes over the screen 2. Click open message
Expected results	The message is shown in the regular message view
Actual results	Correct
Comments	-
Status	OK

Table B.15: Test case 15

Test case ID	16
Name	Send instant message with attachments
Requirement	FR9
Description	Test sending of instant message with attachments
Preconditions	The user is logged in and has at least one image in the gallery
Flow of events	1. Navigate to the instant message view 2. Check if the predefined values are set (if not set them in settings) 3. Press the button for add attachment 4. Select an image from the gallery 5. Press the button for adding attachment again 6. Add GPS coordinates to the message 7. Press the send button
Expected results	The message is sent instantly. Toast notification that says "Message sent" and sent back to the inbox
Actual results	GPS coordinates is attached, but the message attributes somehow get wrong
Comments	Unknown
Status	Image OK, GPS failed

Table B.16: Test case 16

Test case ID	17
Name	Receive instant message outside the application
Requirement	FR12
Description	Test receiving of instant message outside the application
Preconditions	The user is logged in and the application is in the background
Flow of events	1. A red box takes over the screen 2. Click open message
Expected results	The message is shown in the regular message view
Actual results	Correct
Comments	-
Status	OK

Table B.17: Test case 17

Test case ID	18
Name	Widget for instant message
Requirement	FR9
Description	Test the widget for sending an instant message
Preconditions	The user is logged in and the widget is placed on the home screen
Flow of events	1. Click the widget
Expected results	The application opens and the view for sending an instant message is opened
Actual results	Correct
Comments	-
Status	OK

Table B.18: Test case 18

Test case ID	19
Name	Reply
Requirement	FR8
Description	Test reply of a message
Preconditions	The user is logged in and has at least one message in the inbox
Flow of events	1. Navigate to the folders view and look at the inbox 2. Click on a message in the list 3. Press the reply button
Expected results	The send message view opens with the correct fields filled out, the security label is fixed and "Re:" stands before the subject.
Actual results	The correct actions happen, but one gets sent out of the tab system
Comments	This feature has not been implemented perfect
Status	OK-

Table B.19: Test case 19

Test case ID	20
Name	Forward
Requirement	FR8
Description	Test forwarding of a message
Preconditions	The user is logged in and has at least one message in the inbox
Flow of events	1. Navigate to the folders view and look at the inbox 2. Click on a message in the list 3. Press the forward button
Expected results	The send message view opens with the correct fields filled out, the security label is fixed and "Fwd:" before the subject.
Actual results	The correct actions happen, but one gets sent out of the tab system
Comments	OK-
Status	

Table B.20: Test case 20

Test case ID	21
Name	Delete
Requirement	FR8
Description	Test deleting of a message
Preconditions	The user is logged in and has at least one message in the inbox
Flow of events	1. Navigate to the folders view and look at the inbox 2. Click on a message in the list 3. Press the menu button 4. Press delete on the popup menu that shows at the lower part of the screen
Expected results	The view changes to the inbox and the message has disappeared from the list.
Actual results	The message disappears at first, but is still there if you log out and in
Comments	Known deficiency, problems with the Android file system
Status	Failed

Table B.21: Test case 21

Test case ID	22
Name	Delivery report and receipt notification
Requirement	FR11
Description	Test sending of a message where one has requested delivery report and receipt notification
Preconditions	The user is logged in
Flow of events	1. Navigate to the send message view 2. Set "kprothales@gmail.com" as receiver 3. Set "Hello world" as subject 4. Set "UNCLASSIFIED" as security label 5. Enter "Hello everybody" in the message text. 6. Request delivery report and receipt notification 7. Press the send button
Expected results	A toast notification that says "Message sent" and sent back to inbox
Actual results	Feature not implemented
Comments	Feature not implemented
Status	Failed

Table B.22: Test case 22

Test case ID	23
Name	Status of delivery report and receipt notification
Requirement	FR11
Description	Test checking status on a message where one has requested delivery report and receipt notification
Preconditions	The user is logged in and has at least one sent message that has a request for delivery report and receipt notification
Flow of events	1. Navigate to the sent messages 2. Click on a message that has a request for delivery report and receipt notification 3. View the status of the delivery report and receipt notification
Expected results	The message has fields with these attributes, which is red at first but changes to green when the status is updated to deliverd/read
Actual results	Not implemented
Comments	Not implemented
Status	Failed

Table B.23: Test case 23

Test case ID	24
Name	Sort
Requirement	FR13
Description	Test sorting of messages in the inbox based on different criterias
Preconditions	The user is logged in and has at least 4 different messages in the inbox
Flow of events	1. Navigate to folders view 2. Press the sort button and choose different sorting algorithms
Expected results	The inbox messages are sorted correct according to the sorting criteria
Actual results	Correct
Comments	-
Status	OK

Table B.24: Test case 24

Test case ID	25
Name	Search
Requirement	FR14
Description	Test searching of messages in folders
Preconditions	The user is logged in and has at least 4 different messages in the inbox
Flow of events	1. Navigate to folders view 2. Enter text in the search field and press Enter
Expected results	The folder is filtrated to show only the items that match the search condition
Actual results	Not implemented
Comments	Not implemented
Status	Failed

Table B.25: Test case 25

Test case ID	26
Name	Login incorrect input
Requirement	FR1
Description	Test how the application reacts to incorrect login input
Preconditions	The application has started
Tasks	1. Enter bogus username and password 2. Leave the username field empty 3. Leave the password field empty 4. Leave both fields empty
Expected result	An error message showing "Incorrect login information" and the user do not get logged in
Actual result	Tasks 1-4 gives the correct error message and the login is not allowed
Comments	-
Status	OK

Table B.26: Test case 26

Test case ID	27
Name	Receiver incorrect input
Requirement	FR2
Description	Test how the application reacts to incorrect receiver input
Preconditions	The user is logged in and looks at the send message view
Tasks	1. Leave the receiver field empty 2. Enter several words with an arbitrary separator
Expected result	A validation error message in the text field that says "The address is not valid. Please check again".
Actual result	Tasks 1-2 correct
Comments	-
Status	OK

Table B.27: Test case 27

Test case ID	28
Name	Security label incorrect input
Requirement	FR2
Description	Test how the application reacts to incorrect security label input
Preconditions	The user is logged in and looks at the send message view
Tasks	1. Do net set a value for the security label
Expected result	An error message showing "You need to set a security label" and the message does not get sent
Actual result	Correct
Comments	-
Status	OK

Table B.28: Test case 28

APPENDIX C

USABILITY TESTING

C.1 Test execution

C.1.1 Preparation for the test leader

- Make sure you have the latest version of XOXOmail (updated 08.11) installed and Internet connection on your phone/computer
- Have the tasks ready on paper or screen
- Have the SUS questionnaire ready on paper or screen
- Have pen and the observation form on paper or a computer ready to make notes on
- Preconditions for the tests: Make sure there are no mails on the account (kprotesting@gmail.com/kprotes with flash/override as priority. Other messages are OK.
- Put an ID (e.g. the test leader's name) on both the observation form and the SUS questionnaire.
- Note that task 5 is meant to open the message the test person sent to himself/herself. If the test person sent the message to another address or failed to send a message, you need to send a new message with flash/override priority to kprotesting@gmail.com.

C.1.2 Information about the test given to the user

- This is a test to find out if the application is intuitive and user friendly, and not a test of you and your skills.
- The test consists of six (6) tasks and will take approximately 20 minutes.
- Read the instructions for each task and perform the tasks one by one.
- Each task has the same structure: A task number, a task name, a description of what you should do and input data that is needed to solve a task. If you find input fields in the application that do not have a value listed, you can enter an arbitrary value.

- If you cannot figure out how to solve a task this is not your fault, but the application that is not designed in a user friendly way. You can then move on to the next task, but notify the test leader.
- You can ask questions before and after the test, but we cannot help you during the test.
- You can quit the test anytime you want.
- It would be helpful if you could try to think aloud during the test. Try to explain what you see and why you make your choices. This makes it easier for us to figure out how users think and what could be done better in the design.
- After the test we would appreciate if you could fill out a questionnaire and give feedback if you felt that something worked well or not so well.

C.1.3 Information about the application given to the user

XOXOmail is a mail client application for Android phones. It is intended for use in the military or other similar organizations. The application is very much like a regular mail client application, but the difference is that a mail in this application can be given three special attributes: security label, priority and type. Otherwise the application has the ability, like regular mail clients, to send and receive messages with and without attachments.

There is also a distinction between a regular mail and a so-called instant message. An instant message is meant for quicker sending, where you can predefine a receiver and values of the three attributes in the settings. A last thing to note is that messages with high priority are handled differently from regular messages, as they will take over the screen and force an action to either open the message or cancel to carry on with what you were doing.

C.1.4 SUS form

The SUS form can be found in figure C.1 on page 212

C.1.5 Observation form

The observation form template can be found in figure C.2 on page 213

C.2 The results

System Usability Scale

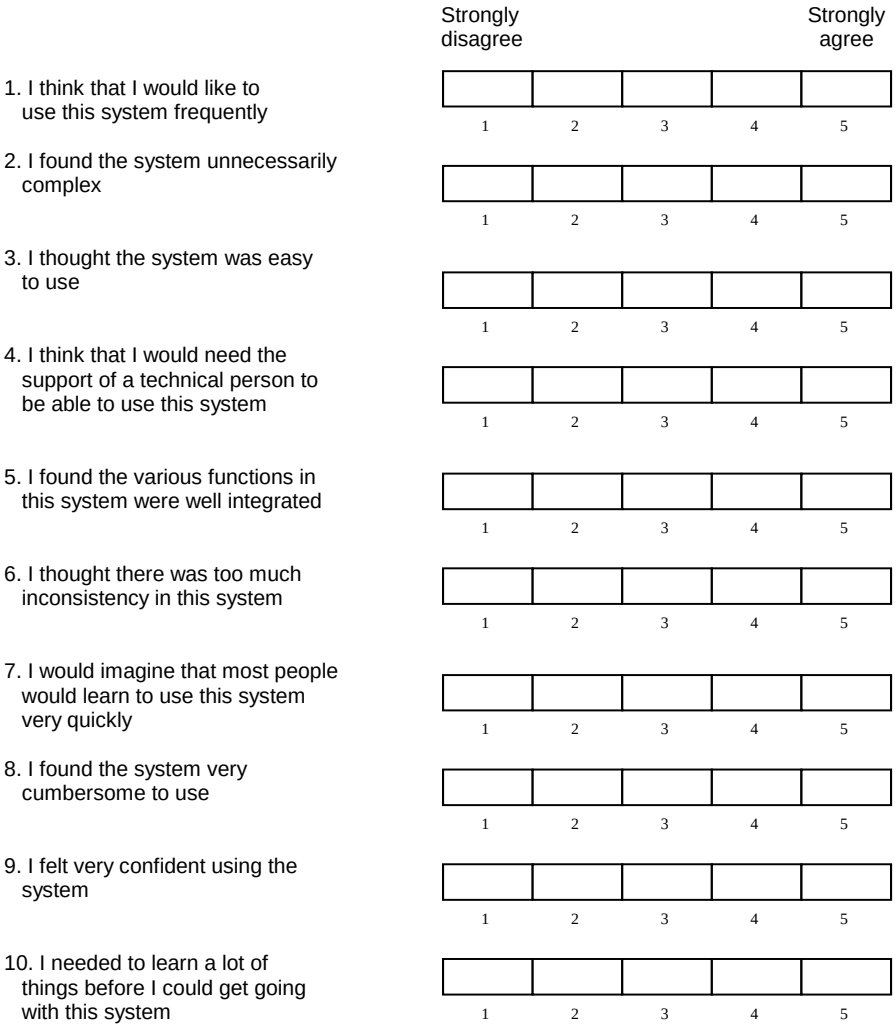


Figure C.1: SUS form

Observation Form for Usability Testing

Test leader:

General information

Participant ID		Date of test	
Gender		Education	
Age		Experience with smart phones/Android	

Test observation

Time	Task number	Observations
	1	
	2	
	3	
	4	
	5	
	6	

Figure C.2: Observation form

Observation Form for Usability Testing

General information

Participant ID	1	**Date of test**	08/11/2012
Gender	Male	**Education**	Computer science major
Age	22	**Android experience**	High experience

Test observation

Time	Task number	Observations
1 min	1	Went well without any problems. Annoyed because the application did not remember his login information after a crash in a later test.
3 min	2	Went well. Found the send message tab immediately. Used contacts button to add the recipient. Reacted to laggy scrolling and the small size of the toast appearing after sending a message.
3 min	3	Found the settings tab immediately. Complained about small font in the settings menu. Confused by security labels, English and Norwegian are mixed together. The application stopped unexpectedly.
1 min	4	Did not go too well. Low affordance on the instant message tab icon. Impossible to send attachment with instant message.
1 min	5	Went well, no problems.
1 min	6	Already did this in the 2nd test.

Question	Answer	Actual
1	3	2
2	2	3
3	4	3
4	1	4
5	3	2
6	2	3
7	4	3
8	3	2
9	4	3
10	1	4
Sum		29
SUS	x 2.5	**72.5**

Observation Form for Usability Testing

General information

Participant ID	2	**Date of test**	08/11/2012
Gender	Male	**Education**	Technology/business major
Age	22	**Android experience**	High experience

Test observation

Time	Task number	Observations
1 min	1	Weird that it says email and one is required to enter username. Went well otherwise.
3 min	2	Thinks + looks like a new message. Tries to tilt the phone, but this redirects to the inbox. The text looks weird, something about the graphics. Annoyed that by not getting capitalized letter after punctuation. Thinks that it is sent, but apparently does not see the toast.
4 min	3	What is the triangle? Looks interesting. Then it must be the last button.. Browses around the settings page, looking. Wants a confirmation that the settings has been saved.
5 min	4	Figures out that he should send something different than last tasks, then it must be the triangle. But does not think instant message from that! "Apparently these attributes are set", before figuring out that he set them in the previous task. "Image from camera" should be "take a picture". Thinks the attachments list looks not clickable. Tries to tilt again.
1 min	5	Looks at the red box, but has already tried to open an attachment, so no testing is required.
2 min	6	Here is a picture with a lot of people - it must be this one.

Question	Answer	Actual
1	3	2
2	2	3
3	4	3
4	2	3
5	3	2
6	2	3
7	3	2
8	2	3
9	4	3
10	2	3
Sum		27
SUS	x 2.5	**67.5**

Observation Form for Usability Testing

General information

Participant ID	3	**Date of test**	09/11/2012
Gender	Female	**Education**	Non technical bachelor
Age	19	**Android experience**	Owns an Android

Test observation

Time	Task number	Observations
1 min	1	No comments, went well.
2 min	2	No comments, went well.
3,5 min	3	Difficulty finding settings, clicking around a bit. Found it at last, but then the app crashed. "What? Now I need to log in again". Figured out how settings worked.
2 min	4	Went well.
0 min	5	Invalid, as the previous did not work
1 min	6	Went well.

Question	Answer	Actual
1	4	3
2	2	3
3	4	3
4	1	4
5	3	2
6	2	3
7	4	3
8	2	3
9	4	3
10	1	4
Sum		31
SUS	x 2.5	**77.5**

Observation Form for Usability Testing

General information

Participant ID	4	**Date of test**	09/11/2012
Gender	Male	**Education/Work**	Electrician
Age	22	**Android experience**	Owns an Android

Test observation

Time	Task number	Observations
1 min	1	Went well without any problems.
2 min	2	Shall send a regular message, guessing on the pen. Somebody else tested at the same time and high priority messages arrived. Shall I send? Yes, I send.
5 min	3	Settings button failed and the app crashed, needed to be closed. Has to log in again... Pushed OK at standard receiver, then the app closed itself but opened with the login information filled in. Crashed again when trying to save more. Should maybe state what you have chosen on everything, or become green when you can make a choice, then push OK.
3 min	4	Does not understand the difference at first. Thinks it is a regular message, but can't figure out a receiver. Reads the test again. Can't close the keyboard.
2 min	5	Went well, no problems. Took some time before the message arrived. There is the picture I took!
2 min	6	Understood quickly where the contacts are. Bad that label disappears when you have pushed it, should maybe become green or something. Somehow make it easier to understand that you have chosen. Weird that you can some places click enter and get to the next field, but in the message text you get a line break.

Question	Answer	Actual
1	4	3
2	1	4
3	5	4
4	1	4
5	5	4
6	3	2
7	4	3
8	1	4
9	4	3
10	1	4
Sum		35
SUS	x 2.5	**87.5**

Observation Form for Usability Testing

General information

Participant ID	5	**Date of test**	09/11/2012
Gender	Male	**Education**	Computer science major
Age	23	**Android experience**	Owns an Android

Test observation

Time	Task number	Observations
1 min	1	The first impression of the user was that the test case specified a username, while the login screen asked for an email. However, the user managed to see the connection and input the correct values and log in.
2 min	2	The user found the "send message" tab at first try, and found no problems in putting in the data specified in the test case scenario.
5 min	3	The user found the "settings" tab without any problems, and proceeded to change the instant message attributes without hassle.
4 min	4	The user did not understand the task, and went along just sending a normal instant message without any attachment, since he could not find anything related to attachments in the 'instant message' tab.
3 min	5	The user had no problems with understanding the flash message popup, and promptly clicked open. The message did however not contain an attachment due to misunderstanding task 4. After task 6, the user was asked if he could open the attachment of the last message he sent, and did so without any trouble.
3 min	6	The user had no problems with attaching an attachment to message before sending.

Question	Answer	Actual
1	3	2
2	2	3
3	4	3
4	1	4
5	5	4
6	1	4
7	4	3
8	1	4
9	5	4
10	1	4
Sum		35
SUS	x 2.5	**87.5**

APPENDIX D

HOW TO BUILD PROJECT

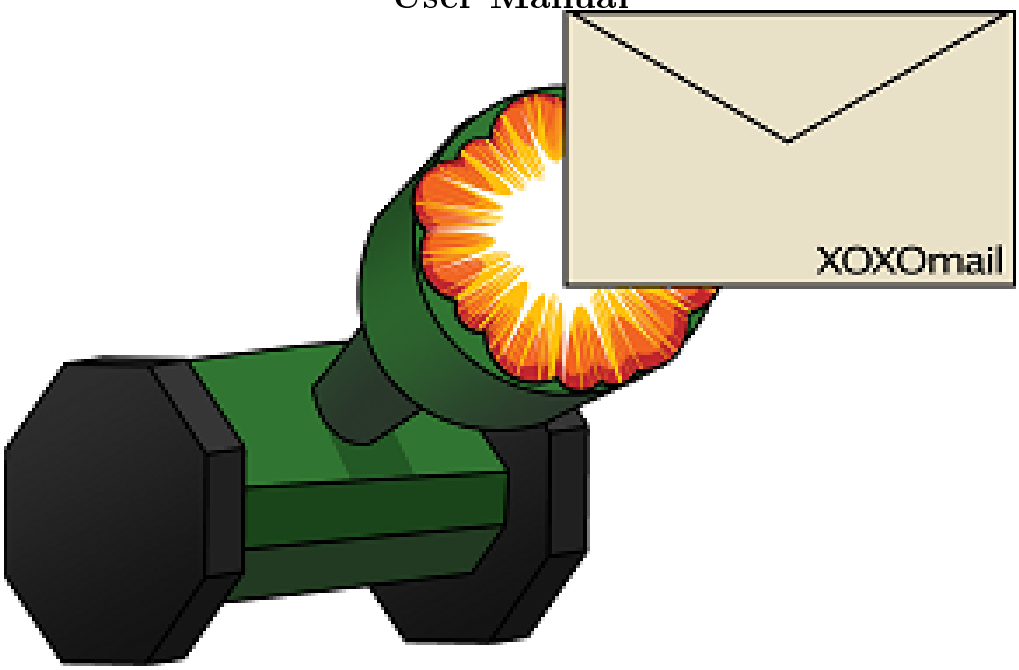
1. Checkout from github (github.com/nutgaard/KPro—Thales)
2. Install Maven 3.0.4, (http://maven.apache.org/)
3. Install dependencies not located in Maven repository (Only needed the first time), all the commands below should be run while in the KPro—Thales directory.
   (a) Install mail.jar
      i. mvn install:install-file -Dfile=mail.jar -DgroupId=javax.mail -DartifactId=mail -Dversion=1.0 -Dpackaging=jar
   (b) Install activation.jar
      i. mvn install:install-file -Dfile=activation.jar -DgroupId=javax.mail -DartifactId=activation -Dversion=1.0 -Dpackaging=jar
   (c) Install additionnal.jar
      i. mvn install:install-file -Dfile=additionnal.jar -DgroupId=javax.mail -DartifactId=additionnal -Dversion=1.0 -Dpackaging=jar
4. Start emulator or connect a phone. A successful connection can be verified by the command 'adb devices', which should return a list of all the connected devices.
5. Run 'mvn clean install && cd kpro-app && mvn android:deploy android:run' from KPro-Thales directory. This command builds the application completely from scratch, runs the tests, deploys it to all available devices and starts the application on the devices.

APPENDIX E

USER MANUAL

XOXOmail

User Manual



Contents

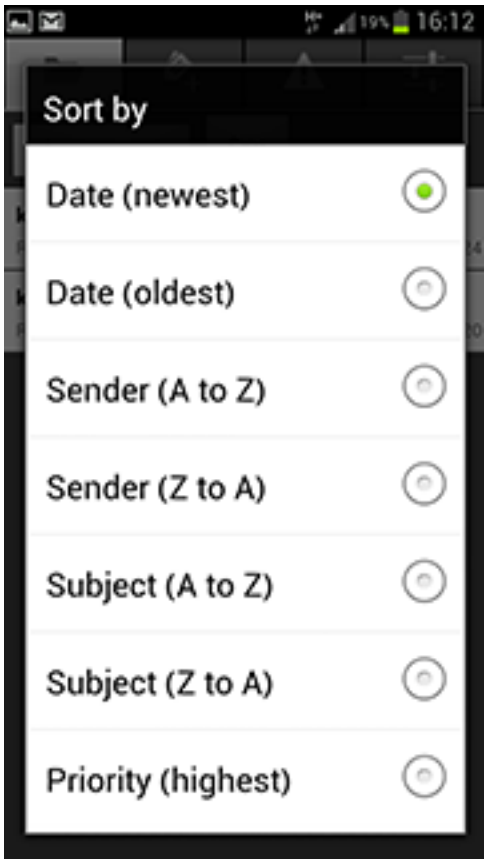
Logging in

After starting the XOXOmail application you will be prompted with a login screen. In order to login, simply press the login button below the password field. There is no need to enter any information in the Email and Password field as this has not yet been implemented.

Viewing the inbox

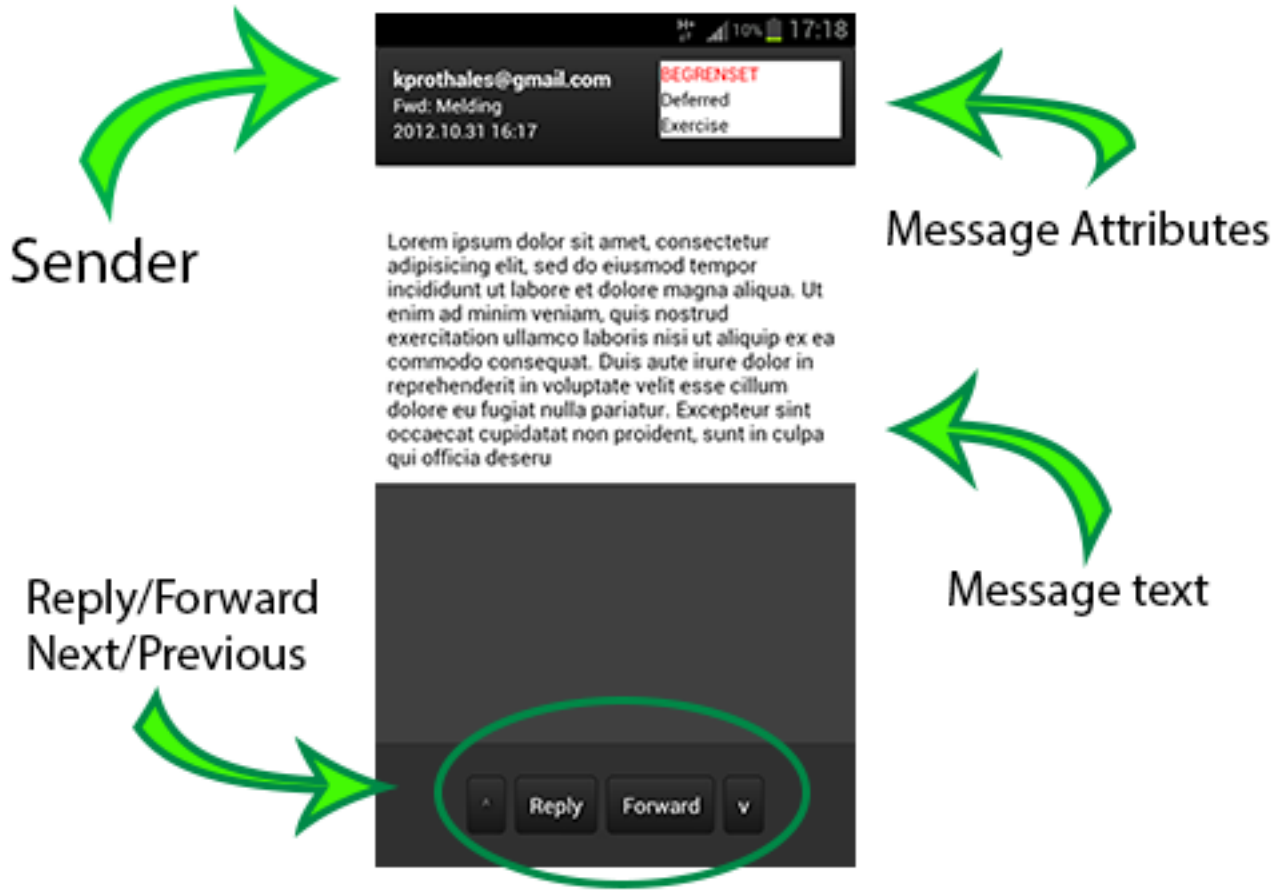
To view the inbox, make sure the inbox tab i selected. You will then see a list of all messages currently in the inbox. If you want to sort these messages, simply press the sort button indicated in the screenshot below.

You can then select the desired sorting algorithm by pressing the different options presented.



If you want to view any of the messages currently in the inbox, you can press the messages displayed on the screen. You will then be brought to the message view.

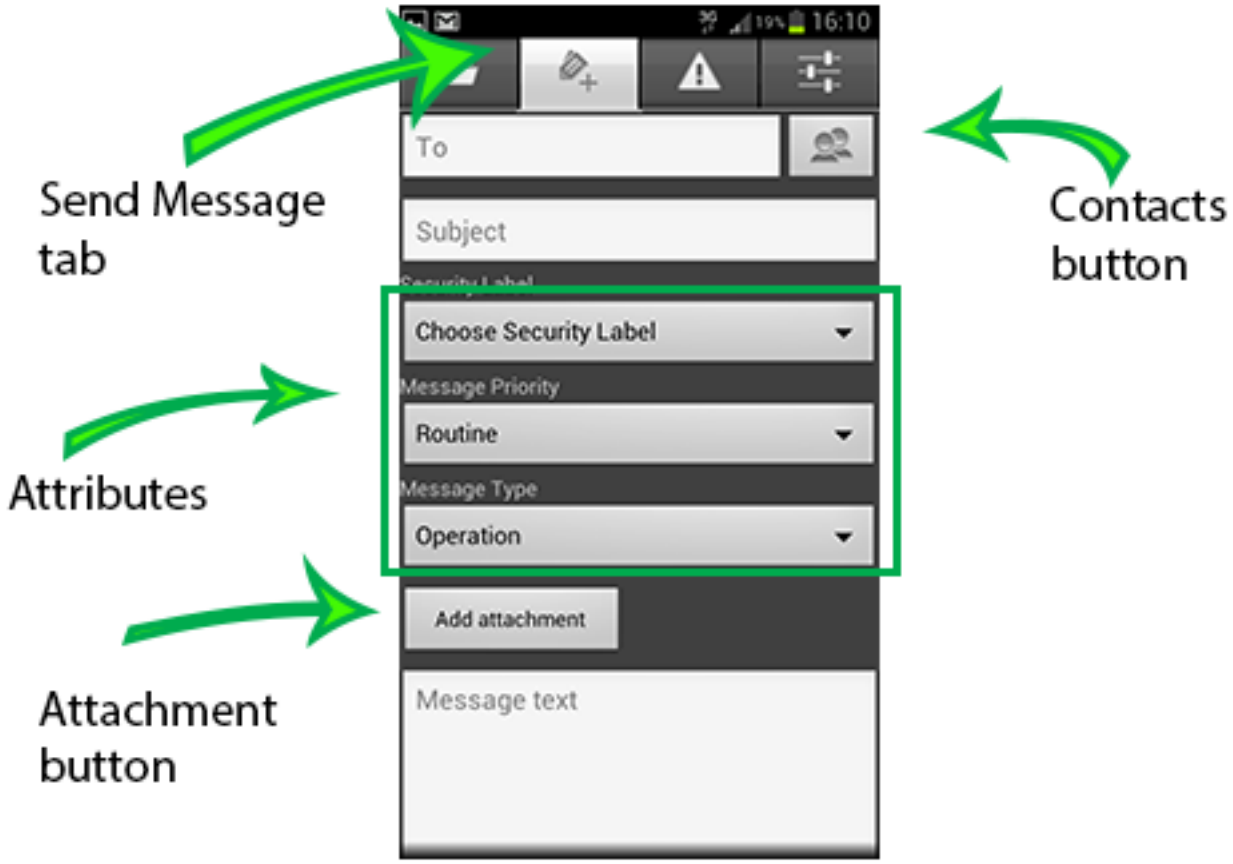
From the message view you can see the sender of the message, the message attributes and the actual message text. You also have the opportunity to change between received messages, reply to a message and forward the received message. This is done by pressing the respective buttons at the bottom of the screen.



Sending a message

To send a new message, you must first make sure you have the send message tab selected. You will then be presented with the send message view.

First you must select the recipient of the message to be sent. This can be done by either typing in the recipient manually or pressing the contacts button.



If the contacts button is pressed, a new window will open displaying your contacts. Pressing one of these contacts will add the address to the "To" field.

After choosing a recipient you must add a subject to your message. This is done by typing in the subject in the field below the recipient field.

The next step is to select which attributes your message will include. You will need to select the label, priority and type of the message, if the attributes are not selected, the default values will be added to the message.

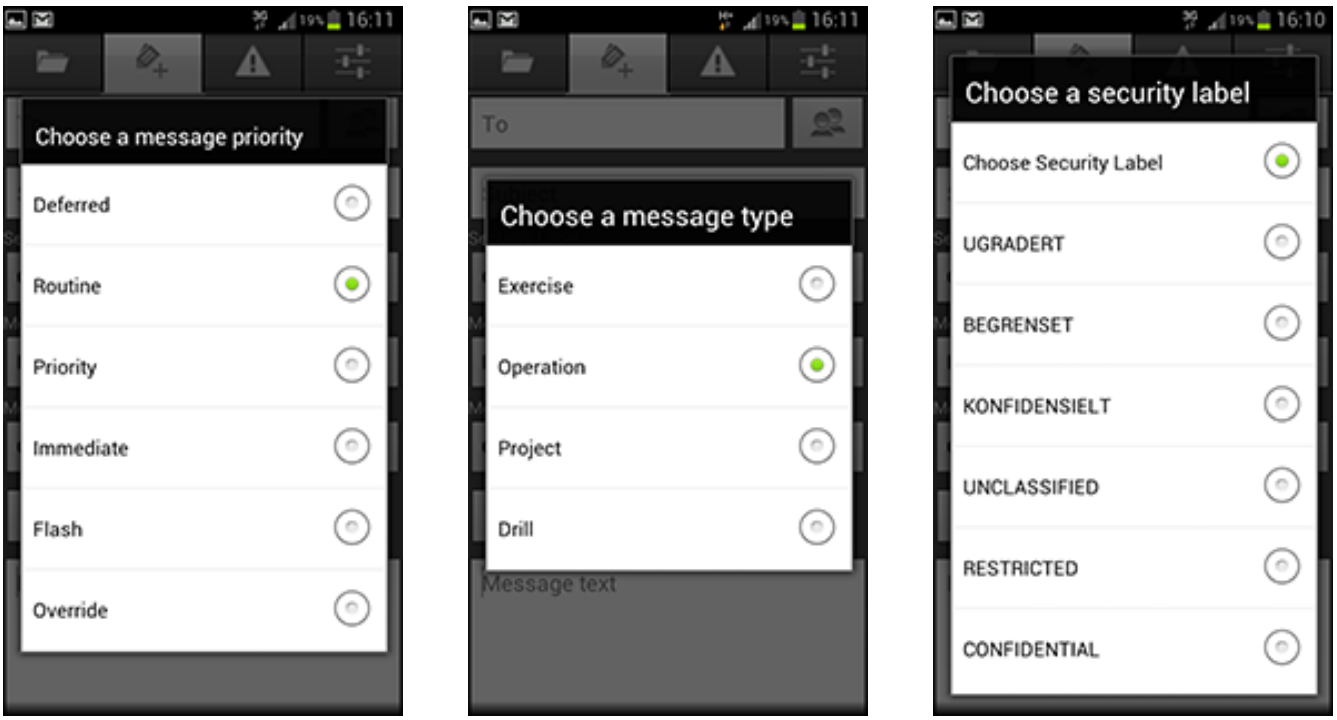
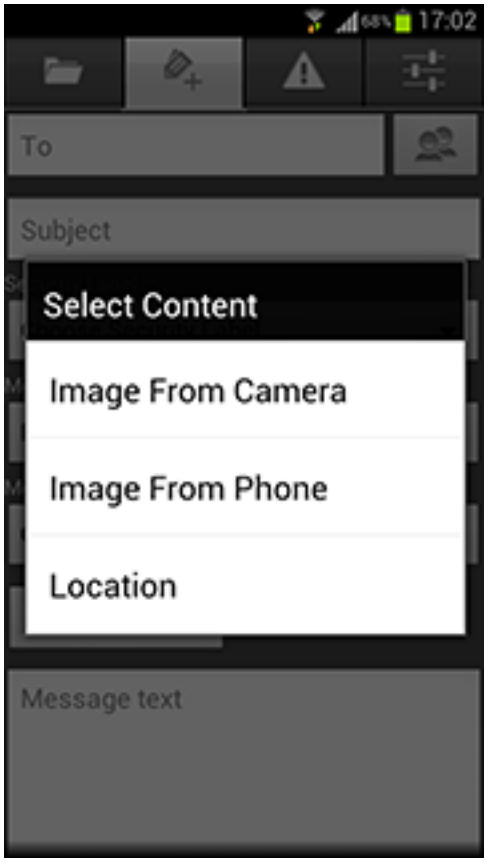


Figure E.1: Choosing attributes

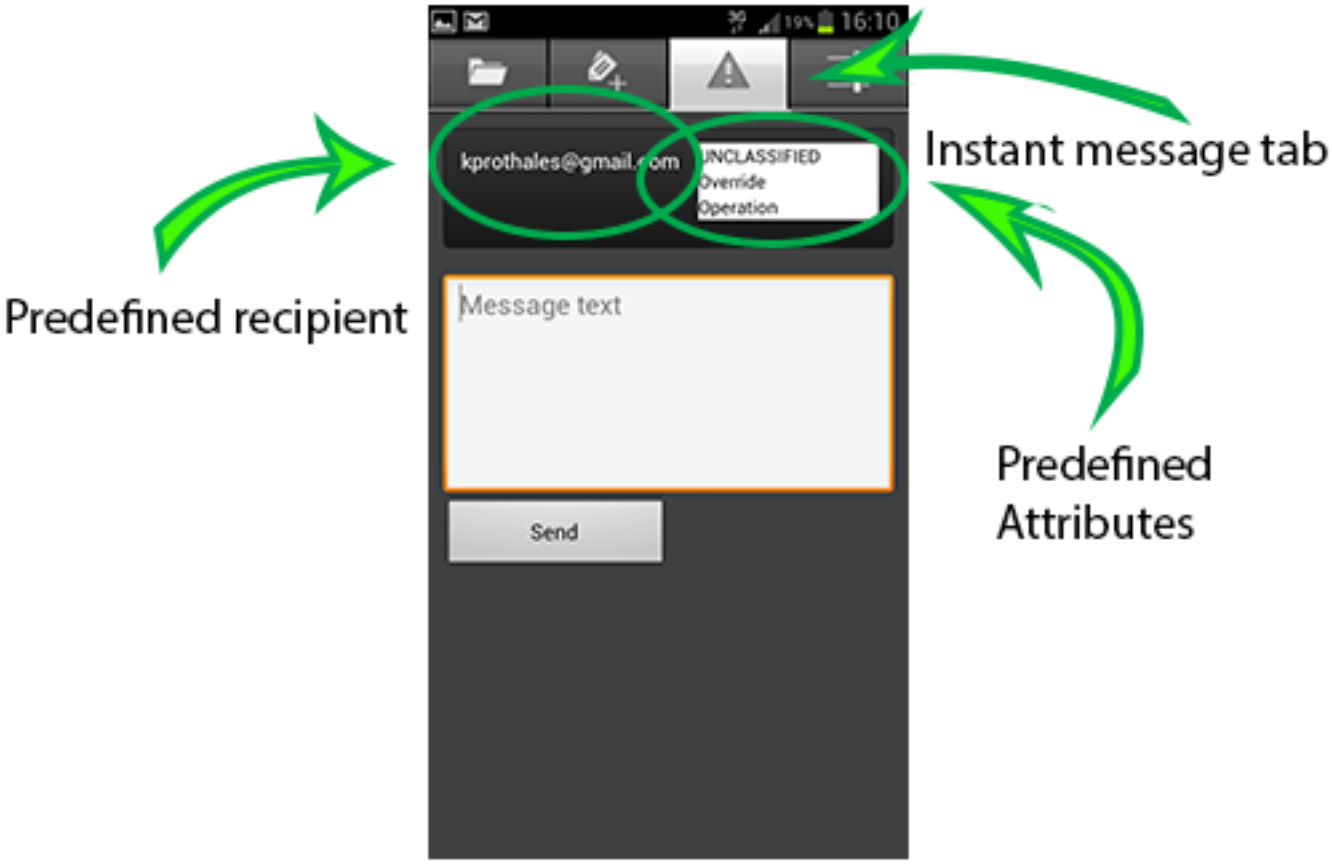
After adding the message attributes you have the opportunity to add an attachment to your message. This can be an image from the camera, an image already stored on the phone or your current GPS location.



Finally the actual message text can be added to the message. This is done by pressing the Message text field and typing in the desired text. To send your message, simply press the send button at the bottom of the screen.

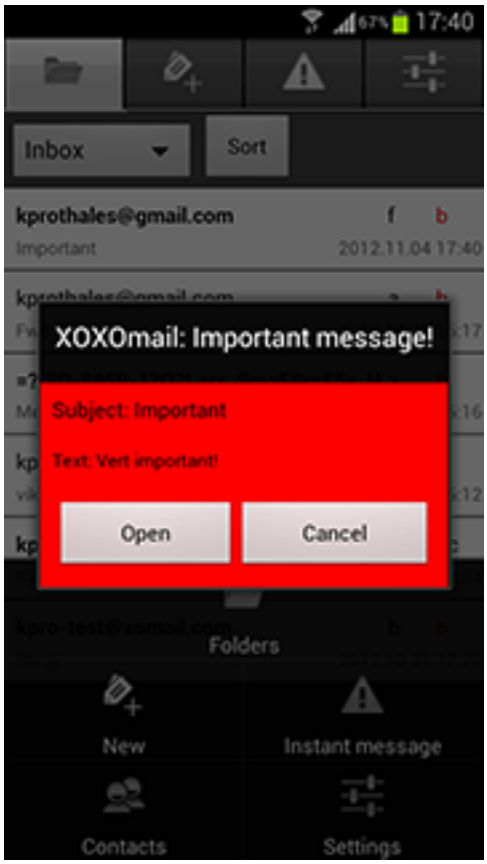
Instant messages

To send an instant message, you must have the instant message tab selected. The instant message view shows you the predefined recipient and attributes. The predefined values are set in the settings menu. To send an instant message you simply type in the message text and press the send button.



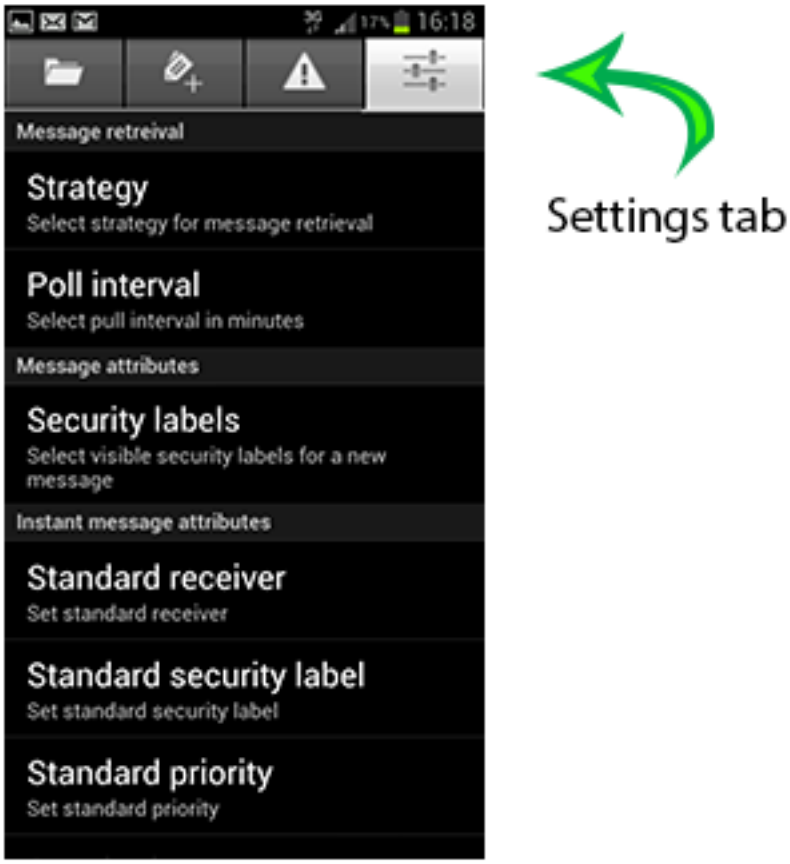
Flash and Override messages

When receiving a message with message priority set to Flash or Override, a red popup will cover your screen. You can then view this message, or press cancel to close the window.



Configuring the settings

To alter the various settings in the application, you must have the settings tab selected.



In the settings tab you have 4 different categories of configurable settings:

- Message retrieval
- Message attributes
- Instant message attributes
- Location data

In the message retrieval section you have the ability to choose the message retrieval strategy and poll interval.

There are 2 message retrieval strategies available at this time: push and pull. When pull is selected, the messages in the inbox are loaded every given interval. The application queries the server for new messages after a set interval has passed.
If push is selected the application will load all the messages in the inbox after startup. When a new message arrives the server will push the message to the application and display it in the inbox.

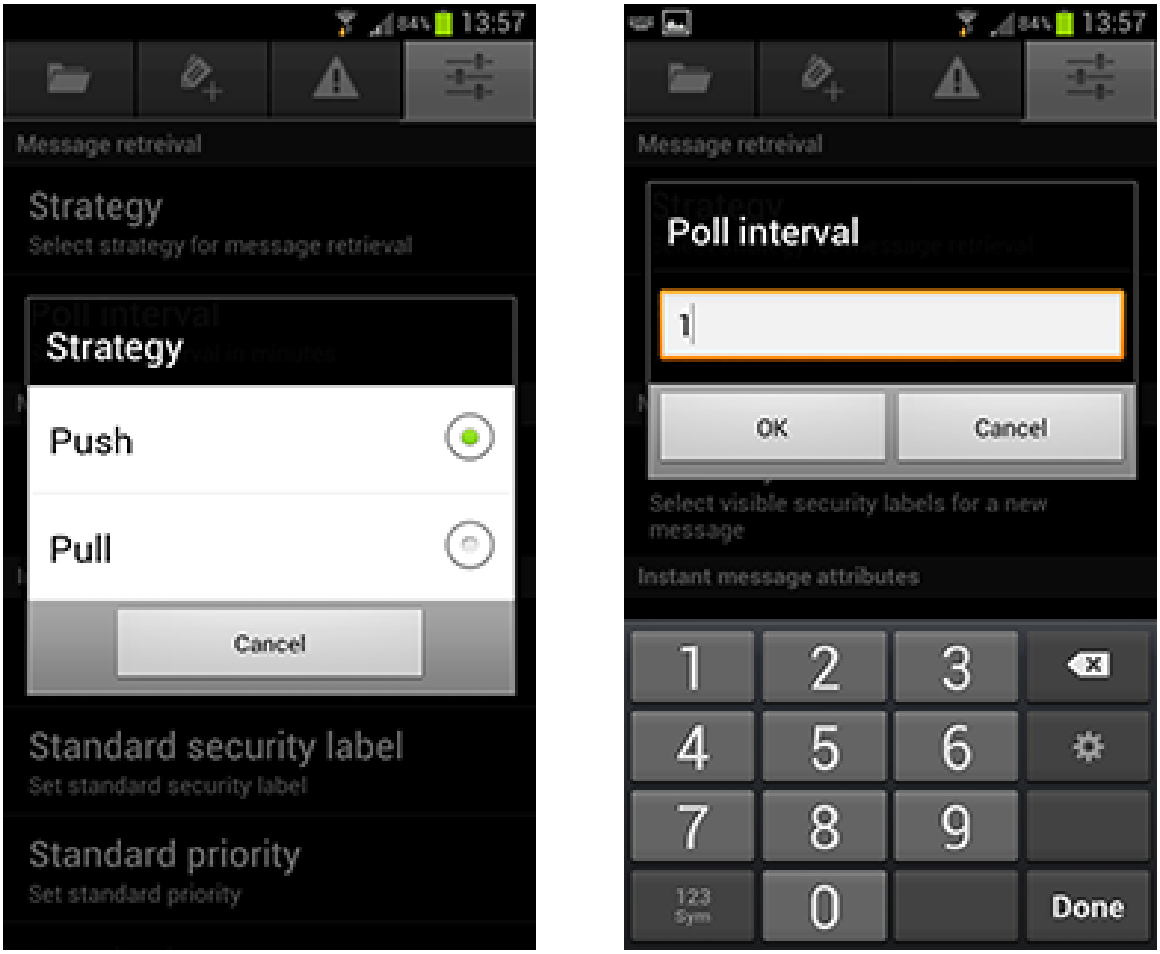
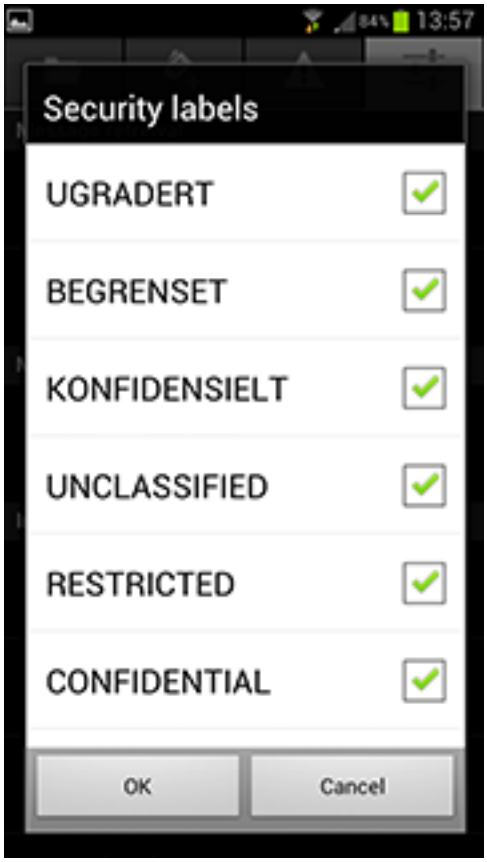


Figure E.2: Pull strategy and poll interval

In the message attributes section you have the ability to select which attributes that you would like to choose amongst when sending a message.



In the instant message attributes section you have the ability to set the different standard values for sending an instant message. These include:

- Standard receiver
- Standard security label
- Standard priority
- Standard type

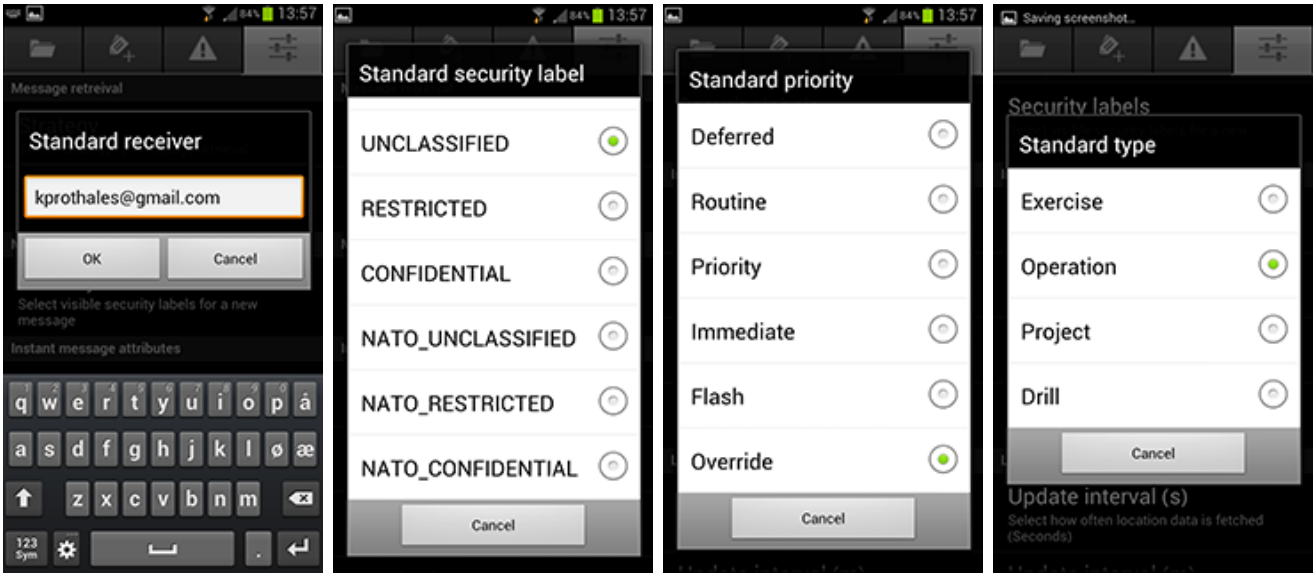


Figure E.3: Instant message standard values

In the location data section you can choose when the gps position is fetched. This is done by setting how often it should try to update the gps position in seconds, and also how far you need to travel in order update the gps coordinates.

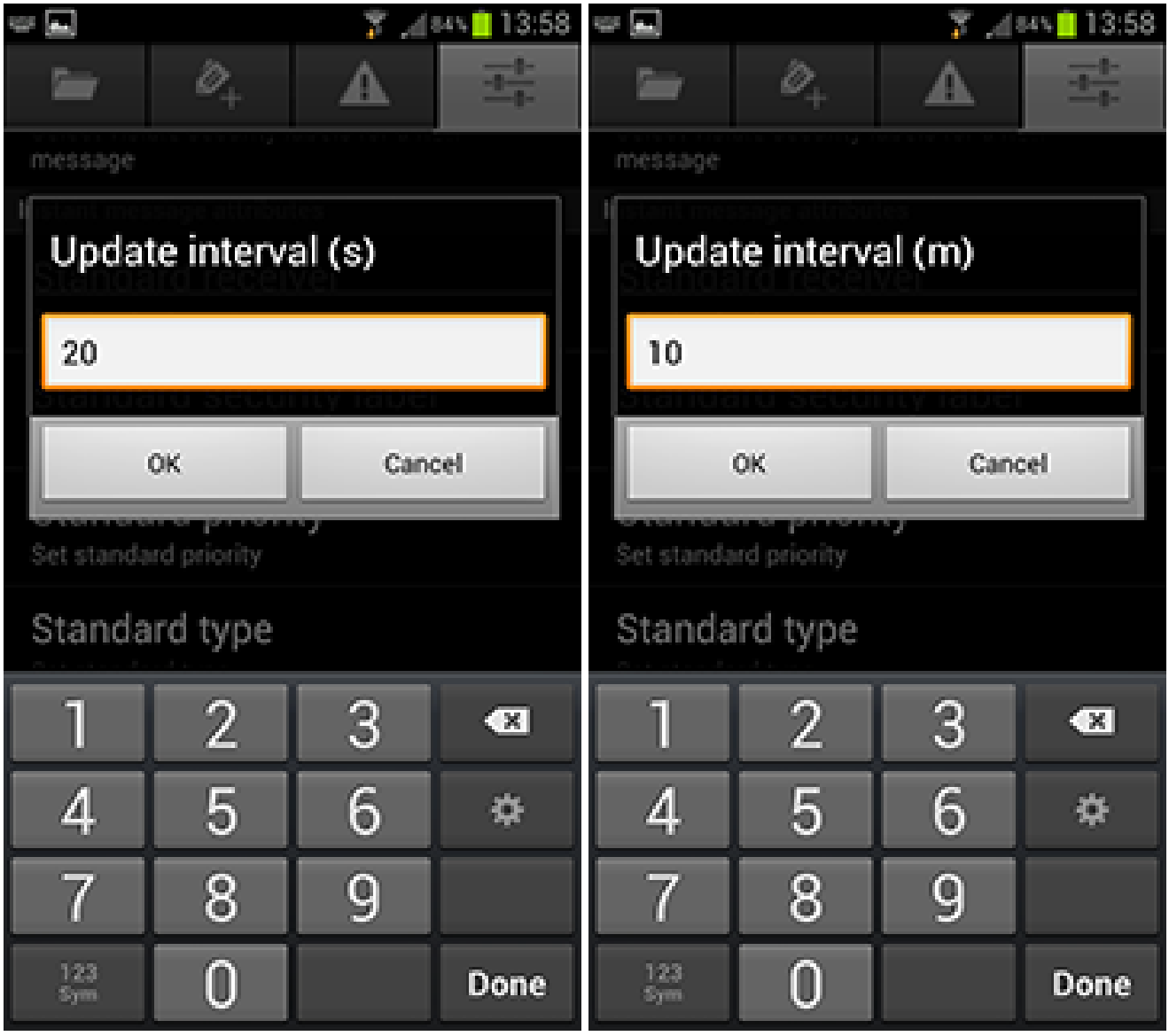


Figure E.4: GPS settings

APPENDIX F

TEMPLATES

Agenda for Advisor Meeting #X

October 2, 2012

F.1 Agendas

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-MM-DD HH:MI
Place	ITV-464
Attendees	Mohsen Anvaari from IDI
	Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Customer Meeting #X

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-MM-DD HH:MI
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last customer meeting

3 Case name

4 Other issues

5 Next meeting

Agenda for Internal Meeting #X

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-MM-DD HH:MI
Place	?
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last internal, customer and advisor meeting

3 Case name

4 Other issues

5 Next meeting

F.2 Weekly Status Report

Weekly Status Report #X

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Written by	Group 15
Week	Week WW
Dates	2012-MM-DD - 2012-MM-DD
Referent	Kristin from Group 15

1 Activities done this week

2 Activities planned next week

3 Challenges ahead

4 Status summary

5 Milestones

Name	Planned	Real	Result

F.3 Minutes

Minutes of meeting for advisor meeting #X

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-MM-DD HH:MI
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Minutes of Meeting for Customer Meeting #X

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-MM-DD HH:MI
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last customer meeting approved

3 Case name

3.1 Sub item

3.2 Sub item

- Point
- Point

4 Other issues

5 Decisions

6 Actions

7 Next meeting

Minutes of Meeting for Internal Meeting #X

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-MM-DD HH:MI
Place	?
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last internal, customer and advisor meeting approved

3 Case name

3.1 Sub item

3.2 Sub item

4 Other issues

5 Next meeting

APPENDIX G

AGENDAS

G.1 Advisor

The following pages include all agendas for advisor meetings.

Agenda for Advisor Meeting #2

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-04 11:15
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created

- Project plan
- Risk plan
- Project report

Meetings

- Meeting with customer 24.08
- Internal meeting 27.08 (planning sprints)

Other activities

- The whole group worked together 28.08 and 29.08 (approx. 6h each day)
- Individual work (pre-studies) in week 35
- Course on testing and agile development 03.09
- Internal meeting 03.09

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

- Have had some problems setting up tools and repository access for some group members
- Is it OK to use last years project delivery as a template for our project?

4.4 Planning of work for the next period

Meetings

- Meeting with customer 05.09

Activities

- Continue with sprint 1

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #3

September 18, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-20 11:30
Place	?
Attendees	Mohsen Anvaari from IDI Kristin, Lars og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created

- Project plan
- Project report

Meetings

- Meeting with customer 05.09 about the sprint planning and demo of their software

Other activities

- The whole group worked together 10.09, 11.09, 12.09 and 18.09 (approx. 6h each day)
- Individual work (pre-studies and programming) in week 36 and 37
- Scrum meetings at the beginning of each work day (Mon-Wed)

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

- Have had some problems setting up tools and repository access for some group members

4.4 Planning of work for the next period

Meetings

- Meeting with customer 19.09, where we will run a demonstration of the result from sprint 1

Activities

- Finish sprint 1 16.09
- Start sprint 2 17.09

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #3

September 23, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-25 11:15
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #4

September 28, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-02 11:15
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #5

October 8, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-09 11:15
Place	ITV-464
Attendees	Reidar Conradi from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #6

October 15, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-16 11:15
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #7

October 22, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-23 11:15
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

Not applicable, as the group did not have a customer meeting last week.

4 Approval of the status report

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Agenda for Advisor Meeting #8

October 29, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-30 11:15
Place	ITV-464
Attendees	Mohsen Anvaarii from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 **Approval of agenda**

2 **Approval of minutes of meeting from last advisor meeting**

3 **Comments to the minutes from last customer meeting**

4 **Approval of the status report**

5 **The group's internal evaluation**

6 **Review/approval of attached phase documents**

7 **Other issues**

8 **Next meeting**

Agenda for Advisor Meeting #9

November 5, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-11-06 11:15
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last advisor meeting

3 Comments to the minutes from last customer meeting

4 Approval of the status report

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

G.2 Customer

The following pages include all agendas for customer meetings.

Agenda for Customer Meeting #3

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-05 12:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales
	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last customer meeting

3 Sprint planning

- We have planned our project in JIRA, and have created a user for you. Christian will receive an e-mail with username and password to the project
- Agree on the sprint planning
- Agree on backlog and priorities
- Agree on task estimation

4 Other issues

5 Next meeting

Agenda for Customer Meeting #4

September 17, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-19 12:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last customer meeting

3 Sum-up of sprint 1 with demonstration

4 Backlog discussion

4.1 Priorities

5 Sprint 2

5.1 What should be included in sprint 2

5.2 Agree on goal from sprint 2

6 Software vs. documentation

What do you want in the documentation?
How much of the time do you expect us to use on the software vs. documentation?

7 Feedback on GUI

8 Other issues

9 Next meeting

Agenda for Customer Meeting #5

October 8, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-10 12:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last customer meeting

3 Sum-up of sprint 2 with demonstration

4 Comments on report delivery

5 Sprint 3

5.1 What should be included in sprint 3

5.2 Agree on goal for sprint 3

6 Other issues

7 Next meeting

Agenda for Customer Meeting #6

October 22, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-24 12:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last customer meeting

3 Status update on sprint 3

4 Connection to XOmail server

5 Keystore

Can the group get access to a key set (with private and public key) that can be verified at Thales' side, so that we can set up a connection?

6 Other issues

7 Next meeting

Agenda for Customer Meeting #7

October 29, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-31 12:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last customer meeting

3 Sprint 3 sum-up with demo

4 Sprint 4 planning

5 Other issues

6 Next meeting

G.3 Internal

The following pages include all agendas for internal meetings.

Agenda for Internal Meeting #1

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-08-27 12:15
Place	?
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last internal, customer and advisor meeting

3 Group role delegation

- Project leader
- Scrum master
- Room booking
- Agendas and minutes
- Contact person against advisor and customer
- Document responsible
- Architecture responsible
- Testing responsible
- Technical responsible
- Quality assurance responsible

4 Weekly meetings

We need to schedule weekly work times

5 Backlog

- Write backlog
- Prioritize backlog

6 Sprint planning

7 Create templates

8 Risk planning

9 Organizing meetings with Thales

10 Use of software

11 Other issues

12 Next meeting

Agenda for Internal Meeting #2

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-08-28 08:15
Place	?
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last internal, customer and advisor meeting

3 Create document templates

- Create templates as described in 2012-08-27-Minutes-Internal
- Alter existing documents to correspond with the templates

4 Start with documentation

5 JIRA organizing

- Start creating tasks in JIRA

6 Risk planning

7 Other issues

8 Next meeting

Agenda for Internal Meeting #3

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-03 12:15
Place	?
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last internal, customer and advisor meeting

3 Write agendas for next meetings

- Finish agenda for meeting with Thales
- Finish agenda for meeting with Mohsen Anvaari
- Translate all agendas and other document

4 Reorganize group roles

5 Structuring internal group meetings

- Report to Scrum master if you are late
- Stand up at the beginning of the work day
  - What have you done?

  - What will you work with now?
  - What do you need to make it work?
  - Who needs to work together the next days?
- How often is it OK to be late?

6 Estimation of all tasks

- Definition of what a finished task is
- Estimate in hours for all tasks in current sprint

7 JIRA

Added task under Meetings for lecture in Customer-Driven project. Show everybody where it is.

8 Git

Ida wants to share her frustration regarding Git

9 Other issues

10 Next meeting

Agenda for Internal Meeting #4

September 17, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-18 08:15
Place	R30
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last internal, customer and advisor meeting

3 Set ground rules for the project

1. Everybody must work (on average) at least 20 hours per week.
2. One must have a good reason (things that are required) to be late, leave early or be absent.
3. Deadlines must be respected, both internal and external, even if this requires extra work.
4. Agendas, minutes and other documents that must be sent Monday 14:00 should be ready before the end of Friday. The rest of the group must review the documents and give comments before Sunday evening.
5. It is understandable that other courses also take time, but this is the same for all of us. It might require us to work after school and in weekends.

6. If somebody needs help, is frustrated or is stuck, that person is responsible for telling the project leader or the group.

4 Role discussion

- Are people happy with the role allocation? There is an uneven distribution of roles.
- Everybody must probably participate in many different tasks (especially towards the end)

5 Sprint 1 review

- What is the status of the demo?
- What was good in sprint 1?
- What could have been done better in sprint 1?

6 Sprint 2 planning

- Suggest sprint plan (Thales must be integrated more)

7 Comments from Thales

- Lacking point in backlog
- Wants discussion on priorities before sprint 2 planning

8 Other issues

- Documents on Dropbox
- Issue log
- Frustrations and problems
- What do you think is important to prioritize next?
- What makes you happy?

9 Next meeting

Agenda for Internal Meeting #5

September 28, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-01 12:15
Place	L-Galleri 5
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Approval of agenda

2 Approval of minutes of meeting from last internal, customer and advisor meeting

3 Work hours

We have now worked two weeks since we decided on the hours per week. How did people do with regards to the required hours per week?

4 Pre-delivery and documentation in general

Ida will explain the status of the project report and what must be done before the predelivery 14.10.

5 Progress of project

How are we doing? The burndown chart for sprint 2 does not look good! Sprint 2 will finish this week, and we have a lot to do.

6 JIRA tasks

- Everybody needs to update the status of the tasks, e.g. if they have started a task or finished it.
- Some of the tasks need a better explanation, e.g. "Back-end documentation" - what does this task involve?
- We should try to outline better in the log what we have done on an implementation task, so others can step in if the task is not finished.

7 Other issues

8 Next meeting

APPENDIX H

MINUTES

H.1 Internal

The following pages include minutes from all the advisorl meetings.

Minutes of meeting for advisor meeting #1

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-08-22 15:15-15.30
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created
Meetings
Other activities

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

5 Review/approval of attached phase documents

6 Other issues

- The group must read the compendium to know what is expected and what should be delivered.
- Weekly meetings are scheduled to take place at Tuesdays from 11-12. The agenda for the advisor meeting should be sent to our advisor at least the day before the meeting takes place. The agenda should containt which problems or questions the group want to discuss at the meeting.
- After each meeting the group must write a minutes of meeting.
- The group has a pre-delivery at October 14th, where some parts of the final report must be ready.
- The group must create a risk plan with concrete and practical solutions, a work-breakdown-structure and a project plan.
- The final document is really important. If the group runs out of time and some things could have been implemented better or have been omitted, this should be documented.

7 Next meeting

Tuesday 2012-09-04 at ITV-464

Minutes of meeting for advisor meeting #2

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-04 11:15-11.30
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

The last customer meeting was mostly a discussion of details of the requirements document the group have received from Thales. The group will send this document to Anvaari with the next minutes of meeting.

4 Approval of the status report

Anvaari thinks the project plan delivered is very high-level, and group 15 will create a more detailed project plan before the next advisor meeting. The group has made a product backlog and decided tasks to be contained in sprint 1.
Concerning the risk plan, Anvaari recommends the group to have a plan for

when a problem occurs, not just how to avoid it. The group may need to have several people responsible for a risk, e.g. if the first responsible person is absent.
Anvaari's advice was to write documentation early.

5 Review/approval of attached phase documents

6 Other issues

- The group wondered whether it is OK to use previous years' reports as inspiration for headers and organization. Anvaari responded that the group might look at the reports to get an idea what can be included, but it is probably not a good idea to copy it uncritically. The reports may have missing parts and different focus from the group's project.
- Anvaari gave the advice to document both product and process. Similar solutions must be explained, e.g. in a table to compare the different solutions and their strengths and weaknesses.
- The group is going to program in Java for the Android platform

7 Next meeting

Tuesday 2012-09-18 11:15 at ITV-464

Minutes of meeting for advisor meeting #3

September 28, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-25 11:15-11:30
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

4 Approval of the status report

4.1 Summary

4.2 Work done in this period

Status of the documents that are being created:

Group 15 thinks that they are ahead of the schedule when it comes to documentation. The group will make architecture (probably according to the 4+1 method) within the next advisor meeting.

4.3 Problems

What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

The challenge is to create architectural views and implement them.

4.4 Planning of work for the next period

Meetings
Activities

4.5 Other

- Task distribution: All group members can pick available tasks at JIRA, but someone is working more back-end and some more front-end. We have one person responsible for the documentation, but everybody must write their parts.
- Our document review procedure is that everybody reads the documentation and make comments in Google Docs, before we convert the documents to Latex.
- The group has written a test plan, although this is a high-level plan.
- The group asked Anvaari if they can add content after the pre-delivery. Anvaari responded that we could very well do that, as the pre-delivery is to show that we are on the right track.
- The problem description can be very high-level, perhaps a paragraph.

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Minutes of meeting for advisor meeting #4

October 2, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-02 11:15-12:00
Place	ITV-464
Attendees	Reidar Conradi from IDI Aleksander, Ida, Kristin, Lars, Magnus og Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

Not applicable as the advisor had not received the minutes.

3 Comments to the minutes from last customer meeting

Not applicable as the advisor had not received the minutes.

4 Status

4.1 Organization

- The group has customer meetings approximately every second week. The rest of the communication goes through email.

- The customer is mostly interested in meeting us at the start of a sprint to participate in the planning and at the end of a sprint to see the sprint result.
- The work load is almost evenly distributed. The group has decided on some ground rules about how many hours each group member should contribute with each week. Some group members are more interested in getting a good grade than others, so there will always be someone who works more than others.
- The group has assigned a document responsible that keeps an overview over what has been and must be written. She delegates the writing tasks to group members that are responsible for the separate areas. Each part of the whole project report is correction read at least once.

4.2 Tips from the advisor

- LaTeX has the opportunity to use a macro log. The group will consider using this.
- The advisor finds it demanding to maintain two text formats, as the group uses both Google Docs and LaTeX.

4.3 Project mood and moral

The group finds the project to be a bit exhausting, as there is always something to do and we also have other courses. The advisor commented that it often turns out to be more work than expected and that the group should try to have some slack where possible.

5 Review/approval of attached phase documents

The advisor gave comments on the project report and delivered this to the group.

6 Other issues

7 Next meeting

Minutes of meeting for advisor meeting #5

October 9, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-09 11:15-12:00
Place	ITV-464
Attendees	Reidar Conradi and Mohsen Anvaari from IDI Aleksander, Ida and Kristin from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

Not applicable as there has not been a new customer meeting since the last advisor meeting.

4 Status report

4.1 Summary

The document has been restructured from the last meeting. It has also been restructured after this week's delivery, to get more and shorter chapters instead of long and unclear chapters.

Regarding the application, the focus lately has been on the GUI. The group

has tried to make the application a bit more easy on the eye. The application is now able to list incoming and outgoing messages, and to send, reply and forward a message. All messages have the three important attributes: Security label, priority and type.

4.2 The advisors' questions and comments

- Anvaari asked whether the group has documented the security studies and the process flow of the user interface. The group responded that this is included in the report, which will be sent on email to Anvaari right after the meeting.
- Anvaari asked when the group thinks they will be done with the predelivery, and the group responded that they will try to have it ready by Friday 12th.
- The group works together 18 hours per week, the rest (approximately 2-7 hours per person) is done in pairs or alone. The group meets six hours every Monday, Tuesday and Wednesday.
- Conradi asked how the group has organized the backlog and plans the coming sprints. The group responded that the backlog contains high-level user stories, that are expanded into several tasks when they are included in a sprint.
- Anvaari commented that the status report for last week just said "almost everything has been implemented", and stated that the group should detail what tasks were not finished.
- Conradi commented that the report does not include any documentation about how the group will test the usability or any rules that the GUI should follow.
- Conradi asked whether the group will implement a search function in the message folders. The group responded that they want to implement a basic search function, but it is not decided when this will be done.
- Anvaari asked what the future challenges with the implementation will be. The group said that saving of password (for e.g. auto login) and the saving and encryption of data on the phone are the most complex parts. At the moment the mails are sent in a safe way, but none of the other security features have been implemented. The group had to make some assumptions about the phone, e.g. that the phone is not rooted.

- Anvaari asked how the group checks out the different possibilties with regards to the security studies. The group responded that they have done research on the web and in Android/Linux documentation.
- Conradi commented that the testing of a task should be done right away and not be delayed to a later sprint.

4.3 The plan from now on

The group is meeting the customer 10.10, where the demo will be shown. Sprint 3 will be planned based on what Thales thinks is most important.

With the regards to the documentation, the group will spend this week correcting the report according to the advisors' and customer's feedback and preparing for the pre-delivery.

5 Review/approval of attached phase documents

Conradi delivered a copy of the project report with written comments.

6 Other issues

7 Next meeting

Minutes of meeting for advisor meeting #6

October 17, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-16 11:15-11:45
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida and Kristin from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

4 Status report

4.1 Summary

4.2 Evaluation

The evaluation criterias are the report (most important), the product and the presentation. The advisor recommended that the group writes test cases based on requirements and write about the test results in the report.

4.3 Testing

The gorup has written specific tests back-end, but not tests for usability and UI testing. The advisor recommends that the group should have at least unit tests and integration tests, and said that UI testing should be done in collaboration with the customer.

The advisor recommended that the group sets up a comparison table of security measures and their tradeoffs. One might use a template for this.

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Tuesday 23.10 11:15 at ITV-464

Minutes of meeting for advisor meeting #7

October 24, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-23 11:15-11:45
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

4 Status report

- The advisor commented that there was no activity for testing in the status report and asked if the group has a test plan for next week. The group commented that they created 11 test cases today and will create more.
- The advisor commented that the group should make test cases on functional requirements etc. Non functional tests are also important.
- The advisor asked how the group wants to test security and performance.

This week the group has written a lot of code, and done some minor adjustments and getting the functionality right for the presentation for Thales.

4.1 Questions from the advisor

- The advisor asked if the group finds that the architecture has changed a lot. The group responded that there have just been minor adjustments. The basic structure is still there, but it has gotten some more modules.
- The advisor asked about the status of the compression studies. Ida explained the details of these studies.
- The advisor asked what Thales' security concerns are. Nicklas responded that they are concerned about the storage on the phone and the connection. The app use login to get access to the data, encryption on the data and checking what data we receive, to be sure it is not malicious.
- The advisor asked if Thales is more interested in security than the app. The group responded that they are interested in both. We currently have basic encryption, but struggle with the key store and certificate store.
- The group is using a lot of open source projects and articles on security to be able to say something about how to solve our problem.
- The advisor asked how the group will compare the different methods. The group responded that the study is finished now. At this point we are just implementing what we can.
- The advisor asked if the group has changed basic functionality of Android. The group responded that they have not changed any basic functionality of Android, because it would give us more problems than benefits.

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Tuesday 30.10 11:15 at ITV-464

Minutes of meeting for advisor meeting #8

November 2, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-30 11:15-11:45
Place	ITV-464
Attendees	Mohsen Anvaari from IDI Aleksander, Ida and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last advisor meeting approved

3 Comments to the minutes from last customer meeting

4 Status report

4.1 Internal group evaluation

The evaluation shows that some got an average effort score of 21 and another has 4. This shows that there is an uneven distribution of work load in the group and that some group members are not working enough. The group has had a lot of discussions about effort and has tried to take actions to improve the situation. The situation has gotten better, but is still not optimal.

4.2 Status of program

The group has a demo for the customer tomorrow (31.10) and needs to fix some bugs before that.

4.3 Last customer meeting

The advisor asked how the previous customer meeting went. The group answered that the customer seemed pleased with the app. We were able to send and receive messages from their server. The signing and verification will be tested via Outlook, Gmail or such.

The advisor asked whether the group has considered the change of connecting to Thales' server. The group responded that this should (at least in theory) be simple, as it should just be a change of IP.

4.4 Report

- The advisor said that the group should create flow chart diagrams, as these give a better overview of the program.
- The advisor said that the group should write about attachment limit and connection limit of Gmail, e.g. that the size of attachments can not be larger than 50 MB.

4.5 Status of project

The group will not be finished with everything they planned, but the requirements will be either in the app or written about in the documentation.

4.6 Customer testing

The group will give the customer the opportunity to test the app and give feedback. The customer has already provided valuable feedback on the GUI.

5 Review/approval of attached phase documents

6 Other issues

7 Next meeting

Tuesday 06.10 11:15 at ITV-464

H.2 Customer

The following pages include minutes from all the customer meetings.

Minutes of Meeting for Customer Meeting #1

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-08-21 HH:MI
Place	KJL21
Attendees	Christian, Sølve and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

Not applicable

2 Minutes of meeting from last customer meeting approved

Not applicable

3 The project task

The task given from Thales is to make an Android mail client for mobiles (from now on called "the app"), with certain extra services. Thales' work is mainly focused on military areas, but the app can also potentially be used in callouts and search operations.
The main focus is on the GUI, which means that the app should be fast and easy to use, but ideally the focus should also be on security.
The task is twofold: On one side the focus is on developing a prototype and on the other side to explore the possibilities for further development, e.g. what solutions exists and what is required to implement the desired security.

4 Details about the task

The app is supposed to communicate with a SMTP-server via POP3/IMAP protocols. Messages of high importance should warn the user, typically in emergency situations.
Two important features the app should support are priorities of messages and security grading.
The app should also support an address book, but this can be done in several ways. In Thales' current systems, the software can fetch addresses from the server one is connected to. In this project it is possible to assume that an address book exists, but the group need to find a way of fetching this.
The two main features of security the group are supposed to do reasearch and possibly implementation of are signing and verification. However, sometimes it is more important that the messages arrive fast than that they are encrypted.
The messages should have ability to give information about whether the message has been delivered and whether the recipient has read the message. The question is if the group needs to deal with this or if the functionality possible comes with the IMAP protocol. IMAP has certain issues when it comes to bandwidth, so the group must consider other push solutions.
The group needs to document the choices being made and what third party code they want to use. It is important to keep track of the licence agreements.
It is important to think about whether the phone is private or locked, routed or not.

5 Other issues

6 Decisions

7 Actions

1. Thales will send suggestions of a requirements document on mail.

8 Next meeting

Friday 2012-08-24 12:15 at Lerkendal

Minutes of Meeting for Customer Meeting #2

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-08-24 12:15-13:00
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales
	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

Not applicable

2 Minutes of meeting from last customer meeting approved

3 Requirements

3.1 Address Book

A protocol for addressing is LDAP (Lightweight Directory Access Protocol). The group can assume that the address book comes as a list in a text file. Relevant information for a contact can be name, e-mail address, capabilities of their device, phone number. The group should investigate the possibilities for attaching attributes to each address.

3.2 Goal with the project suggested by Thales

- Test user interface for message-based communication on a handheld device

- Make a prototype that can give inspiration for further product development and be used as a demonstration internally and to customers.

3.3 Details about messages

The message size must be considered and file attachments must be supported. One can imagine limiting what content the user can send based upon network connection and type. Some attributes can be left out of Thales' format.
The group asked whether it is required to implement threaded messages (a conversation), but Thales responded that this would be nice-to-have, although not necessary.
Thales detailed the different priorities the messages can have, and these are (in increasing order of importance) deffered, routine, priority, immediate, flash, override. FLASH and OVERRIDE have the highest priority and must be managed in a special way.
The group asked whether the subject or sender were most important, but Thales thought that they were equally important.
All security gradings must be selectable (NATO, English, Norwegian), but only one can be selected per message.
The security labeling is coded on a special format. Thales will send valid headers for each of the relevant security labels, since this would require a lot of code to parse.
An "outbox" (sent messages) is important because of the message status. The different statuses are whether the message is delivered and read.
The group may investigate the possibility for sending SMS when there is no network available.

3.4 Encryption

The relevant protocols and standards are OpenSSL/SSL and S/MIME. The signing and verification will disappear in Thales' system, so the group should rather test through NTNU's servers or alike. It is possible to inspect encrypted network traffic in Wireshark. However, this is not straight-forward and Stig may be of help in case of problems.

4 Other issues

5 Decisions

1. Questions and meeting agendas must be sent 1-2 days prior to the meeting, and the minutes of meeting as soon as possible after the meeting.

2. If something must be done (a required action), write what should be done and who should do it.

6 Actions

1. Thales will send valid headers for each of the relevant security labels.

7 Next meeting

Group 15 will send suggestions for the next meeting on e-mail.

Minutes of Meeting for Customer Meeting #3

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-05 12:15-13:00
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last customer meeting approved

Thales had some questions about the minutes, but they were resolved through mail exchange with the referent.

3 Sprint planning

3.1 Comments to JIRA

Thales had sent some comments to group 15 about the content in JIRA prior to the meeting. Their main concern was that they could not see a general, overall plan of how group 15 wants to reach the goals of the project. Thales also thought that tasks concerning pre-studies/analysis were missing in JIRA. Group 15 has to make decisions about choice of technology, use of third party code and design choices, at least at the end of sprint 1.

3.2 Backlog document

The discussion continued to the backlog document from Group 15. Thales wanted Group 15 to decide on the goal with each sprint, as both the developers and customer then has something concrete to relate to.
Thales wanted BL-5 (Message template) to be given less priority.

3.3 Estimate of tasks

Thales thought the estimates for the tasks might be a little optimistic. Group 15 will change to using story points instead of time estimates, as it is probably easier to estimate tasks in complexity instead of in hours, and it is also closer to the Scrum methodology. Group 15 will therefore make an updated version of the backlog and send this to Thales by October 13th.
Thales first impression of the backlog document is that it seems OK, and will send further comments by e-mail.

3.4 Presentation of architecture

Nicklas presented the current design, including interfaces and patterns. Thales will provide further comments on mail.

4 Other issues

4.1 XOmail demonstration

Thales showed a demo of their XOmail client. Group 15 found it interesting to see the complete version of the XOmail. However, the demo was very quick and the program complex and with a lot of functionality, and Group 15 does not see how we can use this experience at the moment.

4.2 Password storing

The discussion continued with details about storing passwords. Group 15 needs to find a smart way of storing passwords, e.g. find ways to get the OS to support it.

5 Decisions

1. Further questions and comments will be exchanged by e-mail until the next meeting.

6 Actions

1. Group 15 will send digital versions of the backlog and architecture document to Thales.
2. Thales will send further comments about the backlog and architecture document to Group 15
3. Group 15 will send an updated version of the product backlog with priorities by October 13th.

7 Next meeting

Wednesday 2012-09-19 12:15 at Lerkendal

Minutes of Meeting for Customer Meeting #4

September 21, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-12 12:15-13:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last customer meeting approved

3 Sum-up of sprint 1 with demonstration

Group 15 admits that the first sprint was a process of trying and failing, but with a very steep learning curve. The group will take the new knowledge and experience and hopefully be more productive in the upcoming sprints.

Group 15 showed a demonstration of the sprint result, which was a very simple application that could send a message, show all received messages in a list and open a message from the list. The group has accomplished the basic functionality of sending and receiving messages, but the user interface has not been the focus and is therefore very basic.

4 Backlog

4.1 Address book

Thales suggested that the group can embed the address book as a list or a text file for now. A solution for later might be to send the adress book in special management messages that need to be recognized and handled by the application.

4.2 Security

Thales has called for a better documentation, especially of the security part of the project. The group needs to think about e.g. how the signing of messages and securing of keys and password can and will be done. Thales has given URLs to documentation that the group can study further

Thales gave the advice to state reasons for choice of design and architecture, preferrably with requirements as the basis.

Thales is mostly interested in documentation of the limitations of Android and what would need further development if they were to create a similar application.

4.3 Scrum tips

Thales made a comment that user stories may be split into multiple stories when a user story is too large to fit into a sprint, or if all its related functionality is considered to have a different priority. In Scrum, a task is either finished or not finished, in a sprint or not in a sprint.

Thales commented that "Open message" and "Send message" should be separate tasks.

4.4 User interface details

- All views should include a security label at the top right. The labels are red or black. Unclassified levels are black, higher classifications are red. The color is included in the message headers and Thales will send examples of this on mail.
- Classification (not grading!) should always be visible in the upper right corner.

- The choices for priorities, classifications and types are static.
- Thales suggests that regular e-mail messages is omitted from the project. Standard e-mails have different priority levels (e.g. lowest, low, normal, high, highest) than military messages and differences in the headers. Adding support at a later time should be simple.
- Operation and routine are standard type and priority. The default values for the different attributes may be included in a preference page.
- Attributes that should be included in the list of messages are From, Subject, Priority (perhaps a symbol), Label (short format, Thales will give examples), Time (DTG format).

5 Sprint 2

Thales wanted completion of the opening/reading and sending of a message to be highly prioritized. The group need to pick the most important tasks (based on how much they think can be done) from what Thales thinks is most important.

6 Software vs. documentation

It is important that the group documents assumptions and reasons for design choices and tradeoffs. Thales is interested in the software, but will only get the full value of it if the group has made documentation about the different opportunities and the reasons for decisions made.

7 GUI prototype

Group 15 has made a simple prototype that is available on the following URL:
`https://www.fluidui.com/editor/live/preview/p_u7V8Fx6ipDz6ZCHBLbR5dLTJVkGsE1TF.1347888261683`
NB: Backwards navigation can be done by pressing Backspace key!

8 Other issues

9 Decisions

10 Actions

1. Thales will send information about security label coloring by mail.

11 Next meeting

Wednesday 2012-09-26 12:15 at Lerkendal (tentative, the group will notify Thales on Monday if the meeting will take place)

Minutes of Meeting for Customer Meeting #5

October 11, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-10 12:15-13:15
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last customer meeting approved

3 Sum-up of sprint 2 with demonstration

3.1 The result

The result from sprint 2 is a functioning application with an improved user interface. The relevant attributes (label, priority and type) have been added to a message. The group has also done a lot of work with researching and documenting security issues.

3.2 Demonstration

The group showed the current version of XOXOmail. There is currently a tab menu at the top with the choices to view folders, write a new message and show contacts.
In folders one can select to view inbox or sent messages as a list where each element has the following attributes: sender, subject, date, label (short

version) and priority (number). One can click at a message and view the mentioned attributes (in full version) and the message body. One can navigate to the next and previous message in the list via buttons.
In the new message view one can set all the relevant attributes. One has to make a choice for the security label, but the default values for priority and type are set to routine and operation. Currently it is not allowed to send a message with an empty subject field or message body.
The contacts list is currently implemented as a sorted list of created email addresses, and nothing happens when one tries to click on a contact item.

3.3 Comments on the application from Thales

- Thales said it should be possible to have an empty message body and subject.
- Thales said that the group should not care about whether a person has the right classification to read a message. This will be fixed in Thales' systems.
- Thales said that the name at the top of the application, XOMail, should be written XOmail. The group is aware of that and will change it to the project name (XOXOmail) or remove the line completely.
- Thales thinks the menu tabs are too big. It might be an idea to remove the text and just keep the icons.
- Thales said the security labels should be with capital letters and non-breaking space.
- Thales asked about an update button in the inbox. The group responded that this is on the way.

3.4 What is next?

3.4.1 Connecting to Thales' system

The group should soon try to connect to Thales' system and send messages through their system. Christian will set up forwarding from kpro@xomail.com to kprothales@gmail.com.

3.4.2 Password storing

There are several options of how to store the password. One option is to give the user the possibility to say that the application should remember the password as long as the phone is on, but this might require restricting the use of other applications and locking of the phone with e.g. PIN code. If the user does not want this he must type the password each time he opens it.

3.4.3 Miscellaneous

- The group suggested to have a widget for the instant message, and Thales was not against this idea.
- Thales thinks it should be possible to set up standard receivers in the settings.
- Thales suggested that it could be possible to include a "Share with XOXOmail" option e.g. when you take a picture. It could be possible to also have "Share with XOXOmail standard receiver".

4 Comments on report delivery

None yet.

5 Sprint 3

As previously mentioned, the group should try to send messages via XOmail very soon, both with and without attachments. To test the application against Thales' client is an important task.

The group should try to set up Wireshark and investigate what uses data traffic. One can set up monitoring of everything that goes in and out of the device and filtrate on port number. The group should look at used bandwidth and overhead, as these numbers will determine the scope of the work with compression. There should be an upper size limit of when the application downloads a message and also a maximum number of messages in the inbox.

The following points were gone through in the backlog (from JIRA):

1. **Login:** Should be included in sprint 3.

2. **Address book:** The implementation of an address book is likely to be postphoned to sprint 4. It would be nice to have autocomplete when the user starts to write an address.

3. **Reply and forward:** This should be included in sprint 3 and requires a reference to the message ID. The security label is set from the message one replies to or forwards and cannot be changed.

4. **Create secure communications channel:** Nicklas explained that this comes with the use of Gmail and suggested that this task will be deleted.

5. **Extend communication interface:** Likely candidate for sprint 3.

6. **Hardware abstraction layer:** Postphoned to sprint 4.

7. **Signing of messages:** The group should start with this in sprint 3. It might not be possible to finish it this sprint, but the group should look at it and maybe have the API ready. Thales' server does not have signing yet, but the technical solution is still interesting.

8. **Compression:** The group should look at the different possibilties. A military standard is setlink (?).

9. **Settings:** Possible settings could be what security labels the user wants to use, set standard receiver and if the application should use "Idle" or move to some "pull each X minutes" instead.

GUI changes must be added as tasks.

As a sum up, the overall goal of sprint 3 is to send and receive messages via XOmail with and without attachments.

6 Other issues

7 Decisions

8 Actions

1. Christian will set up forwarding of mails from kpro@xomail.com to kprothales@gmail.com.

9 Next meeting

Wednesday 2012-10-17 12:15 at Lerkendal.

Minutes of Meeting for Customer Meeting #6

October 25, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-24 12:15-13:00
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales Aleksander, Ida, Kristin, Lars and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last customer meeting approved

3 Status update on sprint 3

The group has realized that we probably will not get finished with everything we planned, but we will work hard to implement what is left.

The different points that were discussed are listed below:

- The app now runs on different threads as it should, and just the GUI uses the UI thread.
- Storage is now OK.
- Thales' server does not support signing, and the group should test it through Gmail's server or alike.
- Signed messages will not be stopped in Thales' system.

- Encryption of mail can be complicated. In Thales' system the link is encrypted and not single messages. In the case of our app, it might be interesting to look at as the underlying link might not be as reliable.
- It is possible to encrypt for a certain receiver, e.g. the SMTP server.

4 Connection to XOmail server

The group has tried to send messages to kpro@xomail.com and has gotten messages through. Nicklas did not receive a message when he tried, but this seems to have been caused by a loop detector.

5 Keystore

Not relevant as Thales' system does not support signing.

6 Other issues

The group should try to send in via Thales' SMTP server. Nicklas commented that this should be straightforward if Thales' server is similar to Gmail's server. Christian will set up and provide login information so that the group can send messages.

We tried to send a message with the app that actually got accepted in the XOmail system with the correct attributes!

7 Decisions

8 Actions

Christian will set up so that the group can send messages in via the SMTP server.

9 Next meeting

Wednesday 2012-10-31 12:15 at Lerkendal (lunch 11:15)

Minutes of Meeting for Customer Meeting #7

November 2, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-31 12:00-13:00
Place	Thales' offices at Lerkendal
Attendees	Christian and Stig from Thales (in addition two other employees from Thales attended the demo) Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Demonstration of XOXOmail

The group showed a demonstration of the XOXOmail app, including sending and receiving messages from Thales' system. The app managed to send an image that was received in Thales' system, but something went wrong when receiving a message with an attachment from Thales' system.

2 Sprint 4

The group will give the documentation the most focus, but will also include some minor implementation tasks in sprint 4:

- Receiving attachments from Thales' system
- Correct mapping of message priority, Thales will send these by email
- "Share with XOXOmail"-option in other applications (at least one, as it might be complicated to make this function dynamic)
- Bug fixing
- Signing has been started on, but it will be up to the group if this task should be continued

3 Guide for running the app

The group will create a guide for what you need to have and do to run the application.

Thales will try the app and give feedback about bugs etc. by email.

4 Decisions

5 Actions

Christian will send the priority abbreviations for the group to use (already done in the time of writing)

6 Next meeting

Wednesday 2012-11-14 12:15 at Lerkendal. This is tentative date for "rehearsal" for the final presentation.

H.3 Internal

The following pages include minutes from all the internal meetings.

Minutes of Meeting for Internal Meeting #1

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-08-27 12:15
Place	ITV-464
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last internal, customer and advisor meeting approved

3 Group role delegation

- Project leader: Aleksander
- Scrum master: Aleksander
- Room booking: Ida
- Agendas and meetings: Kristin
- Contact person against advisor and customer: Aleksander
- Document responsible: Ida
- Graphical responsible in the project report: Nicklas
- Arcitecture responsible: Nicklas
- Test responsible: Magnus

- Technical responsible: Nicklas
- Quality assurance: Nicklas
- Design/GUI: Kristin
- Lead programmer: Magnus

4 Use of software

- We need to decide on standard programs for all tasks
- The group members should look at different possibilties, pick their favorites and we will decide along the way
- Graphviz
- Cacoo (diagrams)

5 Work times

- Monday 12-18
- Tuesday 8-14
- Wednesday 12-18

6 Backlog

- Phases: Tasks, In Progress, Ready for testing, Done
- Backlog with priorities in separate document

7 Sprint planning

- 4 sprints of 3 weeks each

  27.08-16.09

  17.09-07.10

  08.10-28.10

  29.10-18.11

8 Other issues

- Ida gets familiar with the documentation, starts delegating tasks
- Everybody must document what they are responsible of!
- Important dates: Lectures (find dates)
- The group agreed to pay for JIRA and GreenHopper
- Aleksander sends time suggestions for meetings with Thales

  Friday 14-15

  Monday 12-13

  Wednesday 12-13
- Subdued until tomorrow

  Write templates

  Risk planning

  Further sprint planning

9 Next meeting

Minutes of Meeting for Internal Meeting #2

September 14, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-03 12:15
Place	?
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last internal, customer and advisor meeting approved

3 Write agendas, work status

What has been done

- Kristin: GUI (simple screens, can soon be used)
- Nicklas: Service part
- Magnus: Save to disk, makes it simple first, save file structure later
- The group need model classes (decide together?)

Today: Translate all agendas and documents into English.

4 Group roles

- Ida becomes document responsible
- Kristin has responsibility for GUI

5 Structure

- Common meeting at first, split up after
- Everybody meets at the time we have decided
- If you are going to be late, tell Aleksander or write on Facebook group page at least 30 min prior to the meeting and try to be there as soon as possible!
- NOT OK to be late several times
- Standing Scrum meeting

  What have you done?

  What are you going to do today?

  What do you need for this task?

  Max 10 minutes

  Does not to be in the morning

6 Estimate of tasks

- What is "Done"?
- JUnit examples: Nicklas puts out a link to this
- Will go through tomorrow
- Hours for tasks in current sprint

7 JIRA

- Log hours for the lecture today

8 Git

- Use Git Extension if you want

9 Other issues

10 Next meeting

Tuesday 2012-09-04 08:15

Minutes of Meeting for Internal Meeting #4

October 1, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-09-18 08:15-09:30
Place	R30
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last internal, customer and advisor meeting approved

3 Agreed ground rules

1. Everybody must work (on average per two weeks) minimum 20 hours per week, except Magnus who must work 17 hours per week.

2. You must notify the group the day before if you know that you will be absent from the work hours we have set. The project leader has the right to ask for a reason for the absence.

3. Deadlines must be respected, both internal and external, even if this requires extra work.

4. Agendas, minutes and other documents that must be ready on Monday should be finished before the end of Friday (preferrably Wednesday, but cases may arrive later in the week as well). These documents will be sent to the group by mail and must be reviewed and commented before Sunday evening.

5. If somebody needs help or is frustrated, then that person is responsible for telling the person(s) it applies to, the project leader or taking it up with the group.

6. Everyone must log work (how much and what) on JIRA before they leave. Aleksander will remind everyone at the end of the day.

4 Role discussion

Most people are happy with their roles. Lars will help Kristin with the agenda and minutes. The group roles do not mean that you alone have the responsibility for doing everything, but the overall responsibility for getting it done.

5 Sprint 1 review

- The demo will (hopefully) be finished during the day.
- Positive in sprint 1

  The internal meetings are open and constructive.

  The user stories we created were OK.

  We have learned a lot about Scrum and the different frameworks we have planned to use. Some mistakes were made in the beginning, but we managed to take corrective actions.

  The project report has good progress.

- Negative in sprint 1

  We made some mistakes in the planning phase and had to replan and reestimate (create user stories and storypoints instead of tasks and hour estimates).

  We have missed some deadlines regarding documents to the advisor and did not know until last week that we are supposed to hand in status reports and table of effort registration weekly.

  We might have too much bureaucracy, could probably focus our discussions more and become more productive when we meet.

6 Sprint 2 planning

We postphone the planning until after the customer meeting tomorrow.

7 Comments from Thales

Kristin will send the feedback from Thales on the backlog on mail.

8 Other issues

- Agendas and minutes have been converted to Latex documents and can be found on Dropbox. The documentation should still be written on Google Docs.
- A document on Google Docs (Weekly Routines) outlines what must be done and delivered each week of documents.
- Frustration and problems: Nicklas feels that he gets to little done as he must help everyone else.
- Documentation is the most important for the grade, but we must make the customer happy as well. We think pre-studies and analysis is the most important documentation for the customer.
- Nicklas, Lars and Kristin will meet Mohsen Thursday 11:30. Place will be sent on mail.
- We have created a mailing list, so everyone need to check this often!

9 Next meeting

Minutes of Meeting for Internal Meeting #5

October 1, 2012

Project name	Formal and secure messaging on a mobile platform
Calling by	Group 15
Time and date	2012-10-01 12:15-13:00
Place	L-Galleri 5
Attendees	Aleksander, Ida, Kristin, Lars, Magnus and Nicklas from Group 15
Referent	Kristin from Group 15

1 Agenda approved

2 Minutes of meeting from last internal, customer and advisor meeting approved

3 Work hours

Half of the group members have worked the agreed number of hours or more. Everybody knows how they are doing, and will try to improve the numbers this week.

120 hours this week should be achievable.

4 Pre-delivery status

Ida explained that we have finished almost all documents that should be included in the pre-delivery. The missing points are architecture and licenses (Nicklas will take care of these), in addition to minor lacks in some other documents. Certain documents also need some more correction reading.

5 Project progress

- Implementation

  Inbox: Lacking icons/text for label and priority

  Read message: Colors must be set in code

  Reply/forward/delete: Not started

  Menu: We will use Android's embedded menu element.

- Report

  Limitations of Android system will be difficult

6 JIRA points

- Everybody needs to update the status of the tasks, e.g. if they have started a task or finished it.
- Some of the tasks need a better explanation, e.g. "Back-end documentation" - what does this task involve?
- We should try to outline better in the log what we have done on an implementation task, so others can step in if the task is not finished.

7 Other issues

8 Next meeting