Department of
Computer and Information Science

TDT4290 Customer Driven Project

USNeuroNav

SINTEF

and

The National Competence Center For Ultrasound And Image Guided Therapy

Group 18
Stian Weie
Marthe Bekkevold
Kristoffer Hagen

Customer representative
Frank Lindseth

Supervisor
Anh Nguyen Duc

Abstract

This paper describes the planning, design, development and implementation of the "USNeuroNav" application with its supporting infrastructure. USNeuroNav is an application for mobile devices that enables its users to access media, recorded during surgeries where ultrasound navigation was utilized, based on multiple attibutes specific to each of the surgical cases.

Despite ultrasound navigation providing several benefits over the traditional intraoperative MRI imaging, and despite the fact that the technology required is already in use, ultrasound-guided navigation is not very widespread. SINTEF and The National Competence Center for Ultrasound and Image Guided Therapy are among the pioneers in this field and believe this method would gain popularity if the relevant audience could gain access and learn about the technology in a convenient manner.

In this report, the entire process of developing the application will be discussed, from the first day of planning to the last day of the implementation and testing. The resultant product (thus far) is an iPhone application that can browse or search for media based on keywords, surgery categories, or locations. The application includes implemented support in the form of a remote database for storage of metadata as well as a webserver for the corresponding case media files.

Preface

This report, together with the code written, is the product in the master-level course TDT4290 Customer Driven Project, given by Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU). The project was executed from August 21st through November 21st of 2013.

The project team consisted of three members from the Department of Computer and Information Science at NTNU. The task was to develop an application for SINTEF that would aid in the teaching and distribution of knowledge regarding the use of ultrasound navigation during surgery. The concrete task was to develop an application for mobile devices that can display media such as video and images from several different cases that SINTEF used, and continues to use, for data collection.

We would like to thank the SINTEF for providing this opportunity and allowing access to the hardware and software required to develop this application. Frank Lindseth represented SINTEF and acted as the central customer contact during the project; many thanks are owed him. We also want to show our gratitude toward Erlend Dahl for sharing his knowledge during the planning phase of the project.

The team would also like to thank our main supervisor Anh Nguyen Duc, for his input and invaluable feedback during the course.

Contents

List of Tables

List of Figures

Part I

Planning and Requirements

1 Introduction

This section will give a short introduction to this project, including the customer and stakeholders, project goal, purpose and background, and the duration of the project.

1.1 Project Customer

The project customers are SINTEF, and The National Competence Center for Ultrasound and Image Guided Therapy.

SINTEF

SINTEF[1] is an independent, non-commercial research organization. The organization operates in partnership with NTNU and aims to "create value through knowledge generation, research and innovation, and develop technological solutions that are brought into practical use.".

The National Competence Center for Ultrasound and Image Guided Therapy

The National Competence Center for Ultrasound and Image Guided Therapy[2] is a group of scientists, engineers and surgeons in Trondheim that collaborate closely in order to improve current methods and techniques for minimally-invasive surgeries. The group consists of experts from SINTEF, NTNU and St. Olav's Hospital.

1.2 Stakeholders

The term stakeholder refers to "a person, group or organization that has interest or concern in an organization."[3] The project sponsor and customer was SINTEF together with The National Competence Center for Ultrasound and Image Guided Therapy, with Frank Lindseth and Ole Solheim as the primary

contact persons. Additional stakeholders were the development team, comprised of three NTNU students: Kristoffer Hagen, Marthe Frogner Bekkevold, and Stian Weie, as well as the project supervisor, Anh Nguyen Duc.

1.3 Project Goal

The goal of the project was to create a prototype application that the customer can use for proof-of-concept and further development. (The application had to be developed for iOS devices but on a cross-platform framework so that it could be expanded to other platforms at a later time.) The primary goal was to create a stable, professional-looking application with the core functionality specified, such as viewing images and videos, and the infrastructure around it. The development of the product will continue post-production, so expandability and versatility were high priorities, such that new features could be implemented at later dates. As the prototype is developeded further by our successors, the end goal is for it to aid the customer in making their knowledge and expertise in the field of ultrasound navigation publicly available.

1.4 Project Background and Purpose

The title given to this project was "USNeuroNav App."

The National Competence Center for Ultrasound and Image Guided Therapy was established in 1995 and is a collaboration between SINTEF, NTNU and St. Olav University Hospital. Within the center, ultrasound-based navigation technology was being developed and used in various clinical fields in order to guide surgical instruments safely into the human body. One of the primary challenges in order to make use of this technology was sufficient knowledge about interpreting ultrasound images.

SINTEF has, for many years, pioneered the use of intraoperative ultrasound navigation. The technology they have created is already in use, but in order for its application to become more widespread, they must first dissemi-

nate the knowledge necessary for its use. SINTEF believes that an application for mobile devices will enable relevant groups to acquire this knowledge in a convenient manner.

The National Competence Center for Ultrasound and Image Guided Therapy collects unique data material on a weekly basis. If this data was presented in an straightforward and accessible way, selected parts of the anonymous material could be used to teach relevant groups to interpret ultrasound images in a convenient manner. An app for mobile devices (iOS and possibly Android) (and possibly also a web-site) has huge potential to this end. In addition to presenting the available material, there is the possibility of developing the app further into a serious game.

1.5 Project Duration

The project started on August 21st and will conclude on November 21st. The duration of the project is 14 weeks.

2 Planning

This section will discuss the project plan, terms, and team organization. It will describe the projected time to be used on the project, the general definitions and terms related to the project, and a work breakdown structure. The organization of the team and the risk management protocol will be presented, followed lastly by quality assurance protocol.

2.1 Project Plan

2.1.1 Measurement of project effect

The goal of the project was to aid our customer in making their knowledge and expertise in the field of ultrasound navigation publicly available. This was achieved by creating an application that worked as a foundation for further development. The application delivered does provide the basic tools to publicly share some of the knowledge, but mainly it was meant as a modular expandable prototype that our customer could develop further into an ideal product for their purpose. The project effect can be measured in two ways. First, "as-is" upon delivery, it can be used internally to share and provide easy access to case information and media on mobile devices. It also functions as a proof-of-concept that a mobile application is a good method of sharing knowledge publicly. Secondly, it can be expanded with new features and funtionality, as it has been created on a modular platform with options for expansion and good documentation to ensure our successors will know what they are working with.

2.1.2 General terms

The customer had a very definite requirement that the project result in a mobile application for iOS. In addition, the customer saw it as a bonus if this app could easily be extended to additional platforms, for example Android, either by this group, if time should allow, or by another developer. For this

to be a possibility, an Apple developer license had to be procured, in addition to at least one Mac for code-building purposes. Depending on the choice of platforms, additional licenses could be necessary. All this had to be provided by the customer by the beginning of sprint 2, which was when programming was scheduled to begin.

The specific tools for developing this application were scheduled to be determined together with the customer on 04.09.2013. Many frameworks had been investigated and tested, and prior to the meeting, two clear alternatives remained, Xamarin[4] and Appcelerated's Titanium[5]. Both alternatives provided cross-platform development had the possibility of creating applications that look and feel like native applications. In order to make the applcation look and feel native, a separate user interface would have to be developed for each platform, but the customer thought that this was an acceptable tradeoff, sacrificing some development time for the option for cross-platform possibility. While Titanium is a free tool, Xamarin requires a license for each developer. Both require the use of a Mac for building the application, but Xamarin allows doing this remotely. This could prove a significant advantage, since accessibility to Mac computers was limited.

2.1.3 Planned effort

TDT4290 Customer Driven Project is a course that rewards 15 credits. The course compendium suggests that we work 25 hours each week. The project was started on August 21 and will conclude on November 21, which entails a run-time of 14 weeks. Three team members working 25 hours each week for 14 weeks brought this to a total of (3 x 25 x 14=) 1050 hours. While the actual time spent is covered in the evaluation section of the following report, this (1050 hours) was a working estimate that was suitable for our purposes.

Regular working hours were Wednesdays, Thursdays and Fridays from 9.00 to 16.00; these were the times that best coincided with the three team members' existing schedules. Three full days each week reserved for working

on the project was advantageous in that it allowed for greater communication and cooperation than trying to collaborate over the internet.This schedule also left the possibility to meet Mondays and/or Tuesdays should any unforeseen critical issues have arisen.

Our working hours did not conclude when the product was delivered to the customer, but continued until the final report was delivered.

The plan was to deliver the final product to the customer on the 13th of November. This would leave us with a week to focus solely on the report. It should be noted that the report has been continuously updated as the project developed as well.

2.1.4 Schedule of result

Scrum is being used for the development phase. After each sprint there will be a delivery, and all of these deliveries should contribute towards a working system. 21th of Nowember will be the final presentation, so the product and report have to be finnished befor that.

2.1.5 Work breakdown structure

On the August 21st the group met the customer for the first time and started working. The planning and pre-study phase was planned to last for four weeks, since we knew early on that we did not have the required equipment (Mac). The planning phase included planning the project, requirements specification, risk analysis, design of GUI and architecture. The pre-study phase included planning, management and researching the technologies needed for the project. The group planned to have 4 sprints of 2 weeks' duration each. Four sprints fit perfectly into the time limit, and by this plan, work efficiency would be maximized. Each sprint included a planning phase, an implementation phase, and a testing phase. The table below gives an general outline of the sprints.

#	Start date - End date	Description	Planning/ report	Coding/ testing
Sprint 1	18.09.2013 - 01.10.2013	Implement viewing of categories and cases, and their images and videos	Stian, Marthe, Kristoffer	Stian, Marthe, Kristoffer
Sprint 2	02.10.2013 - 15.10.2013	Report	Stian, Marthe, Kristoffer	
Sprint 3	16.10.2013 - 29.10.2013	Implement separation and search	Stian, Marthe, Kristoffer	Stian, Kristoffer
Sprint 4	30.10.2013 - 08.11.2013	Polish and fix bugs	Stian, Marthe, Kristoffer	Stian, Kristoffer

Table 2.1: Overall sprints

The gantt chart below shows the project plan. The work breakdown structure can be seen to the left and to the right is the tasks distributed over time, with the arrows showing the connection between them.

Figure 2.1: Gantt chart - project plan

2.1.6 Milestones

The table below decribes each of the planned milestones and its expected completion date for the project.

#	Date	Category	Description
MR1	30.08.2013	Report	Deliver preliminary project plan to supervisor.
MP1	25.09.2013	Product	Have a running prototype, able to show local images, categorization of cases to enable browsing.
MP2	01.10.2013	Product	Support for video, searching for cases, network support, server communication.
MR2	14.10.2013	Report	Pre-delivery of abstract, introduction, pre-study and life-cycle model choice to supervisor and sensor.
MP3	29.10.2013	Product	Upload of data.
MP4	08.11.2013	Product	Caching of data, finalize features.
MP5	13.11.2013	Product	Product delivery.
MR3	20.11.2013	Report	Complete and print final report.
MR4	21.11.2013	Report	Final project presentation.

Table 2.2: Milestones

2.2 Organization

This section will discuss the organization of the group, and the reasoning behind such organization.

At the beginning of the project, we had a quick discussion of each group member's previous experience, skills and interests. This became the basis for role assignments, so that each group member's strengths were best utilized. The group members' individual strengths are showed in the table below.

	Marthe	Kristoffer	Stian
Java	Medium	High	High
JavaScript	Low	Low	Low
Written English	Medium	High	High
Administration	High	Low	Medium
General computer skills	Low	Medium	High
Scrum experience	High	Low	Medium

Table 2.3: Each team member's strenghts

Since the group only consisted of three people, each group member was responsible for more than one role. The roles were assigned based on the strength table, and a couple of other factors. Stian, for example, was assigned as test manager at the beginning, but was busy with development at the core of the test phase, so Kristoffer eventually took over that role. Stian ultimately took responsibility for Kristoffer's administrative assitant role, resulting in an exchange of roles between them.

Role	Assigned	Responsibility
Project manager	Marthe F. Bekkevold	Responsible for task assignment, organizing meetings and agendas for meetings. The project manager also had the role as Scrum master.
Contact person	Marthe F. Bekkevold	Responsible for keeping customer and supervisor up-to-date regarding project status, and also being the go-to person when the customer or the supervisor had any questions or information for the group.
Administrative Assistant	Stian Weie	Responsible for making sure that oral information was documented and that no communication and information got lost and/or forgotten.
Report supervisor	Kristoffer Hagen	Responsible for supervising that the report was kept up to date. Making certain that the report was ready for delivery and that it met the technical requirements.
Test manager	Kristoffer Hagen	Responsible for setting up, supervising, and documenting tests.
Technical manager	Stian Weie	Responsible for technical research and making certain technical difficulties were solved.

Table 2.4: Roles

The diagram below shows how the roles in the group was organized.

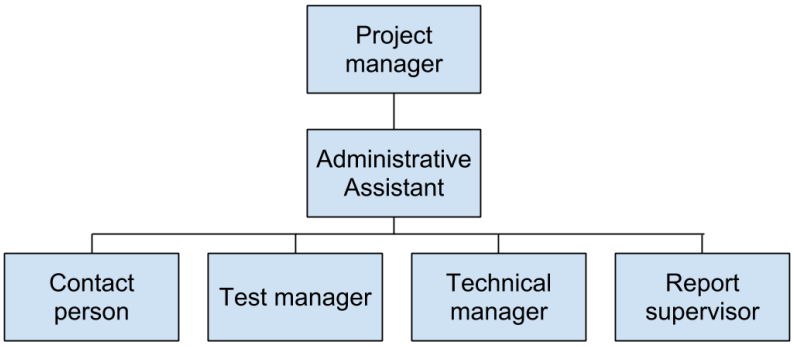

Figure 2.2: Organizational diagram

2.3 Risk Management

To prepare for contingencies that might occur during the project, the group identified risks and potential plans to solve them. For each risk there were some proactive efforts, which stated what to do to prevent the risks from realizing. There were also reaactive efforts which stated what to do if the risks had occurred. The deadline indicates the time period in which each potential risk could have occured. Probability of each risk's occurence, and the severity of its consequences, are denoted by H for 'high', M for 'medium' and L for 'low'.

2.3.1 Risk handling

In the table below, some project risks are identified.

Id	Risk factor	Activity	Probability	Consequence	Pro-active efforts	Re-active efforts	Deadline	Responsible
R1	Sick group member	All	M	H (We are 3 people in the group)	Not work that much overtime, plan with buffer	Re-evaluate scope of project	Continuous	Project manager
R2	Software issues	All	L	M	Thorough pre-study	Not severe: Live with it, Sever: Find alternative tool	Continuous	Technical manager
R3	Hardware issues	Development phase	M (none of the members has a mac)	H	Request necessary hardware from customer		19/09 - 30/10	Technical manager
R4	Miscommunication with customer	All, but mostly planning phase	L	H	Prepare for meetings, and share summary of meeting with customer	Re-plan in cooperation with customer	Continuous	Project manager
R5	Miscommunication with supervisor	All	L	L	Prepare for meetings, and take notes during meetings	More meetings and emails to clear issues	Continuous	Administrative assistant
R6	Miscommunication within the group	All	L	H	Attend group dynamic seminar	Talk about the problems	Continuous	Project manager
R7	Solution does not meet the requirements	All	L	H	Include the customer in the planning phase	Re-plan	Continuous	Project manager
R8	Underestimated time for tasks	All	M	M	Plan with time buffer	Overtime	Continuous	Project manager
R9	Insufficient knowledge	All	M	M	Thorough pre-study	Overtime / simplify	Continuous	Technical manager
R10	Too large scope	All	L	M	Plan with time buffer, and discuss available time frame with customer	Overtime / simplify	Continuous	Project manager
R11	Time lost because of UKA	Development phase	L	M	Plan extra buffer during UKA	Overtime	3/10-27-10	Project manager
R12	Absent group member	All	H	M	Plan with buffer, and inform rest of group within good time if possible	Work while absent / overtime	Continuous	Project manager

Table 2.5: Risk analysis

The risk matrix gives a better picture of how severe the risks for the project were.

Probability			
High		R12	
Medium		R8, R9	R1, R3
Low	R5	R2, R10, R11	R4, R6, R7
Consequence	Low	Medium	High

Table 2.6: Risk matrix

2.4 Project Communication

To ensure high quality communication, we agreed on some guidelines with the customer, supervisor and within the group for the project lifetime.

2.4.1 Customer interaction

- The customer was located in Trondheim, and the meetings would be held at NTNU every other week on Wednesdays at 08:00.
- The agenda for the meeting was to be sent to the customer by 12:00 on the Tuesday before the meeting.
- The minutes of the meeting were to be sent to the customer the same day as the meeting.
- Approval of minutes of customer meetings wereis to be sent as soon as possible, with a maximum of 48 hours.
- Approval of phase documents was to be completed as soon as possible, with a maximum of 48 hours.
- Answering e-mails was to be completed as soon as possible.

2.4.2 Supervisor interaction

- The meetings with the advisor would be held on Wednesdays at 09:30 at the start of the project.
- From September 20 and onwards, meetings would be held on Fridays at 12:15. This is because our sprints start on Wednesdays, so it would be more convenient to have it in the middle of a sprint, as opposed to the start, when we have more questions and challenges. It is also easier to come on track during the development.
- The meetings would last an hour, or as long as needed.
- The minutes for the meeting were to be created the same day as the meeting.
- The minutes of the meeting with the customer were to be sent to the supervisor the same day as the customer meeting.

2.4.3 Group interaction

- Group meetings: Wednesdays at 08:15, Thursdays at 09:15 and Fridays at 09:15
- Unexpected absence was to be reported as soon as possible, preferably before the meetings.
- Planned absence was to be reported as soon as possible, so we could plan accordingly.

2.5 Tools and Technology

This section contains a description of all the tools and technologies used during the project.

2.5.1 Report writing

Lyx and LaTeX The team chose to compose the report in the document markup language LaTeX[6], by using the Lyx document processor[7]. LaTeX was chosen because it is widely used in academia, and the team members saw this as an opportunity to familiarize themselves with the tool they likely will use in composing their masters. In addition, LaTeX generates standard, aesthetically-appealing, and highly customizable pdf's.

The reason why Lyx was chosen for writing the document, is that it provides the user with an easy-to-use interface that still enables the user to make use of all the different possibilities with LaTeX. Another reason was that it is open-source.

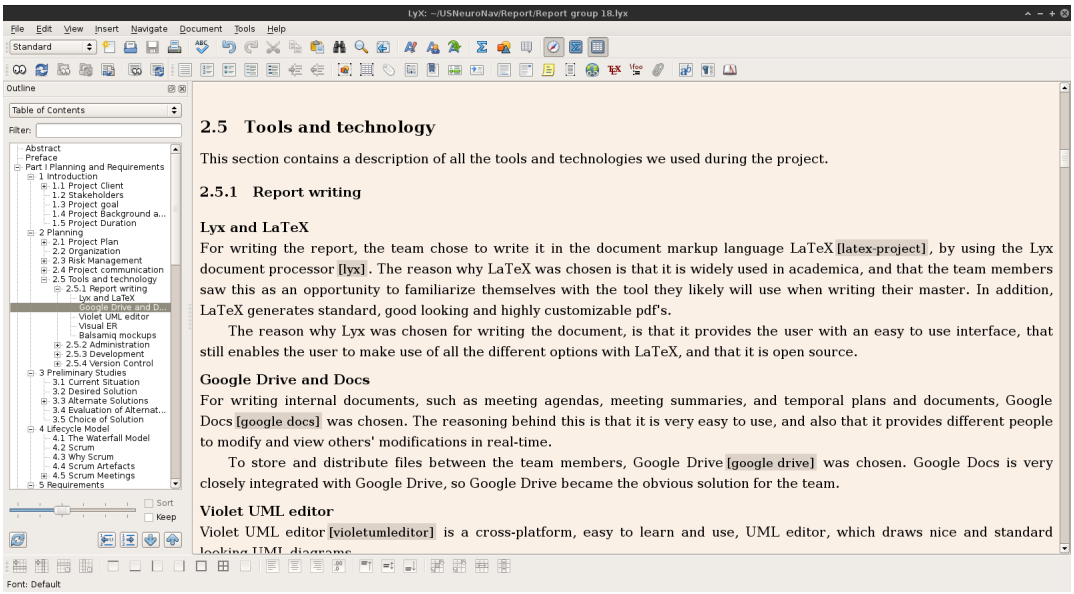

Figure 2.3: Lyx

Google Drive and Docs To write internal documents, such as meeting agendas, meeting summaries, and temporal plans and documents, Google Docs[8] was utilized. The reasoning behind this is that it is very simple to use, and also that it allows different people to modify and view others'

modifications in real-time.

To store and distribute files between the team members, Google Drive[9] was utilized. Google Docs is very closely integrated with Google Drive, so Google Drive was the obvious solution for the team.

Violet UML editor Violet UML editor[10] is a cross-platform, easy to learn and use, UML editor, which draws nice and standard looking UML diagrams. It was chosen because it seemed simpler and more light-weight that other similar tools.

For this project, Violet UML editor was used to draw the use case diagrams, sequence diagrams and class diagram.

Visual ER Visual ER[11] is a very simple Java based tool to create ER diagrams. The team used Visual ER because it was developed by NTNU students, is very simple to use, and because we have had good experiences with it from other courses.

GanttProject GanttProject[12] is a cross-platform desktop tool for project scheduling and management. It gives the ability to create Gantt chart, work breakdown structure, draw dependencies between the tasks, and define milestones. It can also assign human resources to work on tasks. We chose this because it looked simple to use, had all the functionality we needed and it drew a decent locking Gantt chart.

Balsamiq mockups Balsamiq[13] mockups is a tool for creating GUI mockups. It enabled us to quickly create our initial GUI and then easily modify it as we, or the customer, felt any changes were needed or had another improvement that needed to be made. We chose Balsamiq mockups because it had a free trial period, had support for creating iPhone GUIs and offered a simple, intuitive and easy to learn interface for creating and modifying GUI drafts.

2.5.2 Administration

Many of these choices were made because all the group members had experience using these tools and they all provided the features we required. The selection of familiar tools enabled us to put more time into actual development and less time into learning new auxiliary tools.

Facebook For distributing information and Internet URLs between team members, Facebook[14] was used. The group maintained a private group where information was posted on the "wall". In addition, the Facebook's chat function was used for instant messaging between team members. As a second option we all shared our personal mobile phone numbers so that we could get in touch in cases of urgency.

The reason that the team used Facebook was that all team members had experience with Facebook, and were accustomed to checking it regularly.

TeamViewer To utilize a Mac, the team had remote access to one of the customer's own iMacs. The team used TeamViewer[15], a simple software for controlling computers remotely over the Internet.

The team used TeamViewer, because it was free to use (non-comercially), easy to set up, and the team members already had some experience with it.

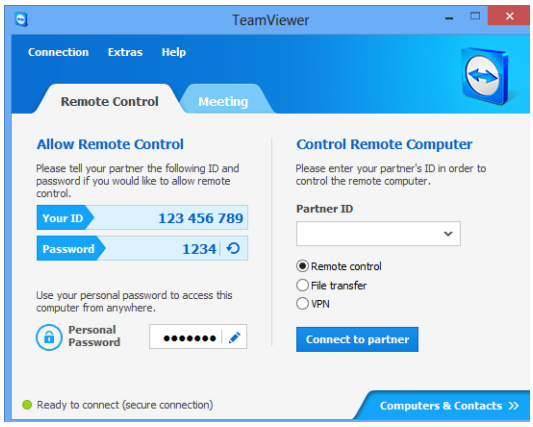

Figure 2.4: TeamViewer[16]

Rally Rally[17] is a cloud-based solution for managing the agile development lifecycle, such as assigning user stories/tasks to sprints and developers. We chose to use Rally because it is relatively easy to use, and because it is free for students.

2.5.3 Development

Titanium and JavaScript To develop the mobile application, the Titanium tool set was chosen. It provides its own IDE, Titanium Studio, for writing the code, which is based on Eclipse[18]. Titanium is a framework for creating cross-platform mobile applications, where the code is written in JavaScript, and compiled to native code for the selected platforms.

To decide on the platform and tools to use, the team spent a lot of time, together with the customer, evaluating different options. Since the customer had very specific requirements, not many available options were sufficient. The customer's requirements were, in short, as follows:

- The application should be cross-platform (at least iOS and Android)
- The application should have a native "look and feel" on all platforms
- The application should have (close to) the same performance as if written in native code
- (The platform should be free to use if possible)

Given these very specific, and quite demanding, requirements, the team was only able to find a couple of legitimate options. Ultimately, these options consisted of Titanium, which was chosen, and Xamarin. Xamarin is a tool similar to Titanium, except that it uses code written in C# instead of JavaScript. Despite the fact that the team had more experience with C#, Titanium was chosen because it is completely free to use, while Xamarin requires a license. Several other options, like PhoneGap, Marmalade and other native apps were evaluated, but none fit the customer's requirement. PhoneGap, for example,

had neither native performance nor "look and feel," while writing the app in native code for iOS and Android would take too much time. Marmalade, together with Xamarin, cost too much for the customer, given that nearly-comparable tools existed and were free. Even though Titanium is not "write once, run everywhere," it came closest to meeting all the customer's needs (weighing pricing quite heavily), so it was chosen. If the code was written carefully, a significant part of it should be re-usable between iOS and Android.

JavaScript[19] is a prototype-based scripting language with dynamic typing. It is commonly used to make web pages dynamic, but in this case is compiled down to native code for mobile platforms by Titanium.

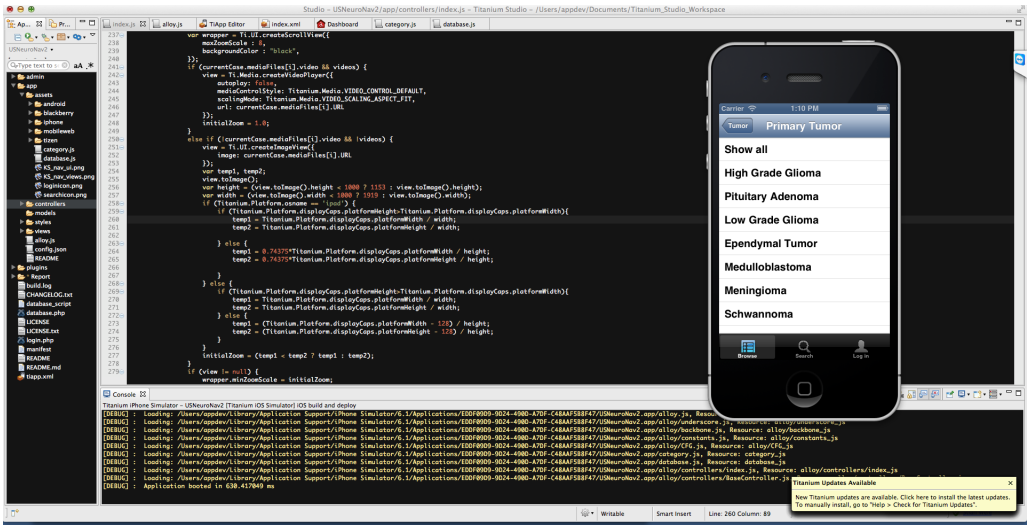

Figure 2.5: Titanium Studio with the iOS simulator

PHP PHP[20] is a server-side scripting language designed for web development. PHP code is interpreted by a web server with a PHP module, which generates the resulting web page.

For this project, PHP was used as a tiny layer between the mobile application and the database. Since Titanium does not support communication with external (My)SQL servers, the app would communicate with a simple script that reads data from the database and sends it to the app.

The reason that PHP was chosen was that it is very simple to use for small scripts, and that it is well-documented.

JSON JSON (JavaScript Object Notation)[21] is a lightweight data-interchange format, which is easy for both humans and computers to read/parse and write.

For this project, JSON was used for the communication between the PHP script and the mobile applications. The JSON objects themselves were structured exactly the same as internally in the database.

The reason that JSON is used for this communication was that it is very well-supported in both PHP and JavaScript, and is simple to use.

MySQL MySQL[22] is a very well-known SQL database, and, in their own words, "the world's most popular open source database."

MySQL was used on this project for storing the categories, and their relations (sub-categories and super-categories), the cases, and information about each case (e.g., which categories it belonged to), and URLs to the cases' media files. In addition, the MySQL database stored all user information.

The reason that the team used MySQL was that it is free to use. Additionally, all team members have had prior experience with it.

Apache HTTP Server Apache[23] is a very common web server. It is free to use, very easy to set up, and works very well with PHP.

The Apache HTTP Server was used in this project to host the media files for the cases, and also to host the PHP script (see above).

Java Java[24] is a concurrent, class-based and object-oriented programming language that is designed to run on as many platforms as possible. We chose to use Java to develop the admin tool, mainly because of the cross-platform feature, but also because all the team members have programmed in Java before, and it was the language all the team members were most comfortable with.

Eclipse Eclipse is a multi-language integrated development environment (IDE). It is very suitable for Java development, and it also has an integrated Git tool.

We chose to use Eclipse for Java development because it is easy to use, but it also has many handy functions. All team members was also more familiar with Eclipse, than for example, NetBeans. Pure text editors like Notepad++, vim, Emacs, etc. were also considered, but ruled out, due to their lack of auto-completion of code, which both Eclipse and NetBeans support.

2.5.4 Version control

Git and GitHub We used Git[25] for version control, both for the textual documents and source code. The Git server we used is GitHub[26]. Git is a distributed revision control and source code management system. We use Git for version control because it is simple to use, it provides both a shell and a GUI, and the team members were already familiar with it, as opposed to, for example, SVN [27]. Git is also much faster than SVN, and provides better auditing of branch and merge events[28].

The reason why we used GitHub as the Git server, is that it was free to use, required no personal servers, and that it provided a very useful web site to manage pull requests, issues, etc. The project's GitHub repository may be found at http://Github.com/stianwe/USNeuroNav.

With Git every group member had a copy of the repository locally, and synchronized their repositories through the server. We chose to use Git because it ensured that all group members could work on the code and report, regardless of the status of the server. This also meant that we had many backups of the entire repository if something went wrong, so the team can ensure that all the files are up-to-date and no information was lost during the project. It is necessary to work on up-to-date data, especially when two people are working on the same file; it was important to ensure that they didn't inadvertently destroy one another's work. Git supports quick branching and

merging, and includes specific tools for navigating a non-linear development history.

2.6 Coding Conventions

When writing code, we decided on a couple of coding conventions, to make the code readable and more easy to understand.

2.6.1 Names

When writing ordinary names of variables, functions, classes, etc., camel case was used. Camel case means that the word is not divided by space, but each new word starts with an upper case character. Example: ThisIsAnExample-OfCamelCase. Constants, on the other hand, were written with all uppercase characters, with underscore to separate words. See table below for details.

Variable name	thisIsAVariable
Function/method names	thisIsAFunction()
Class	ThisIsAClass
Constant	THIS_IS_A_CONSTANT

Table 2.7: Naming conventions

2.6.2 White space

For white space, the following conventions apply:

- Blocks should always be indicated with brackets, even if they are one-liners: {}
- The bracket at the end of a block should always be on its own line
- When inside a block, an extra indent (\t) is to be applied
- There should always be space operators and variables/constants: (b + 1 * (2 - c)) > 0

3 Preliminary Studies

This section will give a description of the current situation, the desired solution, alternate solutions, their evaluation, and our ultimate choice of solution.

3.1 Current Situation

Currently, MRI (Magnetic Resonance Imaging)[29] is the tool doctors and other medical personnel prefer in pre-operative evaluation. The reason for this is that it is relatively easy to understand, and also to identify tumors and other potential medical obstacles. The figure below shows a typical MRI machine.

Figure 3.1: MRI[29, 30]

Recently, however, there has been speculation on the use of DMS (Diagnostic medical Sonography) (ultrasound)[31] either instead of MRI, or together with MRI. The main incentive for this is that DMS is much cheaper, and also more flexible, since the MRI requires its own dedicated room, while DMS is mobile and can be moved from room to room. It is therefore easier to perform a new intraoperative scan with DMS than MRI. The fact that surgeons can now obtain real-time updates on what the conditions inside the body are, is extremely advantageous. As an example, DMS can determine

whether there are any remnants of a tumor still inside the body. On the other hand, DMS is much harder to understand. Therefore, MRI is still preferred by many doctors, since it is what they accustomed to and familiar with. Younger doctors who have only recently graduated from medical school do get some training there, and therefore are more adaptable to the new use of DMS. The main problem with the current situation is simply that experienced doctors have no understanding of the new way of using DMS, and, unfortunately, very few opportunities to learn how to do so.

The figure below shows a typical image from an surgical operation. The leftmost image is an MRI, and the middle image is an ultrasound of the area represented by the rectangle in the leftmost image. The rightmost image is a skull with a representation of where the images were taken in the brain.

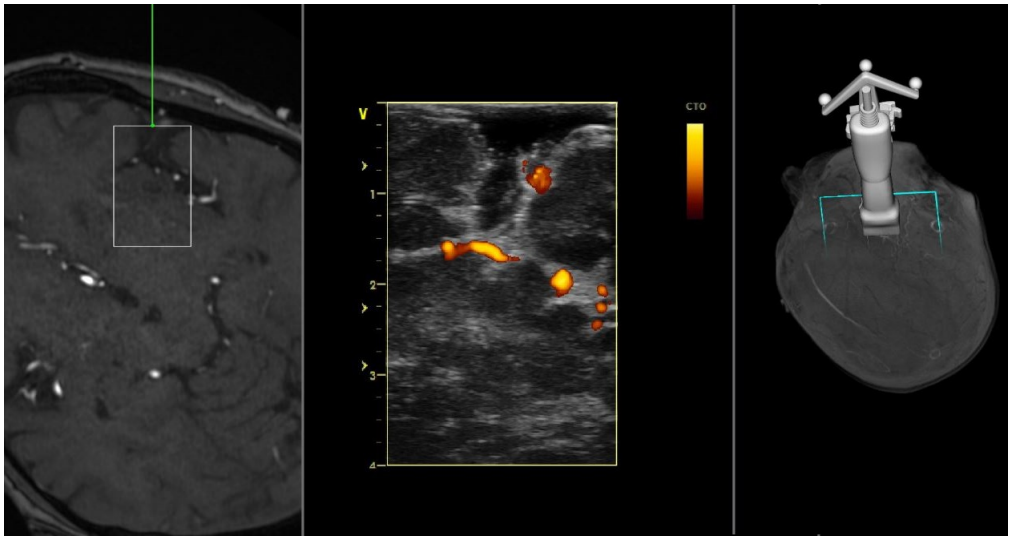

Figure 3.2: MRI - ultrasound image

3.2 Desired Solution

The solution that the customer desires is a simple and easily accessible method for doctors (and others) to familiarize themselves with DMS scans (images and video), and to learn to interpret them, as many doctors now understand

MRI. For this, the customer has planned a mobile application that will be able to view images and videos from DMS scans, and in the future also provide a quiz-like game to challenge the users of the app to try to understand the data shown. The customer already has a lot of relevant data that can be used, and also continuously produces more, so what they need is simply a tool for displaying and distributing it, such that it is easy to find, wherever the user is.

3.3 Alternate Solutions

There are some similar solutions for the USNeuroNav App. Some are available to everyone, and others are only for surgeons and internal staff. Consideration should be given to privacy, therefore not all systems are available to everyone.

3.3.1 EUS - Diagnostic and interventional endoscopic ultrasound app

The EUS - Diagnostic and interventional endoscopic ultrasound app[32] for iPhone/iPad has some similar functionalities to our system.

Figure 3.3: EUS - eiagnostic and interventional endoscopic ultrasound[32]

This app is dedicated to endoscopic ultrasound education and has been developed by Drs. Shyam Varadarajulu, Paul Fockens and Robert Hawes. The app is for endoscopists, pulmonologists and gastroenterology/pulmonology trainees interested in EUS. The EUS App is available for free download in the online Apple store.

The app shows images and videos of ultrasound images.

3.3.2 Helsinki microneurosurgery basics and tricks app

The Helsinki Microneurosurgery Basics and Tricks App[33] for iPhone/iPad was provided by the customer.

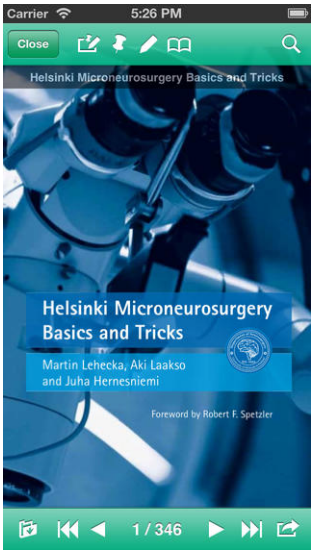

Figure 3.4: Helsinki microneurosurgery basics and tricks[33]

This app for Microneurosurgery contains the "Helsinki Microneurosurgery Basics and Tricks," which is a unique book about the "Hernesniemi school of neurosurgery" written by Martin Lehecka, Aki Laakso and Juha Hernesniemi. Additionally, this app provides many operative videos and images which the book directly refers to, as well as an interesting link to the Hernesniemi-Aescuap Fellowship.

3.4 Evaluation of Alternate Solutions

In the table below there is an overview of the strengths and weaknesses of the alternative solutions, including the application we are planning to develop.

Applications	Strength	Weakness
EUS - Diagnostic and Interventional Endoscopic Ultrasound App	The app have the ability to show images and videos. Since it is for iPhones it is portable,	The app has a focus on lungs and diagnostic, and not ultrasound images during a brain tumor operation. The app also gives information about conferences, which is not important in our case.
Helsinki Microneuro-surgery Basics and Tricks App	The app have the ability to show images and videos. Since it is for iPhones it is portable.	The app is a book, and it does not have the ability to search and browse for cases in a category
USNeuroNav App	The app have the ability to show images and video. It can also browse and search for cases, and can distinguish between internal users with more rights, and external users. It is for iOS devices, which makes it portable.	It was only meant to be a prototype with basic functionality.

Table 3.1: Evaluation of alternative solutions

3.5 Choice of Solution

There are sufficient differences between the system the customer needs and the systems described above, such that we can not choose and modify one of them.

We developed the system with frameworks and tools described in section

2.5.3.

4 Lifecycle Model

In this section, our choice for the lifecycle model is described. First we introduce the waterfall model and why we did not chose that. Following is a description of why we chose scrum, then a disccusion of scrum as a lifecycle model. Lastly, there is a description of scrum artefact and the scrum meetings.

4.1 The Waterfall Model

The waterfall model[34] is a linear-sequential lifecycle model, in which progress is seen as flowing steadily downwards through the phases, like a waterfall. Any phase in the development process begins only if the previous phase is complete, and the phases do not overlap.

The figure below illustrates how waterfall phases are organized.

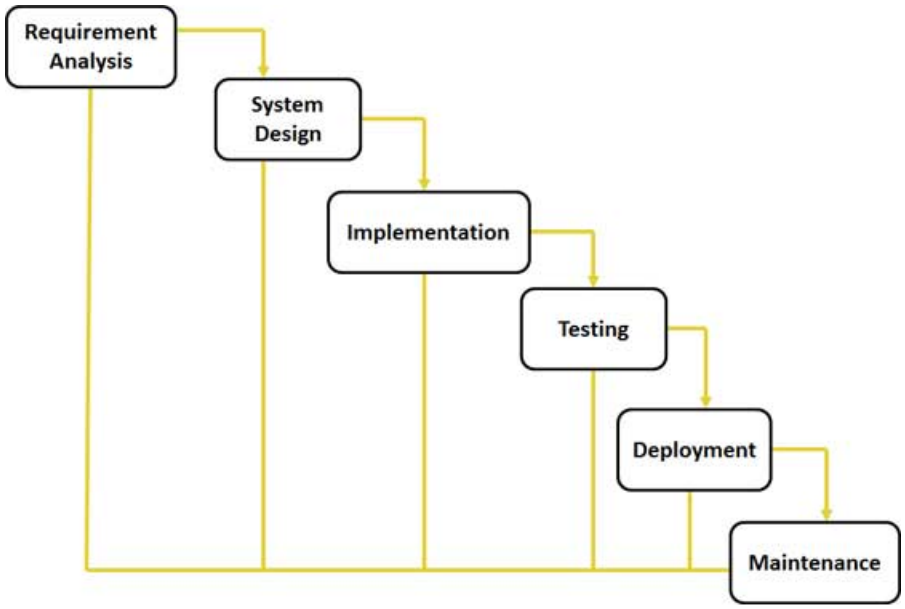

Figure 4.1: Waterfall model[34]

The waterfall model is simple and easy to understand and use, the outcome of one phase acts as the input for the next phase sequentially.
It works well for smaller projects where requirements are very well understood, and with its clearly defined stages it is easy to arrange tasks, understand milestones, and document the process and results.

The disadvantages of the waterfall model are that no working software is produced until late during the lifecycle. It cannot accommodate changing requirements, and, generally speaking, handles all changes poorly.

4.2 Scrum

Scrum[35] is an iterative planning and execution method for software development. Scrum is agile and based on a series of increments of sprints, where requirements, design and product implementation evolve through the project lifetime. A scrum team is quite small, usually about five to ten people. The team is self-organized, which means that the main roles for the project are divided among the members of the team. There is usually a focus on quick communication between members, given that scrum teams are relatively small.

In the figure below, the scrum lifecycle is shown. A description of these elements is given in section 4.4 Scrum Artifacts.

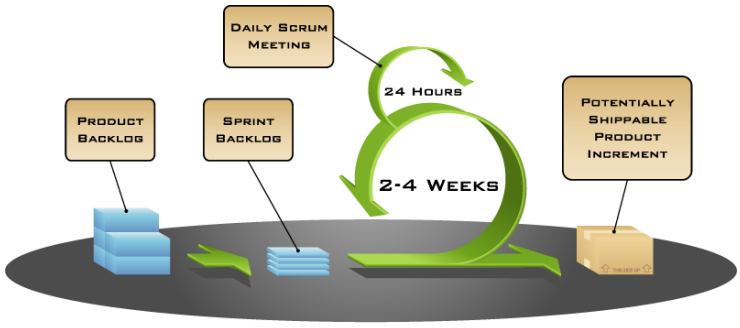

Figure 4.2: Scrum lifecycle[36]

There are many advantages for using scrum. For example, by managing risk and change effectively using small steps and quick feedback (tests, customers), errors from misunderstandings are quickly addressed. Focusing on the most valuable and risky aspects of the project up front reduces the cost of failure, and therefore gives a greater understanding of risk in the project.

4.3 Why Scrum

The table below shows the project's properties, and whether the scrum or the waterfall model can satisfy them.

Project properties	Waterfall	Scrum
Small team	X	X
Uncertain group size		X
Not clearly defined requirements at the beginning		X
Not clearly defined scope of the project at the beginning		X
Hardware uncertainty		X
Communication with customer during the entire semester	X	X

Table 4.1: Properties of the project

Clearly the waterfall model was inappropriate, so we chose an agile model. It works well for small groups, and the scope and requirements for the project were not clearly defined at the time it had to be decided. We also did not know exactly when we would receive access to a Mac from the customer, so a versatile lifecycle model was preferable.

The ability to cope with changes was the major reason that we chose scrum. It was not clear whether we would get an additional team member in our group, so we planned for three people; with scrum we could easily change the plans if we gained a member.

4.4 Scrum Artifacts

There are several artifacts used in scrum, such as product backlog and sprint backlog. The product backlog is a list with all the remaining functionality to implement, while the sprint backlog is a list with functionality that is to be implemented during a sprint.

The project was divided into sprints, which usually lasted between two to four weeks. A sprint began with a planning meeting, wherein the members of the team and the stakeholders/customer selected items from the product backlog that they would try to complete within that sprint, and placed these items into the sprint backlog. Each of these items had an associated estimated effort of their difficulty and time required for completion. The item's estimated effort to complete was updated after each day so that the team was able to see if they were "on track" with time constraints or not. This was visible through the burndown chart, a graph which shows how much effort remained in the sprint on any given day.

Pair programming is an agile software development technique that we would try to take advantage of.

4.5 Scrum Meetings

There are several different meetings in the scrum lifecycle that we used:

4.5.1 Sprint planning meeting

A meeting was held at the start of each sprint. The team decided which items in the product backlog would be moved to the sprint backlog, based on the item's priority. Each item in the sprint backlog would then be assigned an estimated effort on completion time.

4.5.2 Daily standups

A short meeting, not longer than 10 - 15 minutes, was held at the start of each day during a sprint. Each member of the group updated the rest on what had been done since last meeting, what they would do that day, and whether they'd encountered any problems.

4.5.3 Sprint review meeting

After each sprint, a meeting was held to review which tasks from the sprint backlog had been completed. The tasks that had not yet been completed yet were returned to the product backlog. During this meeting, the customer was shown the current state of the product, and gave feedback to the group.

4.5.4 Sprint retrospective meeting

After each sprint, a meeting was held to reflect on the past sprint and discuss how to improve on the next sprint. The questions asked and documented were: "What went well during the sprint?" and "What could be improved in the next sprint?"

5 Requirements

This section will discuss the system requirements. First will be a description of the scope of the project. Next will be a description of the actors, followed by a description of the functional and non-functional requirements. It will close with a description of the product backog.

5.1 Scope

The final system will be identified by the name USNeuroNav App. The system will have two interfaces: one for the administrator and one for users (registered and unregistered).

The administrator will have a simple panel (text-based or GUI) where he can register users, create new categories and cases, and upload media files (images and video) to cases. The administrator will also have the potential to edit the category structure, for example, by specifying subcategories. This panel will be a computer program, not a mobile application.

The interface for users is the main project, namely the mobile application. The interface will be similar for both registered and unregistered users, except from the fact that the user may log in. However, unregistered users will not necessarily be able to view all categories, cases or media files.

For both registered and unregistered users, the app should be able to view a structure of categories, which contains cases. These cases, as noted, contain media files (i.e., images and videos). The app must present these media files in a simple yet informative manner. Navigating the category structure should be very "native." For example, for iOS, a "back" button should be displayed in the upper left corner, with a label which tells the user where "back" actually leads.

5.2 Actors

5.2.1 Unregistered user

This is a public user who is interested in learning about ultrasound images and how to interpret them. These may be students, doctors, or anyone who wants access to such information.

5.2.2 Registered users

These are users within the organization. They have an interest in learning how to interpret ultrasound images, so they can ascertain, for example, that all the cancer was removed during a procedure. These users may be students or doctors.

5.2.3 Administrator

This user has the authority to access the administration panel in the application. The administrator can manage the application by using an interface made for that purpose.

5.3 Use Cases

"A use case is a software and system engineering term that describes how a user uses a system to accomplish a particular goal."[37] The following textual use cases display how the users, or actors, interact with the system in order to achieve a specific goal, or objective. By creating good use cases, we place ourselves in a good position to define the system's requirements. These use cases were created based on several meetings with the customer during the initial phase of the project.

Title:	**Browse categories**
Requirements involved:	FR01
Actors:	Users
Objective:	To give the user the option to browse thorugh surgery categories to find specific cases that match all the selected categories.
Pre-conditions:	The browsing tab of the application is selected.
Post-conditions:	One or several cases of interest have been found.
User story:	As a user, I want to browse through a hierarchy of categories to find cases I might be interested in.

Title:	**Select a case**
Requirements involved:	FR01
Actors:	Users
Objective:	To give the user detailed information on a specific case along with providing options regarding the case, such as viewing images or videos.
Pre-conditions:	Having found a case of interest either through browsing or searching.

Post-conditions:	The page belonging to the case has been opened and information is showing along with the options of showing videos and images.
User story:	As a user, I want to select a case of interest, in the menu, to see more information about the case. If I am logged in, I want to see the additional private information as well.

Title:	**Browse case images**
Requirements involved:	FR02
Actors:	Users
Objective:	To give the user easy and intuitive access to all the images that belong to a selected case.
Pre-conditions:	Having found a case of interest, selected it, and then chosen to view images.
Post-conditions:	Having looked through one or all available images belonging to the case.
User story:	As a user, I want to see all images relevant to the case I chose in an easy and intuitive manner.

Title:	**Browse case videos**
Requirements involved:	FR03

Actors:	Users
Objective:	To give the user the option to play any videos belonging to the selected case.
Pre-conditions:	Having found a case of interest, selected it, and then chosen to view videos.
Post-conditions:	Having watched one or more videos belonging to the case.
User story:	As a user, I want the option of watching any videos relevant to the case I have selected.

Title:	**Search for cases**
Requirements involved:	FR04
Actors:	Users
Objective:	To give the user to option of searching for a case (or cases) of interest based upon category keywords.
Pre-conditions:	The search tab of the application is selected.
Post-conditions:	Any cases matching all the keywords are shown in the menu; if nothing matches the search, nothing is shown.
User story:	As a user, I want to be able to search for any cases that fall under a certain set of categories.

Title:	**Log in**
Requirements involved:	FR05
Actors:	Users
Objective:	To give the user the ability to log in to gain access to private cases and private information concerning public cases.
Pre-conditions:	The search tab of the application is selected. The username and password combination entered has been registered by the administrator.
Post-conditions:	The user has been logged in and now has access to private information.
User story:	As a user, I want to be able to log in in order to gain access to more detailed information on the case(s) as well as to the ability to find private cases.

Title:	**Select cases/videoes/images to download and store localy**
Requirements involved:	FR07, FR08
Actors:	Users
Objective:	To give the user the ability to locally store data for offline use.
Pre-conditions:	Having found a case of interest either through browsing or searching.

Post-conditions:	The case data has been stored locally on the device and is available offline.
User story:	As a user, I want to be able to store case data locally when I am online, so that I can later access it without having to be connected to the Internet.

Title:	**Create new case**
Requirements involved:	FR10
Actors:	Administrator
Objective:	To give the administrator the option to create a new case in the database that will automatically appear in the application without requiring a recompilation of the code.
Pre-conditions:	Case media files have been put in the correct folder and the administrator is connected to the database.
Post-conditions:	A new case has been created and is available through the application.
User story:	As an administrator, I want to upload new cases to the database in reasonable time and without undue much complexity. After I have uploaded it, I want the new case to appear in the application without having to modify the application code.

Title:	**Register new user**
Requirements involved:	FR06, FR10
Actors:	Administrator
Objective:	To give the administrator the option to create a new user in the database that will be usable immediately after registration, without requiring a recompilation of the code.
Pre-conditions:	Having the desired username and password of the new user. Being connected to the database.
Post-conditions:	A new user has been registered and is available for use.
User story:	As an administrator, I want to be able to register a new user with a username and password that I have been given. I want the new user to be ready for use the moment I upload it.

Table 5.2: Use case description

5.4 Use Case Diagrams

The figure below shows the use cases for the application and the administrative tool. These diagrams are a simple representation of the interaction between actors and the system.

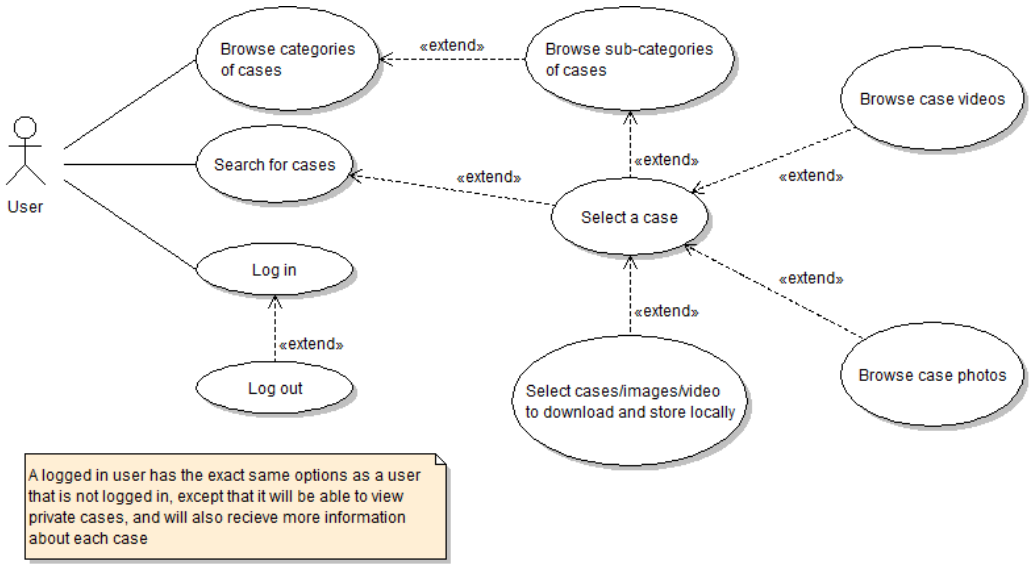

Figure 5.1: Use case application

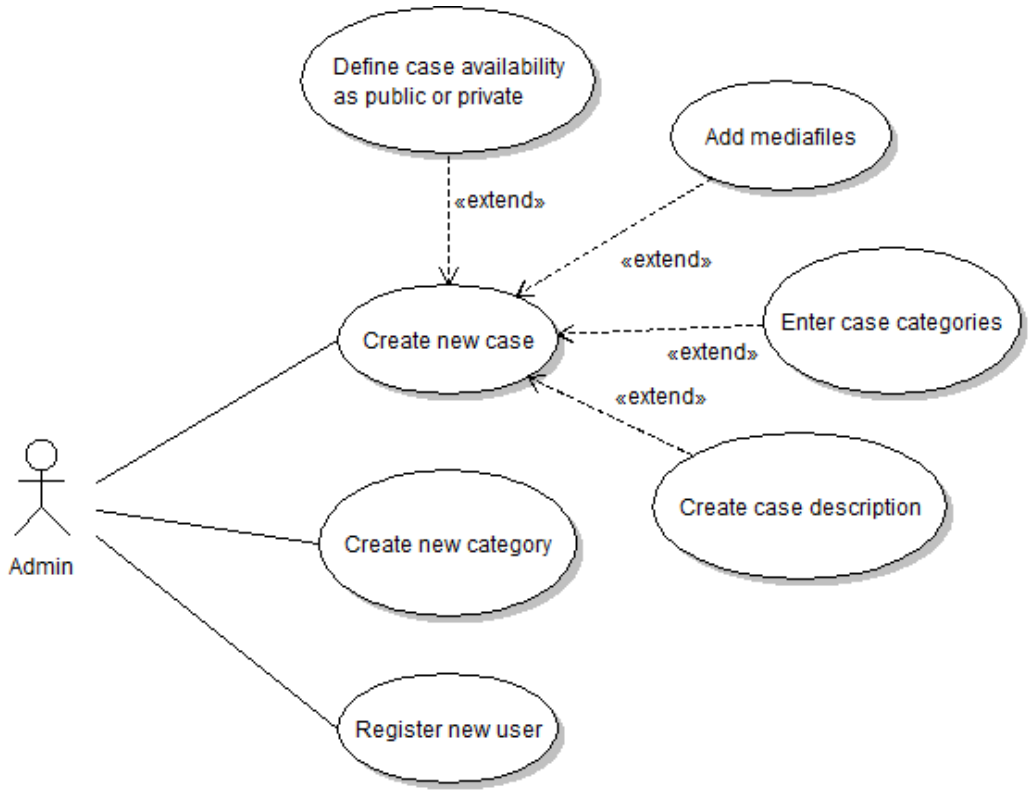

Figure 5.2: Use case admin

5.5 Functional Requirements

Based on the previous section discussing use cases, we constucted a concrete list with both the functional and non-functional requirements of the system. All the requirements are assigned one (out of four) possible priority value: critical, high, medium or low. Critical prioritized requirements are the most important and the system will not be approved if these items are not implemented. High prioritized requirements are important for the customer, but these requirements will be implemented after the critical prioritized requirements. Medium prioritized requirements are not as important for the customer as the critical and high prioritized requirements, and will be implemented after the high prioritized requirements. Low prioritized requirements

are “nice-to-have” requirements, but are not important for the customer, and these requirements will be fulfilled only if there is time left.

The tables below contains a summary of the functional requirements, its priority status, and the identification we will use later for reference.

ID	Functional requirements	Priority			
		Critical	High	Medium	Low
FR01	Browse cases	X			
FR02	Show images	X			
FR03	Play video		X		
FR04	Search for cases		X		
FR05	Log in		X		
FR06	Registration			X	
FR07	Save images to phone			X	
FR08	Save video to phone			X	
FR09	Separation of data (users - no-users)		X		
FR10	Upload data to server		X		
FR11	GUI for uploading data to server			X	
FR12	Quiz-game				X
FR13	"Like" function to images				X

Table 5.3: Overall functional requirements with priority

5.6 Non-Functional Requirements

In this section, the non-functional requirements for the system are stated and explained in detail.

All requirements are assigned one of four possible priority values: critical, high, medium or low.

Critical prioritized requirements are the most important and the system will not be approved if they’re not implemented. High prioritized require-

ments are important to the customer, but these requirements will be implemented after the critical prioritized requirements. Medium prioritized requirements are not as important for the customer as the critical and high prioritized requirements, and will be implemented after the implementation of high prioritized requirements. Low prioritized requirements are "nice-to-have" requirements, but it's not important for the customer, and these requirements will be fulfilled only if there is time left.

A summary table of the non-functional requirements and their priorities is given below.

Id	Non-Functional requirements	Priority			
		Critical	High	Medium	Low
NR01	Platform - Working on iOS	X			
NR02	Platform - Working on Android				X
NR03	Security - Unregistered user will not have access to all data		X		
NR04	Usability - Native look and feel			X	
NR05	Playing video should be smooth, and should not stop due to buffering		X		
NR06	Navigating menus should feel responsive	X			
NR07	Loading and displaying images should not take undue time			X	

Table 5.4: Overall non-functional requirements with priority

5.7 Mapping Requirements Into Product Backlog

The table below shows the project's product backlog which is derived from the project requirements.

Identifier	Req.	Description
BL1		**Categories**
BL1.1		Browse window
BL1.2		Display categories
BL1.3		Display cases
BL1.4		Window navigation
BL1.5	FR01	Browse categories
BL2		**Cases**
BL2.1		Case window
BL2.2		Show case metadata
BL2.3	FR02	Show images
BL2.4	FR03	Play movies
BL3	FR04	**Searching**
BL3.1		Search window
BL3.2		Implement searching
BL4	FR09	**Separation of data**
BL4.1	FR07	User login
BL4.2		User registration
BL5		**Local storing**
BL5.1		GUI for local storing
BL5.2	FR05	Save images locally
BL5.3	FR06	Save video locally
BL6		**Social**
BL6.1		GUI for social
BL6.2	FR12	Quiz game
BL6.3	FR13	"Like" function
BL7	NR04	**GUI design**
BL7.1		GUI for categories
BL7.2		GUI for cases
BL7.3		GUI for searching
BL7.4		Icons and UI elements
BL8		**Admin tool**
BL8.1		Admin application
BL8.2		Add data
BL8.3		Upload by textfiles
BL8.4	FR08	Add new user
BL8.5		Admintool bugfixing
BL8.5		Talk to webserver
BL8.6		Talk to database
BL8.7	FR11	User interface

Table 5.5: Development backlog

Identifier	Req.	Description
BL9	FR10	**Servers**
BL9.1		Apache server
BL9.2		Apache setup
BL9.3		Database server
BL10		**Database**
BL10.1		Insert tables
BL10.2		Insert initial case data
BL10.3		Finalize database
BL11		**Application**
BL11.1	NR05	Final testing
BL11.2		Bug fixing
BL11.3		Distribution
R1		**Report**

Identifier	Req.	Description
R1.1		Create LaTeX document
R1.2		Abstract and preface
R1.3		Introduction
R1.4		Planning
R1.5		Preminary studies
R1.6		Lifecycle model
R1.7		Requirements
R1.8		Software architecture
R1.9		Sprint 1
R1.10		Sprint 2
R1.11		Sprint 3
R1.12		Sprint 4
R1.13		Conclusion
R1.14		Evaluation

Table 5.6: Report backlog

6 Software Architecture

This section will describe the architecture of our application as it was planned before the implementation started. It will describe the different modules of the system and, through different architectural views, show how the planned the structure and modularity of the system.

6.1 Introduction

6.1.1 Purpose

The purpose of this is to give an overview of the architectural structure of the system. This will enable the team to have a clear and planned approach to the system architecture, and will guide the implementation of the system. By being familiar with the architecture, and knowing how every module works, all the current and future developers will be able to work on the modules independently and can rely on the planned architecture when the system is assembled. This will also aid future third parties to further develop and improve the system; with the proper documentation of our architecture it will be easier to change specific modules of the system without having to change the entire code.

6.1.2 Scope

Chapter 6 will cover the important aspects of the architecture of our system. We will describe the architecture through several different views, and with the appropriate amount of detail that is to be expected given the size of the project. The logic and reasoning behind our choices will also be discussed by the end of the chapter.

6.2 Data Storage

6.2.1 Overall

Since the data to be used in this app is from real medical cases, the customer wanted to have full control over the data, and therefore did not want to rent/buy cloud services from third parties. The team therefore suggested that if the customer provided a dedicated server, the team would create the database and other necessary components.

Another important note about data storage: more relevant data for the app is produced every day, so the customer wanted to be able to "push" new data into the app without the users having to download a new version each time. This will be solved by maintaining a database with the different categories and cases, and also the relationships between them, as well as an overview of where to find images and videos for each case. The images and videos themselves will be stored on a web server which the app can access and stream the data from.

Since the customer only wants part of the data to be publicly available, some user identification must be enforced. User information, such as user name and password, must therefore also be stored.

Finally, the customer enquired about the possibility of using the app offline. Since the data for the app will be very large, the videos are several megabytes, and each case contains one video, the team decided that storing all of the data locally on the phone would be too much. As a solution to this, the possibility for selecting some cases, images and/or videos that the phone would download and store locally were proposed, so that some data selected by the user would be available offline.

6.2.2 Categories, cases and users

To store the categories and cases, and also a reference to the media files belonging to each case, a MySQL database will be used. The reason for this was that MySQL was free to use, and was also an SQL database that the team

was familiar with, so that the development process became efficient. Since there are clear relationships between categories and cases, and also between the categories themselves (sub-cases and super-cases), a relational database like SQL fits very well. The MySQL database is also the logical place to store the user identification, since they can be stored in a simple table with a few columns for user name, password, etc.

The design of the MySQL database can be seen in the ER-diagram below:

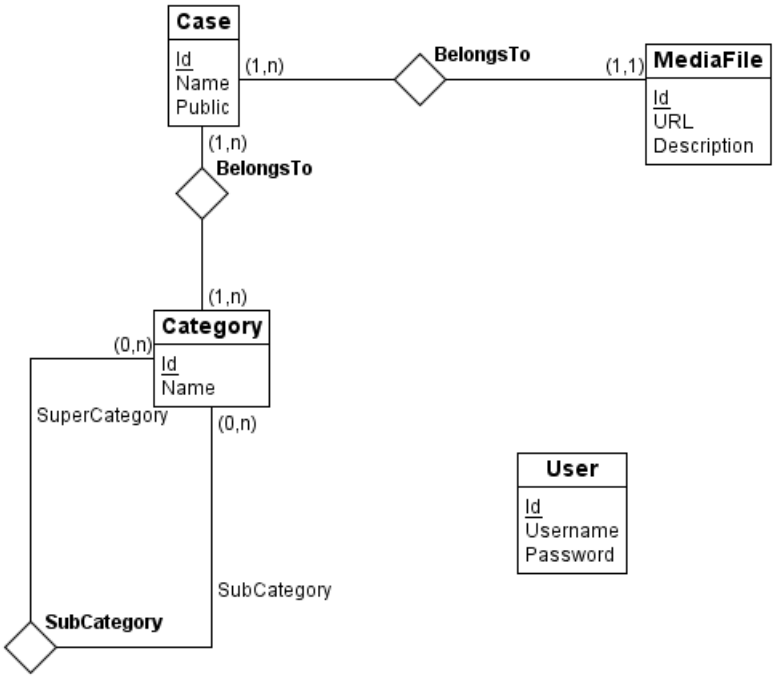

Figure 6.1: ER diagram

The reasoning behind this ER-diagram is as follows: A category must be identified by a name for the user, and must have an ID to use as foreign key. A category can have zero or more subcategories, and zero or more super-categories. Several categories can have the same subcategory, and vice versa,

to allow for reuse of certain categories, for example locations such as “frontal lobe.” To then decide which case belongs to which selection of categories, each case will belong to a tree-structure of categories. Therefore, each case must be connected to all the categories along a path. For instance if case 1 is from a “primary tumor” surgery located in the frontal lobe, case 1 will belong to all of the categories “primary tumor,” “tumor,” and “frontal lobe.”

6.2.3 Images and videos

The MySQL database will only contain URLs to the images and videos, while the images and videos themselves will be stored on a web server to allow streaming the media data directly to the phone application. The media files will be organized after cases, but will not contain any metadata at all. Metadata will come from the database.

6.3 Selections of Architectural Viewpoints

In order to illustrate the architectural viewpoints we have decided to use the “4+1 architectural view model”[38]. This is view model designed for "describing the architecture of software-intensive systems, based on the use of multiple, concurrent views."[38] This model consists of four views of the system seen from different stakeholders, in addition to use cases (the “+1”), in order to accurately portray the architecture of the system.

The different views are:

View	Description	Purpose	Tables
Development view	The development view is concerned with the system from a programmer's perspective; it focuses on software management.	To aid with implementation	Component diagram
Logical view	The logical view describes the functionality that the system provides for its end users.	To create an overview of the system	Sequence diagrams
Process view	The process view illustrates the dynamic aspects of the system; it explains how the system communicates. This view addresses concurrency, performance, and scalability, etc.	To model the runtime behavior of the system	Activity diagram
Physical view	The physical view describes the system from an system-engineering viewpoint. It handles system topology and the connections between the layers of the system.	To show the physical components of the system and how they are connected	Deployment diagram
Scenarios	Scenarios are illustrated through several different use cases and describe the interaction between users and the system.	To provide a demonstration of how the system can be used	Use cases Use case diagram

Table 6.1: Architectural viewpoints

6.4 Views

6.4.1 Develoment view

The overall architecture is such that the phone application primarily communicates with a PHP script at startup in order to fetch the categories, cases, and also some references to the media files (images and videos) for each case. The PHP script receives HTTP GET requests, and responds to them by reading from an underlying SQL database. It then responds to the request by sending back JSON objects representing the categories and cases, using the same structure used in the SQL database to ensure consistency. The reason we need this PHP script between the SQL database and the phone application, is that while Titanium does not support communcation with external SQL servers (only internal SQLITE servers), it does support HTTP communication and JSON parsing. When the user browses to a specific case, the app will go to a web server and stream the media files directly from there. If the user decides to cache som media files locally, the app will simply download them from the web server, and save them in the device's local storage.

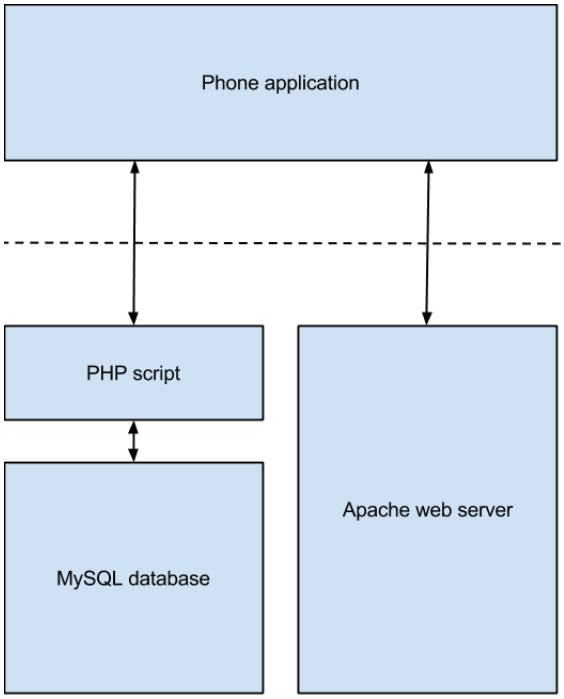

Figure 6.2: Overall architecture

The image above is largely just a "classification" of the ER-diagram, maintaining an internal representation of the category and case hierarchy, and the media file references belonging to each case. It also provides a component for logging in, and one for communication with the database.

6.4.2 Logical view

The following four sequence diagrams show how the system can be used by the end users and how the different modules of the system communicate with each other during use.

Class diagram

This diagram briefly describes the classes, and their relationships, that we used in our application.

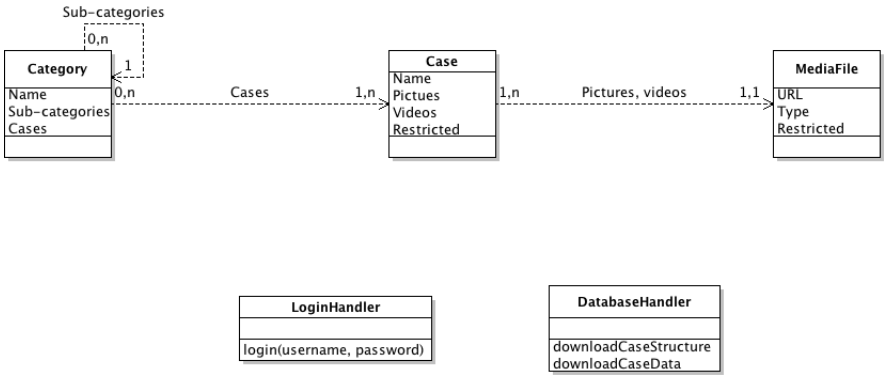

Figure 6.3: Class diagram

Sequence diagram 1

This diagram depicts what happnes when the user of the application is browsing the available categories looking for a case, finds the case, accesses it, and views the media. It works by simply selecting the wanted category, and then browsing through the sub-categories the same way. Finally, when the user has reached a category that does not contain any sub-categories, the user can select a case, and view its media files.

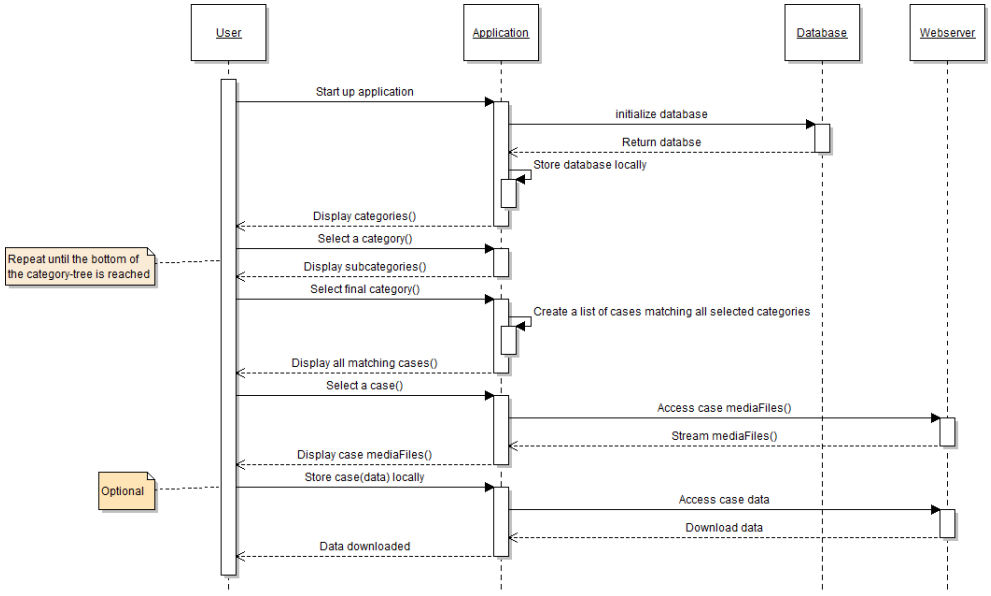

Figure 6.4: Sequence diagram 1

Sequence diagram 2

This diagram describes the use of the search function in our application. The user starts the application which connects to the database and stores the data locally. The user then selects the tab for searching. The application changes to the search interface window and asks for user input. The user then keys in the desired keywords before the application searches for all cases matching the keywords and displays the results to the user. The user then selects a case and is refered to the remote webserver to access the media files belonging to the case. The user then is free to stream all the case media.

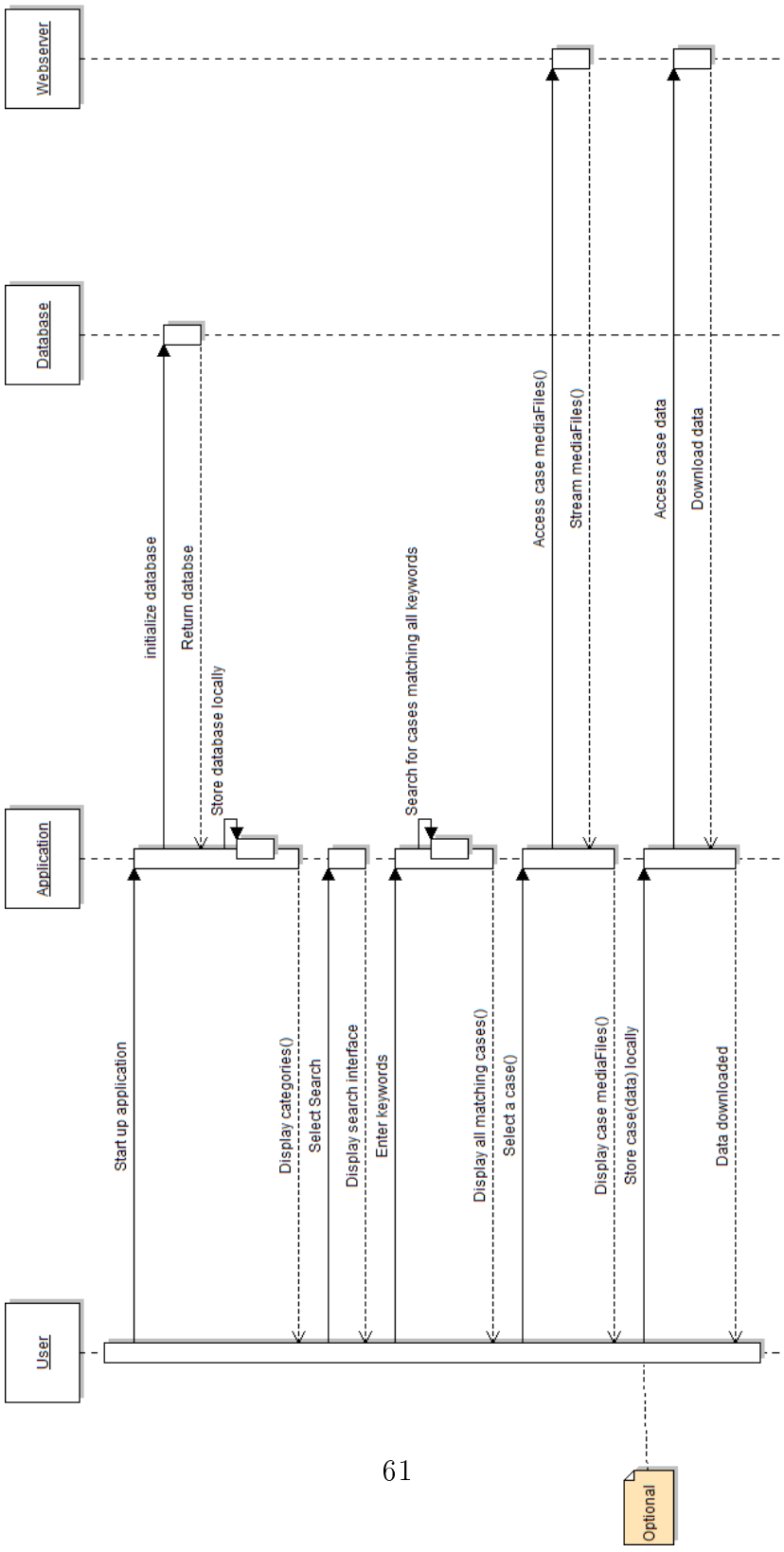

Figure 6.5: Sequence diagram 2

Sequence diagram 3

This diagram shows how the administrator can store new case data in the database and on the webserver through our admin interface. This works by entering case name, and then the new case's sub- and super-categories, before, finally, uploading the case's media files.

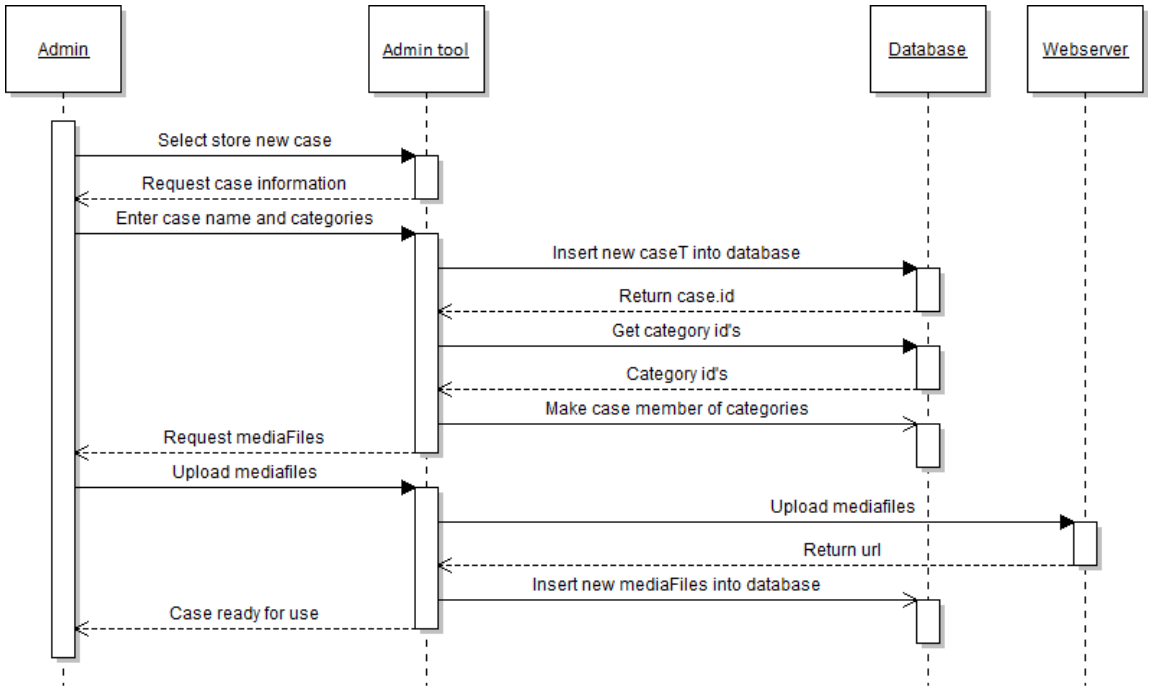

Figure 6.6: Sequence diagram 3

Sequence diagram 4

This diagram shows how the admin can create new categories, should the need arise, through the admin interface. This works by entering the new category's name, before selecting parent and children categories.

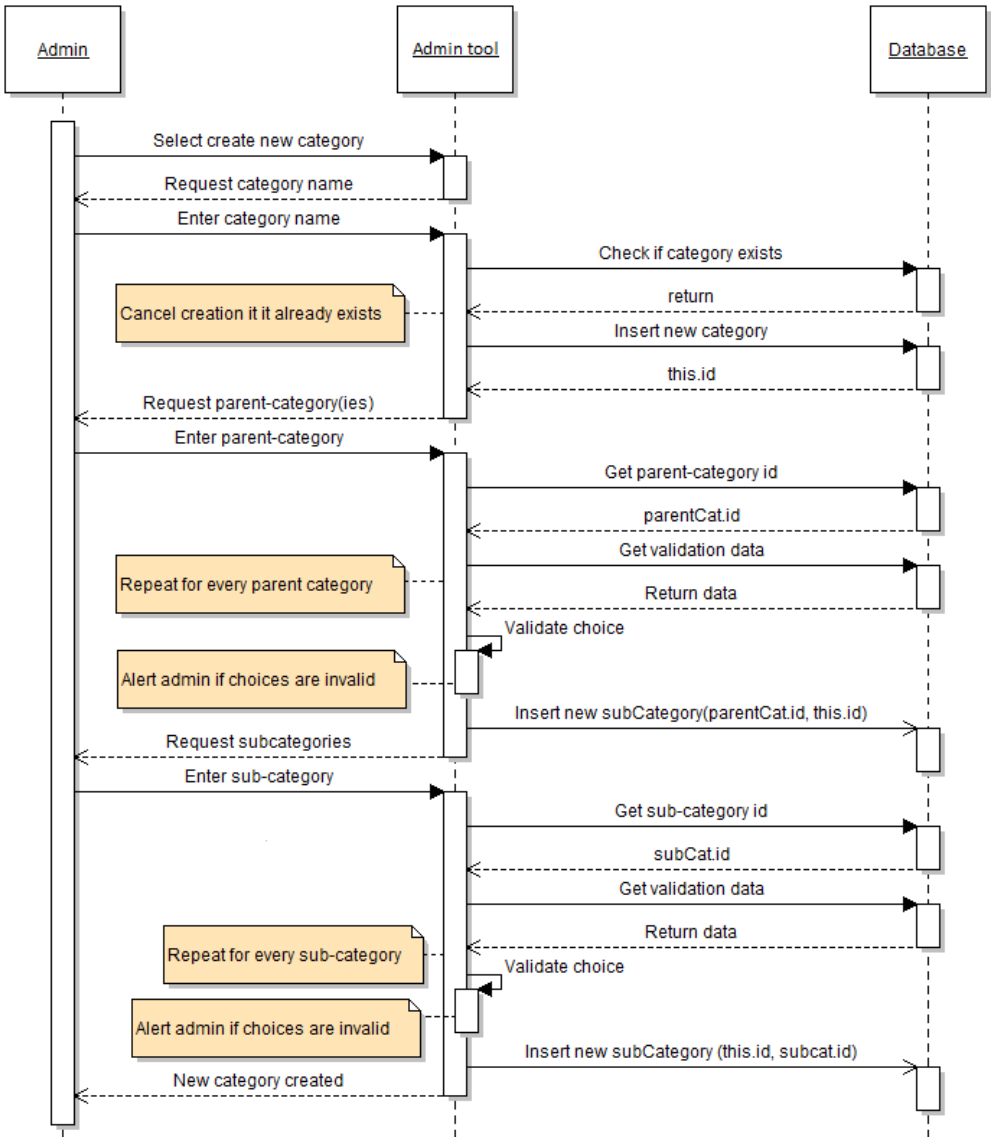

Figure 6.7: Sequence diagram 4

6.4.3 Process view

Browse activity diagram The activity diagram below shows the flow of the mobile application when a regular, or logged-in, user is browsing. The program starts by displaying a list of "main categories." From then, until either "Show all" is selected, or there are no more sub-categories, a new list of categories is displayed for selection, where these categories are the sub-categories of the previously selected one. When there are no more sub-categories or "Show all" was selected, all cases matching the selected categories are displayed. Selecting a case displays the media files that belongs to the case.

In addition, for each step, there is a "back" button, which takes the user back to the previous state. (This is left out of the diagram for simplicity.)

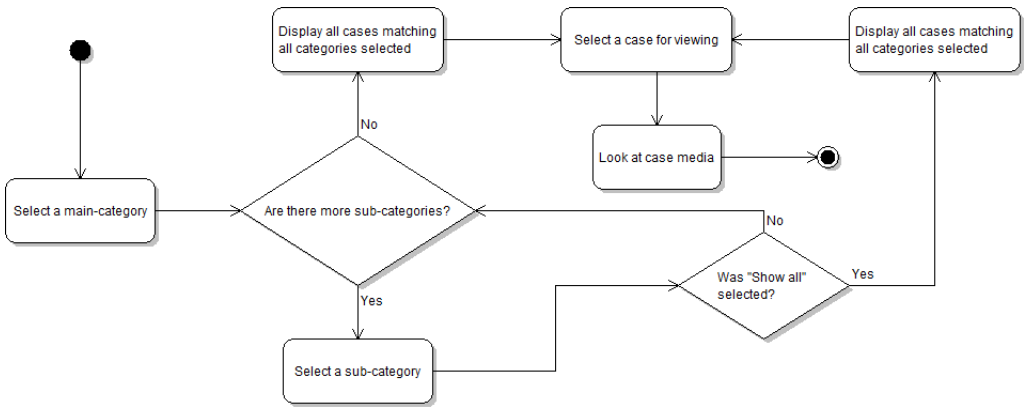

Figure 6.8: Browse activity diagram

Search activity diagram The next activity diagram, shown below, shows the flow of the mobile application when a regular user uses the search function. The app begins by displaying a text field where the user can enter keywords to search for. Following that, cases matching the search keywords are displayed for the user to select among. The user is next allowed to select a case for

viewing, or to modify the search. If the user decides to modify the search, the app returns to the starting state, with a text field for searching. If the user selects a case, however, the case's media files are displayed, the same as if the user was browsing the "regular" way. After viewing a case, the user can opt to view at the other cases that matched the search criteria. If this path is taken, the user is returned to the list of matching cases, as if the search were performed again using the same keywords. When the user decides to not view any more cases or modify the search, the search function is exited.

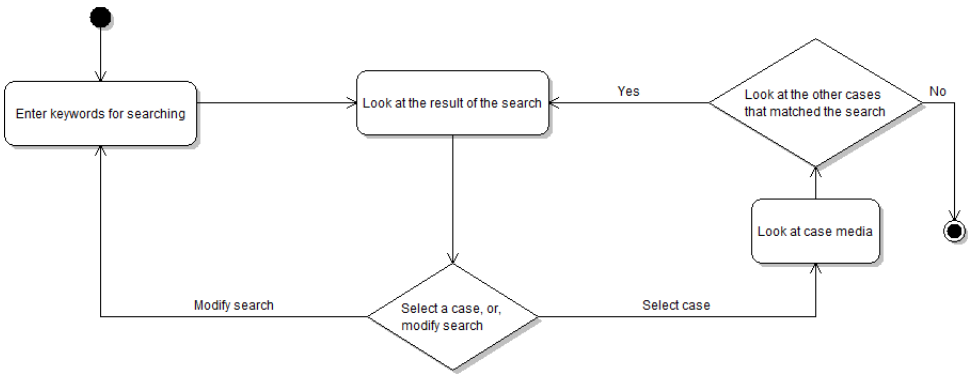

Figure 6.9: Search activity diagram

6.4.4 Physical view

The following deployment diagram describes the overall system architecture of the four components that make up the whole system.

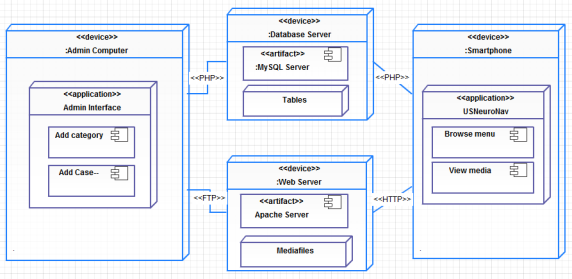

Figure 6.10: Deployment diagram

Admin computer The admin computer is used to add categories and/or cases to the server. Its main functionality is to add new categories and cases to the database server, using a connection to a PHP script. In addition, when adding a case, the admin can add media files to the case. This is done by uploading them to the web server using, for instance, FTP.

Database server The database server's task is to run the MySQL server, and to answer queries to it. The database responds to either select queries from the mobile application, or inserts/edits from the admin computer. The MySQL server consists of several tables, as is referenced in the sub-chapter about "Data Storage." These tables contain information about the categories and cases, and the relationships between them. In addition, a reference to the media files located on the web server is stored for each case.

Web server The web server's task is to store and make available on the Internet the media files associated with every case. This is done by running an Apache web server, which hosts the media files so that they can be streamed or downloaded using the mobile application, using regular HTTP and JSON objects. In addition, the web server must allow the admin computer to upload new media files, either for new cases, or for pre-existing ones. This can be done by using, for instance, FTP. A media file can either be an image or

a video.

USNeuroNav, the mobile application USNeuroNav is the name of the mobile application, which can be run on smartphones. The app communicates with the database server to fetch information about the categories, cases, and media files. This communication goes through a thin layer of PHP script, that actually reads the database, and passes on the information. The media files themselves are fetched from the web server using regular HTTP.

6.5 Architectural Rationale

We tried our best to stick to the traditional Model View Controller (MVC) architecture during planning; this is a tried-and-tested architecture that provides separate modules and provides us with the ability to modify our GUI late in the development should the customer have any change of mind or other reasons to modify our interface. Given the framework we ended up with (Titanium) and our unfamiliarity with the coding language (Javascript), we did run into some minor problems with sticking to MVC, but as we became more familiar with it, the problems seemed to be solved one by one.

7 Testing

This chapter explains how we performed the testing of our project. This includes day-to-day testing of the code we implemented, but also the final acceptance test, and usability test executed with the customer and the future users of the application.

7.1 Unit Testing

"The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and

determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use."[39] Unit testing is one of the cornerstones of agile development. With proper use of unit testing we reduced the amount of potential bugs and created more reliable software. Because of the limited time we had during this project we didnt write all the unit tests before we started programming, but each programmer was responsible for unit testing the part of the code that he/she implemented.

7.2 Integration Testing

Integration testing is the next step when testing software. This tests takes the parts, or units, of code, integrates them and makes sure they work as expected. It is not uncommon for units to function well independently, but somehow fail when they are connected. For this reason it is very important to perform this test. By performing unit tests (see 7.1) and passing them, we ascertained that any errors that appeared when connecting units originated from the interface between the units and not the units themselves. This was an advantage that made it significantly easier to fix any errors.

Again, because of the time constraints on the project, we did not write the integration tests beforehand, but rather executed them as we integrated units. For every new unit added to our system, we made sure an integration test was performed on every part of the system affected by the new unit. We also took caution not to add more then one unit at the time. By programming and testing this way, we minimized the number of errors at every part of the implementation.

7.3 System Testing

System tests are executed to test the functionality of the product. The goal is to make sure the system behaves in the intended and expected way, and

that it provides the functionality that is required of it. System testing should require no knowledge of the inner workings of the system, and should be able to be performed by anyone. It involves the core functionality of the system as seen by an end user.

Since both unit- and integration testing were only done by the programmer working on that particular code, it was important to thoroughly document and perform system tests in every sprint. The system tests were written based on a combination of the requirements and the usecases, making sure to include all the core functionality of the application. The tests were mainly performed by the members of the team, but they were prepared and executed in a manner such that they could have executed by the customer or any other user, should that have been desired. Below is a table containing all our system tests; the complete description of each test can be found in appendix C.

Identifier	Test name	Description	Requirement
T01	Browse categories	Browsing in the categories hierachy	FR01
T02	Select a case	Selecting a case and getting the case view	FR01
T03	Browse images	Selecting images and browsing them	FR02
T04	Play video	Selecting a video and playing it	FR03
T05	Search for case	Searching for a case by category	FR04
T06	Log in	Logging in and confirming additional options	FR05, FR09
T07	Add new case	Admintool core functionality	FR10, FR11
T08	Add new user	Admintool core functionality	FR09, FR10, FR11

Table 7.1: System tests

7.4 Acceptance Test

A user acceptance test is conducted at the very end of the development process to determine whether the system meets the customers requirements. The purpose is to assure that the project is accepted by the customer, and that it at the very least meets the minimal requirements set so that it may be utilzed for its intended purpose. It is common for the intended users of the products to execute the tests so that the customer can be sure that the product safisfies its potential users. The acceptance test "acts as a final verification of the required business functionality and proper functioning of the system, emulating real-world usage conditions on behalf of the paying client or a specific large customer."[40] The test itself is "usually performed by clients or by end-users, [and does] not normally focus on identifying simple problems such as spelling errors and cosmetic problems, nor showstopper defects, such as software crashes; testers and developers previously identify and fix these issues during earlier unit testing, integration testing, and system testing phases."[40]

We held our acceptance test during a meeting with both the customer and a small group of the intended users. We both showcased the product and offered them use of the product and requested that they provide feedback. Details from this meeting can be found in section 13.2.2.

Part II
Sprints

8 Sprint 1

In this section the first sprint is described. First the planning and goals for the sprint, then the burndown chart is presented followed by the implementation. After that the occurring risks, then feedback from the customer, followed by an evaluation of the sprint.

8.1 Sprint Planning

The duration of sprint 1 was from 18th of September to 1th of October.

During a large part of sprint 1, Marthe was away on a planned trip. To accommodate for this, Marthe had responsibility for gathering all the material that we currently had created in Google Drive, and put it together into a single LaTeX document. In addition, she started to looking into the test planning. We chose to move our report into a single LaTeX document so that we had better options for better version control, and also to familiarize ourselves with LaTeX, as t the final report would be written in LaTeX at the end.

Kristoffer and Stian, who were in Trondheim for the duration of the sprint, started on the implementation. To begin with, Stian would set up a raw skeleton for displaying the categories (hardcoded for now), while Kristoffer finnished a demo of the GUI, which would be shown to the customer and the supervisor. The next step was to display cases, along with their images and videos. All categories, cases, images and videos was stored locally to begin with, to focus only on the simple parts first. Before development could start, the iMac, borrowed from the customer, had to be set up with the required software and licenses. At that time the only software required was XCode and Titanium Studio. As Titanium was free to use, no license would be required at this step, but might be required if testing was to be done on a "real" device, and not the iPhone simulator which Titanium makes available.

Since the team were quite inexperienced with scrum, an additional feature was planned, that would be implemented if the initial tasks were completed within the first half of the sprint. That feature was loading the categories,

cases, images and videos from the server. To allow for this, the server had to be set up with a MySQL server that stored the categories and cases, and a web server (Apache) that made the images and videos available. Setting up the server with the required software, and configuring it, was assigned to Stian, as he had some experience with it. Implementing the communication between the client (app) and the server, would be done in cooperation between Stian and Kristoffer, as they were the "developers" for this sprint. In more detail, Stian would do the server side communication, and Stian and Kristoffer together would work on the client side.

The customer provided us with images and videos from several real-life cases (surgical operations), and a description of how all these could be categorized, to enable us to use the actual future content for develolping and testing the system.

For sprint 1, we tried to complete two milestones, MP1, "Have a running prototype, able to show local photos, categorization of cases to enable browsing", and MP2, "Support for video, searching for cases, network support, server communication".

8.1.1 Sprint goals

The main goal of this sprint was to have a running prototype which could display categories and cases, and their images and videos. At that moment, these data were stored locally or with hardcoded URL's. Another goal was to come to an agreement with the customer for a rough GUI specification. Finally we wanted to transfer the report over from Google Drive into a LaTeX documents, together with a plan for the overall structure for the report.

8.1.2 Sprint backlog

The sprint backlog for is shown in the table below. It shows all the tasks we completed and estimated effort that was planned.

ID	Task	Main responsible	Estimated cost	Real cost
BL1	Categories	Stian	8	10
BL1.1	Browse window	Stian	2	2
BL1.5	Browse categories	Stian	8	6
BL1.2	Display categories	Stian	4	4
BL1.3	Display cases	Stian	4	3
BL1.4	Window navigation functions	Stian	4	6
BL2	Cases	Stian	6	4
BL2.1	Case window	Stian	2	1
BL2.3	Show images	Stian	8	2
BL2.4	Play movies	Stian	6	1
BL3.1	Search window	Stian	1	2
BL7	GUI Design	Kristoffer	11	12
BL7.1	GUI for categories	Kristoffer	2	1
BL9	Servers	Stian	6	4
BL9.1	Apache server	Stian	4	2
BL9.2	Apache setup	Stian	2	1
BL9.3	Database server	Stian	2	2
BL10	Database	Kristoffer	4	2
BL10.1	Insert tables	Kristoffer	1	1
BL10.2	Insert initial case data	Kristoffer	3	4
R1	Report	Marthe	30	35
R1.1	Create and setup LaTeX document	Marthe	3	4
R1.7	Requirements	Marthe	10	12
R1.8	Software architecture	Kristoffer	10	18
R1.9	Sprint 1	Stian	8	10
R1.14	Evaluation (Testing)	Kristoffer	10	6
Sum			159	155

Table 8.1: Sprint 1 backlog

8.2 System Burndown Chart

The figure below shows the burndown chart for this sprint. We managed to finish almost every task in this sprint, but we had both overestimated and underestimated the effort of time for many tasks. Therefore the ideal remaining effort line is slightly beneath the actual remaining effort line through most of this sprint. We did most of the work on Wednesdays, Thursdays and Fridays, which explain the horizontal line at day 1, day 2 , day 6, and day 7.We had to do a lot of programming in this sprint since our technical manager would be absent during the next sprint, so a lot of implementatuin was completed this sprint.

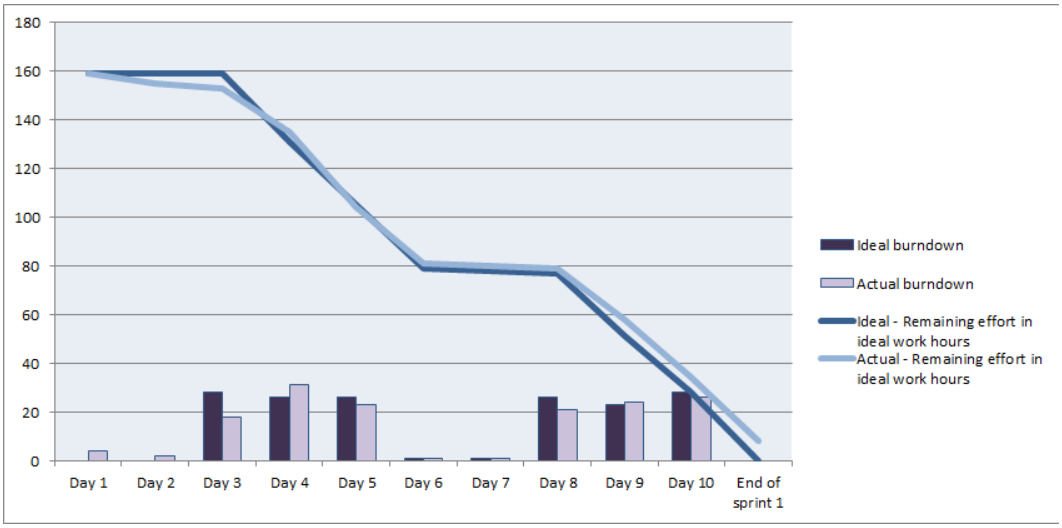

Figure 8.1: Burndown chart for sprint 1

8.3 Implementation

Here is a description of everything that was implemented in this sprint.

8.3.1 User interface

A lot of the implementation for this sprint was to program the user interface for viewing categories and cases, and their images and videos. In the early phase of the sprint, we made a GUI demo with some options, which we intended to show to the customer. However, the customer was unavailable for two and a half week, so no meeting was possible during sprint 1. The demo was therefore presented to him in an email, and he agreed with our design choices, and urged us on.

The first decision which had to be made, was to figure out how we "split" the app into the three main "functions" that it should support, which was browsing categories, searching, and logging in. For this, we decided to use "tabs", one for each "function". See the figure below:

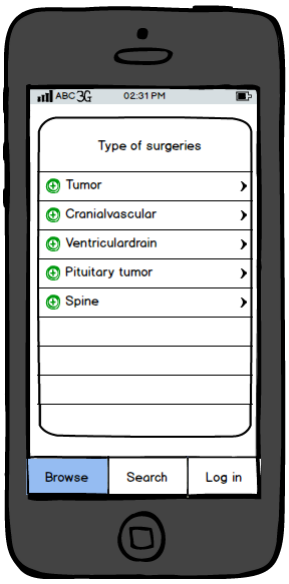

Figure 8.2: GUI demo: Browsing with tabs

The reason for why we chose to use tabs, was because that it is the standard way to split a app into several functions, both for Android and iOS, and

the customer had pointed out that "native look and feel" was very important, and that he wanted an app that looked professional. Tabs were easily supported with Titanium, using a tab group that contains mutliple tabs (three in our case).

Further on, we decided to simply use a list view to that simulates a tree structure to display the categories and the cases. To make the structure of the categories and cases as general as possible, a category could have zero or more sub-categories, and zero or more super-categories, that was "parents". Also, a category could have zero or more cases. Several categories could have the same case, and to decide which case that was shown when navigating the tree structure, we simply took the intersection of the cases for the categories we had selected. Below is a demo of the navigation:

Figure 8.3: GUI demo: Browsing navigation

When a case was selcted, the case's images and videos would be displayed. To begin with, this was done be simply taking the user to a view of the first element (video or image), and from there, the user could swipe between images and videos belonging to the selected case. We had discussed the possibility for a thumbnails view or something similar also, but no decision about this had been made, and at this point we were postponing this until the customer could have his say on the matter.

8.3.2 Client-server communication

Server side As Titanium did not support direct communication with an external MySQL server, a dedicated component to compensate for this had be developed, and run on the server side. Since Titanium supported HTTP connections, a PHP script would run on the server's web server, and would be used to read data from the MySQL server, and send it to the client upon request. In more detail, the server would answer a HTTP GET request, with a JSON list, which was built in the same way as the MySQL server to ensure consistency. The reason for sending JSON objects instead of, for example, XML objects, was that JSON objects typically were smaller, and faster to parse. Since the MySQL server only contained references to the images and videos, the data sent here would not be too large, and it would allow the client to stream the images and videos from the Apache web server, by sending the reference as URL's.

Client side When the application was booting up, the client would poll the server for data once by sending an HTTP GET request, using Titanium's build-in support for this. The client would then receive the JSON list, which Titanium also had built-in support for parsing. Using this data, the app would build an internal structure of the categories in memory, and use this to generate the tree-structured navigation. The JSON object would also contain references to where the images and videos for each case could be streamed from, so these would also be stored locally.

8.4 Testing

The following tests were executed on September 27th 2013.

Test ID	Test name	Requirement	Result	Comments
T01	Browse categories	FR01	Passed	-
T02	Select a case	FR01	Passed	-
T03	Browse images	FR02	Passed	-
T04	Play video	FR03	Passed	-

Table 8.2: Completed tests for sprint 1

8.5 Occurring Risks

The risks that occurred during sprint 1 are summarized in the table below, along with the reactions taken.

ID	Risk factor	Explanation	Actions taken
R1	Sick group member	Kristoffer was sick 1.5 days	The group was already ahead of time, so no actions were necessary
R8	Underestimated time for tasks	We did not finish all the tasks during this sprint, and by looking at the burndown chart, we were slightly behind schedule during the sprint.	Since we had both underestimated and overestimated time requirements for the tasks for this sprint, it did not become a huge problem, since they canceled each other.
R12	Absent group member	Marthe were gone travelling during a large part of the sprint.	Marthe informed the group about the absence in good time before the period in question, so the group was able to plan with this in mind. The agreement was that Marthe was going to work on some pre-defined tasks while she was away
R2	Sofware issues	We had some difficulties with Rally. We did not have all the rights that were required to make a burndown chart.	We created the backlog in Rally, and created the burndown chart in Exel.

Table 8.4: Occurring risks in sprint 1

8.6 Customer Feedback

As the customer was away travelling during the whole sprint, all contact with him was done via email. Prior to his travel, we decided that the sprint delivery, where the team showed what had been done this sprint, and would show a prototype to the customer if possible, were to be postponed until his arrival, somewhere in sprint 2. Thus, the only feedback from the customer during sprint 1 was the feedback he gave on the GUI demo which was sent. His feedback regarding that was that he very much liked the navigation, but he also wanted text on the back-button, explaining where it led. This was exactly what the group intended, but it did not show very well on the demo. He also pointed out that categories would change, which will be fully supported, by reading them from the SQL database. The categories in the demo were only examples. Other than these few comments, the customer agreed with our design choices.

8.7 Sprint Evaluation

Since we only had one iMac we chose that only two people should develop the system.

The implementation of the initial stories, viewing categories and cases, and their images and videos, went very well, and were done within the first half of the sprint. The team therefore moved on to set up MySQL server and Apache web server on the borrowed iMac, to use it for the development phase. This also went without too many issues, even though the team was very unfamiliar with Mac and OSX. Finally, the development of the client-server communication also went well, considering no one in the group had used PHP or Titanium before. As an aside, the development team for the first sprint was not very comfortable with using Titanium and JavaScript, but adapted to this by "learning by doing". The team was still not certain that the right development tools were chosen (Titanium vs Xamarin), as they felt they would be more productive writing code in C# (Xamarin) than JavaScript

(Titanium), considering that JavaScript in theory does not support classes.

We had some difficulties with Rally during this sprint. We managed to create the backlog and manage it thereafter, but we did not have all the rights to add the data required to make a burndown chart. So we maintained the backlog in Rally, but created the burndown chart in Exel. We continued this procedure in the next sprints as well.

The milestones MP1 and MP2 was completed within time.

8.7.1 What went well during this sprint

The development phase proceeded well. We had absolutely no prior experience with Titanium, JavaScript or Mac before the sprint. We also didn't have access to more than one Mac. All things considered, we were pleased with the results of the coding.

We didn't have much progress on the report early in the sprint, but during the last week of the sprint, a surge of content was produced, which we were pleased with.

8.7.2 What could be improved for the next sprint

We had a slightly slow start with the document-writing this sprint, primarily because only the "developers" for this sprint were fully available during the whole sprint.

9 Sprint 2

In this section the second sprint is discussed. First the planning and goals for the sprint are presented, followed by the burndown chart. After that, the occurring risks, then feedback from the customer, and lastly an evauation of the sprint are described.

9.1 Sprint Planning

The duration of sprint 2 was from the 2nd of October to the 15th of October.

We managed to complete nearly every task from the previous sprint. There were only two tasks that had to be pushed to the backlog of sprint 2, R1.8, "Software architecture," and R1.9, "Sprint 1."

Our technical manager, Stian, was gone on a planned trip during half the sprint, and while he was away he worked on the report. Kristoffer and Marthe also focused on the report as there was a pre delivery on the 14th of October, and we wanted to get as much feedback on the report as possible.

We decided that there would not be any new features implemented during this sprint. This was because of the pre delivery of the report, and also due to the fact that Stian was gone most of the sprint.

The customer returned from his trip the 9th of October, which was in the middle of our sprint. After that we agreed to try to have regular meetings every second week.

For sprint 2, we planned to complete one milestone, MR2, "Support for video, searching for cases, network support, server communication."

9.1.1 Sprint goals

The main goal of the sprint was to have written as much as possible of the report. Since our technical manager was gone for half the sprint, the report had our full attention.

On the report, we wanted to be finnish writing the entire Part 1, and the

chapters of Sprint 1. We also wanted to begin writing the chapters of Sprint 2.

9.1.2 Sprint backlog

The sprint backlog for this sprint is shown in the table below. It shows all the use case stories, with the tasks and estimated effort that were planned for this sprint. We learned from the last sprint that we had underestimated the tasks, so we attempted to estimate more accurately during this sprint. This sprint's backlog also contains the items that were pushed over from sprint 1.

ID	Task	Main responsible	Estimated cost	Real cost
R1	Report	Marthe	40	44
R1.9	Sprint 1	Stian	5	5
R1.2	Abstract and preface	Kristoffer	8	9
R1.3	Introduction	Kristoffer	11	11
R1.4	Planning	Stian	14	12
R1.5	Preminary studies	Stian	8	8
R1.6	Lifecycle model	Marthe	6	5
R1.8	Software Architecture	Kristoffer	3	4
R1.10	Sprint 2	Marthe	16	15
Sum			111	113

Table 9.1: Sprint 2 backlog

9.2 System Burndown Chart

The figure below shows the burndown chart for this sprint. We managed to finish nearly every task in this sprint, and we had estimated the effort more accurately. The actual remaining effort line is beneath the ideal remaining effort line for almost the entire duration of the sprint, indicating that we were not behind schedule. We felt we had made good progress, but since we did not implement anything this sprint, the progress did not benefit the application.

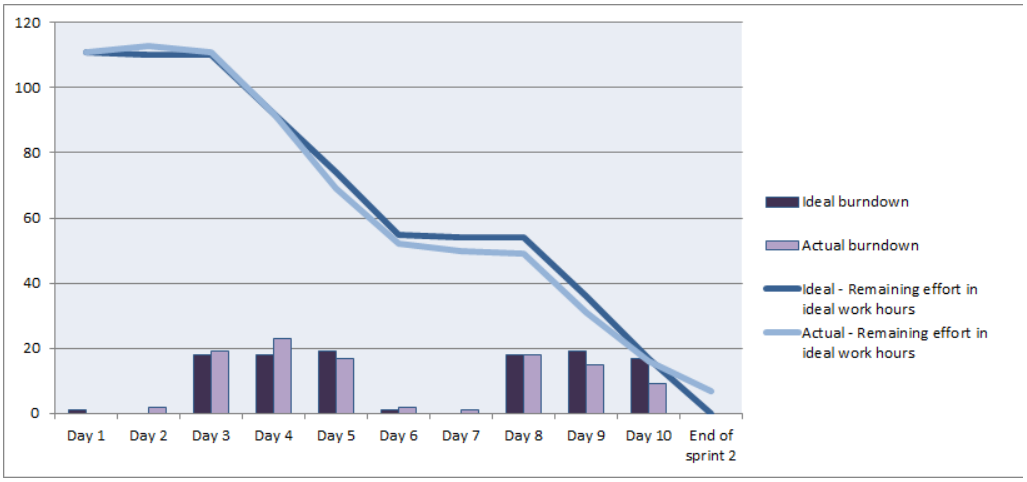

Figure 9.1: Burndown chart for sprint 2

9.3 Occurring Risks

The risks that realized during sprint 2 are summarized in the table below, along with the reactions.

ID	Risk factor	Explanation	Actions taken
R12	Absent group member	Stian was gone traveling during a large part of the sprint.	Stian informed the group about the absence a good time before the period in question, so the group was able to plan with this in mind.

Table 9.3: Occurring risks in sprint 2

9.4 Customer Feedback

We had a meeting with the customer on 9th of October, and he did approve the progress to date. He made some comments on how the application would work on iPhone, specifically with zooming and tilting of images and videos. We agreed on the further work and the priorities of the remaining tasks, as shown below:

Id	Task	Previous priority	New priority
FR07	Save images locally on the phone	Medium	Low
FR08	Save video locally on the phone	Medium	Low
FR06	Registration	Medium	Low

Table 9.4: New piority of requirements

We did not have a meeting with the customer at the end of sprint 2, since no implementation was done.

9.5 Sprint Evaluation

We made good progress with the report this sprint; we managed to write everything we had planned for the pre delivery of the report, so the milestone MR2 was completed within the time constraints.

9.5.1 What went well during this sprint

The progress on the report was very positive. We managed to complete nearly all the tasks of this sprint's backlog, and there was no rush during this sprint.

9.5.2 What could be improved for the next sprint

Since we were not a complete group the whole sprint, we planned for too little work. In the next sprint, we had to bear this in mind during the planning so that we could complete the most important requirements from the customer.

10 Sprint 3

In this section, the third sprint is described. First the planning and goals for the sprint, then the burndown chart is presented, followed by the implementation. After that, the occurring risks, then feedback from the customer, followed by an evauation of the sprint are all discussed.

10.1 Sprint Planning

The duration of sprint 3 was from the 16th of October to the 29th of October.

We completed nearly every task from the previous sprint, and there was only one task that had to be pushed to the backlog of sprint 3: the R1.10, "Sprint 2."

The plan for sprint 3 was to pick up development where we left off at the end of sprint 1. During a meeting with the customer in sprint 2, we received some feedback regarding the display of media files. The most important feedback was that the customer wanted to be able to "tilt" the phone and have the image/video to rotate. He also wanted to be able to zoom using "pinching." Finally he wanted us to separate the videos and images for a case into two categories, "videos" and "images," and add some meta data. These things were to be implemented during sprint 3.

In addition, we planned to implement separation of data with log in, so that registered users have access to all cases, while unregistered users only have access to public cases. We also planned to add functionality to search for cases using category keywords, for instance "Tumor" or "Primary Tumor." Finally, we decided to start work on a standalone program that could aid the future administrator of the database to upload new items. We felt that the process of creating new cases and categories in the database was too complex and time-consuming to do manually. This, and the fact that a GUI for uploading was already requested by the customer, made us shift some resources towards this new program, which we will refer to as "Admin tool."

The reason we set such ambitious goals regarding the amount of implementation for sprint 3 is that we laid a very thorough foundation in sprint 1 that made it fairly simple to add new features, and that we did not have the time or opportunity to do any coding during sprint 2.

For sprint 3, we tried to complete milestone MP3, "Upload of data."

10.1.1 Sprint goals

The goals for sprint 3 were to implement the feature requests from the customer, and to have some progress on the functional requirements, by implementing separation.

10.1.2 Sprint backlog

The sprint backlog for this sprint is shown in the table below. It shows all the use case stories, with tasks and estimated effort planned for this sprint.

ID	Task	Main responsible	Estimated cost	Real cost
BL2.2	Show case metadata	Stian	8	12
BL3	Searching	Stian	2	2
BL3.2	Implement searching	Stian	10	12
BL4	Separation of data	Stian	18	20
BL4.1	User login	Stian	2	3
BL4.2	User registration	Stian	4	2
BL7.2	GUI for cases	Kristoffer	3	2
BL7.3	GUI for searching	Kristoffer	3	3
BL8	Admin tool	Kristoffer	6	10
BL8.1	Admin application	Kristoffer	10	14
BL8.2	Add data	Kristoffer	6	6
BL8.4	Add new user	Kristoffer	2	2
BL8.5	Talk to webserver	Stian	4	2
BL8.6	Talk to database	Stian	4	2
R1	Report	Marthe	20	18
R1.10	Sprint 2	Marthe	7	7
R1.11	Sprint 3	Marthe	15	12
Sum			124	129

Table 10.1: Sprint 3 backlog

10.2 System Burndown Chart

The figure below shows the burndown chart for this sprint. We managed to finish every task in this sprint, and we felt that we had estimated the effort very realistically. We made good progress with the work, but were slightly behind schedule in the first week. We ultimately caught up with the plan in the last week of the sprint.

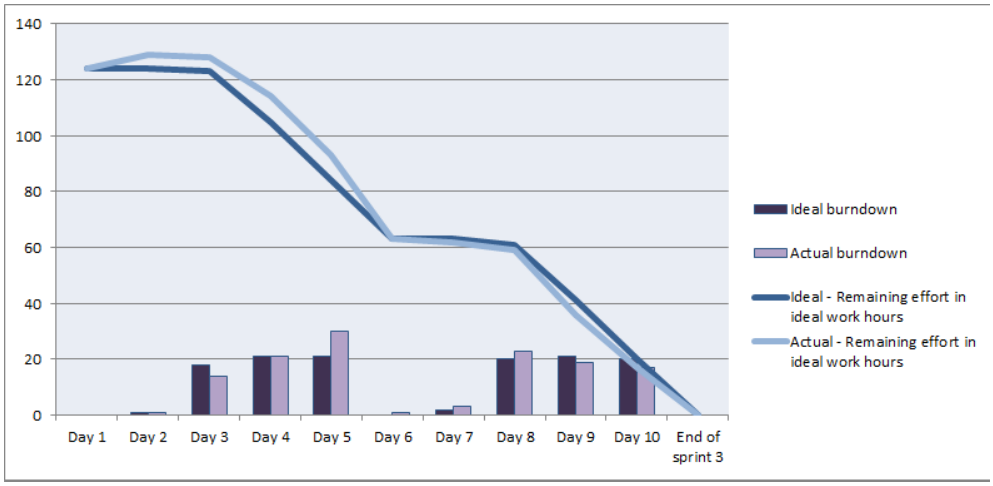

Figure 10.1: Burndown chart for sprint 3

10.3 Implementation

Following are descriptions of items that were implemented in this sprint.

10.3.1 Searching

Searching for cases was one of the customer's top requirements, and therefore the highest prioritized feature for sprint 3. It was implemented on a separate "tab."

10.3.2 Separation

Separation of data was another of the customer's high priority requirements. The intention was not to provide airtight security for the private cases, but rather to show only a selection of cases to unregistered users. All cases were therefore left as they were on the web server, and sent over to the phone on app startup, however, only the cases marked as public would be displayed if the user wasn't logged in. The reason that we decided to send all cases on startup, instead of only sending the public ones, was that we would potentially have to send more cases after log-in, and then rebuild all the menus. This

would have caused additional loading time after logging in. Since security in itself was not a very high priority for the customer, it was decided that loading all cases on startup was the best solution, prioritizing performance over security.

Another much-discussed point was the registration of new users. The customer had no intention of allowing "random" users to register, so registering users using the admin panel (discussed in section 9.3.4) was deemed appropriate.

The log-in process itself was placed on a separate tab, and consisted of only fields for username and password and a "Log In" button. After the user has logged in, this tab displays only a "Log Out" button.

10.3.3 Viewing of media files

As previously mentioned, we received some feedback on the viewing of media files. We therefore implemented "tilting" and zooming using a pinching gesture. Both were relatively straightforward, and required only a couple of code lines. Tilting was simply activated by setting a certain parameter on the application windows that should be "tiltable." Zooming was implemented by putting all the media files inside a ScrollPane, which performed the zooming job. The only issue with this was that the view with the image was by default zoomed in upon selecting a case. This was solved by finding the relation between the screen size (which can vary with different phone/tablet models) and the original image size.

In addition we split the media files into two categories, "videos" and "images," which appears when selecting a case. Also, on this new view where the user could choose between videos and images, we added a field for metadata. This field was split in two, after receiving feedback from the customer, to show only some metadata for the unregistered (not logged in) user, and extensive metadata for a registered user. This was implemented by splitting the column for "description" in the SQL server into "publicDescription" and

"privateDescrption," and letting the user specify both when uploading a case.

10.3.4 Admin tool

As mentioned earlier, we realized that a small program, with the purpose of simplifying the upload process, would be necessary. We decided to write it in Java and such that it would be executable on any machine and operating system. We also had libraries to support communication with our MySQL server availble for Java, and lastly, we were all experienced with Java as a programming language, therefore we were confident that we could complete it in reasonable time with the functionality that was necessary. By writing a program to handle the uploads, many parts of the otherwise-tedious insertions into the MySQL database could be automated. For example, to add a new case with Admin tool, the administrator could type in the name of the new case, the super- and subcategories it belongs to, and the description and the path where the media files were stored. The program would then create the new case with the description specified, create the categories if they didn't exist, add the proper relationship between the categories, add the case to the categories, and finally add all the media files with a pointer to the case. Manually this amounts to 40-50 insertions and 8-10 lookups in the database; with Admin tool it is entirely automated. Below is a sequence diagram that illustrates the process of creating a new case and a new category in Admin tool. A more detailed description of the Admin tool is given in Appendix B.

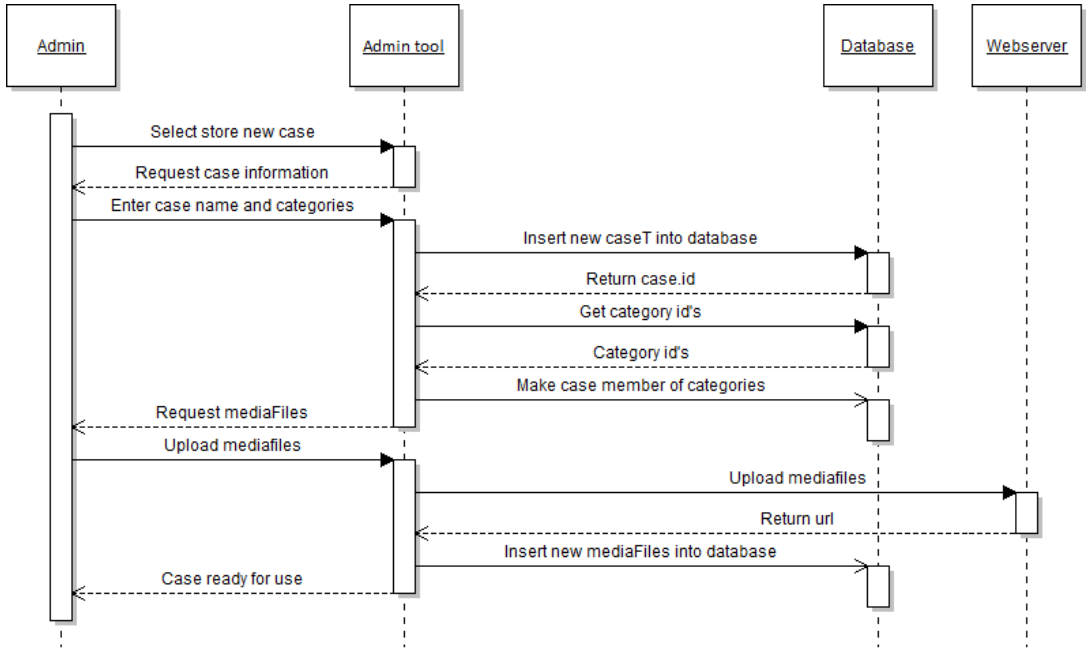

Figure 10.2: Admin tool sequence diagram for new case

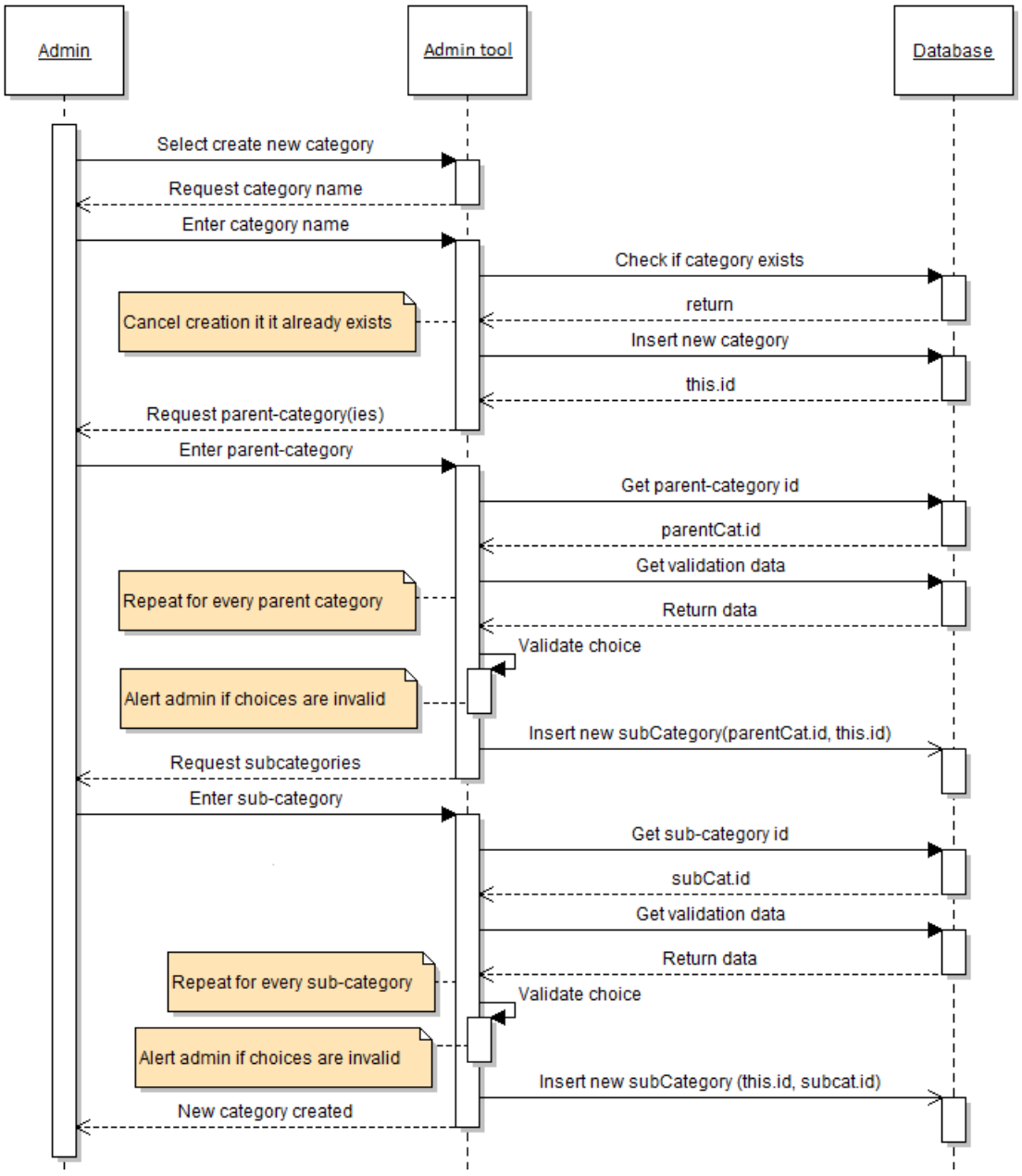

Figure 10.3: Admin tool sequence diagram for new category

10.4 Testing

The following tests were executed on October 28th 2013:

Test ID	Test name	Requirement	Result	Comments
T05	Search for case	FR04	Passed	-
T06	Log in	FR05, FR09	Passed	-
T07	Add new case	FR10, FR11	Passed	-
T08	Add new user	FR09,FR10, FR11	Passed	-

Table 10.2: Completed tests for sprint 3

10.5 Occurring Risks

The risks that realized during sprint 3 are summarized in the table below, along with the reactions taken.

ID	Risk factor	Explanation	Actions taken
R1	Sick group member	Marthe, Kristoffer	The group members worked a little in the evening to complete their tasks.

Table 10.4: Occurring risks in sprint 3

10.6 Customer Feedback

Halfway through the sprint, we had a meeting with the customer to show the progress to-date. We showed that we had implemented zooming and tilting, as requested, and also split videos and images into two categories, and displaying metadata. The customer was satisfied with the changes, but wanted the metadata to be split into two parts, one for the "normal" (not

logged in) user, and one for the registered user. We told him that this would not be a problem, and promised to implement it during this sprint, which we did.

The customer was also very eager to see the application on a real device, not just the emulator, but as Marthe was traveling during sprint 3, and owned the team's only iDevice, this was not possible. We mentioned, however, that we had tested it on her phone, and that the videos were not running very smoothly when streaming over the internet. The main reason for this was that the videos had a very high resolution. The customer did not want to reduce this, so he suggested that we look into the opportunity to download the videos to the device before playing it. As we were approaching the delivery deadline, we made no promises regarding this other than to look into it.

During the meeting, the possibility of demonstrating the application for some real users was discussed. As the application was close to completion, we had no objections to this, but the method for distributing the application to the test users was identified as an issue. Since Apple doesn't allow "unfinished" applications on the App Store, alternative methods had to be researched.

At the end of sprint 3, we had another meeting with the customer to demonstrate our progress and to discuss the goals of sprint 4. The customer liked the idea of the admin tool for uploading new cases, registering new users, etc., but thought navigating through the menus was a little of a hassle. He wanted the admin tool to be able to read information from a text file, and upload cases that way. Other than this, and a few bugs that the team already was aware of (see sprint 4 for details), he was pleased with the application, and thought that no new features were necessary.

10.7 Sprint Evaluation

The targeted milestone, MP3, was completed during sprint 3. Also, all new feature and fine-tuning requests from the customer were implemented. So were all planned new features. We were very pleased with this progress, and

it actually placed us at an even speed with the schedule, despite the fact that we didn't complete any programming in sprint 2.

The work items left over from sprint 2 were completed (i.e., writing about sprint 2 in the report, etc.). Some additional work, primarily formatting and structuring, was also done on the report.

10.7.1 What went well during this sprint

Sprint 3 was very focused on the development aspect. We made excellent progress both on the mobile application itself and on the admin tool that we created. The mobile application was approaching a state where the team felt that all major requirements had been implemented, and only bug-fixing and polishing remained. Considering that we opted to add a fourth sprint wherein some development would occur, we were ahead of schedule.

10.7.2 What could be improved for the next sprint

During sprint 3, Marthe was the only person assigned to work with the report. Prior to sprint 3, we felt that we were ahead of schedule regarding the report, so we did not place particular stress on completing the report. Marthe was, however, traveling during the entire week, and was also sick a couple of days, so the report saw minimal progress. This might have been a mistake, and one more person should perhaps have used half the sprint to work on the report, so as not to fall too far behind. This meant that for sprint 4, the final sprint, we might have to de-prioritize bug-fixing, and instead focus solely on the report.

11 Sprint 4

In this section the fourth sprint is described. First the planning and goals for the sprint are presented, followed by the burndown chart and implementation. After that, the occurring risks, then feedback from the customer, and lastly an evauation of the sprint are discussed.

11.1 Sprint Planning

The duration of sprint 3 was from the 30th of October to the 8th of November.

During sprint 4, the final sprint, the plan was to get the application as polished as possible. This involved fixing several bugs:

#	Bug
1	The application occasionally crashed when opening the "images" section of a case with many images.
2	Video playback did not stop when leaving the video view.
3	The application rotates with minimal "tilting," making it seem a bit unstable.
4	When opening the images of a case for the first time, the initial zoom was too large, and it was not possible to zoom out.
5	After entering full screen when viewing a video, zooming is no longer possible, even when leaving full screen.
6	Selecting "images" on a case with many images, makes the application seem to "hang". In sprint 3, we added a loading animation, which worked very well on the iPhone, but were a bit too bright on the iPhone.
7	The customer did not fully agree on the category hierarchy.

Table 11.2: Discovered bugs

In addition, we intended to implement the admin tool's new feature, suggested by the customer at the end of sprint 3, to be able to read information

from a text file to upload new cases. It was also requested that we update the database with new cases and a new category hierarchy specified by the customer.

Finally, the customer wanted us to show the mobile application to some of his coworkers. These users were in the target user group. In order to demonstrate the app, we had to find a way to distribute it without connecting all the devices to a Mac.

Also, we planned to make significant progress on the report, by writing the remainder about the sprints and conclusion, in addition to starting on the evaluation section.

For sprint 4, we tried to complete milestone MP4, "Caching of data, finalize features."

11.1.1 Sprint goals

The goals for sprint 4 was to polish the app a little by fixing some bugs, and to make headway on the report.

11.1.2 Sprint backlog

The sprint backlog is shown in the table below. It shows all the use case stories, with tasks and estimated effort that was planned for this sprint.

ID	Task	Main responsible	Estimated cost	Real cost
BL2.3	Caching of images	Stian	3	1
BL7.4	Icons and UI elements	Kristoffer	3	4
BL8.3	Upload by textfiles	Kristoffer	2	1
BL8.5	Admintool bugfixing	Kristoffer	5	6
BL10.3	Finalize database contents	Kristoffer	2	2
BL11.1	Final testing	Kristoffer	12	10
BL11.2	Bugfixing	Stian	32	36
BL11.3	Make app distributable	Stian	8	7
R1.12	Sprint 4	Marthe	5	4
R1.13	Conclusion	Marthe	7	6
R1.14	Evaluation	Marthe	4	5
R1	Report	Marthe	20	21
Sum			103	103

Table 11.3: Sprint 4 backlog

11.2 System Burndown Chart

The figure below shows the burndown chart for this sprint. We managed to finish every task in this sprint, and we felt that we had estimated the effort very accurately. We made good progress with the work, and at the end of the sprint we managed to work in accordance with ideal burndown. We work nearly every day during this sprint, as can be seen by the almost-linear line for the actual remaining effort.

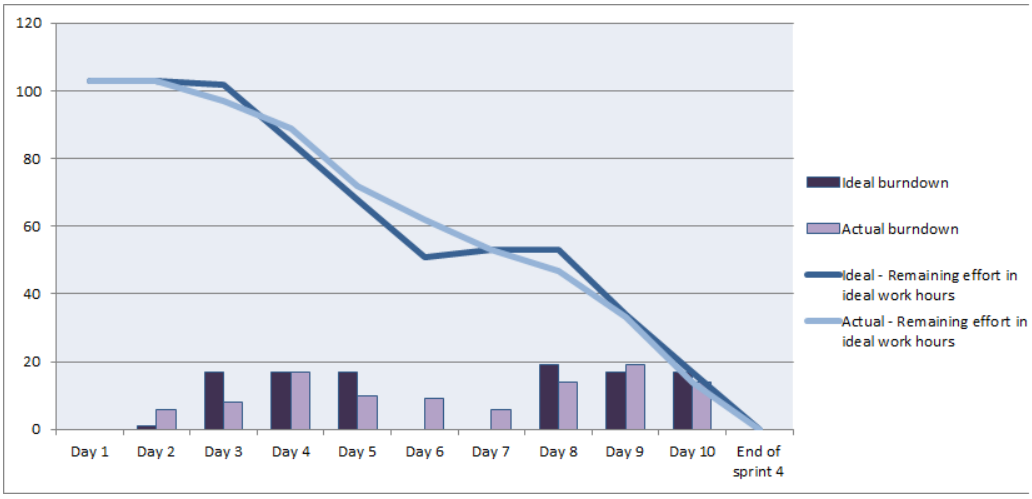

Figure 11.1: Burndown chart for sprint 3

11.3 Implementation

Following is a description of the items that were implemented in this sprint.

11.3.1 Bug fixing

Bug #1 was a result of long loading time when opening many images at once with a slow internet conneciton. When an iOS application does not repsond within 10 seconds or so, it is killed by the operating system. As we saw it, we had two possible solutions for this:

1. Implement some sort of caching or buffering of images.
2. Reduce the number of images for some large cases.

As was concluded at a meeting with the surgeons, any given case should not contain too many images, in order that the case not become clouded by too much information. Option 2 was therefore a very tempting solution, although not particularly elegant. We implemented a very simple form of caching, wherein the most recently opened case always was cached locally, to reduce the loading time and the chance of the application to being killed.

Bug #4 was quite hard to replicate. Apparently, when opening the images of a case for the first time ever (after installing the application, not after restart), the image was zoomed in, without the ability to zoom out. Since the images had a very high resolution, they had to be scaled down somewhat before being displayed. This usually worked as expected, but not the very first time. This was solved by caching the image locally, and then calculating the size and the zoom, rather than doing it on the fly as before.

Bug #7 was reduced by implementing local caching, but was not fully fixed.

Bug #8 was actually not a software bug, but a miscommunication with the customer when entering categories into the database. This was fixed very easily by altering the database content slightly.

The team was not able to fix the remaining bugs in sprint 4 due to lack of time.

Final bug status, after sprint 4:

#	Bug	Status
1	The application crashed sometimes when opening the "images" section of a case with many images.	Partly fixed
2	Video playback did not stop when leaving the video view.	Not fixed
3	The application rotates with only minimal "tilting," making it seem a bit unstable.	Not fixed
4	When opening the images of a case for the first time, the initial zoom was too large, and it was not possible to zoom out.	Fixed
5	After entering full screen when viewing a video, zooming was no longer possible, even when leaving full screen.	Not fixed
6	Selecting "images" on a case with many images, makes the application seem to "hang." In sprint 3, we added a loading animation, which worked very well on the iPhone, but were a bit too bright on the iPad.	Partly fixed
7	The customer did not fully agree with the category hierarchy.	Fixed

Table 11.5: Sprint 4 final bug report

11.3.2 Admin tool

Making the admin tool read from a text file and create cases using that information was implemented simply by creating a thin layer over the admin tool. This layer simply read from a file, converted the information there into commands that the admin tool expected, and ran the admin tool. The format of the input text file must be as follows:

case name
categories
public description
private description
path to media files
visibility of case

Table 11.7: Admin tool input file format

11.3.3 Application distribution

To distribute the mobile application without using a Mac, we used a web service called Test Flight [41], which allowed us to upload a mobile application build to the web, add the UDID's of the devices that should have access, and then download the application from the web using a mobile phone.

11.4 Testing

During sprint 4 we repeated all our previous system tests in order to prepare for the acceptance test and meeting with the customer. Since we also updated our database to the new desired structure, the tests became slightly different. See appendix C.2 for a overview of the data we used during testing.

The following tests were executed on November 8th, 2013.

ID	Test name	Requirements	Result	Comments
T01	Browse categories	FR01	Passed	Updated database caused change
T02	Select a case	FR01	Passed	New case name: "Tumor (Real Case)" New subcategories
T03	Browse images	FR02	Passed*	New case name: "Tumor (Real Case)"
T04	Play video	FR03	Passed	New case name: "Tumor (Real Case)"
T05	Search for case	FR04	Passed	Search: "High grade glioma" New case name: "Tumor (Real Case)"
T06	Log in	FR05, FR09	Passed	-
T07	Add new case	FR10, FR11	Passed	-
T08	Add new user	FR09,FR10, FR11	Passed	-

Table 11.8: Completed tests for sprint 4

* If on a slow Internet connection, the application occasionally crashed (killed by the operating system) when browsing the images belonging to cases with very high quantities of images.

11.5 Occurring Risks

The problems that occured during sprint 4 are summarized in the table below, along with the actions taken.

ID	Risk factor	Explanation	Actions taken
R1	Sick group member	Marthe and Kristoffer were sick for a couple of days each	The group members worked a little in the evening to complete the assigned tasks.
R12	Absent group member	Marthe was away for one week	Marthe informed the group about the absence with good time before the period in question, so the group was able to plan with this in mind. The agreement was that Marthe was going to work on the report while she was away.
-	Noise from construction work	We had our office in the IT building, and there was a lot of noise from construction work at that time.	Sometimes we had to leave the office, and work in the cafeteria or other locations where there was sufficient space. We could not move the iMac, as we assuredwith the customer that it would at all times be locked in the office. When working with the application, we had to work in the office despite the noise from the construction.

Table 11.10: Occurring risks in sprint 4

11.6 Customer Feedback

The feedback from the customer after sprint 4 was very positive. He was very pleased with the number features implemented, and their appearance. Understandably, he was not thrilled that the application was not bug-free, so we promised to look at them if we could make the time. Other than that, he thought the application was a good foundation to build on, and even asked if we wanted to continue with the project when writing our masters.

11.7 Sprint Evaluation

The milestone MP4 was not completed within time. We did not prioritize caching of data, as, in accordance with the customer, we had other higher-priority items. Some minor bugs were also not fixed due to lack of time. The admin tool was completed during sprint 4, which should be very useful for the customer. It allows him to upload new cases without any knowledge of databases or SQL.

All in all we were satisfied with the progress of sprint 4. The admin tool turned out quite well, and we had a very productive meeting with the customer and his co-workers. The application at its current state was deemed finished, and most requirements were complete.

11.7.1 What went well during this sprint

In advance of the meeting with the surgeons, wherein we were to present the application and do a demonstration, we discovered quite a few of bugs in cooperation with the customer. This was late the night before the presentation, so we didn't have much time to fix or hide them. Despite the time constraints, we were able to fix a many of the major bugs in the time before the presentation.

We were also pleased at the meeting with the surgeons. It was very satisfying for us to see that our work was important to some people, and that

they were very interested. In addition, it was nice to get additional feedback from new perspectives, both on what we had so far, but also on potential future features. The surgeons discussed amongst themselves what features they thought would be useful, which was interesting for us to listen in on, even if we didn't have time to implement anything further.

11.7.2 What could be improved for the next sprint

Sprint 4 was the last sprint, but we discovered several things that could have been improved for potential future sprints. These are discussed here.

A lot of time in sprint 4 was used to prepare for the meeting with the surgeons, both on finding a way to distribute the application, and on planning what to do and say. Also, a large chunk of time was spent coordinating the meeting with the customer. The meeting took a large part of one day. When estimating tasks for sprint 4, we did not plan to use this much time on the meeting and its preparation. That was clearly a mistake.

All the time spent on the meeting resulted in fewer bugs being fixed than we had planned for. We also would have liked to have made greater progress on our report. This would likely result in some overtime in the last couple of weeks of the project.

Part III

Results, Evaluation and Conclusion

12 Results

This section gives a description of the final system architecture, the results of the project, and futher work.

12.1 Final System Architecture

The architecture of our solution was designed at the beginning of the process and has been followed closely throughout the implementation. However, there have been some changes.

Since JavaScript/Titanium didn't really support classes, the class diagram in figure 6.3 was merely used as a starting point. Using JavaScript, we were, however, able to simulate classes by using functions with attributes and with their own functions. This way, we were able to implement a class-like architecture for the Category, Case, and MediaFile classes in the class diagram. Some changes were also made to the attributes of the three classes to reflect the changes in the database. See below.

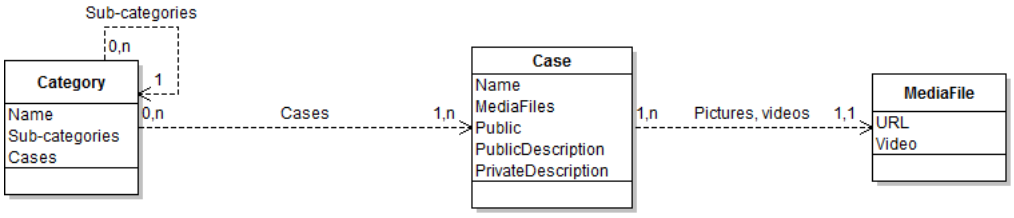

Figure 12.1: Modified class diagram

The database architecture was also altered slightly. The description field of MediaFile was moved to Case, and split into two new fields, privateDescription and publicDescription. This was done to better accomodate what the customer wanted. A new field was also added to MediaFile, called video. This is a simple boolean (true/false) field that explains if the given MediaFile is a video or not. If video is false, the MediaFile must be an image. The video field was added to simplify the support for multiple video formats.

With this solution, the application itslef doesn't need to be changed at all to support new video formats, as long as the underlying framework (Titanium and iOS) supports them. Other than these minor changes, the database was implemented as originally designed.

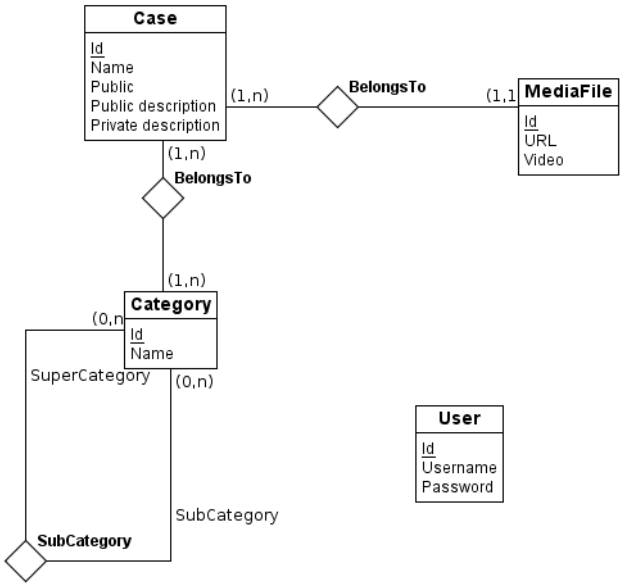

Figure 12.2: Modified ER diagram

The overall architecture, as seen in figure 6.2, was followed without any modifications, and will therefore not be discussed in this section. See chapter 6 for details.

As for the implementation of GUI with MVC, it showed to be a little harder to do with Titanium and JavaScript, since no classes exists. In addition, the creation of GUI elements in Titanium is a little awkward. To try to still follow the MVC pattern, we decided to create own functions to display categories and cases with. These functions in practice make out the View part of MVC. We also created specific Controller functions to respond to any user interaction with the GUI. The model part, however, were a little more challenging. To keep the state of the browsing we created a global vari-

able to hold this information. In a more object-oriented language, we would have used a dedicated class for this, but this did not feel very natural with Titanium.

12.2 Project Results

In this section, the system, as it was at the end of the project, will be described in depth.

The system consists of several parts, where the main part is the mobile application. In addition, server software was created, and also a simple command-line based admin tool.

12.2.1 The application

The application itslef is divided into three tabs, which represents the application's three main functions:

1. Browsing, where the user can browse categories and cases
2. Searching, where the user can search for specific cases using categories as keywords
3. Login, where the user can enter login information to log in, to be able to view private cases, and private descriptions.

The "Browse" tab is the application's "main" tab, and where users probably will spend the most time. The navigation is structured as a hierarchy, where selecting a category will display all its sub-categories. Selecting a category which has no sub-categories, will display all cases that belongs to all the categories selected on the current "path". The user can also select "Show all" at any point in a "path" to view all cases belonging to the categories on the "path" so far. A "path" is a set of the categories which has been selected by the user. For instance, if the user selects "Tumor", and then "Primary tumor", the path will look like this: Tumor -> Primary tumor. Selecting "Show all" with

this path will display only cases which belong to both “Tumor” and “Primary tumor”.

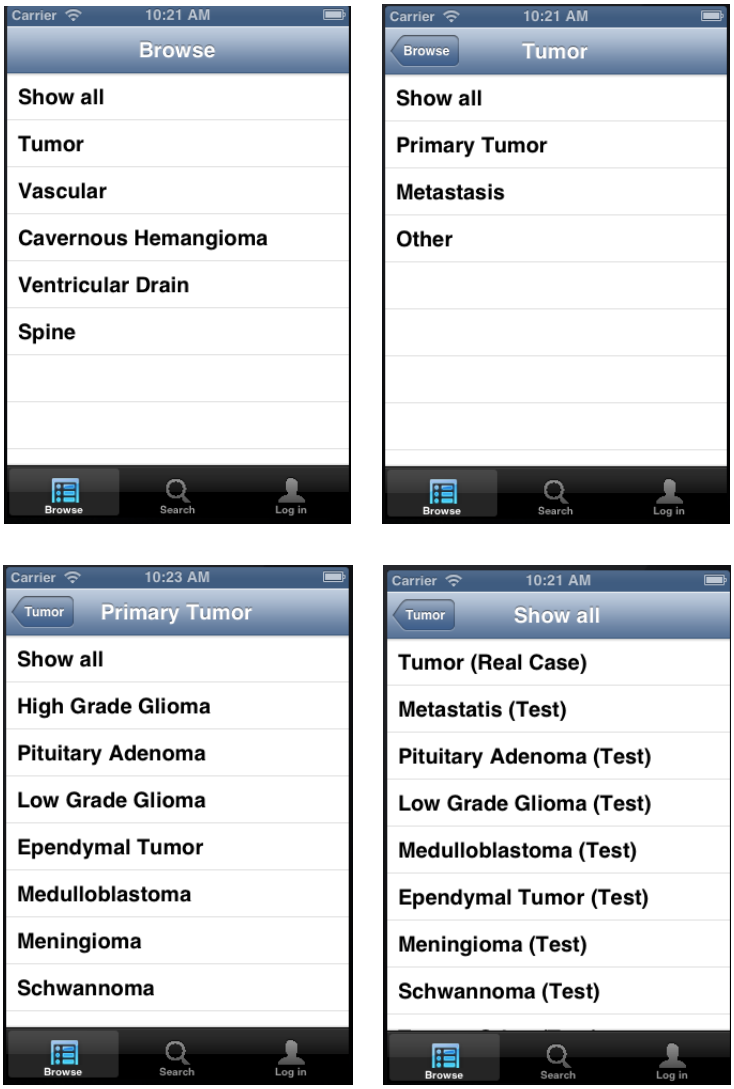

Figure 12.3: Browsing for cases

The “Search” tab is another alternative to find cases. The first window consists simply of a help label, explaining the allowed format for searching, a text area where the user can enter the search keywords, and a search button. Completing the search, by either clicking the search button or by tapping

search on the keyboard (which is displayed when selecing the text area), the user is taken to a search results window. This window will display cases that matches the search keywords. Selecting a case will take the user to the same case view as when browsing.

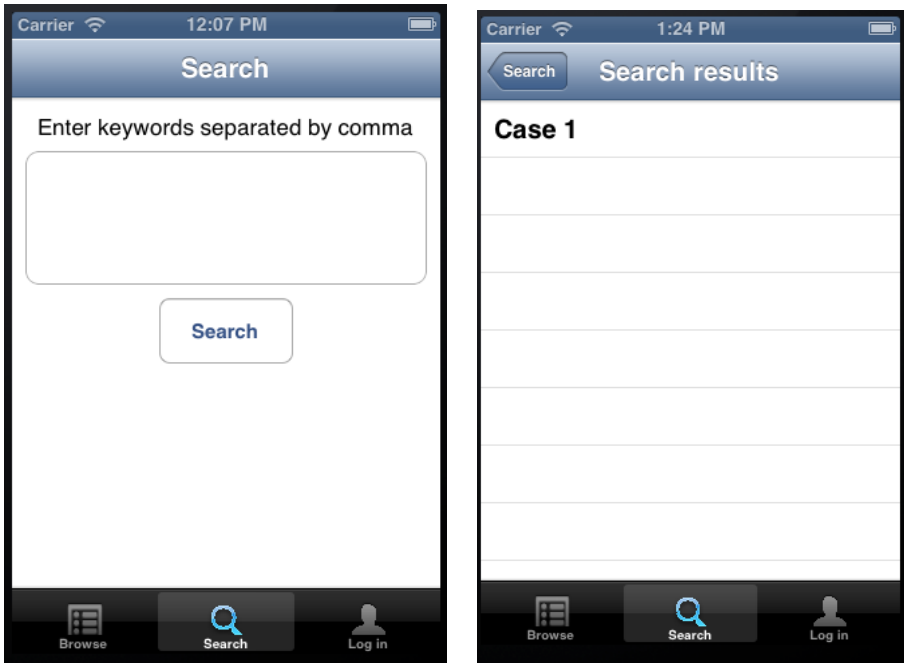

Figure 12.4: Searching for cases

The "Log in" tab is very simple, and only contains fields for supplying username and password. After logging in, the tab is simply offering the option to log out. When a user has logged in, it is able to see both public and private cases, and both public and private descriptions for all cases.

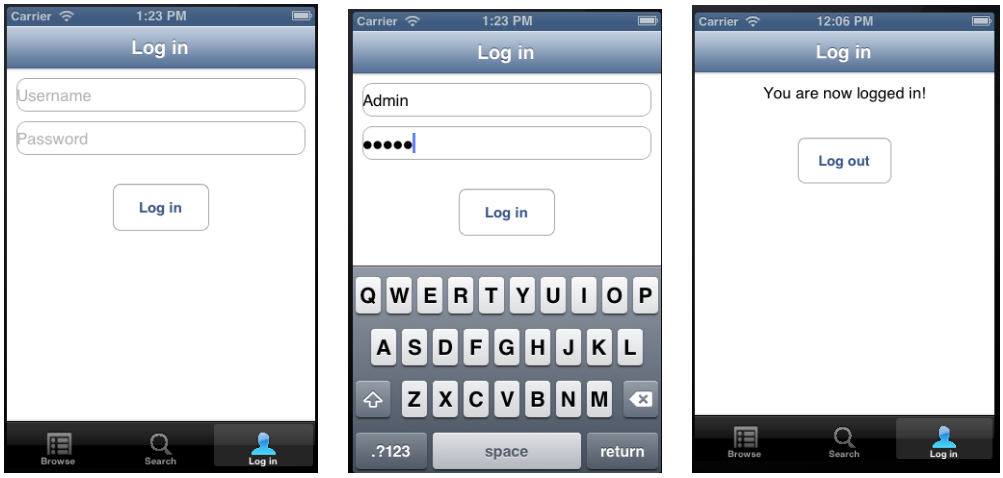

Figure 12.5: Logging in

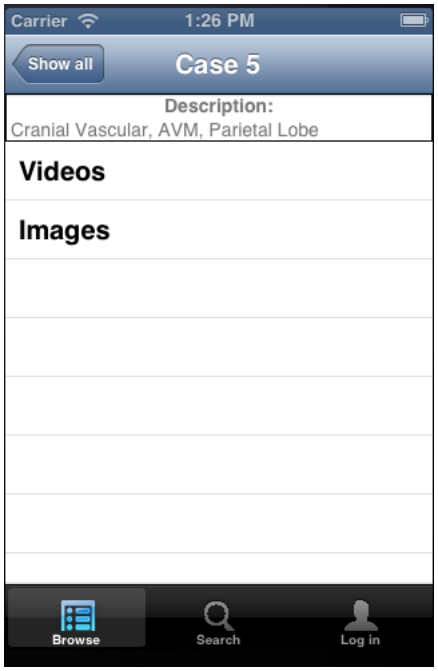

Figure 12.6: Case view, "Case 5" has been selected

Viewing of cases is exactly the same when selecting a case when browsing, and selecting a case from the search results. On the top, the descriptions will

be displayed. If the user is logged in, both public and private description is shown, if not, only public. Below, up to two choices are available, videos and images. These are only shown if they contain anything, for example, if the selected case only contains images, the video choice is not displayed, and vice verca.

Selecting either "Videos" or "Images" will allow the user to browse the media files of the selected type, belonging to the selected case. Navigating between different videos or images is done by "swiping" left or right.

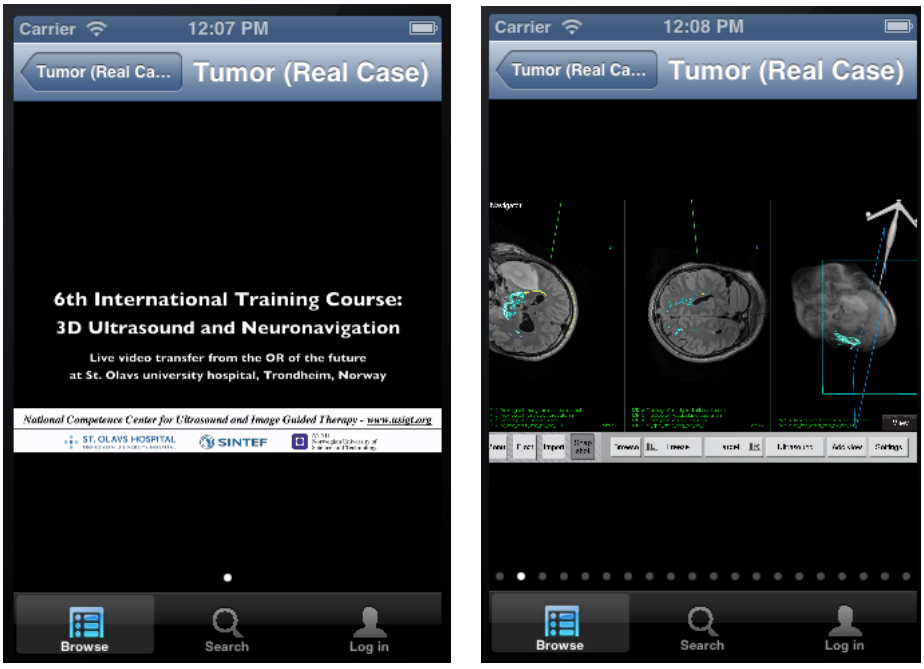

Figure 12.7: Displaying video (left) and image (right)

12.2.2 The server

The server software contains of two different PHP scripts which runs ontop of pre-existing software. The purpose of the two PHP scripts is simply to load all categories and cases from the database, and to verify if user login information is corrrect. These runs ontop of Apache with PHP support. Apache is also

used to make the media files available on the Internet. The server also run a MySQL instance, which is the database used to store the cases, categories, media file references, and user information.

12.2.3 The admin tool

The admin tool was created to simplify the process of uploading new cases and registering users, without the need for any database or SQL knowledge. It is purely command-line based, and can be used in two ways:

1. The program can be "started", and used in an interactive way where you select choices from a menu, and then follows the instructions, step-by-step to create a case and upload data, or create a new user.

2. The program can also be run with a file path as argument. If this file follows a specified format, it can be used to upload a new case, without the need of navigating menus and such.

```
Mainmenu
1: Create new case
2: Create new category
3: Create new user
4: Exit program
Select your choice:
3
Insert the username for the new user:
test
Username: test

Is this correct?
1: Yes
2: No
1
SELECT id from user WHERE username ="test"
Insert the password for user for the user: test
test
The password is: test

Is this correct?
1: Yes
2: No
1
INSERT INTO user (username, password) VALUES ("test","test")
User test has been added to the database
Exiting to main menu
```

Figure 12.8: Registering new user using the admin tool (debug mode on and showing SQL statements)

12.3 Further Work

The application project for this group was never supposed to be a standalone application, but a foundation for a more entertaining application. The goal was to create an application which were able to read data from a server, and display it in an intuitive way. Therefore it is obviously a lot of work left to do with the application.

12.3.1 Unfinished requirements

Not all of the requirements that was established with the customer was finished, we created more requirements than we thought would be manageabe, so this was not a surprise. The reason why we did that was that we wanted a more complete picture of what the customer wanted, so that we could atleast lay the foundation for it, even if not able to implement it all.

The requirements regarding local storage/caching, FR07 "Save images to phone" and FR08 "Save video to phone" was not implemented at all. These requirements were downgraded in priority by the customer, when it became clear that we were not able to fulfill all requirements. This was done because it doesn't really give anything new to the application, but just makes it usable without an Internet connection. A function to download files were, however, implemented as a foundation for these, so the requriements should be quite simple for further development.

Requirements FR06 "Registration" and FR10 "Upload data to server" were fulfilled in accordance with the customer, but no GUI were created for them. This was to save time, and was actually the customer's idea, since he thought it would be sufficient with a command line based tool to do these things, for now.

The final requirements, FR11 "Quiz-Game" and FR12 "'Like' function to images", were not something the group and the customer expected to be completed during this project, and they were not. These are two of the major requirements for further development.

An overview of the total progress of the requirements can be seen in the table below.

ID	Functional Requirement	Status	Priority
FR01	Browse cases	Complete	Critical
FR02	Show images	Complete	Critical
FR03	Play video	Complete	High
FR04	Search for cases	Complete	High
FR05	Log in	Complete	High
FR06	Registration	Done, but no GUI	Low
FR07	Save images to phone	Not complete	Low
FR08	Save videos to phone	Not complete	Low
FR09	Separation of data	Complete	High
FR10	Upload data to server	Done, but no GUI	High
FR11	Quiz-Game	Not complete	Low
FR12	"Like" function to images	Not complete	Low

Table 12.1: Status of the functional requirements

12.3.2 Other functionality

In addition to the functionality that we were not finished with, there is many other features that could be developed to improve the system:

- Add text to an image. Users could add and save notes to an image and store it locally, writing down their own thoughts and descriptions. This could be helpful for own learning.
- For every image there could be an indication showing the location in the brain that it was taken.
- A list showing all the images belonging to a case in miniature (thumbnails).
- The possibility to publish articles in the application so users could learn more about ultrasound. It could be used to give advise about how to use ultrasound imaging during an operation.

- It could be possible to refer to or give links to other sites with more detailed information about different topics relating to ultrasound navigation.

13 Evaluation

In this section the evaluation of the group, the project and the technology are discussed.

13.1 Process and Group Evaluation

13.1.1 Communication

All the members in the group and the customer spoke Norwegian, so there was never any difficulty understanding each other. Our supervisor spoke English, but we did not feel like there was a language barrier.

We had a good group dynamic throughout the project. We did not have any major disagreements, and in the minor ones we had, we confronted each other and solved them quickly. We had agreed to communicate through certain channels, and we managed to be available almost all the time.

The customer was available by email, and we could always reach him that way. We met him approximately every other week, but occasionally he was unavailable because of travel. He always informed us in advance of his trips, so we could plan with that in mind. The feedback we received from the customer after meetings, and especially after the demonstrations of our product, was very beneficial and helped us stay on the right track.

The main communication channels with our supervisor were email and our regular meetings. We prepared for these meetings with questions, a status on the feedback from the customer, and the most recent version of our report and product. The supervisor was also occasionally traveling, but he was available via email.

13.1.2 Actual effort

As we were only three people working on a project that, to us, seemed intended for 5-6 people, we were somewhat overwhelmed at the start of the project. Due to a mis-estimation our (and others') group ended up being

about half the intended size. After a discussion with the course organizers, we accepted the fact that we would have to complete the course with only three members and tried to work around it. Whether or not the customer was informed of the smaller group size is unknown, but an unfortunate consequence of this was that we simply had too few "man-hours" to complete all the features the customer wanted in the project. We not only had fewer people to develop the application, but also fewer people to write the report, which is quite a substantial part of the project and should be allotted significant time. Despite the smaller workforce, we got off to a good start as all members were motivated to do a good job, shared similar goals, and agreed on the amount of work to be put into the project.

We planned to use a total of 1050 person-hours during this project, spead out over 14 weeks and three people. We tracked our time use on each part of the project, in addition to our work on the product backlog.

Task	Estimated time (in hours)	Actual time spent (in hours)
Planning	120	96
Meetings	60	~30
Lectures	30	~37
Self study	60	~50+
Report	390	451
Development	390	427
Sum	1050	1091

Table 13.1: Actual time used

As shown, we finished the project with 1091 person-hours used, just a bit above the estimated time, with the biggest difference time estimated versus used on the report. The following figure shows our activity on Github, which isn't a one-to-one relationship with time spent, but does give an idea of how the intensity of our work progressed throughout the project. The graph shows the amount of commits to master distributed over time.

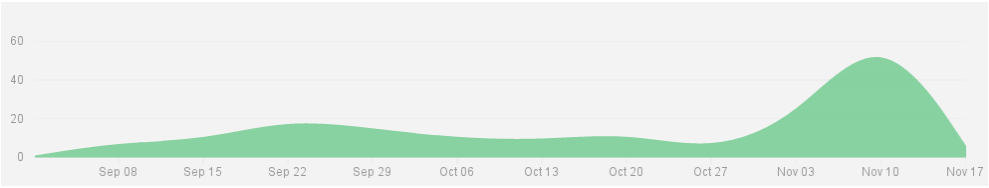

Figure 13.1: Github activity

During the late parts of the project we had to work overtime to get all our tasks done; this was not unexpected and we feel that our estimation was reasonably good.

13.1.3 Roles

We specified, in the planning chapter, different roles for the the three member of the group, with different responsabilities and tasks. We quickly realized that with such a small group, the tasks and responsibilities would be very floating and vary from day to day. In light of this, we tried to keep our role assignments as a general guideline, but we decided to not take them too literally and just communicate frequently about what we were doing and who was responsible for what. In retrospect, we believe that this caused us to be more efficient and spend less time on delegating work and coordinating our efforts.

13.1.4 Risks

In the risk evaluation, we did not predict that we would use three weeks on deciding the platform, nor that we wouldbe driven away from our office due to construction noise. We had to leave the iMac in the office and use TeamViewer to communicate with it when we sat in another place. There were more sicknesses among the group members than predicted, but we did not lose much time due to that, since we had planned with a buffer, and the group members could work other days and in the evenings. The planned absence of group members did not cause problems, since it was informed well

in advance so we could plan with that in mind. Most of the action consisted of the absent group member completing the work before he or she returned.

13.1.5 Prestudy

Given the terms of the customer, mainly the prerequisite of multiplatform possibilities and the preference towards Titanium as the development tool, it was difficult to find a similar or usable code for our project. If we had the opportunity to develop for a single platform, it would have been easier to find reusable code.

To help us choose the development platform, the customer brought in an external person, Erlend Dahl. He had previously developed an iOS application for the customer, and had some experience with a couple of our alternative platforms. He thought that both Titanium and Xamarin would work, but did not have that much experience with Titanium, and therefore could not recommend one over the other, but highly recommended Xamarin, which he thought was very good. Nevertheless, we chose Titanium, since we were not able to give enough "pros" for Xamarin to justify the license costs.

13.1.6 Scrum

The decision to use the scrum method was determined to be a good one. Some from our group had experience with scrum from other projects, so the particular scrum artifacts were known. The process of learning scrum went quite well, but we did not follow protocol as strictly as we first predicted. We kept the meetings short, and the topics were the same as described in daily stand up meetings. The sprint planning meetings were held the same day as the sprint evaluation, because the customer had more time available when our sprint started.

As the project evolved, we were pleased with the choice of the lifecycle. We were unaware that sprint 2 would not contain any programming. Furthermore, the time required to obtaint he necessary hardware was also unknown.

The time for deciding the development platform was also not predicted to take three weeks. Scrum did an excellent job at allowing for the uncertainties and changes we experienced.

We maintained the burndown chart for each sprint's backlog in a common document, as we had a problem with Rally. We logged the time used for the tasks in the backlog, and continuously monitored how the progress was going.

We used pair programming approximately half of the time we programmed. It was helpful, and we learned more from each other than we would have otherwise. It came quite naturally, and it was fairly efficient when one wrote code and the other watched.

13.1.7 Requirements

During the planning phase it became obvious that the customer had created the project with more than three people over three months in mind. The customer was, however, very accommodating and realistic when confronted with the small group size, and we agreed upon feasible goals. Since the deadline for registering for the course was well into the project, our supervisor told us that we might get one additional group member if we were a lucky. We therefore decided to keep a couple of "less practical" requirements for a group of three, like FR11, FR12, and FR13, but we gave them a very low priority. It was also useful for us to have the extra future requirements when creating the general foundation for the app, as previously discussed. It enabled us to see what should be supported without too many changes. We therefore do not see it as a failure that these requirements were not completed.

13.1.8 Architechture

When the customer came to us late in sprint 4 and wanted some changes in the category structure, we were relieved that we created the database to as flexible as it was. The changes required no recompilation, and they were completed with only a couple of SQL operations in a very short time. In the

end, we felt very successful with our data structure. With our design, the customer could add as many structures and sub-structures of categories and cases as he wants.

The few small changes we made with our architecture, as discussed in 12.1 (Final System Architecture) were to acommodate more specific requirements from the customer, and to simplify some implementation details. All in all, we followed the initial architecture very closely, and were pleased with it. It also allows the web server and the database server to be set up on the same computer or separated on several computers as the workload changes. This gives the system very good scalability to allow many users to connect at the same time.

When developing the graphical user interface (GUI) for the mobile application, we decided to follow the model view controller (MVC) software architecture pattern, as discussed in chapter 6.5. Unfortunately, this was not a simple task with Titanium and JavaScript. We ended up with a variation of MVC, as discussed in 12.1 Final System Architecture, where the three components were separated by functions, but not by classes or files. We were not entirely happy with this, but it was the best we were able to do. The consequence of this is that the code is perhaps not as readable as it could have been.

13.1.9 Testing

As mentioned in Chapter 7 (Testing), our limited time and resources made us lessen the amount of pre-written tests and focus more on unit- and integration testing as an integrated part of the programming. This worked well for us and saved us several hours of planning and testing throughout the project. The number of bugs at any time during the implementation was always manageble and the final polish and debugging process occured with less difficulty then expected. We credit this to the thourough unit- and integration testing throughout the development.

We decided against having a "normal" useability test. The technicians and surgeons in particular were very busy, and it was difficult to justify using their time for this. In addition, the application is not a "final product," but only an application which provides base functionality to be built upon. To compensate for this we had good contact with the customer throughout the whole project, and we asked for frequency feedback. During the meeting with the surgeons, everybody had the opportunity to test the app, and we obtained valuable feedback from that as well.

13.2 Project Evaluation

13.2.1 Evaluation criteria

As mentioned in the introduction, the goal of the project was to create an application for our customer that can be used for proof-of-concept and further development. The evaluation criteria was therefore primarily how satisfied the customer was with our end product and how well it fit and satisfied the requirements identified. The possibility for implementation of additional features in the future and a general expansion of the application was also an important criterion. In order for the application to function as a proof-of-concept, we were certain that it would have to prove useful for the intended users when we were done with the prototype. In order to test this, we would have to meet with some of the potential users.

13.2.2 Final meeting with the customers

The meeting took place at St. Olav's Hospital on the 8th of November. Our meeting with the surgeons was the first part of an internal weekly meeting for their team. Present at the meeting were 2 surgeons, 3 technicians, the customer, and the three group members. The meeting's primary function was to obtain feedback on the project from perspectives other than that of the customer. We wanted feedback both on the current application that was

demonstrated, as well as potential uses they identified for the app, and lastly, potential future features.

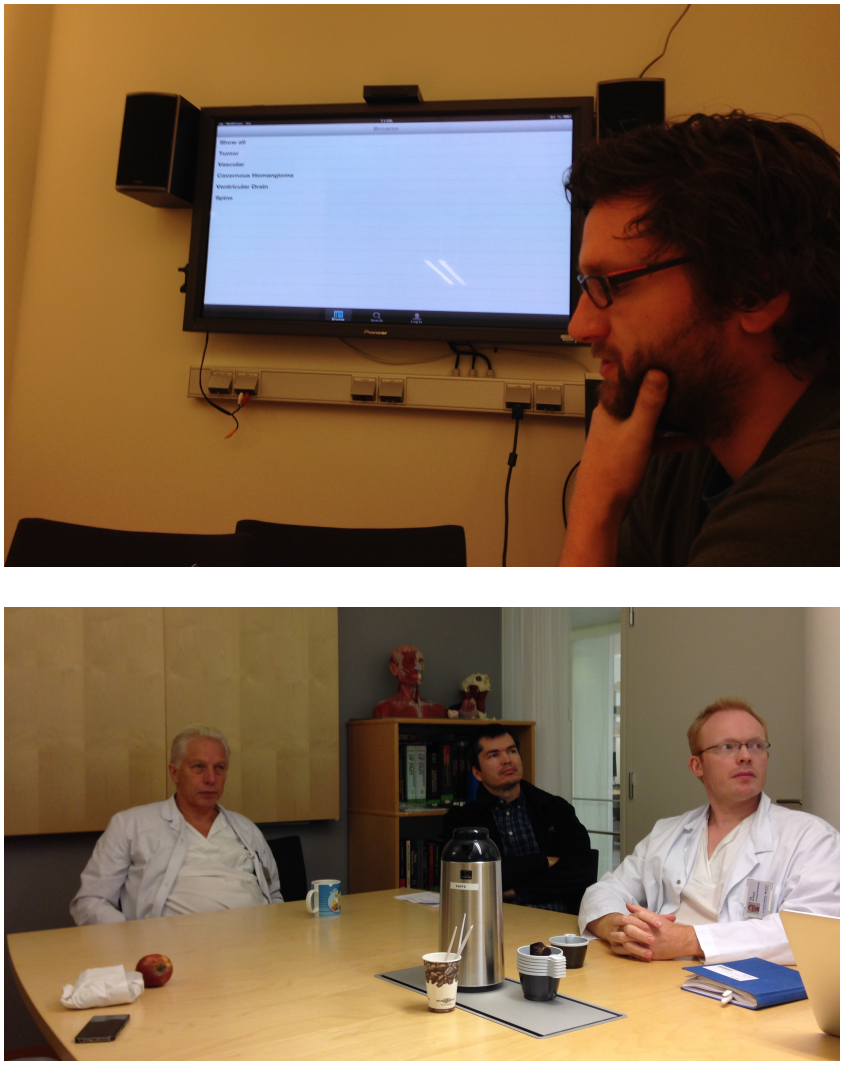

Figure 13.2: The final meeting with the customer and his co-workers

The meeting was divided into two parts. First, the application was demonstrated on a large screen. The demo consisted of showing all the features, and passing an iPhone, with the app installed, around the room for all the participants to try. The demonstration started with regular browsing through

categories, and concluded in a case where both video (streamed directly from a server) and images from an surgical procedure were shown. Next, we logged in to the application to demonstrate the differences between the registered and unregistered user experiences. These differences included additional metadata for all cases, as well as some complete cases that are only available to registered users. Finally, the search function was briefly demonstrated, by searching for a keyword, "tumor," and then browsing through the cases that matched the keyword.

The second part of our meeting with the suregons was a discussion, wherein the surgeons and technicians discussed their interpretations of application that was demonstrated, and that they had been able to test themselves on the iPhone that was passed around the room. They suggested with a lot of possible additional features that they thought would be useful.

During the meeting, the surgeons found one bug, wherein the application crashed when displaying the images of a case with many images. Other than this, they had no negative comments on the current state of the application, and they seemed pleased with it. They thought that the application, in its current state, could already be useful. They thought that the basic functionality provided by the application was very good.

The meeting attendants had many suggestions for additional features, as summarized in the table below.

Feature	Reason
Metadata for each case image	If the image documents something in particular, it should be possible to point it out to the user. Information about where on the body the image is taken from could also be supplied if not obvious.
Personal notes on images	Help with self-learning by allowing the users to write down their thoughts.
A figure of a human body with the location of the images highlighted	Show the user where the image (and perhaps also video) is taken from on the body.
A view of thumbnails for all images belonging to a case	Allow the user to select the image he/she is interested in, without having to "swipe" through all other images.
Possibility to view published articles on the application	Let the user find useful information in convenient and easy way.
Possibility to add links to web sites in descriptions etc.	Let the user find other useful information in a convenient and easy way.
News feed of relevant news	Let the user get up to speed on the latest available information in a convenient and easy way.

Table 13.3: Feature suggestions

13.2.3 Evaluation of the project

The project evaluation is largely based on feedback from the customer, and from the surgeons that we presented the application for, and how their feedback compared to the evaluation criteria.

The application itself was not complete when presented to the customer and his co-workers, and this is obviously something that should have reme-

died. The bugs, however, were not in the foundation part of the application, only in the presentation part, so they were not very critical.

At the presentation, we received positive feedback on the features implemented so far, and they were impressed by what we had been able to do with limited time, as well as by with the possibilities that the foundation provided.

As a proof-of-concept, the application worked very well. We were able to demonstrate many features, like browsing categories, viewing case images and video, searching for cases, and separation of data for registered and non-registered users. The test users decided that the application at its current level (minus the bugs), would be very useful without further development, and they inquired both how possible it would be to publish the application on the App Store, and how the process of adding new cases worked. This was positive feedback, and we ultimately concluded that the customer was pleased with the product.

We proceeded through the implementation process bearing in mind the fact that the application would be used as a foundation for future development. Neither the customer nor the end users have any insight into this fact, but we think that we did a good job of making our code as flexible as possible to allow for new features. The methods to retrieve data internally within the app is also very straightforward.

Milestone MP5, "Product delivery" would have been met at its deadline if the customer had not been traveling on that date. The milestone was completed shortly after the customer returned from his trip.

The final two milestones, MR3, "Complete and print final report," and MR4, "Final project presentation," were met at their set deadlines as planned.

13.3 Technology Evaluation

In this section, some of the technologies used during the project are discussed. The technologies evaluated are those that actually had an impact on the project work, both positive and negative. All technologies were discussed in

the tools and technology section, so the others are not evaluated here.

13.3.1 Apple, iOS and Mac

In the early phase of the project we spent a lot of time discussing which platforms the application should support, and which frameworks would fit the project. The customer was adamant that iOS should be the main platform, but the team did not have acess to a Mac. Since a Mac is required to develop and test an application for iOS, a solution had to be worked out.

Within a couple of weeks, the customer was able to loan us an iMac. This enabled us to target the iOS platform, but only one person was able to write/test code at a time. This increased the development time by two thirds, and was a restriction by Apple that felt very unecessary. In addition, to run the application on an actual device an Apple developer license was required. The customer already had one of these, but it was also being used by another developer at the time. The fact that it was already in use limited us to quite an extent. We were dependent on receiving a private encryption key from the other developer, which took almost a month. This limitation delayed testing on an actual device.

13.3.2 Titanium and JavaScript

At first glance, Titanium with JavaScript seemed like an ideal tool to develop a cross-platform application for mobile devices, since it compiles down to native code, and since almost all of the GUI code can be used for multiple platforms.

Ultimately, they both showed themselves to be a bit of a pain to work with. First of all, all the team members were very unfamiliar with JavaScript. This was a major point, and we spent quite some time learning and getting familiar with JavaScript before being productive. In addition, JavaScript does not have direct support for classes, and even though its name is very similar to Java, the two languages do not look very much alike at all. We were also

unpleasantly surprised with Titanium's development environment. Titanium studio is a free IDE built on top of Eclipse, which allows the developer to test the application on an iOS simulator. It has, however, a couple of flaws. First and most important is its total lack of useful debugging support. When running an app on the iOS simulator through Titanium studio, the error codes that pop up do not make sense at all, and the line numbers are typically off. In addition, Auto-completion when writing code is almost non-existant, and only work in the simplest cases. These major shortcomings resulted in extended development time and a much longer testing and debugging phase.

13.3.3 Eclipse and Java

After developing with Titanium and JavaScript for several weeks, developing the admin tool in Java with ordinary Eclipse was a actually quite a pleasure. It made us appreciate the available auto-completion of code and the accurate, realiable error messages and exceptions.

14 Conclusion

The team was pleased with the product. The application presented very nicely, and received good feedback from both the customer and the surgeons. Even though we were not able to implement all the features that the customer initially requested, we came quite far on the prototype.

The server software that we implemented worked exceptionally well. The admin tool that was created to simplify upload of new data and registering of users was only command line based, but this was at the customter's own suggestion, and we were happy to oblige.

Finally, we were also pleased with the team work, the communication with the customer, and our supervisor. The team met at least three days a week to work for several hours, and we usually made good progress. Even if somebody was not able to meet, we divided tasks well to keep all team members busy doing something useful. The team also had meetings with the supervisor nearly weekly, and received a lot of useful feedback from him, both on the product and on the report. The meetings with the customer were usually at the end of each sprint, or when either of us felt the need for a meeting. These meetings usually consisted of showing the customer the state of the product, and receiving feedback on it. In addition, the plan for the next sprint was usually revised to accomodate the customer's wishes. These meetings were very useful, both to ascertain we were on the right track and to determine whether the customer's priority of new features had changed.

15 Definitions, Acronyms and Abbreviations

- Unregistered users: A person who uses the system on the client side with a limited access to data.
- Registered users: A person who uses the system on the client side with access to all data.
- Administrator: A person who will have full access privileges, in the database and the system as well.
- Use Case: Technique that specifies the interaction between the user and the system.
- Case: Usually refers to a surgery performed by the surgeons and staff connected to our customer. From each surgery, or case, data is collected in the form of videos, images and textual information.
- App: Short for application.
- SSH: Secure shell.
- DMS: Diagnostic medical sonography or ultrasound.
- MRI or MR: Magnetic resonance imaging.
- NTNU: Norwegian University of Science and Technology.
- MVC: Model View Controller.
- GUI: Graphical User Interface.

References

[1] SINTEF http://www.sintef.no/home/About-us/ 1.1

[2] The National Competence Center for Ultrasound and Image Guided Therapy http://www.usigt.org/ 1.1

[3] Stakeholder definition http://www.businessdictionary.com/definition/stakeholder.html 1.2

[4] Xamarin http://xamarin.com 2.1.2

[5] Appcelerator http://www.appcelerator.com 2.1.2

[6] LaTeX-project http://www.latex-project.org 2.5.1

[7] Lyx http://www.lyx.org 2.5.1

[8] Google docs http://docs.google.com 2.5.1

[9] Google drive http://drive.google.com 2.5.1

[10] Violet UML editor http://alexdp.free.fr/violetumleditor/page.php 2.5.1

[11] VisualER http://www.idi.ntnu.no/emner/tdt4145/programvare/VisualER/ 2.5.1

[12] GanttProject http://www.ganttproject.biz/ 2.5.1

[13] Balsamiq http://balsamiq.com/products/mockups/ 2.5.1

[14] Facebook http://www.facebook.com 2.5.2

[15] TeamViewer http://www.teamviewer.com 2.5.2

[16] Wikipedia - TeamViewer image http://en.wikipedia.org/wiki/TeamViewer (document), 2.4

[17] Rally http://rallydev.com 2.5.2

[18] Eclipse http://www.eclipse.org 2.5.3

[19] Wikipedia - JavaScript http://en.wikipedia.org/wiki/JavaScript 2.5.3

[20] PHP http://php.net 2.5.3

[21] JSON http://www.json.org 2.5.3

[22] MySQL http://www.mysql.com 2.5.3

[23] Apache http://www.apache.com 2.5.3

[24] Wikipedia - Java http://en.wikipedia.org/wiki/Java_(programming_language) 2.5.3

[25] Git-scm http://Git-scm.com 2.5.4

[26] Github http://www.Github.com 2.5.4

[27] Subversion http://subversion.apache.org/ 2.5.4

[28] SVN versus Git https://git.wiki.kernel.org/index.php/GitSvnComparison 2.5.4

[29] Wikipedia - MRI http://en.wikipedia.org/wiki/Magnetic_resonance_imaging (document), 3.1, 3.1

[30] MRI image http://www.healthcare.philips.com/main/products/mri/systems/achieva3t/ (document), 3.1

[31] Wikipedia - DMS http://en.wikipedia.org/wiki/Medical_ultrasonography 3.1

[32] EUS - Diagnostic and Interventional Endoscopic Ultrasound https://itunes.apple.com/us/app/eus-diagnostic-interventional/id527085806?mt=8 (document), 3.3.1, 3.3

[33] Helsinki Microneurosurgery Basics and Tricks https://itunes.apple.com/us/app/helsinki-microneurosurgery/id506365864?mt=8 (document), 3.3.2, 3.4

[34] Waterfall Model design http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm (document), 4.1, 4.1

[35] Scrum https://www.scrum.org/ 4.2

[36] Scrum image http://epf.eclipse.org/wikis/scrum/Scrum/guidances/supportingmaterials/scrum_overview_610E45C2.html (document), 4.2

[37] Use Case Defenition http://www.techopedia.com/definition/25813/use-case 5.3

[38] Kruchten, Philippe (1995, November) http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf 6.3

[39] Microsoft Developer Network http://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx9 7.1

[40] Wikipedia - Acceptance test http://en.wikipedia.org/wiki/Acceptance_testing 7.4

[41] Test Flight http://testflightapp.com 11.3.3

[42] Wikipedia - SSH http://en.wikipedia.org/wiki/Secure_Shell B

[43] MySQL Workbench http://www.mysql.com/products/workbench/ B.5

Part IV

Appendix

A USNeuroNav

USNeuroNav is a mobile application which runs on iOS. Its purpose is to display images and videos from ultrasound surgery, together with relevant information, in a simple and intuitive way. The application is divided into three separate "tabs," which provide the three main functions of the application. These tabs can be switched between by selecting their respective titles at the bottom of the application.

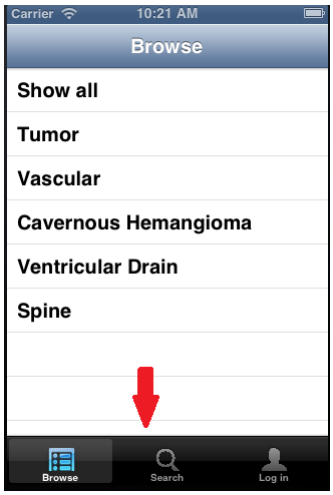

Figure A.1: Tabs in USNeuroNav

A.1 Browsing Through Categories

The first tab that the user is confronted with when starting the application is the browsing tab. Here, the user is presented several categories to choose between. To select a category, simply tap it with a finger. Also, at the top of the list of categories is another option, called "Show all." Selecting this option will display all cases.

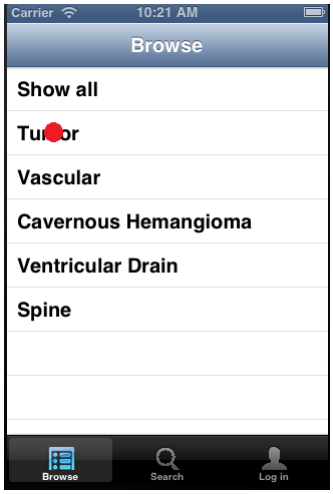

Figure A.2: Selecting a category

After selecting a category, sub-categories of that category might be displayed. If that is the case, these can be selected as with the previous view. If the user wants to show all cases that belong to the category he/she chose at the beginning, selecting "Show all" on that view will do that.

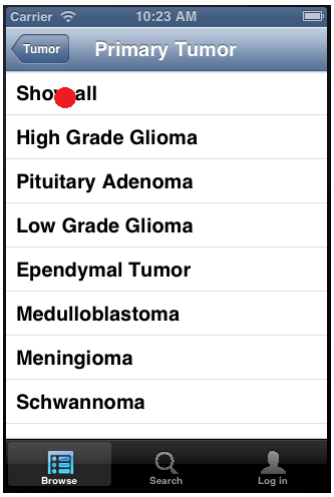

Figure A.3: Selecting "Show all"

After selecting "Show all," or by selecting a category that does not have

any sub-categories, the user is presented with a list of cases. A case can be selected the same way as a category, by simply tapping on it with a finger. Doing that will take the user to the case's own view. This view consists of a description field at the top, showing information relevant to the case. Under the information field, a couple of options might be displayed, such as "Videos" and "Images."

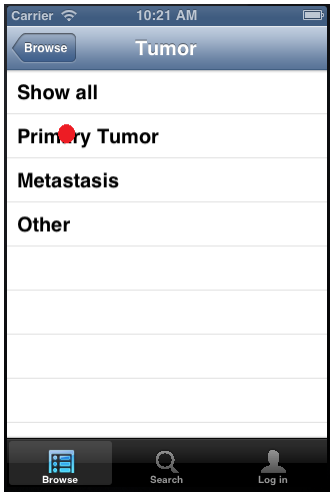

Figure A.4: Selecting a sub-category

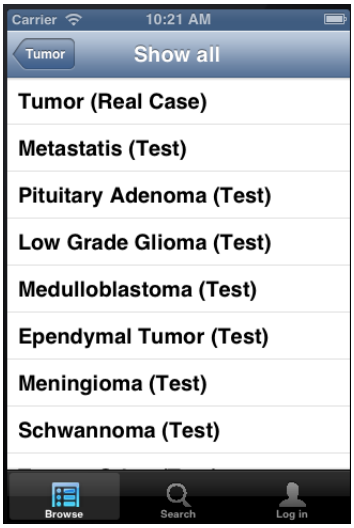

Figure A.5: Viewing cases

Selecting "Videos," if present (the option will only be available for cases that have any videos), will take the user to a screen wherein the user can play a video using the standard iOS video player. Tap the "play" button to initiate the video.

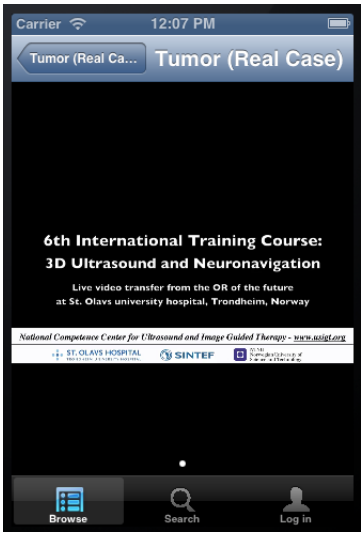

Figure A.6: Viewing video

Selecting "Images," if present (the option will only be available for cases that have any images), will take the user to a screen wherein the user can view images. To switch between images, simply "swipe" left or right with your finger over the screen. To zoom in or out on a image, simply use two fingers to "pinch," either inwards to zoom in, or outwards to zoom out.

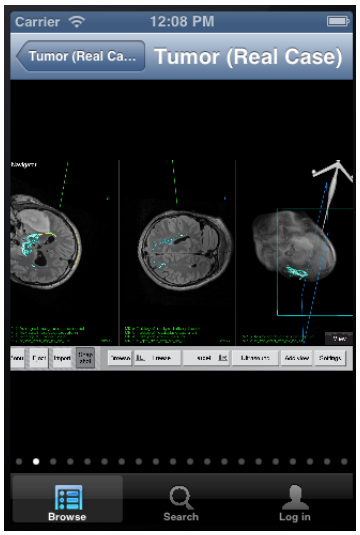

Figure A.7: Viewing image

At any time, the user can return to the previous screen by tapping the button at the upper left corner, which is shaped as an arrow, pointing backwards.

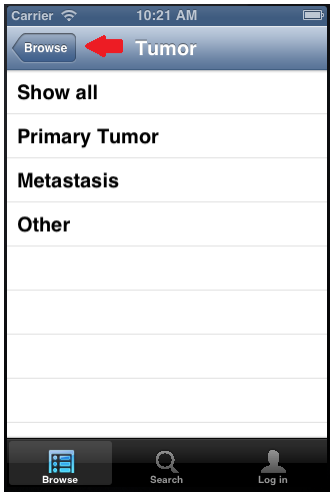

Figure A.8: "Back" button

A.2 Searching For Cases

The second tab is called "Search," and can be selected from the bottom menu. This tab prompts the user to type in one or more search keywords in a text field. To enable the on-screen keyboard, simply tap the text field with a finger. Then, type in the keywords to search for, separated by comma, ",". For example, "tumor, frontal lobe". To search, simply tap on the button labeled "Search."

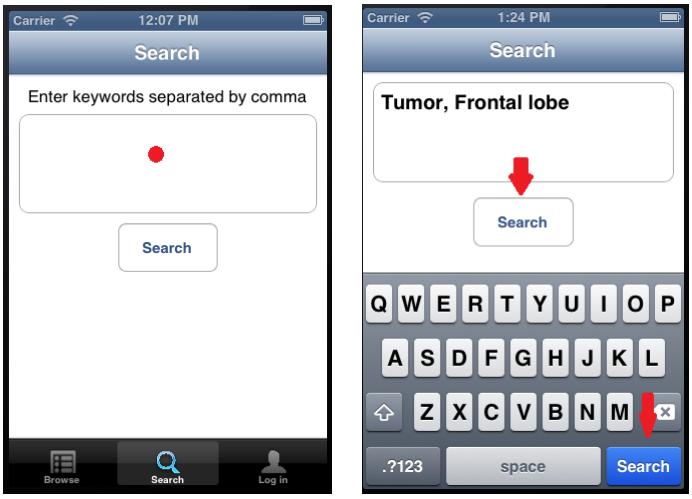

Figure A.9: Searching

After selecting "Search," the user will be taken to a new screen presenting the search results. The search results are structured exactly the same way as navigation through categories, for simplicity. Selecting a case and viewing its data is done the same as described in the previous section.

A.3 Logging In

If the user is registered in the USNeuroNav database, they can log in by navigating to the third tab at the bottom menu. Doing so will take the user to a window where the user can supply user name and password. Selecting "Log in," will verify the credentials, and then take the user to a new screen, where the user can, for instance, choose to log out again, if necessary.

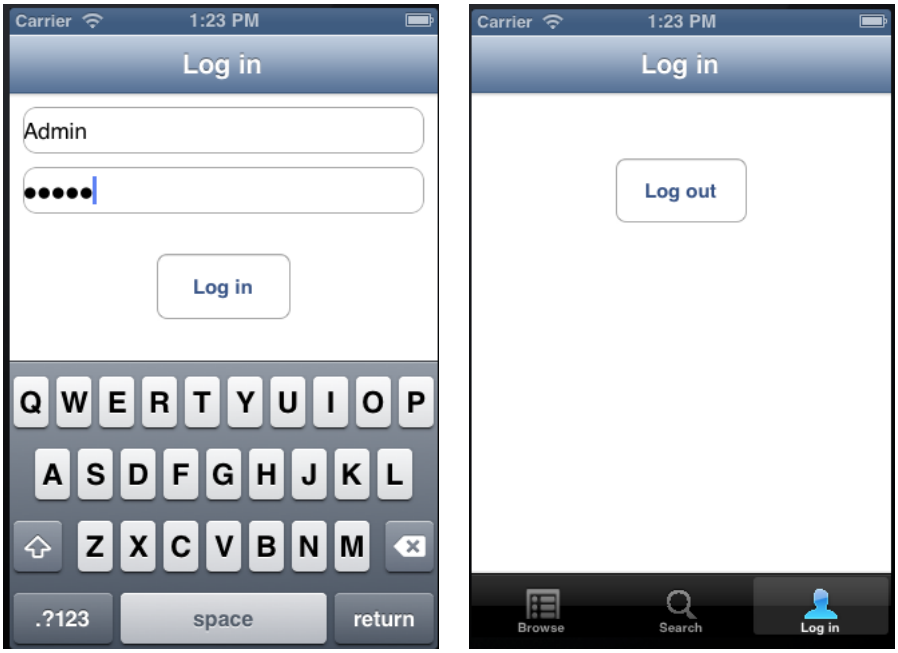

Figure A.10: Logging in

After logging in, the user can return to the browsing or search tab, and find additional cases. Also, some cases may have extra information in the description field after logging in.

B Admin Tool User Manual

The admin tool can only be run locally from the server which runs the database. To do this, SSH [42] is the preferred method. Connecting to the server can be done from a UNIX (Mac, Linux, etc.) terminal by typing "ssh [user name]@[server IP address]." The admin tool can then be launched by typing "java -jar Admintool.jar" in the directory where it is located. This is, by default, in the home directory, so no directory changing is needed. To make the admin tool run commands from a text file (described below), simply run "java -jar Admintool.jar [path to text file]."

To upload media files when creating a new case, the files must be stored on

the server, and they must be situated such that the Apache web server is able to locate them, e.g. in "/Library/WebServer/Documents/media/[casename]." Based on testing, we recommend movies to be uploaded in the .mp4 format with a resolution of 960x540, a frame rate of 29-30 fps, and a total bitrate of 1500-2000 bbps. Images must be of the .jpeg filetype, the best resolution is 1919x1153 pixels.

B.1 Adding a New Case From Text File

1. Create a new text file on the server
2. Open the newly created text file
3. Add the following lines:
   (a) Case name
   (b) The names of the categories that the case should belong to (Make sure that the order and spelling of the categories is correct, separate categories with a comma) (Example: "Tumor, Primary Tumor, Low Grade Glioma")
   (c) Public case description
   (d) Private case description
   (e) Path to where the media files are stored (locally on the server)
   (f) "private" or "public" to decide which users are able to view the case
4. Save the file
5. Run the admin tool with the file path as argument (see above)

B.2 Adding a New Case

1. In the main menu select "1: Create new case"

2. Confirm selection
3. Enter case name (This is the name of the case that will be shown in the application)
4. Confirm name
5. Enter the categories the case belong to (Make sure that the order and spelling of the categories is correct, separate categories with a comma.) (Example: "Tumor, Primary Tumor, Low Grade Glioma")
6. Confirm categories (You can also restart the step, or ask to see all categories in the database if you are uncertain about spelling, etc.)
7. Enter case public description (if you type "default" the description will become the categories of the case as entered in step 5.)
8. Confirm description
9. Enter case private description (this will only be visible to registered users that are logged in)
10. Confirm description
11. Enter path to where the case mediafiles are stored (the current path we use is: "/Library/WebServer/Documents/media/casename")
12. Confirm path
13. Select public or private case by typing "1" or "2" (private cases will only be visible to registered users that are logged in)
14. Confirm selection
15. New case have now been created

Example input:

1. “1” (Select Create new case)
2. “1” (Confirm)
3. “Case 12” (Case name)
4. “1” (Confirm)
5. “Vascular, Aneurysm,” (Categories)
6. “1” (Confirm)
7. “This is the public description” (public description)
8. “1” (Confirm)
9. “This is the private description” (private description)
10. “1” (Confirm)
11. “/Library/WebServer/Documents/media/case12data” (path to mediafiles)
12. “1” (Confirm)
13. “1” (public case)
14. “1” (Confirm)

This will create a new case with the name “Case 12,” with the public and private description entered, belonging to the categories entered. All mediafiles with ending with (".mov",".mp4",".jpg”) in the specified folder will be added to this case in the database.

B.3 Adding a New Category

1. In the main menu select “2: Create new category”
2. Confirm selection

3. Enter the name of the category (if you enter a name that already exists, the identification key will be fetched from the database so that you can modify the category)
4. Confirm name
5. Enter the parent (super-) categories of the case (If you want to add a new main-category that shows at the top of the hierarchy, simply leave this empty, then it will automaticly be added to "root")
6. Confirm supercategory
7. Enter the child (sub-) categories of the case
8. Confirm subcategory
9. A new category has been added

Example input:

1. "2" (Select Create new category)
2. "1" (Confirm)
3. "New type of primary tumor" (Category name)
4. "1" (Confirm)
5. "Tumor, Primary Tumor" (Supercategories)
6. "1" (Confirm)
7. "" (Subcategories, empty in this case, but locations such as Frontal Lobe could have been added here)
8. "1" (Confirm)

This will create a new category called "New type of primary tumor" as the subcategory of "Primary Tumor".

B.4 Adding a New User

1. In the main menu select "3: Create new user"
2. Confirm selection
3. Enter username
4. Confirm username
5. Enter password
6. Confirm password
7. A new user has been created

Example input:

1. "3" (Select Create new user)
2. "1" (Confirm)
3. "User" (Username)
4. "1" (Confirm)
5. "password" (Password)
6. "1" (Confirm)

This will create a new user with the username "User" and the password "password."

B.5 Additional Tools

Admin tool will only assist in updating and maintaining the database; it is still necessary to communicate with the database directly for several different actions, such as deleting or modifying data. In order to do this, the administrator needs either to use standard SQL queries directly to the database server, or a third-party database management program can be utilized. We found the MySQL Workbench[43] covered all the necessary features, so we would recommend that in addition to our own Admin tool to maintain and expand the database.

C Testing

C.1 System Tests

Test ID	T01
Test name	Browse categories
Description	1. Select the "Browse" tab 2. Select on "Tumor" 3. Select on "Primary tumor"
Requirements involved	FR01
Completion criteria	1. Browse interface is showed 2. Sub-categories of "Tumor" are displayed 3. Sub-categories of "Primary tumor" are displayed
Results	1. Accepted
	2. Accepted
	3. Accepted
Comments	

Table C.1: Test T01

Test ID	T02
Test name	Select a case
Description	1. Select the “Browse” tab 2. Select on “Tumor” 3. Select on “Primary tumor” 4. Select “Frontal Lobe” 5. Select “Case 1”
Requirements involved	FR01
Completion criteria	1. Browse interface is showed 2. Sub-categories of “Tumor” are displayed 3. Sub-categories of “Primary tumor” are displayed 4. “Case 1” shows in menu over cases 5. The interface for Case 1 shows
Results	1. Accepted 2. Accepted 3. Accepted 4. Accepted 5. Accepted
Comments	

Table C.2: Test T02

Test ID	T03
Test name	Browse images
Description	1. Select “Case 1” 2. Select “Images” 3. Swipe to browse to next image
Requirements involved	FR01, FR02
Preconditions	Case 1 must show in the menu (Case 1 can be found as per Test 02)
Completion criteria	1. The interface for Case 1 shows 2. An image belonging to case 1 shows 3. Swiping causes new images to appear
Results	1. Accepted 2. Accepted 3. Accepted
Comments	

Table C.3: Test T03

Test ID	T04
Test name	Play video
Description	1. Select “Case 1” 2. Select “Videos”
Requirements involved	FR01, FR03
Preconditions	Case 1 must show in the menu (Case 1 can be found as per Test 02)
Completion criteria	1. The interface for Case 1 shows 2. A video belonging to case 1 starts playing
Results	1. Accepted 2. Accepted
Comments	

Table C.4: Test T04

Test ID	T05
Test name	Search for case
Description	1. Select the “Search” tab 2. Enter “Primary Tumor”
Requirements involved	FR04
Completion criteria	1. The interface for searching shows 2. “Case 1” and “Case 2”
Results	1. Accepted 2. Accepted
Comments	

Table C.5: Test T05

Test ID	T06
Test name	Log in
Description	1. Select the “Log in” tab 2. Enter “Admin” as both username and password 3. Click on the “Log in” button
Requirements involved	FR05
Completion criteria	1. The interface for logging in shows 3. “You are now logged in” shows up in the tab
Results	1. Accepted 3. Accpeted
Comments	

Table C.6: Test T06

C.2 Test Data

Case ID	Case name	Category	Sub-Category	Location
1	Case 1	Tumor	Primary Tumor	Frontal Lobe
2	Case 2	Tumor	Primary Tumor	Unspecified
3	Case 3	Tumor	Gliomas	Unspecified
4	Case 4	Cranial Vascular	Aneu	Frontal Lobe
5	Case 5	Cranial Vascular	AVM	Parietal Lobe
6	Case 6	Spine		Brain Stem
7	Case 7	Spine		Brain Stem
8	Case 8	Ventricular Drain		Occipital Lobe
9	Case 9	Pituitary Tumor		Frontal Lobe
10	Case 10	Pituitary Tumor		Parietal Lobe

Table C.7: Test data

Case ID	Case name	Category	Sub-Category	Sub-Sub-Category
1	Tumor (Real Case)	Tumor	Primary Tumor	High Grade Glioma
2	Aneurysm (Real Case)	Vascular	Aneurysm	
3	AVM (Real Case)	Vascular	AVM	

Table C.8: Test data during, and after, sprint 4

D Category Hierarchy At Delivery

Top level	Mid level	Low level
Tumor		
	Primary Tumor	
		High Grade Glioma
		Low Grade Glioma
		Pituitary Adenoma
		Ependymal Tumor
		Medulloblastoma
		Meningioma
		Schwannoma
	Cavernous Hemangioma	
	Metastasis	
	Other	
Vascular		
	Aneurysm	
	AVM	
Ventricular Drain		
Spine		
	Prolaps	
	Tumor (Spine)	