NTNU – Trondheim
Norwegian University of
Science and Technology

IT2901 Informatikk Prosjektarbeid II
Midterm Report

Vitensenteret: A Mobile Game

Group 16:

Bjor Løyning Byrkjedal
David André Årthun Bæsje
Didrik Pemmer Aalen
Kristian Svoren
Nicolas Almagro Tonne
Oscar Thån Conrad

March 16, 2016

Contents

1 Introduction **5**
- 1.1 Assignment description . . . 5
- 1.2 Brainstorm . . . 5
  - 1.2.1 Virtual reality . . . 5
  - 1.2.2 Augmented reality . . . 5
- 1.3 Solution . . . 6
  - 1.3.1 Minigames . . . 6
- 1.4 Status . . . 8
  - 1.4.1 Waterflow game . . . 8
  - 1.4.2 Robot parts overview . . . 8

2 Pre-Study **10**
- 2.1 Considerations . . . 10
  - 2.1.1 AltBeacon . . . 10
  - 2.1.2 libGDX . . . 10
  - 2.1.3 Xamarin . . . 10
- 2.2 Technology and frameworks . . . 10
  - 2.2.1 Cordova . . . 11
  - 2.2.2 Ionic . . . 11
  - 2.2.3 AngularJS . . . 12
  - 2.2.4 Web . . . 12
  - 2.2.5 NodeJS . . . 12
  - 2.2.6 Beacons . . . 13
  - 2.2.7 Scss . . . 13
- 2.3 Tools . . . 13
  - 2.3.1 GitHub . . . 13
  - 2.3.2 Trello . . . 13
  - 2.3.3 Sharelatex . . . 13
  - 2.3.4 Google Drive . . . 14

3 Team Organization **15**
- 3.1 Scrum . . . 15
- 3.2 Planning Poker . . . 15
- 3.3 Roles . . . 15
- 3.4 Group communication . . . 16
  - 3.4.1 Facebook (Chat and groups) . . . 16
  - 3.4.2 Customer communication . . . 17

3.4.3 Supervisor communication . . . 17
3.5 Time table . . . 17

4 Requirements **18**
4.1 Functional Requirements . . . 18
4.2 Non-functional Requirements . . . 20

5 Design and Implementation **23**
5.1 System design overview . . . 23
5.1.1 Architechtural Model . . . 23
5.2 Implementation overview . . . 24
5.2.1 Organization . . . 24
5.2.2 User interface . . . 25
5.2.3 Hardware support . . . 25
5.2.4 Frameworks . . . 26
5.3 Backend . . . 26

6 Design **27**
6.1 User interface design . . . 27
6.2 Designing the application . . . 28
6.2.1 sketches . . . 28
6.2.2 Elements of design . . . 32
6.2.3 Continuity and color selection . . . 32

7 Testing **33**
7.1 Testing methods . . . 33
7.1.1 Unit testing . . . 33
7.1.2 Graphical user interface testing . . . 33
7.1.3 Usability testing . . . 34
7.1.4 Integration testing . . . 34
7.1.5 System testing . . . 34
7.2 Test Plan . . . 34

8 Project Management **37**

9 Evaluation and Conclusion **38**
9.1 The Assignment . . . 38
9.2 Choices . . . 38
9.3 The group . . . 38
9.4 Conclusion . . . 39

Abstract

The main goal with this course is to give the students practical experience on software development, and the entire process around it, for a specific customer. This includes project planning, communication with the customer and the course supervisor.

This report describes the process and everything around the project in the course IT2901 Informatics Project Work II. The project team consists of six Informatics students from the department of computer and information science at Norwegian University of Science and Technology. The customer for this projects is Trondheim Science Centre (also known as Vitensenteret). Trondheim Science Centre wanted an application for Android and iOS that could be an interactive guide and a supplement to the exhibition.

1 Introduction

1.1 Assignment description

The group was given an assignment from the Science museum in Trondheim, also known as Vitensenteret. Vitensenteret wanted an application that could be used as an interactive guide throughout the museum, or as a supplement to the different exhibitions they have to offer. Vitensenteret further specified that the main audience should be families with children in the range of 6 – 10 years. The group was told to get familiar with the museum, and to present some suggestions on the next meeting. As a backup, Vitensenteret had some ideas of their own.

1.2 Brainstorm

The first two weeks were dedicated to brainstorming for presentable ideas. In this phase, the team discussed various concepts. Only one of the ideas were presented to the museum.

1.2.1 Virtual reality

The group discussed the use of virtual reality in the application. The main idea was to make an application that gives an overview of different objects in the universe, accurately displaying the size of objects relative the size of other objects, e.g. the earth, or the solar system. This could be a very powerful experience, as the universe contains objects that is vastly bigger than what many can comprehend, especially children. However, it would require the team to use 3D modelling and to learn to program 3D applications, as well as virtual reality functionality. Therefore it seemed hard to predict the workload, and the idea was discarded.

1.2.2 Augmented reality

Using augmented reality was something that the staff at the museum suggested. After looking at some different libraries and frameworks that would allow for augmented reality, we concluded that it seemed impractical/hard. Some of the solutions that seemed viable was not free and often too expensive.

1.3 Solution

After spending a couple of afternoons at Vitensenteret, the crew found a solution that satisfied the requirements, and after discussing it with Vitensenteret, it was accepted after enthusiastic response. The task at hand was to make a cross-platform application for smart phones, consisting of a quiz and a selection of six different minigames. Each minigame should "activate" a different section of the museum.
To do this, the functionality of the minigames should be connected to either the theme in that section, or to parts of the exhibition, and in some areas the app should communicate with bluetooth beacons in order to send push notifications to the users. An example would be when a visitor enters a certain room, they could receive a notification saying that there is questions in the quiz that is connected to the room, giving an incentive to read up on something in the museum. Push notifications from the bluetooth beacons should also give the users a notification, that tells the user that some robot part can be found in that area, by playing through a mini-game.

1.3.1 Minigames

On entering the application, the user would be presented with a dilemma. A picture of a robot is displayed with a text bubble above his head. The robot asks the user to help him build his robot friend, but the parts are all missing. As the user completes minigames, more parts will be discovered, and added to a view dedicated to displaying the robot parts. After completing all games, including the quiz, the robot would be built.

1.3.1.1 Body parts overview

All collected and collectible parts will be displayed here, with the player's robot at the top. Only the collected parts will be visible on the robot. When clicking a robot part the player is taken to the map, with the room one can find the part in highlighted. From this view one can go to the map, the language selection screen and the modify robot screen, where the player can modify which robot part to use on their robot.

1.3.1.2 Waterflow

The waterflow game consist of a crane that when activated, releases water. It is a grid-based game, with each field containing a tube that is randomly turned. The user has to turn all the tubes such that the crane is connected to a drain via the tubes.

1.3.1.3 Light game

The light game is tied together with a light mixing station in the exhibition where a person can mix red, green, blue and white together with sliders to create all different colors. In the game the player is given a color and has to move the sliders in the exhibition to find what values are needed to create that color. The player then has

to input these values into the game, depicting a large combination-lock. After three iterations of correct color-matches, the lock is opened and a camera/eyes part is given.

1.3.1.4 Table of elements game

At the Science Center there is a big wall dedicated to the table of elements. Every element is there, and most of the non-radioactive elements are represented by items. In the game the user will be presented with a series of pictures of the items used to represent some of the elements, and nine buttons with symbol for different elements, one of them being the element on the picture. The user is supposed to find the symbol for the element on the wall, and press the corresponding button. A pop-up will appear with information regarding the correctness of the answer, and a couple of facts about the element the user pressed, regardless of correctness. The game is not completed until every element has been answered correctly.

1.3.1.5 Memory game

The memory game will be familiar to a lot of people. The screen will have four buttons with different colors. When the game begins, the buttons will light up in sequence. First one button. The user then pushes this one to get a confirmation that it was correct. Then two buttons light up, the user then has to press the same buttons in the same order. When the user is able recreate a pattern of 10, he/she will be rewarded with a robot part, and given the opportunity to continue or quit. If the user chooses to continue the sequence of buttons will continue to rise until the user is unable to imitate it. If time allows us, we might have a high-score list.

1.3.1.6 Sound imitation game

The sound imitation game is a game that is an addition to an already existing element in the exhibition at Trondheim Science Center. The element in the exhibition consists of a rotating arm and five lose steel pipes, of different heights. The pipes can be placed in random order around the arm. When the rotating arm hits the pipe it will make a sound. Since the pipes have different heights each pipe will have its own tone. The app will play a sound for you, and then the goal in this game is to place the pipes in the correct place and order.

1.3.1.7 Quiz

The quiz, like the minigames also rewards the user with a robot part upon completion. The purpose is to give the user an extra incentive to look for information in the museum. The group has considered multiple possibilities regarding the implementation and functionality of the quiz, for instance whether the quiz should be connected to a database, allowing Vitensenteret to pick prize-winners and manage the quiz-questions with ease. However, the main priority should be to focus on completing the minigames and the basic quiz functionality.

1.4 Status

The project status is that the customer has approved of the idea that was presented for them. The team has decided on programming language and technology. An agreement with Nordic Semiconductor has been made, in order to get beacons.

At the moment one of the minigames is completely finished. Three minigames are almost done, as well as the home-screen and menu-screen are very close to finished. Two minigames have not yet been started on. Learning how the beacons work and setting them up is also a big task that is remaining.

In the end the target situation is that meets all the requirements that have been set.It will be fully functional on both Android devices and iOS devices. All the functionality that has been implemented should work, with or without the use of beacons. The most important thing is that Vitensenteret finds the application so good that they want to apply it to their exhibition.

1.4.1 Waterflow game

The waterflow game is nearly done. The win-condition and game-logic is implemented, and the rest of the functionality can be considered to be fast to implement. Currently the game lacks proper design implementation. Before implementing the design of this game, we need a spritesheet for the tubes. The most likely approach here is that we make a spritesheet consisting of 4 or 5 images displaying an empty tube, partly filled with water tubes and a completely filled with water tube. We also need the crane and the drain sprites.

1.4.2 Robot parts overview

The robot with dynamic body parts has been made, but is not finished. The list with parts is finished with minor modifications. Is missing link to map view and a proper "change body parts" screen.

Figure 1.1: Screenshot of the current development of the parts overview view, taken in browser simulation.

2 Pre-Study

This chapter contains information about all the technology, tools and work methods which was used in this project, aswell as considerations about other solutions.

2.1 Considerations

Vitensenteret did any restrictions regarding the development tools, other than that it should support beacons for android, although cross-platform functionality was preferred. The group decided that cross-platform functionality should be included anyway, that is, if a viable solution allowed for it. Therefore, most of the alternatives that were left out, was left out because they did not not allow for beacon support in addition to cross-platform development.

2.1.1 AltBeacon

Library that adds support for beacons. It was left out because it only works on android devices.

2.1.2 libGDX

LibGDX is a free and open source game development platform written mainly in java. It provides cross-platform support and simplifies game development. However, to gain support for beacons we would, according to the groups research, have to use AltBeacon to gain support for beacons, which implies that the cross-platform feature would be left out.

2.1.3 Xamarin

Xamarin makes it possible to program cross-platform web-apps with c#. It has automated testing tools and monitoring services for managing user activity and performance. It can be noted that beacon support for both android and iOS devices seems possible to achieve. This tool was left out because the majority voted for another solution.

2.2 Technology and frameworks

The two main focuses of this project was to make the application work cross-platform and for the application to be compatible with beacons. Because of the otherwise simple nature of the application, the best option was to use Cordova.

2.2.1 Cordova

[6]
Cordova is an open source mobile application development framework that allows cross-platform development of mobile applications with the use of HTML, CSS and Javascript. This as opposed to developing platform specific applications using platform specific tools.

Cordova-projects consists of platform specific code for the chosen platforms in a project. The code a developer writes in HTML, CSS and Javascript is wrapped in platform specific code containers, extending the features of the code to work with the device. [7][10]

2.2.1.1 Plugins

A Cordova plugin is third party code which provides Javascript interface to native components. Plugins enables the application to use native device capabilities.

2.2.1.2 Android

Android applications are usually written in Java, but it's not the only programming language that is possible to use for android development. It is however recommended by google to use Java.
When making an android application in Cordova, the HTML, CSS and Javascript are wrapped in a Java container to make the application able to work with the android operating system.

2.2.1.3 iOS

Applications for iOS are written in Objective-C, C or Swift. Most applications for iOS are written in Objective-C, but more and more are written in Swift. Swift is Apples own programming language for development for Apples devices.
When making iOS applications in Cordova, the HTML, CSS and Javascript is wrapped in a Objective-C container to make the application work with the iOS operationg system.

2.2.2 Ionic

[2]
Ionic is an open source HTML5 mobile application development framework. It builds on top of Cordova and uses AngularJS. Ionic can be thought of as a front-end user interface framework that handles the look and feel of the user interface interactions. The functional part of Ionic comes from AngularJS, but AngularJS is not required for Ionic to function. Without AngularJS, Ionic consists of CSS and Sass modules which can be used for designing elements of the application.

Ionic is a framework that supports developing applications with MVC architecture. This is because of AngularJS, which is a Model View Whatever (MVW) framework. MVW supports all Model View architectures. [4][15]

2.2.3 AngularJS

AngularJS, commonly known as Angular is an open-source web application framework developed mainly by Google. Angular simplifies the process between testing and development by providing a framework which is based on the model-view-controller architecture.

The way Angular works is that it interprets a HTML page which consists of custom HTMLtag attributes. Angular then uses those attributes to bind directives to different parts of the page to a model, which is represented by regular JavaScript. Angular is a powerful tool which many big corporations use for their website today. [9][8]

2.2.4 Web

This application is built with webtechnologies.

2.2.4.1 HTML5

Hyper Text Markup Language 5, also known as HTML5 is a markup language used for structuring and presenting content on various devices. The backbone of this application is the HTML5. Without HTML5, the CSS and javascript have nothing to work with. [13]

2.2.4.2 Cascading Style Sheets

Cascading Style Sheets commonly known as just CSS. CSS is a style language which is used to describe how a document written in most mark up languages is presented. It is most often used style web pages written in HTML and XHTML. Today CSS is the most used technology to when creating the visuals to a website. [11]

2.2.4.3 Javascript

JavaScript is a high-level, light-weight and object-oriented programming language.It is best known language for scripting webpages JavaScript is also a dynamic language which uses first-class functions.

The language is prototype-based making it a multi-paradigm language. Today it is supported in most web-browsers. [16]

2.2.5 NodeJS

To install Cordova, Ionic and other dependencies, NodeJS is required. NodeJS has a package community, npm, which is the largest community of open source libraries in the

world. Npm is used to get and install both Cordova and Ionic from the open source community.[22]

2.2.6 Beacons

Beacons are small low-cost devices that transmit small amounts of data over Bluetooth Low Energy (BLE).[32] [26]

2.2.7 Scss

Sassy CSS also known as Scss is a professional CSS extension. Scss adds a series of functions, including nested rules, variables, selector inheritance and more. Scss is transformed into well formatted CSS with the help of a framework-plugin.[17][23][3][28]

2.3 Tools

2.3.1 GitHub

[21] For collaborating on the source code, GitHub will be used. This allows the group to have version control over the codebase, so in case something goes wrong, or if there are conflicting commits, it is easy to roll back to a previous version of code.

Github works, in a large extent, as a cloud based storage drive, for our source code. Local backups will be made, as a precautionary measure.

2.3.2 Trello

[18] Trello is a free to use web-based project management application that is mainly used to manage tasks. In Trello, projects are represented as boards which contains cards that represent the tasks. The reason for picking Trello over other project management tools is that it is simple and easy to use, and has the necessary functionality that the group needs to manage different aspects of the project using Scrum.

2.3.3 Sharelatex

[25] Sharelatex is a real-time online collaboration tool. Sharelatex is going to be used to write the final report. This makes it easy to collaborate on the report, and gives the final result a professional, and neat look.

In addition to being a collaborative report writing tool, Sharelatex allows stores all the documents in the cloud, for easy access at all times. It does have version control, so it is easy to roll back a previous version of the document, if necessary.

2.3.4 Google Drive

[24] Google has a broad set of collaborating tools, which we are going to use throughout the project. We are mainly going to use Google Drive to store the report, diagrams, illustrations and other documents, so that every group member can access, and add files, at all times.

We are using Google Sheets for managing our time sheet, so that every person in the group can write down how many hours they have been working on the project consecutively.

3 Team Organization

3.1 Scrum

The group will be using scrum the basis for development methodology. This fits the project well because the team is small, and the project is split into parts with completely separate functionality. The group will meet with the customer every other week. This means a natural duration of the sprints will be two weeks. The goal is to be able to show the customer some new functionality and/or new minigames every time we meet. This means the customer is able to continuously give feedback on the completed work, and an indication on whether they were picturing something else.

Scrum has a fairly strict structure when it comes to daily meetings, stand up meetings, retrospectives and other types of meetings. The group will not follow these guidelines 100%, but adapt them to the project, customer, team and time capacity. The group will meet at least 3 times a week to discuss the progress and plan forward. These meetings have mostly had the form of a stand up meeting, but more comprehensive. The meetings will not be standing, but sitting. We will be discussing what we have done since last, what we will do until next time, and what might be prohibiting us from moving forward.

3.2 Planning Poker

To hopefully make a realistic time estimate for the various tasks, the group played a game called planning poker (also referred to as Scrum poker), which is a consensus based, gamified technique for estimating workloads/sprints in terms of hours. In this game, everyone gets to estimate the time a task takes, the two individuals with the highest and lowest estimates will then explain their reasoning. Before proceeding to estimate the next task, the process is repeated for the same task until an estimate is agreed upon.

3.3 Roles

Traditionally a team consists of members with specific and clear roles. These include, but are not limited to architect, designer, developer and tester. Seeing as this is a student project, the members roles are not as clear because all members want to gain an understanding for the different aspects of the project and acquire the necessary skills and knowledge to be able to participate in any given role in the future. However, areas of responsibility have been distributed did to the team members that are best equipped

within the different areas of expertise.

The roles are; an architect to oversee that all communication within the system is done in a correct, consistent way. A designer, responsible for consistency in the graphical layout of the application, as well as it being intuitive and easy to use. A group leader, that will function as scrum master, responsible for making the team perform at full capacity and efficiency. In order for the end result to be working properly the group also have a test leader, whose responsibility is to make the team keep in mind anything that the user can do to crash the application, or anything that can go wrong with the code. The two team members without a specific role are labeled developers. This does not mean they do not engage themselves in the other areas of development, it only means that they do not have a specific responsibility that no one else has.

Team members

- Bjor Løyning Byrkjedal: Has experience in 2D game-development including comprehensive architectural design in java and c# using various libraries and frameworks. Is most comfortable in using object-oriented languages. Role: System architect.
- David André Årthun Bakke: Some experience with web-developing, designing and developing. Most comfortable writing Java, back- and frontend. Have been managing databasesystems as well. Role: Designer.
- Didrik Pemmer Aalen: Some experience with webtechnologies, but mainstrengths are backend programming mainly with Java. Is also good at organizing. Role: Group leader.
- Kristian Svoren: Some experience with webtechnologies, java, and low level programming. Most comfortable in front-end, but ready for any task. Role: Test leader.
- Nicolas Almagro Tonne: Experienced with web technologies like AngularJS, Django, PHP, as well as designing with CSS. Has experience in backend modelling and development as well, but stronger in front-end development.
  Role: Developer.
- Oscar Thån Conrad: Role: Developer. Has experience with front-end development and webtechnolgies. Is also experienced with group leadership and group management

3.4 Group communication

3.4.1 Facebook (Chat and groups)

is going to be our main communication channel for discussing issues that might arise outside of the group meetings, in addition to planning meetings if there are unexpected

matters that must be addressed.

3.4.2 Customer communication

Our communication with the customer is going to be by e-mail, phone calls or sms.

3.4.3 Supervisor communication

Communication with the supervisor will be done by e-mail.

3.5 Time table

We have used planning poker to estimate how much time the development/implementation is going to take, but it is not seen in context with the whole project.

task	**Estimation in hours**
Learn technologies (Self study) (Cordova)	32
Design general layout	18
implement functionallity for the general layout	20
Design color-minigame layout	24
implement functionallity for color-minigame	5
Design layout for quiz-minigame	10
implement functionallity for quiz-minigame	40
Design layout for shortest path-minigame	10
implement functionallity for shortest path-minigame	16
Design layout for table of elements-minigame	10
implement functionallity for table of elements-minigames	7
Design layout for water-minigame	24
implement funtionallity for water-minigame	23
Design layout for memory-minigame	16
implement functionallity for memory-minigame	18
Design layout for sound-minigame	25
implement functionallity for sound-minigame	22
Design different robot-parts	?
Setting up database	22
Handle data from database in application	25
Learn beacon-technology	?
Make the application communicate with beacons	?
Make superclasses with possibilities for inheritance	35
Testing	110
Total	512

The tasks estimated with a question mark are tasks we did not feel confident in estimating.

4 Requirements

This chapter explains and elaborates the different requirements for this application, both functional and non-functional. The functional requirements explains what the application should do and the non-functional requirements explains how the application is supposed to behave [1].

4.1 Functional Requirements

Name of functional requirement	Hints in minigames
Priority	High
Threat	If kids get stuck without the possibility to get help, they might give up finishing the game, and perhaps the other games as well.
Requirement	If kids get stuck in an arbitrary minigame, they should have the possibility to get a hint that helps them finish the minigame.
Action	Hint-button that give kids hint if pushed.
Use case	1

Name of functional requirement	Quiz
Priority	Medium
Threat	Customers might not be motivated to read the questions already posted on the walls of the museum.
Requirement	There should be a quiz-minigame.
Action	Implement the quiz-minigame
Use case	2

Name of functional requirement	Mimic the color-minigame
Priority	Medium
Requirement	There should be a Mimic the color-minigame.
Action	Implement Mimic the color-minigame
Use case	2

[1] http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

Name of functional requirement	Shortest path-minigame
Priority	Medium
Requirement	There should be a shortest path-minigame.
Action	Implement the shortest path-minigame
Use case	2

Name of functional requirement	Table of elements-minigame/quiz
Priority	Medium
Requirement	There should be a table of elements-minigame.
Action	Implement the table of elements-minigame
Use case	2

Name of functional requirement	Guide water through the pipelines-minigame
Priority	Medium
Requirement	There should be a guide the water through the pipelines-minigame.
Action	Implement the guide the water through the pipelines-minigame
Use case	2

Name of functional requirement	Mimic the sound-minigame
Priority	Medium
Requirement	There should be a mimic the sound-minigame.
Action	Implement the mimic the sound-minigame
Use case	2

Name of functional requirement	Memory-minigame
Priority	Medium
Requirement	There should be a memory-minigame.
Action	Implement the memory-minigame
Use case	2

Name of functional requirement	Collect robot-parts
Priority	High
Requirement	Every completed minigame should be rewarded with one of the robots missing parts
Action	Implement rewarding “ceremony” after each completed minigame.

Name of functional requirement	Map
Priority	Low
Requirement	The application should have a map that shows which rooms are finished and which rooms are not.
Action	Implement the map.
Use case	3

4.2 Non-functional Requirements

Name of non-functional requirement	Cross-platform
Priority	Medium
Threat	Excluding user if we stick to only one platform.
Requirement	The application should be developed for both android and iOS.
Action	Cross-platform development with Cordova.

Name of non-functional requirement	Beacons
Priority	High
Threat	People forget to use the application when going through the exhibition.
Requirement	The application should be able to get push-notifications through beacons
Action	Enable the application for beacon-push notifications

Name of non-functional requirement	No beacons
Priority	High
Threat	The beacons might cease to function, or the user does not wish to use them.
Requirement	If one or all beacons cease to function, the application should still be working.
Action	Make the application working without beacons.

Name of non-functional requirement	Age 6-10
Priority	Medium-High
Threat	The application can be too advanced for children in the age 6-10
Requirement	The application should be easy to use for children aged 6-10
Action	Make an easy user interface, and not too difficult minigames.

Name of non-functional requirement	Change elements of some minigames
Priority	Low
Threat	The games can easily be repeated because there are no changes in some of the minigames, e.g same questions in the quiz.
Requirement	The owner of the application should have the possibility to change elements of some of the minigames
Action	Have a database where questions for the quiz can be changed.

Name of non-functional requirement	Everything on the phone
Priority	Medium
Threat	The database where the changeable content is, can go down.
Requirement	If the database goes down, the app should still be able to run as normal
Action	Have all information downloaded to the phone, so it's not depending on the database.

Name of non-functional requirement	Language
Priority	High
Threat	Non-norwegian users could be using the application
Requirement	The application should have the option to change language from norwegian to english and the other way around.
Action	Implement an english version as well.

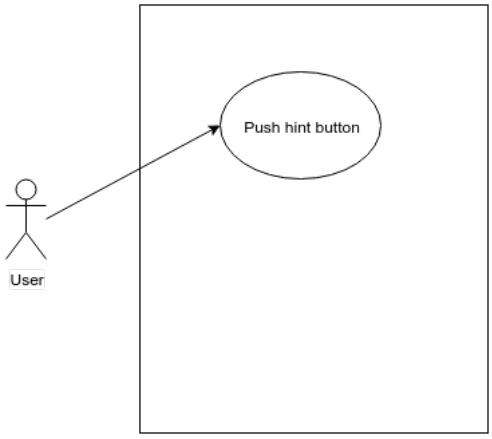

Figure 4.1: Use case 1 - use case for obtaining a hint in a minigame

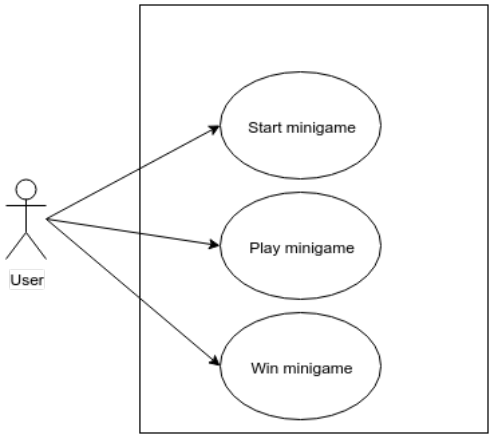

Figure 4.2: Use case 2 - Use case for playing a minigame

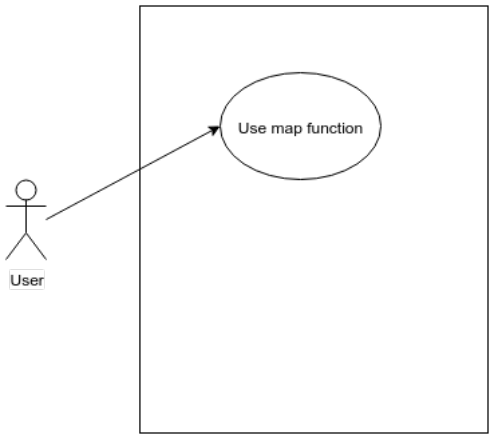

Figure 4.3: Use case 3 - Use case for using the map function

5 Design and Implementation

5.1 System design overview

5.1.1 Architechtural Model

Angular JS uses an architecture model known as MVW (Model View Whatever). This is a term used mainly by Google when describing the Angular JS architecture.[1]MVW is similar to MVC (Model View Controller) and MVVM (Model View ViewModel) [2] but instead of having to use either a Controller or a ViewModel one can use "Whatever works for you".

The system architecture the group has chosen to use is MVC (Model View Controller [3]). This is a fairly common architecture in mobile applications, as well as other platforms for software development where one wants to have changes in the view update the model and process them in the controller.

As you can see in the figure 5.1, The MVC architecture divides the system into three interconnected parts; Model, View and Controller.

The view is what the user sees. It is the graphical interface that presents the data in the model. The view contains all buttons and other parts of the UI that creates interactivity. However, when this interactivity is activated the signal and information goes directly to the controller, which then decides what happens.

The controller is the part of the application that communicates with databases and transmits data between the modules of the application. It processes the data and sends it to the model, which updates the view with the given data. The controller can communicate directly with the view as well. This is usually done when no information is changed, for instance in the case of scrolling; all the information is already in the view, it is just not on the screen.

The model is responsible for editing and updating the information in the view. It receives data/information from the controller, stores it, and updates this in the part of the view where this is supposed to go.

[1]https://plus.google.com/+AngularJS/posts/aZNVhj355G2
[2]https://en.wikipedia.org/wiki/Model–view–viewmodel
[3]https://en.wikipedia.org/wiki/Model

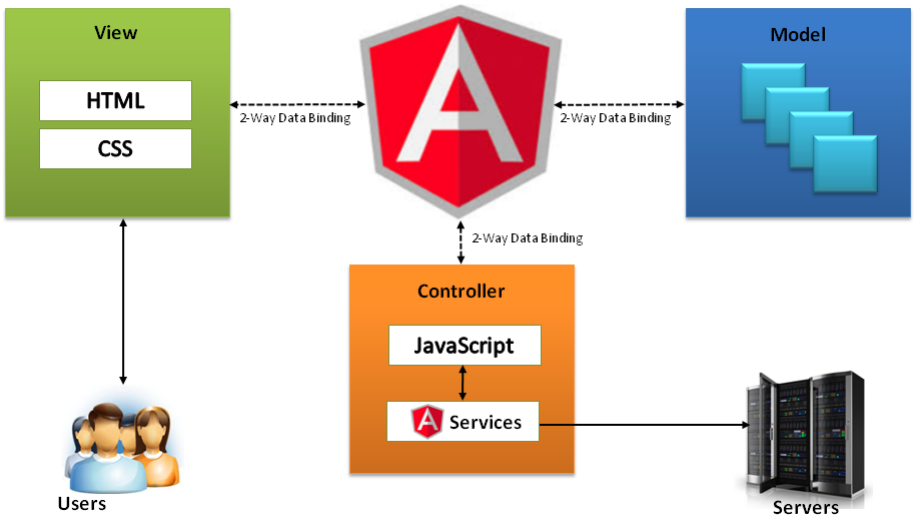

Figure 5.1: The AngularJS MVC architecture. User interacts with the view (HTML) and changes the data (Model), calls the controller (interaction), Controller modifies the Model, interacts with Servers via services. AngularJS uses 2-way bindings to detect model changes and update the view.

5.2 Implementation overview

The system is implemented as a web-application running with some native functionality on either iOS or Android through the Cordova framework which makes the web-application available offline as a normal application on smartphones. In addition to the normal web technology like HTML, CSS and JS the system uses Angular JS, and Ionic Framework.

5.2.1 Organization

The project is organized first as a cordova application, with config files, plug-ins, and other application-building related files. It is inside the `www` folder that our project lives. We have organized common files by filetype, and files for the different games inside its own folder called `app`.

Each part of the game the group has made is separated into its own folder. Each part, referred to as an 'app' inside the game, has its own view and controller files. The view is a HTML template into which data is injected, parsed and read by the Angular controller.

The project has a central module which decides which controller goes with which view, and in what structure they are. It is here that the inheritance is defined.

When "compiled" most of the project files are concatenated into one big file, which then is linked in `index.html`, which is the file first opened by when starting the app.

5.2.2 User interface

The user interface views use inheritance to make reusable design possible. The current design uses tabs at the bottom or top of the screen, depending on the OS it runs in. Each of these tabs links to its own view which is displayed by default, from where one can explore the sub views if they exist. This can be seen in fig 5.2.
When going to sub-pages, your current "position" is displayed at the top of the nav-bar, with a back button to take you back to the parent view.

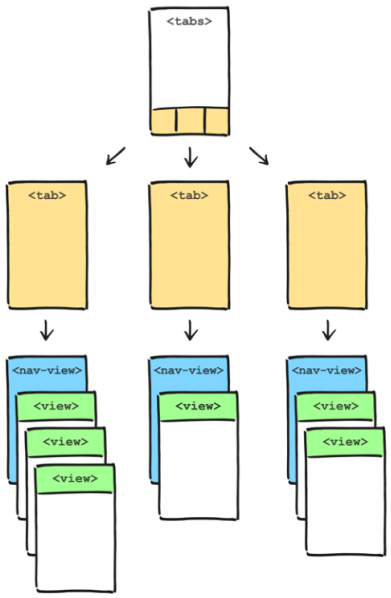

Figure 5.2: View structure in the Ionic tabs layout

5.2.3 Hardware support

By using Cordova the application has the possibility of running on a multitude of platforms, like `Amazon Fire OS, Android, Blackberry, Firefox OS, iOS, Ubuntu and Windows Phone`, with native hardware support for functionality like `accelerometer, camera, compass, contacts, geolocation, notification, storage and vibration`.[4]
The group will however focus our efforts on making the application work properly on Android and iOS before considering supporting other platforms. Hardware functionality like camera, compass and accelerometer might be used in the future, but the core of the project does not rely on any of these.

[4]https://cordova.apache.org/docs/en/4.0.0/guide/support/

5.2.4 Frameworks

Cordova is the fundamental part of the system, on top of which Ionic is implemented as a facilitator of structure and consistent design elements. Ionic uses Angular JS as a integral part of its workings and can almost be said to be a angular project written as a plug-in for Cordova. When Ionic is run Gulp compiles the SCSS code into normal CSS. The SCSS code gives us easy control over the overall look of the app, without having to change values throughout the groups entire style code.

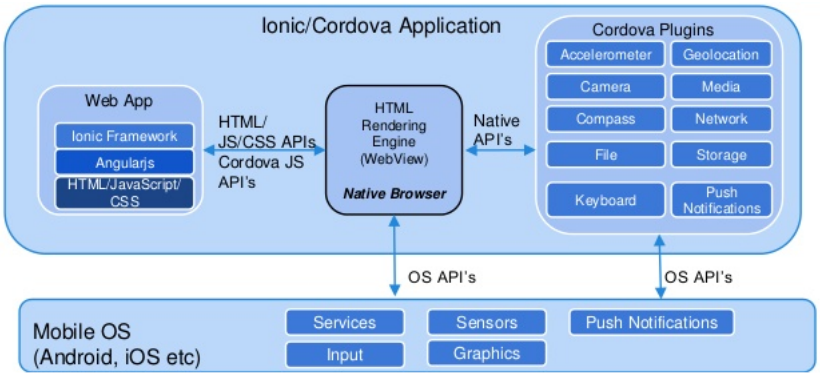

Figure 5.3: Architecture of a normal Cordova application using Ionic

5.3 Backend

The backend for the project has not been developed yet, as it is not part of the essential core functionality of the application, but extra functionality for the owners of the project. The group has, however, discussed what solutions will be used and how to do it, if the group is to implement the backend.

6 Design

6.1 User interface design

In the starting phase, the general functionality of the app was discussed with the customer. The customer did have some ideas, but wanted the group to come up with their own, to see if they had any better solutions. After the meeting, the group went through the entire exhibition, and made some thoughts on what the application should contain. The group later presented their ideas, and it was well received by the customer.

In the first meetings with the customer, the user interface, overall design and functionality of the application was discussed. The customer gave the group a design manual, which included what colors to use, guidelines for using their logo, and other general design related matters. The customer did not have any design proposals, and gave the group a pretty open task. They did, however, have a few inputs on the target audience, which was going to be children between six and ten years. From this information, some assumptions could be made:

- The application interface has to be simple and intuitive
- The colors, and graphics should be "childish"
- All information presented in the application should be written in a language targeting children.

The group spent the first weeks designing, and improving their starting ideas. After some discussion within the group, some clear points of design had been made, based on user interface design basics. [31]

- The interface should be consistent, and use common UI elements, such as a standard menu button, and standard page-navigation inside the application.
- The application should have a design with a purpose, and elements should be logically placed throughout the application
- Fonts, and font sizes should be used as a tool, to create a sense of hierarchy and structure.
- The users should not wander in the dark. It is important to provide the user with enough information to understand how they are supposed to use the application.

The text and buttons should be explained precise enough for the user to know what the next step is.

- The system should also communicate what is happening. If a button is clicked, or a setting is switched, the system should give some kind of feedback.

6.2 Designing the application

6.2.1 sketches

As the application is described in the introduction section, it consists of a set of mini games, and an overview screen that shows what robot parts have been collected. The group discussed the outline of these different parts of the application, and made some rough sketches to have something to start from. When these sketches were done, it was shown to the customer, and they gave their approval of the outline.

Figure 6.1: Welcome Screen

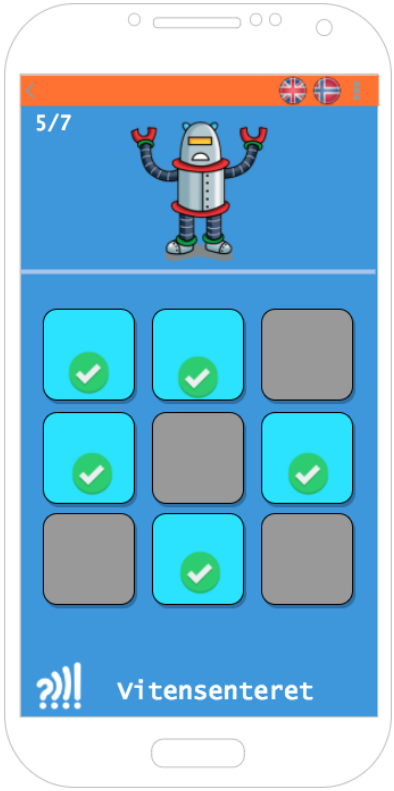

Figure 6.2: Overview screen

Figure 6.3: Quiz mini-game

Figure 6.4: Shortest path mini-game

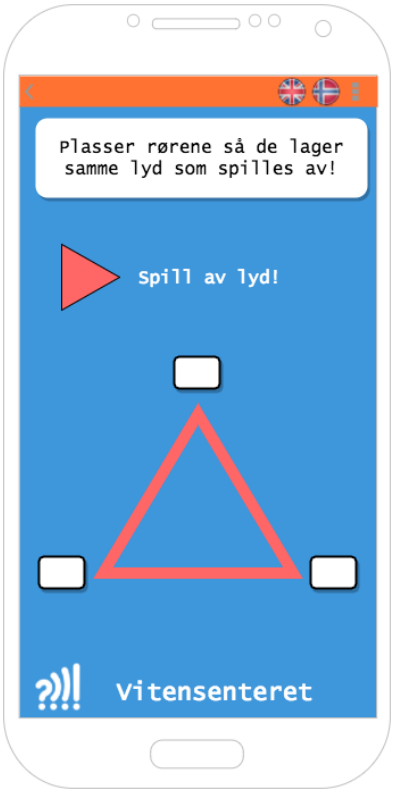

Figure 6.5: Sound mini-game

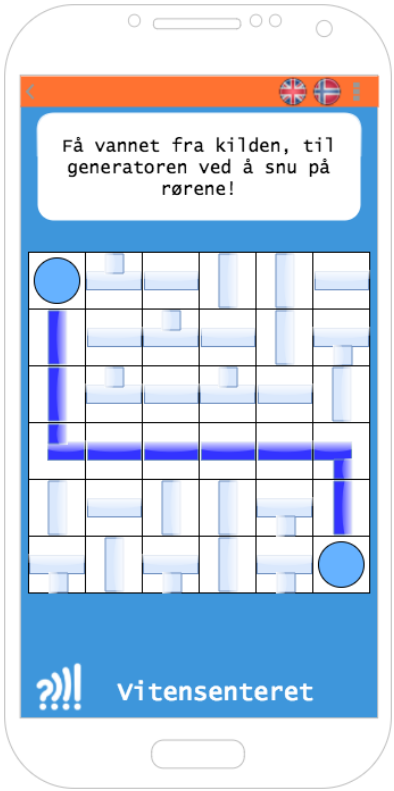

Figure 6.6: Water flow mini-game

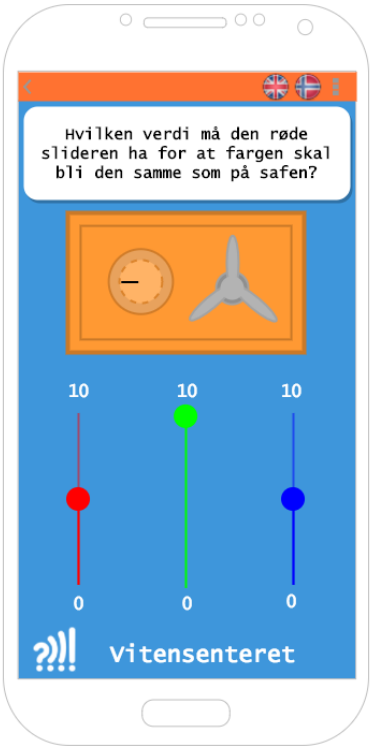

Figure 6.7: Color slider mini-game

Figure 6.8: Memory mini-game

From the time these sketches were made, until present day, some changes had to be made, to make the process a little more effective for the group. By using the built-in UI elements of the framework the group could do the development of the graphical design faster than if all the elements had to be made from scratch. There has also been a couple of changes from the original design, but the layout, and idea is the same. Below, some of the implementations can be viewed. There are still some minor changes that deviate from the sketches. The final design is under development, and the group are discussing minor changes.

Figure 6.9: Welcome Screen

Figure 6.10: Robot part main screen

Figure 6.11: Table of elements mini-game

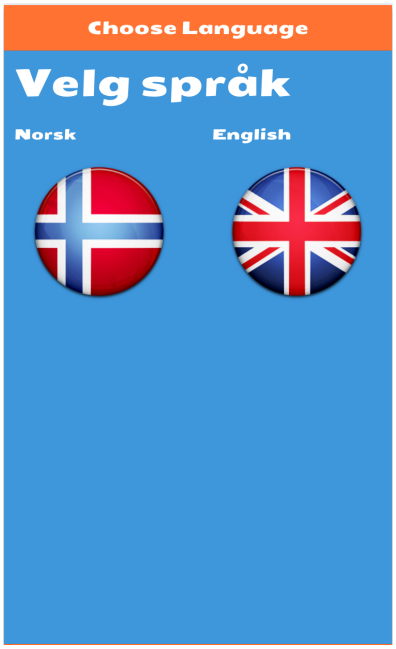

Figure 6.12: Choose language, welcoming screen

6.2.2 Elements of design

As of now, all graphical elements shown in the sketches, and in the different parts of the app, are downloaded from the Internet, for testing and outlining purposes. the group have communicated with the customer, and their graphical designer will deliver the graphical elements needed for the application.

6.2.3 Continuity and color selection

As seen in all of the sketches, there are some elements of continuity. The information boxes, have a rounded design, to make it more appealing to children. The font size i big enough to read easily. The navigation bar is present throughout the whole application, and is a familiar item, that people can relate to, and understand the usage of.

The buttons viewed in the application, also have a rounded design, for the same reasons as the text boxes. They have been given a shadow, to make it easy to understand that it is something that can be pressed.

As you can see from the sketches, the color orange, and blue are the main colors used throughout the application. These are the colors from to the customers design manual. With approval from the customer, some changes had been made to their color palette, to make the application "pop" a little more. A couple more colors were also added for variety.

7 Testing

Testing is a useful tool to ensure the quality of the end product. In an iterative project like ours it is especially important to make sure the functional and non functional requirements are been fulfilled throughout the entire project. Testing will be used to show the customer that the product being created is the product they want, as well as getting rid of bugs and other issues that might occur.

7.1 Testing methods

7.1.1 Unit testing

Unit testing is done with the smallest components of a system. A unit test can be used to ensure that something as little as a single line of code does what it is supposed to, or an entire module. Unit testing is done by writing small programs that uses the parts of the code you want to test. You then compare the results with the expected outcome. [1] [19]

Unit testing is mainly used to find problems early in the development cycle, but because the application is being developed iteratively, every sprint can be considered a cycle, so unit testing will be an important tool throughout the project period.

7.1.2 Graphical user interface testing

Graphical user interface testing is the process testing weather the system and interface meets its specifications and functional requirements. This is often done by generating test cases to be executed by test subjects within the user group you are trying to reach.[12]

These test cases are developed by test designers. The designers should attempt to make the test cases cover the entirety of the functionality of the GUI.[**?**]

So far in the project the GUI-testing have been done by the group, sometimes in cooperation with the customer. We are also in contact with an educator employed at the Science Center with insight on necessities when it comes to communicating with children. This is a very useful recourse for the group, considering that the user group is families with children, ages 6-10.

7.1.3 Usability testing

Usability testing is a way of evaluating the usability of a product by testing it on users. This is very useful because it gives direct feedback on how the user perceives the product. Usability testing is often done in laboratories with close observation.[20]
The user is given a scenario where he/she is going to complete a list of tasks that are part of the functionality of the application. The tasks are concrete, but not detailed. The tests are done to see weather the user can navigate easily within the application and find what he/she is looking for and complete the given task. After completing the tests the user is asked to report what he thinks about how it went, any difficulties they might have encountered. The same goes for the observers. The observers time how long each individual task takes to finish, and other notable information.[30]
Towards the end of the project the customer will give us an opportunity to test the application with actual customers in the exhibition. This will not be a straight-forward usability test, because we do not have a laboratory, and the application is tied closely to the exhibition. Usability testing might also be hard to do with 6 year old children in an as controlled environment as a laboratory.

7.1.4 Integration testing

Integration testing is done when new modules or components are merged into already working parts of the system. This is done to discover any issues that might occur between the working program and the new modules. [14] [5]

In this project integration testing is done by merging the code from the "master-branch" in GitHub into a members own code and test it thoroughly before pushing your own code up to master the master branch. This is done in all the branches whenever something new has been added to the master branch to make sure the newly added code does not interfere with what everyone else is doing.

7.1.5 System testing

System testing it the most extensive form of testing. This is where you get to see weather all functional and non-functional requirements are met. [27]System testing often include stress-, and performance testing.[29] At this point in the project there is not any part of the system to stress test, because the application runs locally on the users phone. The only part of the system the group considers necessary to stress test is the database the application send the "robots" to. This will most likely not be a problem, considering the small user mass at the science center.

7.2 Test Plan

As previously mentioned the application is being developed iteratively. This means there are unit tests being performed throughout the project, whenever a new part of

executable code is written, and there is a testing phase at the end of each iteration.

The majority of the tests are done by the group in preparation for the meetings with the customer. First everyone conduct their own GUI tests for the part of the application they have completed during the iteration. If the GUI tests are passed, integration testing can begin.

Integration tests are done in order. One by one the team members that have something to integrate merge the master branch into their own branch and test weather all of their functionality remains intact, before pushing merging the branch back into master. The reason for doing it this and not merging everything to master directly is that this makes it easier to figure out where/when an error occurs. This also means the last person to integrate conducts a full GUI test in collaboration with the rest of the team.

The tests below are the graphical user interface tests written to ensure all the necessary functionality is present for the table of elements minigame is present. They are also written to test that the user cannot crash the application by doing something unexpected.

At this point of the project this is the only part of the system that is finished, and ready for a complete GUI-test, but the same kind of tests will be conducted on every part of the GUI.

Test	Table of elements
Test ID	ToE01
Requirement	When selecting an element, the user will get feedback regartding the correctness of the answer, and some relevant facts about the selected element. The popup should also contain a "next element" button
Precondition	The user has entered the minigame
Approach	• Select an element when it is incorrect. • Press next element. • Repeat for every element available. • Select an element when it is correct. • Press next element. • Repeat for every element available.

Test	Table of elements
Test ID	ToE02
Requirement	If the selected element is correct, it should not reappear in the game, but if an element is incorrect, it should be but back in the stack, and reappear when you have been through all the elements.
Precondition	The user has entered the minigame
Approach	• Select an element that is correct. • Select the wrong element until you are at the place in the sequence where the correct element should have been. • If the correct element is gone, and all the incorrect ones are reappearing, the minigame has passed the test.

Test	Table of elements
Test ID	ToE03
Requirement	If the user has selected all the correct elements to the corresponding pictures, the user will receive a robot part and be returned to the main view
Precondition	The user has entered the minigame
Approach	• Select all the correct elements. • A popup should appear that gives you the correct part, with an "accept" button. • When the button is pressed the user should be returned to the main view. • Navigate to the view that contains the overview of all the acquired robot parts to verify that the part has been given.

8 Project Management

Project management documented: possible deviations and how they have been handled, examples of activity plan and status report. Major lessons learned so far.

In the beginning of the project we decided to have a group meeting every Monday at noon. We also agreed to meet the customer and the supervisor every other week. Although those meeting were not held in the same weeks. This made it a little bit harder for us to plan our sprints. We chose to do it this way in order to provide the best service to the customer. Therefor we planned our sprints based on when our customer meetings were held.

We have decided that our first and second week will conform to week six and seven in the week numbers of the year. We had two weeks before week one and two which we filled with brainstorming for the application. This is because the task we got from our customer was vaguely defined. The customer had some ideas for the application, but wanted to see what we could come up with. So we used two weeks just coming up with ideas, then after two weeks we had a new customer meeting where our ideas got approved. This opened up for us to start planning the executing of the project. We used most of week one on planning and finishing the preliminary report.

After only a few weeks we realized how valuable our group meeting were, and how much work er were able to get done when the whole group was working together. We then decided to meet three more times a week just to be able to work together.

In the middle of the third week we decided to start using Ionic, which is a powerful framework that builds on top of Cordova. This introduced some new knowledge for us to acquire and caused some delay since we had to get basic understanding of how it works. This also caused us to discard a lot of the work we had already done, because the implementation of Ionic, our previous work was not compatible with the rest of our code.

According to our activity plan that we made for week 3 and 4. We were able to finish most of our planned assignments. In week 3 we had two member of the group who were ill. This was the reason why we were not able to complete all the planned tasks.

The biggest lesson we have learned so far is the fact that is when we work the whole group together as one unit. We are able to do much more than when we work individually.

9 Evaluation and Conclusion

9.1 The Assignment

On our first customer meeting, we had no idea of what to expect. And from our point of view it seemed like the customer did not know what to expect either. We did not know much about our assignment and we got very excited when they told us that we could do almost whatever we wanted, as long as we created something cool and made use of new and interesting technology. Our first thought was that our assignment was great, we could create whatever we wanted. However, on the other hand it became hard to limit ourselves, and not go over board with ideas and plans.

In the end we managed to come up with the plan that both we and the customer found interesting. It took advantage of interesting technology as well, as it was not to complicated and unrealistic to complete.

9.2 Choices

So far during the project we have made two major choices which have benefited the group greatly. Our first decision was to begin to make use of the framework Ionic. At first we had some trouble setting it up and understanding how to use it. In retrospect we now see that it has helped us a great deal, and spared us for a lot of work.

Our second major choice was to meet three more times a week just to work together, and not once a week just for group meetings. We quickly realized that we worked much better when the whole group was present at the same time.

9.3 The group

When two members of the group where sick for a whole week at the same time, we saw that the work progression did not go according to plan. We had not really accounted for two members being sick at the same time for such a long time. But we made up for lost time in the following weeks, and are now back on schedule.

When it comes to the group, the atmosphere is very good. We all share the goal of achieving a good grade. In order to do that we have signed a contract which specifies how we are going to work together to reach our goals.

9.4 Conclusion

In conclusion we feel that we are doing very well. We are a group that works hard together to reach our goals. So far we have done a lot, but we also have a lot of future work to do in order to reach the deadline.

Bibliography

[1] http://code.tutsplus.com/articles/the-beginners-guide-to-unit-testing-what-is-unit-testing-wp 25728. *The Beginner's Guide to Unit Testing: What Is Unit Testing?* 2016.

[2] http://ionicframework.com/. *Ionic framework.*

[3] http://ionicframework.com/docs/cli/sass.html. *Scss and Ionic.*

[4] http://ionicframework.com/docs/guide/preface.html. *Ionic.*

[5] http://istqbexamcertification.com/what-is-integration testing/. *Integration testing.* 2016.

[6] https://cordova.apache.org/docs/en/5.4.0/guide/overview/. *Apache cordova overview.* 2016.

[7] https://cordova.apache.org/docs/en/latest/guide/overview/. *Cordova.*

[8] https://docs.angularjs.org/guide/introduction. *Angularjs.*

[9] https://en.wikipedia.org/wiki/AngularJS. *Angularjs.*

[10] https://en.wikipedia.org/wiki/Apache$_{C}ordova.Cordova$.

[11] https://en.wikipedia.org/wiki/Cascading$_{S}tyle_{S}heets.CSS$.

[12] https://en.wikipedia.org/wiki/Graphical_user_interface_testing. *Graphical user interface testing.* 2016.

[13] https://en.wikipedia.org/wiki/HTML. *HTML.*

[14] https://en.wikipedia.org/wiki/Integration_testing. *Integration testing.* 2016.

[15] https://en.wikipedia.org/wiki/Ionic$_{(}mobile_{a}pp_{f}ramework).Ionic$.

[16] https://en.wikipedia.org/wiki/JavaScript. *Javascript.*

[17] https://en.wikipedia.org/wiki/Sass$_{(}stylesheet_{l}anguage).Scss$.

[18] https://en.wikipedia.org/wiki/Trello. *Trello.*

[19] https://en.wikipedia.org/wiki/Unit_testing. *Unit Testing.* 2016.

[20] https://en.wikipedia.org/wiki/Usability_testing. *Usability testing.* 2016.

[21] https://github.com/. *Github*.

[22] https://nodejs.org/en/. *Nodejs*.

[23] https://responsivedesign.is/articles/difference-between-sass-and scss. *Scss*.

[24] https://www.drive.google.com/. *Google drive*.

[25] https://www.sharelatex.com/. *Sharelatex*.

[26] http://uk.businessinsider.com/beacons-and-ibeacons-create-a-new-market 2013-12?r=USIR=T. *Beacons*.

[27] http://www.guru99.com/system testing.html. *What is system testing*. 2016.

[28] http://www.mindscapehq.com/products/web-workbench/what-is sass. *Scss*.

[29] http://www.softwaretestingclass.com/system-testing-what-why how/. *System Testing: What? Why? & How?* 2016.

[30] http://www.usability.gov/how-to-and-tools/methods/usability testing.html. *Usability testing*. 2016.

[31] http://www.usability.gov/what-and-why/user-interface design.html. *User interface design basics*. 2016.

[32] http://www.webopedia.com/TERM/B/beacon.html. *Beacons*.