IT2901 - Informatics Project II
Final Report

Vitensenteret: The RobOtto Trail

Group 16:

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

May 30, 2016

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 as well as the actual development of the software.

This report describes the process and the result of 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 hereafter known as Vitensenteret. They wanted an application for Android and iOS that could be an interactive guide and a supplement to the exhibition.

Contents

Contents

1. Introduction

This is the project report for the course Informatics Project II - IT2901 written the spring semester, 2016 at the Norwegian University of Science and Technology. This report will give the reader an overview of the project. The assignment was to create a mobile application for Vitensenteret.

1.1. The Course

Informatics Project II - IT2901 (1) is a mandatory course in NTNU's informatics bachelor program. The course is the bachelor's thesis for the students attending the informatics program and is usually taken during the sixth and last semester. The purpose behind this course is to give the students practical experience with a real software development process for a real customer. This includes obtaining experience on the complete software development process. The students are supposed learn techniques within software engineering, including different programming languages, development tools and project management tools.

1.2. The Group

The group consists of six students studying informatics at NTNU. All the members has different backgrounds and possess different set of skills.

- **Bjor Løyning Byrkjedal:**
  Bjor has experience with 2D game-development and architectural design in java and c# using various libraries and frameworks. He is most experienced with object-oriented languages.
- **David André Årthun Bakke:**
  David has some experience with web-development, design, and database management. He is comfortable writing both backend and frontend, mainly in Java.
- **Didrik Pemmer Aalen:**
  Didrik is experienced with web technologies, but his primary strengths are backend programming, mainly with Java, and team leadership.
- **Kristian Svoren:**
  Kristian has experience with web technologies, java, and low level programming, but is most comfortable in front-end.

- **Nicolas Almagro Tonne:**
  Nicolas is experienced with web technologies such as AngularJS, Django, PHP, as well as designing with CSS. He also has experience in backend modelling and development, but is stronger in front-end development.

- **Oscar Thån Conrad:**
  Oscar has experience with front-end development and web technologies. He also has some experience with group leadership.

1.3. The Customer

Vitensenteret(2) is a popular scientific visitor center in Trondheim. The museum is located in Trondheim city center and offers many different activities such as robot lab, planetarium, invention workshops and more(3). This project will focus on their the main exhibition. Vitensenteret's motto is "technology and science for everyone!"

1.4. Project description

The initial assignment was described as follows:

"Vitensenteret has made several pathways through the center where you can experience the exhibitions based on themes and interests. We would like to build these pathways into apps for smartphones and combine them with mini-games/ puzzles/ in-depth knowledge etc, that allows visitors to have new experiences in the science center and enrich the experience of the models on display. The pathways will be developed together with the student group."

On the first meeting, the customer specified that the main audience for the application should be families with children between ages six and ten.

1.5. The goal

The goal with this project was to make a fun and interesting application for Vitensenteret. They wanted an application that would motivate the visitors to interact with the exhibition. They had observed that many visitors tended not to visit every part of the exhibition. To solve this they wanted an application that incentivized the visitors to explore more of the features at the museum.

1.6. Definitions and abbreviations

This section contains the definition of terms, as well as all abbreviations needed to understand this report.

- **Activate**
  Throughout the report, the verb "activate" has been used about items in the exhibition. In this context, it is meant as a means of incentivizing users to interact or take interest in parts of the museum.

- **Angular JS**
  Is a Javascript MVW framework.

- **Apache**
  The world's most used web server software.

- **C++**
  General purpose programming language known for performance with low-level memory manipulation.

- **C#**
  A multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic and object-oriented, developed by Microsoft.

- **C**
  Low-level and efficient computer programming language.

- **CSS**
  Cascading Style Sheets are for styling web pages.

- **GUI**
  Graphical User Interface

- **Gulp**
  Build system for automating tasks: minification and copying of all JavaScript files, static images

- **HTML**
  HyperText Markup Language; a markup language for defining the structure of web pages.

- **IDE**
  Integrated development environment.

- **iOS**
  Apple's operating system for mobile devices.

- **Java**
  General-purpose, class-based computer programming language.

- **JDK**
  Java Standard Edition development kit.

- **JSON**
  JavaScript Object notation. It is a lightweight data-interchange format.
- **JS**
  JavaScript is a high-level, dynamic, untyped, and interpreted programming language for adding dynamic functionality to web pages.
- **Linux**
  A family of operating systems which are UNIX-like, such as Debian and Ubuntu.
- **MVC**
  Model, View, Controller is a software architectural pattern for implementing user interfaces on computers.
- **MVW**
  Model View Whatever is the same as MVC, but one can use "Whatever" instead of a controller.
- **MYSQL**
  Is an open-source relational database management system.
- **NTNU**
  Norges Teknisk Naturvitenskapelige Universitet or Norwegian University of Science and Technology
- **OS X**
  Apple's operating system for desktop and laptop computers.
- **PHP**
  Is a general purpose scripting language.
- **Python**
  A high-level, general-purpose, interpreted, dynamic programming language.
- **SCSS**
  Syntactically Awesome Style Sheets
- **SDK**
  Software development kit
- **UI**
  User interface; the space where interactions between humans and machines occur.
- **Vitensenteret**
  Vitensenteret is the norwegian name for Trondheim Science Museum.
- **Windows**
  Microsoft Windows operating system

- **XAML**
  Extensible Application Markup Language
- **Xcode**
  Apple's IDE for development of applications for Apple devices.

2. Prestudy

Given the problem Vitensenteret presented to the group, the group decided to look at solutions to similar problems in different museums, as well as conducting a brainstorming session to come up with ideas. It was also important to gather knowledge about different technologies, frameworks, and different development methods. This chapter will elaborate on the different ideas, technologies and development methods that were considered for this project.

2.1. Ideas for the application

After receiving the project description from NTNU, the group had the impression that Vitensenteret already had a clear vision for the application.The first meeting between the group and Vitensenteret proved otherwise. Vitensenteret told the group that they wanted a mobile application, but they did not specify how they wanted it to be, although they did have some ideas. These ideas were considered and some of them were used in the final version of the application. This section explains sources of inspiration and the ideas that were considered for the application.

2.1.1. Similar applications

To find inspiration for the application, the group searched for solutions to similar problems. This subsection will go through the different applications that were found and how they inspired the group.

2.1.1.1. Science Museum in London

The Science Museum in London (4) had several applications for the group to look at for inspiration. One of the applications involved using beacons to interact with the exhibition(5). The group was intrigued by this idea and started to look more into the beacon technology. Read more about beacons in section 2.1.4

Another application let the user design their own space rover(6). This gave the group an idea to use the same concept, but in a different manner. The group saw the opportunity to use this as a rewarding mechanism in the application. Read more about rewards in the application in section 2.1.7.

2.1.1.2. American Museum of Natural History

The American Museum of Natural History(7) had several applications, one of which the group found interesting. It was meant to be a guide throughout the exhibition, and to give the user a new experience in the museum(8). This gave the group a similar idea. Read more about this concept in section 2.1.6.

2.1.2. Virtual reality

Virtual reality is a technology that the team thought would bring more customers to Vitensenteret in that the group considered that the technology itself was appealing. The idea was to make an application which takes the user on a trip through the universe while describing and visually displaying objects such as the earth, stars, and galaxies. By comparing the properties of these objects like the size and weight, the app could provide a sensational experience and a new perspective of how big the universe is. Using virtual reality for this could be a good idea because it immerses the user into the environment of the application. To implement virtual reality applications, knowledge about 3D-modeling is required(9).

2.1.3. Augmented reality

Vitenseteret suggested that the group could use augmented reality in the project. Their reasoning was that there had been recent progress in augmented reality technologies that could be innovative to use at a museum. The concept of augmented reality is capturing the real-world environment, and augmenting it with software. Augumented reality is known for blurring the line between what is real and what is computer generated(10). For instance, the app could capture reality with a camera, augment it, and display an enhanced version. Implementing this kind of augmented reality application would require knowledge about 3D-modeling.

2.1.4. Beacons

The customer wanted the group to make an outstanding application, preferably with new technology. This made the group think of beacons. Beacons are small Bluetooth-devices that constantly emits data to tablets, smartphones and other Bluetooth devices within range. The data sent by the beacons can be modified and used to trigger actions on a device that receives the package. This action can be anything from opening a web page to having an app do something specific(11)(12)
.

2.1.5. Minigames

The idea to structure the application as a series of minigames was suggested in the original project description. When the group was brainstorming in the exhibition there

were many minigame ideas. The different sections in the exhibition have different themes, therefore, the group thought of making minigames that fit the themes of the exhibition.

2.1.6. Exhibition guide

Making the application a guide through the exhibition was quite similar to the minigames-idea. The group got this idea from Vitensenteret, with inspiration from the application from the American Museum of Natural History. One idea was to make the application have a storyline that would guide the user through the museum. The group though that this could make the application more interesting and hopefully trigger curiosity in the user, which in turn could make the user want to go through the whole museum.

2.1.7. Rewards

In order to make the application more appealing towards the targeted audience, the group considered rewarding the user for making progress.

The group discussed the possibilities for rewards in the form of discounts in Vitensenterets gift shop with Vitensenteret. Potentially, the children who finished the application could show their device to the staff managing the gift shop and get either a small box of candy or discount on a small item in the gift shop.

Having rewards in the application itself was also considered. Whenever progress is made in the application, the user could get something in the application such as puzzle pieces or parts for a map. When all parts are collected, the user could receive some sort of reward.

2.2. Frameworks and technologies

Most of the ideas that were considered for the application would require usage of different technologies. The group explored the technologies which made the different ideas possible. This section introduces these technologies.

2.2.1. Native platform

The first technologies that were considered were based on development for the native platforms. The native platforms are iOS, Android, and Windows Phone.

2.2.1.1. iOS

Native development for iOS can be done with multiple programming languages, but is mainly done in Objective-C and Swift(13)(14). To develop native iOS applications, it is necessary to have a device running OS X, with Xcode installed(15). One also has to buy the Apple developers license.

Because Apple's iPhone holds approximately 28% of the Norwegian smartphone market share(16), it makes sense to develop an application that works on this operating system. Another reason for an application compatible with iOS is that it supports beacons, virtual reality, and augmented reality.

One problem with this native iOS development is that it excludes other platforms, and special tools are needed to develop native iOS applications.

2.2.1.2. Android

Native Android applications can be implemented with Java, C, or C++, although Java is the promoted programming language by Google(17).

Because Android holds a big share of approximately 62% of the Norwegian smartphone market(16), it makes sense to develop for the Android operating system. Unlike native iOS application development, Android application development does not require any special tools. All that is needed is a JDK, an SDK, and an IDE. Native development for Android supports beacons, virtual reality, and augmented reality.

One problem with developing a native Android application is that the other platforms would be excluded.

2.2.1.3. Windows Phone

Native application development for Windows Phone can be done using C# or Visual Basic for the code part, and XAML for the visuals. The tools for making Windows Phone applications are most functional when using the Windows desktop operating system, but Microsoft has made it possible to make applications on UNIX based operating systems such as Linux and OS X as well.

Unlike Android and iPhone, there aren't that many Windows Phones on the market. As of April 2016, windows phone has roughly 4% of the market share(18). It is, therefore, not very attractive to develop native for Windows Phone.

2.2.2. Google VR

The Google VR SDK is a software development kit provided by Google for the development of virtual reality applications for both iOS and Android. The SDK allows for development of virtual reality applications for Google Cardboard and Google Daydream(19)(20), which is Google's own virtual reality devices.

Google Cardboard is Google's cheap virtual reality device, which allows anyone with a smartphone to develop and experience virtual reality applications.

Google Daydream is Google's high-end virtual reality device. It is more expensive than

Google Cardboard, and comes with some extra features like a controller that makes it easy to navigate and control the virtual environment. Google Daydream, like Google Cardboard, uses a smartphone to portray the virtual reality.

2.2.3. Cross platform

An alternative to developing native applications is developing cross platform applications. There are many frameworks and libraries that enable this, some of which were considered for this project.

2.2.3.1. Xamarin

Xamarin is an application development platform which enables making cross-platform applications. It is based on the programming language C# and can be used to develop applications for Android, iOS, and Windows Phone.

Xamarin is compiled for native performance, something that cannot be achieved with other cross-platform development tools. It also utilizes native libraries and APIs. This means that every library or API that is made for either Android, iOS, or Windows Phone can be used by Xamarin applications. Xamarin is a tool that is free to use for developing open-source applications, but that otherwise costs money(21).

2.2.3.2. Cordova

Cordova is a free and open-source framework for cross-platform application development(22). It is issued under The Apache Software Foundation, which supports open-source development. Cordova is based on wrapping web technologies in a native wrapper. This enables a developer to use HTML, CSS, and JavaScript to develop applications. Cordova then wraps it in the code for the wanted platforms(23). It is possible to make applications for several different platforms, including Android, iOS, and Windows Phone. Cordova has support for plugins that enable native functionality, and has plugins that support iBeacons(24), and Eddystone beacons(25).
Developing with Cordova gives the applications some restrictions. The main restriction is performance, because it is not developed natively(26).

2.2.3.3. Ionic

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 that can be used for designing elements of the application(27).

Ionic is a framework that supports developing applications with an MVC architecture.

This is because it builds on AngularJS, which utilizes Model View Whatever (MVW) architecture (see 8.1). One of the key features of Ionic is that it makes standardized design easier. Applications developed with Ionic will not perform as good as native applications(28)(29).

2.2.4. Beacons

Amongst multiple choices of beacon technology in the industry, the group found two of the alternatives more attractive.

iBeacon is a beacon standard developed by Apple. This technology allows both Android and iOS mobile devices to receive Bluetooth-signals broadcaste from an iBeacon(30).

Eddystone is Google's new open source beacon standard. It deviates from Apple's iBeacon standard by being able to transmit larger data packages. This means that the beacon can send an external URL, from which the user can download content. This opens for a variety of possibilities that the iBeacon does not support. For instance, iBeacon can only work as an identifier that the application would need to read in order to do an operation. With Eddystone you can download content from the url provided in the package(31).

2.3. Development methods

To find a development method that fit the project, the team made an effort in considering several development methods.

2.3.1. Scrum

Scrum is an agile and iterative development method. The method is designed to help small teams develop complex products. Scrum is mostly used in the technology sector, but it is also found elsewhere. A scrum team usually consists of approximately seven people. One of the mantras of scrum is "inspect and adapt". Scrum teams are known for being determined to constantly improve their work process and the product they are developing(32).

2.3.1.1. Sprints

In scrum, the team works together in sprints. A sprint is a static period of time, typically between one week and a month, where a specific workload has to be completed and ready for review within the end of the sprint. The duration of the sprint, as well as the planned work is determined by the team, sometimes in collaboration with the customer. A project developed in scrum consists of many sprints, where one sprint builds upon the previous(33).

2.3.1.2. Roles

Scrum defines a set of roles: product owner, scrum master and development team(32).

Product owner: The product owner holds the vision of the product, and represents the customer. The product owner is responsible for the communicating the customer's needs and wants to the scrum team.
Scrum master: The scrum master is the expert on scrum and adviser. The scrum master helps the team follow the scrum guidelines and overcome any obstacles or interferences that might occur in the development process. It is important to note that the scrum master is not there to be the team's boss, but to help the team organize the project and preform to the best of their ability.

Development team: The development team consists of all the developers. They are the ones that have full control and authority over what is done and what is going to be done. The developers are responsible for all the actual work, including prototyping, design, development, and testing.

2.3.1.3. Scrum cycle

The standard Scrum cycle is to have daily meetings, called "stand up meetings". They are typically conducted standing, to limit the time use. Each team member presents what they have done since the last stand up meeting, what they are going to do until the next meeting, and any potential problems prohibiting them from moving forward. At the end of each sprint, the outcome is presented to the customer. This is called a "sprint demo".This is a good opportunity for the customer to give the team feedback. After the sprint demo the group should conduct a sprint reflection meeting called "retrospect". In the retrospect, the team discuss what could have been done differently in the sprint, what they did well, and the current situation of the development process (32).

Figure 2.1 illustrates the cycle of a sprint. The product backlog is an overview containing all the different tasks, and has to be done in order for the project to be completed. The sprint backlog contains tasks from the product backlog that are supposed to be completed within a given sprint.

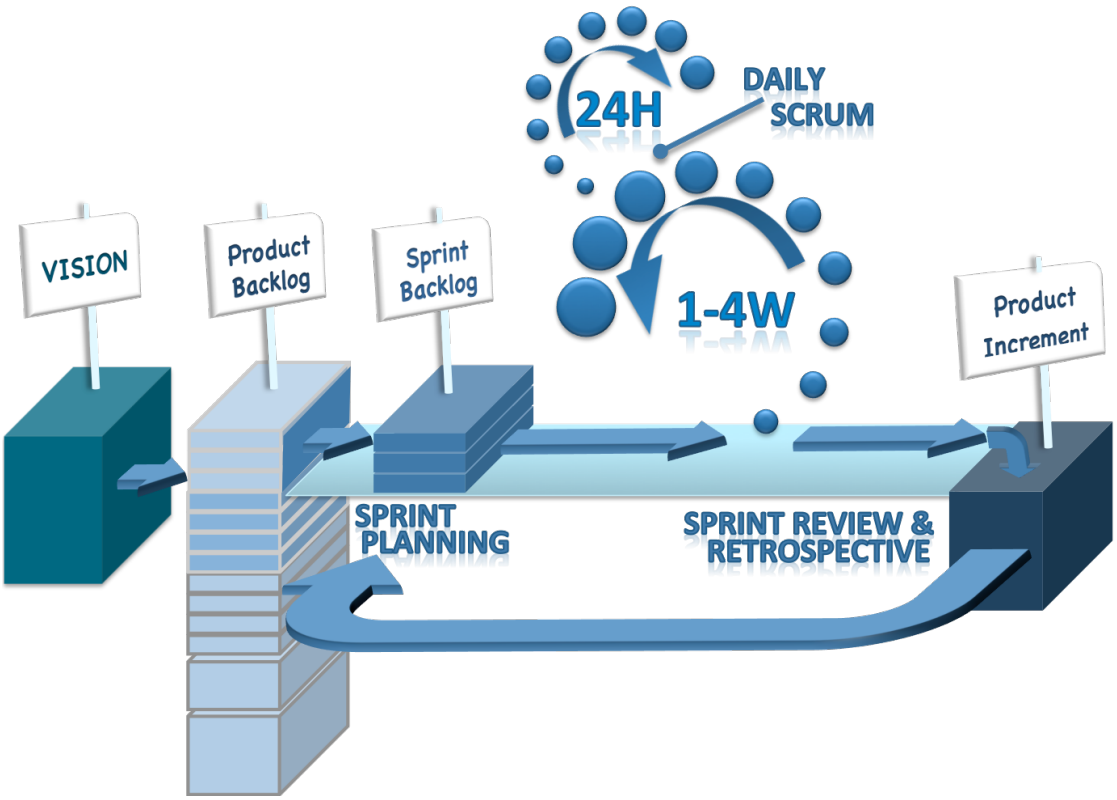


Figure 2.1.: The Scrum cycle

2.3.1.4. Reflection

The goal with Scrum is that the team should be able to present new functionality to the customer, and get feedback throughout the development process. This reduces the possibility of disagreements and unsatisfied customers when the product is to be released. Another advantage about Scrum is the flexibility and freedom that comes with having a board of tasks to choose from. Scrum can be criticized for the amount of effort and knowledge required by all the members for it to be efficient. A workaround to this is to define the contents and procedure of the meetings as it fits the team.

2.3.2. Waterfall

The waterfall method is a development method that was very popular for a long time. It was one of the first development models, and is also known as linear-sequential life cycle model. The waterfall method is common in smaller projects where the requirements are simple, detailed and when there are no uncertainties(34).

2.3.2.1. Sequential progress

The waterfall model provides a framework that aims to complete the development cycle sequentially. The standard waterfall model defines five phases: requirements, design, implementation, verification and maintenance, which are to be completed in that order(see figure 2.2). A phase cannot be initiated before the preceding phase has been completed. When all the phases are completed, the product is finished. In contrast to the Scrum model described in the previous subsection, the waterfall model is characterized by strict planning procedures and documentation. This is because any change in the requirements could force the developers to redo a phase that is marked as completed, which could be expensive(34).

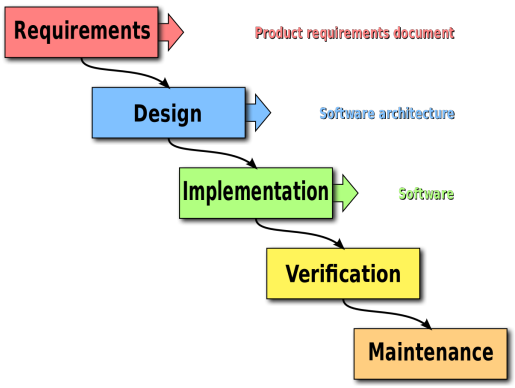


Figure 2.2.: The Waterfall method.

2.3.2.2. Reflection

Whilst agile development methods require good communication to be effective, the sequential waterfall model relies more on good documentation analysis than on communication. Generally, the bigger the team - the worse the communication in the team is going to be. By this logic, the waterfall model might be suitable for some of the big development teams. If the task is to develop a system that in any case would require a lot of analysis and documentation, the disadvantage of adding additional work in the documentation would cancel out. In the requirements phase, the development team analyze the requirements provided by the customer. Since the customer relies on the team to accurately analyze the requirements, and the customer is not included in the development process, the risk of surprises and discontent customers increase with the complexity of the requirements. It could, however, work well if the requirements are static and easy to analyze.

2.3.3. Lean software development

Lean software development is an agile software development method created by Mary and Tom Poppendieck. It is based on seven principles(35):

Eliminate waste: Get rid of everything that does not have any value to the customer, e.g. partially done work.

Amplify learning: The development process should be customized to amplify learning for the developers. This is done by using short iteration cycles and by focusing on testing code as soon as it is written.

Decide as late as possible: Delaying decisions as much as possible to be able to make decisions based on facts and not uncertain assumptions and predictions. This would result in better decisions being made.

Deliver as fast as possible: If the project is delivered sooner, it will result in getting feedback earlier than one would with a late delivery.

Empower the team: Instead of managers telling the developers how to do their work, the roles were turned. The managers are taught to listen to the developers, so that they can better explain what actions should be made.

Build integrity in: The customer needs to have insight and opportunity to influence how the system is advertised, delivered, deployed, accessed, how intuitive it is to use, the price, and how well it solves the problem. This gives the customer a comprehensive experience of the system.

See the whole: Decompose big tasks in the project into smaller tasks. This makes it easier to find and eliminate defects.

2.3.4. Pair programming

Pair programming is an agile development technique in which two developers work together on a single workstation. The two developers take turns writing code while the other developer reviews each line of code that is being written. The programmer is often called "the driver" and the reviewer is often called "the observer". The driver's main objective is to write the code and complete the current task, while the observer is responsible for verifying the code. Verifying the code includes suggesting different methods of solving problems, and keeping in mind how the code interacts with the rest of the system(36).

This method of programming comes with many advantages. Research shows that pairs tend to spend 15% more time on programming than individuals. However, the result

is usually a program with 15% less unwanted features and bugs. Today, one of the biggest expenses in the digital world in is solving and repairing software problems. Pair programming results in code with higher quality which again results in a reduction of resources spent on repairing software.

Two programmers working together are more likely to come up with a more future-oriented solution. It is also shown that programmers enjoy their work more while working in pairs, and provides better design. Since knowledge is constantly shared between the developers it is also likely that the learning outcome is bigger than when working alone(37).

2.3.5. D3: Design Driven Development

D3 is an agile approach to development. It adds another dimension to the software development by making innovation, design and usability central in the development process. D3 turns design practices into games to bring different people with different sets of skills together to make design decisions together(38). The games are put into five different categories(39):

Startup: These are games designed to motivate the team for the project.

Understand: These are games developed to make the team understand the existing system and its problems by exploring the domain, business, customer, users, and context.

Question: It is important to ask questions in order to be innovative. The games in this category are designed to make the team question the existing business models, systems and processes.

Design: Explore some of the elements of design, for example interaction and information.

Experience: Design is not a one time project, but a process of improvement and refinement.

D3 is an agile development method that resembles Scrum and Extreme programming with its iterative ways, but instead of bringing agility to overall management, estimation, planning, or engineering activities, D3 brings agility to solution design and innovation(40). See Figure 2.3 for a visual representation of how D3 functions as an agile development method.

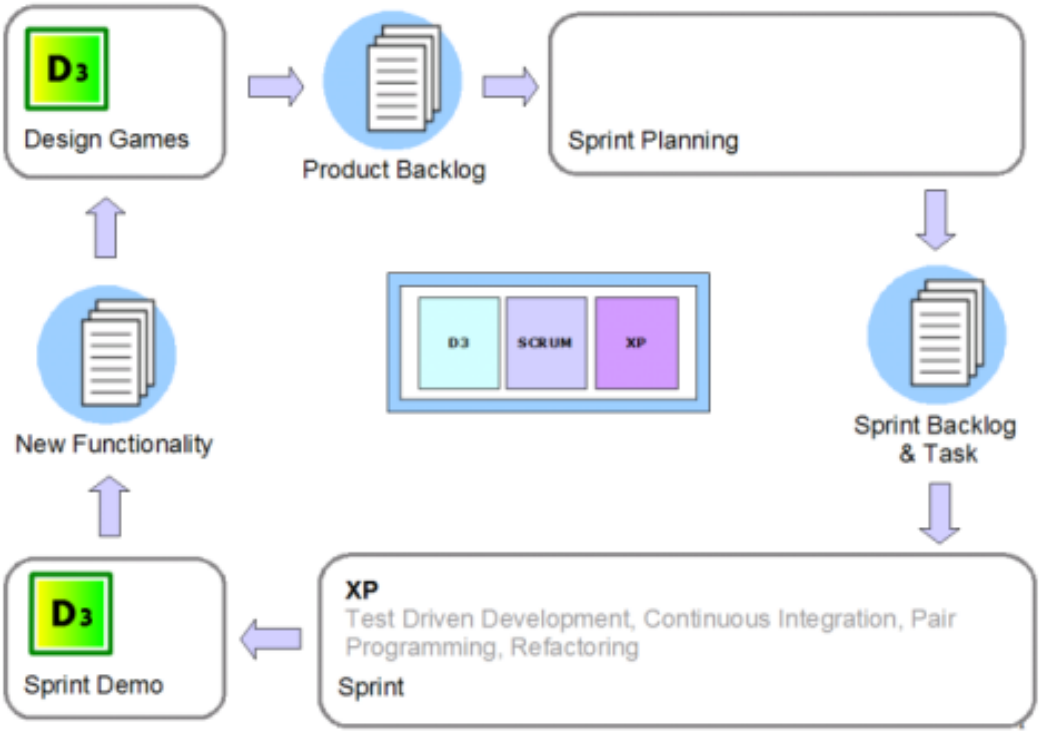


Figure 2.3.: The D3 method

2.3.6. Extreme programming

Extreme programming (XP) is an agile development method that is defined as a set of rules and values. The values are simplicity, communication, feedback, respect, and courage. XP is also characterized by it's incremental and iterative development in which there is a new version/release on every iteration(41).

2.3.6.1. User stories

In the planning phase, the customer provides the developers with user stories. User stories consist of 3 sentences of text and briefly describe a feature. The description should avoid technical details such as technology and algorithms, but should contain enough information for the developers to roughly estimate the implementation time in terms of weeks. The user stories are also used to create test scenarios for GUI tests. When it is time to implement a user story, the development team contacts the customer to get a full description of the requirements (42).

2.3.6.2. Iterative development

The work is divided into about a dozen iterations of 1 to 3 weeks in length. The duration of the iterations is to be kept constant throughout the project because it makes measuring progress and planning simple and reliable. All the code needs unit tests, and all the unit tests must be passed before it can be released for customer approval. There are also rules for iteration planning, acceptance testing, changes in terms of architecture and time estimations (43).

2.3.6.3. Customer involvement

One of the biggest requirements for XP to work is customer presence in the project. Unlike Scrum, where the developers and customers meet at the end of each iteration, in XP, the customer is required to be present every day as every phase in the standard XP requires customer involvement(44). For this reason, XP might be more suitable when the customer is located close by and has the time and interest to be involved in such a process, than if the customer is busy, and is located far away.

3. Solution

After the brainstorming session, the group had to choose between the ideas presented in the prestudy chapter. Furthermore, the group had to decide on which technologies and frameworks they wanted to use. The group made several considerations about what ideas and features that should be included in the solution. This chapter explains the reasoning behind the choices and the solution of the project.

3.1. Choice of ideas

In considering the ideas for the application, certain realities about competence and estimation of workload and time limitations had to be addressed. In this section, the rationale behind the choices that were made is presented.

3.1.1. Augmented and virtual reality

Developing with augmented reality and virtual reality require knowledge about 3D modelling and programming in a 3D environment. It was agreed that the members of the group did not have enough knowledge or experience to make a decent application in this field. Augmented reality applications does not necessarily need any special equipment, but the group was not able to find any frameworks that made it possible to develop these kinds of applications. Development with these technologies were therefore dropped.

3.1.2. Exhibition guide and minigames

A guide through the exhibition described in chapter 2.1.6 was one of the ideas that seemed possible to develop within the time frame of the semester. The problem with this idea was that it focused on guiding the user through the museum, while providing no interactivity to the user. To add some interactivity, the idea was modified to having minigames in the application rather than a static guide. This was similar to the first idea of making a guide in that the goal was to activate certain elements within the museum. Since the target user group was children between the ages of 6-10 years old, creating an interactive application rather than an a guide seemed more appropriate as the group thought that children tend to focus more on interactivity and learning by doing.

From the groups perspective, dividing the labour would also be much easier if the goal was making several minigames than if it was making a single guide throughout the museum. It was agreed upon that it would be simple to keep track of the development process and estimate the remaining work, as well as having some new functionality for

every other week to present to the customer in the case of using Scrum as a development method. Based on this, the group decided to base the application on minigames with roots in different paths of the exhibition.

3.1.3. Beacons

Both Vitensenteret and the group agreed that using beacon technology would add an interesting aspect to the application. The idea here was that the beacon technology would help activating certain features or themes of the museum. It was therefore decided that the group would try to implement beacon functionality in the application.

One of the group members had a contact in Nordic Semiconductor(45), and got acquired some hardware for testing different beacon technology. There was discussed that in a final version, Vitensenteret could get the needed hardware from Nordic Semiconductor, in exchange for some advertising in the exhibition, and in the app.

3.1.4. Web technologies and frameworks

Due to the solution the group ended up with, some technologies proved incompatible with the project. This includes the technology for augmented reality and virtual reality. This subsection will explain the reasons why some technologies were chosen.

All types of native development was rejected mostly due to excluding a lot of users. Development for iOS was never a real option because of the required tools, but it was considered because of the high number of iOS users. This leaves the cross-platform development technologies. The options were Xamarin and Cordova to begin with. Xamarin was rejected because it would cost money to develop an application with Xamarin, unless the group chose to do it open source, which was not an option.

The group chose to develop the application with Cordova. Even though Cordova does not give the same performance as the other technologies, it was sufficient enough for the planned application. Cordova has a lot of plugins that enables different native functions, such as communication with beacons. Virtually all the group members had experience with web technologies, which is what Cordova uses to develop applications. This meant that more time could be used to write code instead of learning new technologies.

After deciding to go for Cordova, the group discovered that there are other frameworks that are based on Cordova that makes developing easier. One of these frameworks are Ionic. After discussing the pros and cons of using Ionic and Cordova versus only using Cordova, the group decided that they would rather spend some time learning to use Ionic, as it would potentially simplify the development process.

3.2. The application

This section contains more detailed information and reasoning about the solution that was chosen, including the different minigames and views available to the user. Practical information on how to install and use the application can be found in Appendix G (installation guide), and in Appendix F (user manual).

3.2.1. Idea

Before deciding upon what minigames and features that should be included in the application, the group considered an underlying theme and story that would make the app more friendly towards children. In collaboration with Vitensenteret, the group decided that the story should be about Rob and Otto and their desire to build their new robot friend. Rob and Otto are both robots and have gathered some robot-parts for their robot, but some bad guys have stolen them. The user is presented with this dilemma, and is encouraged to solve the different minigames to retrieve the lost parts to build Rob and Otto's new friend.

3.2.2. Rewards

In order to motivate the user to complete the game, a reward system was planned for the application. It was decided that upon finishing a game, the user should receive an instant reward in the form of a robot-part.

3.2.3. Overview screen

It was first planned that walking into the different rooms in the museum would trigger the corresponding minigame to start on the app. This should be achieved using beacon technology. However, for users that have not enabled Bluetooth, having an overview screen would provide certain features that the group deemed necessary. The intention of making an overview screen was to provide the user with an overview of the progress of the game, and a selection screen where the user could start any of the minigames.

The other option would be having the games be played sequentially, with each game being played in a certain order. Since the visitors at the museum are able to move between rooms in any order, it would be desirable that could be played in the order that the visitor wanted as well.

3.2.4. Map screen

The intention of the map screen was to make Vitensenteret's map more available and to let the user know where the user is in the museum by using beacon technology, as well as informing the user about what rooms that remains to be visited using some sort of highlighting.

3.2.5. Robot customization screen

The idea of having a screen with all the currently gathered robot parts originated from the reward system. To gain a robot part would not be as satisfiable if you weren't able to view them together. To increase the motivation for gathering robot parts, the group decided that the user should be able to customize the robot by having a selection of parts to chose from. It was later decided that the user should be able to adjust the color and brightness of the individual parts as well, using sliders.

3.2.6. Quiz

Throughout the exhibition there are notes on hanging on the walls. On each of these notes there is a question. To view the answer of a question, the visitor has to lift the note up, and the answer is written on the wall. In order to activate these notes via the app, the group decided to include a quiz as one of the minigames. On the quiz, some of exact same questions are displayed, and the user is incentivized to find the notes in the museum and read the attached answer.

3.2.7. Pipes

The pipes game was intended to fit the theme of a room in Vitensenteret about the science of water and hydraulic power technology. The plan for the game was to have a grid with randomly turned pipes, a water-source and a turbine. To complete the game, the user would have to turn the tubes such that the tubes connect the water-source and the turbine. It was debated whether the tube-grid should be made using fixed tube-locations or by being generated using an algorithm. Because of simplicity and time restrictions, the group decided to use fixed locations for the different types of tubes. It was later decided that there should be three levels. After completing all the levels, the user would receive a reward. This was to extend the duration of a rather simple game, and to have the difficulty increase with each level.

3.2.8. Simon says

The Simon says game was a response to the part of the museum dedicated to the human body. This room contains, among other items, an interactive item for measuring reaction time, and a plastic version of the human body where one can reassemble the stomach. The group found that it would be a good idea to have a memory game because it was fitting to the theme and the purpose of this room. Ideally, the game would have some sort memory test in which the user had to remember the order of an occurrence, and repeat it into the app. It was decided that the memory game should have 3 levels, with each level getting slightly harder.

3.2.9. Elements

Vitensenteret has a big shelf displaying the table of elements. Each of the elements has its own space, where a picture or an actual piece of material is displayed inside a little

box with the front side protected by a layer of glass. The elements-game was intended to activate this part of the museum by providing incentives to read the descriptions and look at the items at this shelf. The plan for the game was to have a picture of an element or an item, and have the user choose the corresponding element that the item mostly consists of. It was decided that there should be nine fixed alternatives that sticks to the screen, with a new item appearing on top of the screen after the user answers a question. If the user would answer wrong, the next item would appear as normal, but the item that was answered wrong would be placed in the back of the queue. The user would ultimately have to answer all the questions correctly to finish the game.

3.2.10. Color lock

In the museum there is a section dedicated to colors, light, and mirrors. One of the interactive items in the room is about making a color based on three sliders that resembles mixing red green and blue. The color lock game was supposed to take this a step further by adding additional levels and confirmation when the user would get the answer right. The idea was that the user should be presented with a color that he/she has to create using three sliders. After creating the correct color within a reasonable margin, the user would have the option to "Unlock" and progress to the next level, hence the name the game - color lock.

3.2.11. Shortest-path

Vitensenteret has a room containing several puzzles, one of which is a shortest-path problem. This puzzle is about finding a path between 10 different cities in Europe in an order that minimizes the length of the path to the minimum, but the puzzle gives no signals on whether the user has found the correct answer. The shortest-path minigame was meant to help the user confirm that his answer was right; the puzzle exists in the museum, but the answers are filled into the app.

3.2.12. Melody game

One of the rooms in Vitensenteret contains an installation consisting of 5 steel pipes, a spinning metal arm, on top of a plastic circle with three different shapes printed on it (triangle, rectangle and pentagon). The pipes are different height and thickness, which means they make a certain sound when the spinning arm hits them in a circular clockwise motion. The minigame provides the user with a sound recording of the pipes placed in a particular order being hit by the arm. It is the visitors task to recreate the sound by placing the pipes in the same order as the recording. The game has three different stages, one with three pipes, placed on the triangle, one with four pipes placed on the rectangle, and the last one with all five pipes placed on the pentagon.

4. Project Management

The following chapter goes into detail about the project management and the decisions that were made. This includes decisions regarding what development methods to use, responsibilities, communication, planning, and risk analysis.

4.1. Development method

In order to find a suitable development method that fit both the customer, the group, and the task at hand, research on different development methods was required. This chapter will cover the discussion considering the different approaches regarding project management.

4.1.1. Considerations

The team suspected that changes in the requirements might happen during the project. With customer meetings every other week, the group had to be open for different kinds of changes and feedback. Therefore the group had to choose a development method that could handle sudden changes in the requirements.

4.1.2. Reasoning

Based on the prestudy, described in Chapter 2, in addition to the considerations described above, the team made some decisions about the development method.
The team agreed on the fact that an agile development method was preferred in order to be prepared for changes in the requirements. Because of this, the Waterfall model was not an option for this project. In the prestudy chapter, extreme programming (XP) was described as an agile method. One of the requirements for XP is that the customer must be available, preferably every work day of the development process. Since the team already had agreed upon meeting every other week with Vitensenteret, and since the team would most likely have to ask for a workplace in their building in order to use this development process, such customer involvement was not possible. The team also wanted to have the possibility of working individually or in pairs, either at home or with the rest of the team. Therefore XP as development method was discarded.

The group found that Scrum as the development method was a better idea for this project. With the agile properties and the iterative work process scrum has, it fit very good into the project. Since the requirements were not clearly defined in advance, the group needed to have a good dialog with the customer. The close communication with

the customer was considered essential in order to make an application that satisfied Vitensenterets wishes. Pair programming was also widely used. This was done in order to share knowledge and to hopefully achieve the advantages of Pair Programming listed in chapter 2.3.4.

4.1.2.1. Adaptations

In order for scrum to work as good as possible for the group, a few adaptions were made. Since the meetings with the customer were every other week, the duration of a sprint was set to two weeks. The sprint ended right before the customer meeting, and a new one started the day after. This was done in order to be able to present new functionality to the customer, or discuss changes at every meeting. Since all the team members took two other courses in parallel to this project, it was decided to have daily scrum meetings four times a week, instead of every day. Instead of having daily stand-up meetings, an activity plan was made using cards on Trello. The activity plan kept track on the status of the tasks. This includes a description of the task, a label indicating the progress, and names of the group members involved in the task. The activity plan also replaced the standard scrum board one often finds in scrum projects.

4.2. Team organization

It is important to have a structured and well organized team. This section contains information about the different aspects of team organization, and how the group organized themselves.

4.2.1. Agreement of cooperation

All the team members agreed on the fact that a good grade was something to aim at. The goal was a top grade. In order to easier achieve that, a contract was made. The contract set the terms for how the group was going to work; when the group meetings should be, what was considered valid absence, how many hours each member was expected to spend on the project, and etc. The contract was made in order to minimize the extent of conflicts, disagreements and misunderstandings. The contract was written early in the period and signed by everyone right after. The contract can be found in Appendix E

4.2.2. Team roles

Traditionally a team consists of members with specific and clear roles. These include, but are not limited to group leader, architect, designer, developer, and tester. Seeing this as a student project, the members' roles are not as clear because all members wanted to obtain an understanding of 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 were distributed to the team members that were best equipped

within the different areas of expertise. The team found that it was important to divide the different roles between the group members, so that everyone would know who to contact if they needed help.

Group leader: The group leader was responsible for contacting the customer and the supervisor. Also to ensure project progress and to meet deadlines. The group leader was the chairman of the group meetings.

System architect: The system architect was responsible for making any decisions regarding the architecture of the software. In the beginning of a project, the system architect set up the basic architecture so that the team could start coding.
Designer: The designer was responsible for making a consistent graphical layout for the application, and to make the application intuitive to use.

Tester: The tester had responsibilities regarding the testing done in the project. This included writing tests, making sure the other group members wrote tests, and conducted user tests and acceptance tests with the customer.

Backend developer: Main responsibility was to develop the backend for the project. Although this was a rather small task, it was important to have someone in charge for this task.

Frontend developer: The frontend developer was responsible for the development of the frontend. He was responsible for making the idea and design into code.

4.2.2.1. Role distribution

This was how the roles were distributed in the proejct.

Group leader: Didrik Pemmer Aalen

Architect: Bjor Løyning Byrkjedal

Designer: David André Årthun Bakke

Test leader: Kristian Svoren

Backend developer: Nicolas Almagro Tonne

Frontend developer: Oscar Conrad

4.2.3. Communication

It was important to choose communication tools that worked for all members of the group. The group decided to use Facebook as the main source of communication inside the group. In the beginning both Facebook groups and Facebook chat was utilized by the group, but as time went by, the usage of the Facebook group declined. This was because of the simplicity of using the Facebook chat.

The main channel for communication with both the customer and the supervisor was through email. This was also used for some communication within the group, e.g. when information from the supervisor or the customer had to be spread to the rest of the group.

In addition to communicating with the customer per email, there were biweekly meetings with them. When communicating with Vitensenteret, the group was in contact with five different people. The director, Arnfinn Stendal Rokne, the IT manager Markus Nygård, the educator, Rannvei Sæther, the designer, Svein Erling Lode and Martin Kulhawczuk, head of communication.

4.3. Project planning

Planning a project consists of dividing bigger tasks into smaller tasks, making an overview over all the tasks in the project, estimating the time of the project as a whole, and estimating how much time every small task will take.

4.3.1. Work breakdown structure

A work breakdown structure (WBS) is hierarchical decomposition of the work to be done in a project. It is composed of two elements: the WBS tree and a table which defines the chunks of work to be done (46)(47).

In the beginning of the project, after deciding what idea to use, the group tried to break the project into packages. This resulted in the work breakdown structure tree in Figure 4.1. Each part in the work breakdown structure tree is described in detail in Table 4.1.

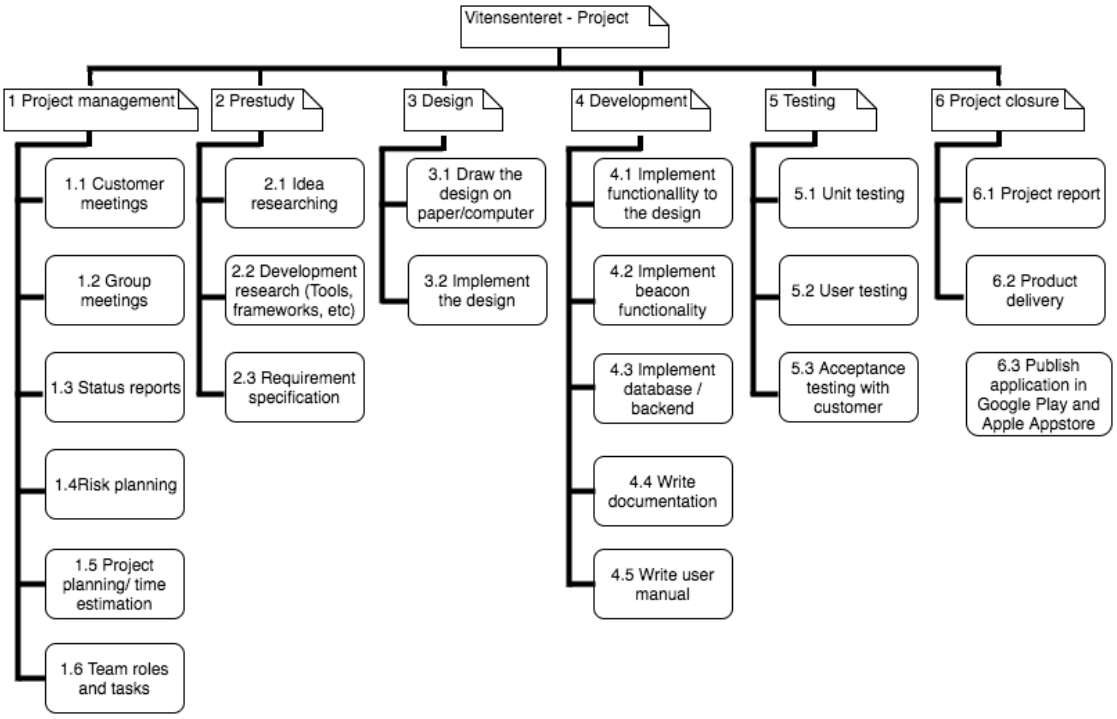


Figure 4.1.: Work Breakdown Structure tree

Table 4.1.: Work breakdown structure

WBS code	**Name**	**Definition**
	Vitensenteret - Application	All the work done in order to create the application that Vitensenteret wanted for both iOS and Android.
1	Project management	All the work done with planning and managing the project.
1.1	Customer meetings	Meetings with the customer to show project status and discuss ideas, requirements and changes.
1.2	Group meetings	Group meetings, have discussions about the project, and distribute work and work together with the project.
1.3	Status reports	Writing status reports.
1.4	Risk planning	Defining the risk for the project and analysing them.
1.5	Time estimation	Estimate the time needed to complete the different tasks.
1.6	Team roles	Distribute roles to the team members.
2	Prestudy	The work done in order to gather information about the project.
2.1	Idea researching	The work done with coming up with ideas for the application.
2.2	Development research (Tools, frameworks, etc)	The work done to find the right tools and framework to develop the application with.
2.3	Requirement specification	Defining the requirements for the application and making use cases for the different requirements.
3	Design	The total workload for designing the application.

Continues on the next page.

Continued from the previous page.

WBS code	Name	Definition
3.1	Sketching the design on paper / computer	The initial design process. Make the design for the application before implementing it.
3.2	Implement the design	Write code for the application design.
4	Development	Write the code for the functionality for the application, as well as writing documentation.
4.1	Implement functionality to the design	Create logic for the implemented design, so that the application works.
4.2	Implement beacon functionality	Make the application work with beacons.
4.3	Implement database / backend functionality	Create the database that holds the robots in the application.
4.4	Write Documentation	Write installation guide and user manual.
5	Testing	All the work put into testing the application.
5.1	Unit testing	Test every part of the application separately.
5.2	User testing	Let people in the targeted group test the application and come with feedback.
5.3	System testing	Test the system when all units are combined.
5.4	Acceptance testing with Vitensenteret	Let Vitensenteret try the application and come with feedback.
6	Project closure	The work put into completing the project.
6.1	Project report	Finish the final report.
6.2	Product delivery	Deliver the product to the customer and the course staff.
6.3	Publish application in Google Play and Apple Appstore	Publish the application so that it can be used.

4.3.2. Time estimation

An important aspect of project planning is time estimation. This includes everything from estimating the project as a whole to estimating the smaller tasks in the project. The group started out with making a GANTT-diagram based on the work breakdown structure.

4.3.2.1. GANTT

The group tried to estimate how much time each of the packages would take in the project. This resulted in the GANTT-diagram in Figure 4.2. See Appendix I.2 for a full-sized GANTT diagram.

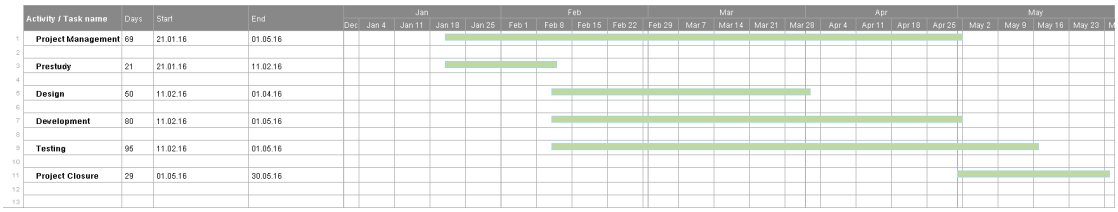


Figure 4.2.: GANTT diagram.

4.3.2.2. Task estimation

After estimating the project as a whole, the group tried to break the design and development process into smaller and more manageable tasks. The group decided to use planning poker to estimate the smaller tasks. Planning poker was a consensus based, gamified technique for estimating workload in terms of hours. The initial estimations made with planning poker can be seen in Table 4.2. In this estimation the group tried to find all the tasks linked to design and development in the project, but there were some additions and deletions of tasks later in the process. The tasks the group did not feel confident in estimating, because of the lack of experience with beacon technology. The design of the different robot-parts were also hard to estimate since the task was done by Vitensenteret.

The first activity plan the group created was based on the estimations made with planning poker, but this was the only activity plan made based on these estimations. After finishing the first activity plan, the group discovered that the initial estimations done with planning poker were not accurate, and therefore conducted estimations based on experience, which gave more precise results.

After the first activity plan was finished, the group was more experienced with the

Table 4.2.: Initial estimation made with planning poker

task	**Estimation in hours**
Learn technologies (Self study) (Cordova)	32
Design general layout	18
implement functionality for the general layout	20
Design color-minigame layout	24
implement functionality for color-minigame	5
Design layout for quiz-minigame	10
implement functionality for quiz-minigame	40
Design layout for shortest path-minigame	10
implement functionality for shortest path-minigame	16
Design layout for table of elements-minigame	10
implement functionality for table of elements-minigames	7
Design layout for water-minigame	24
implement functionality for water-minigame	23
Design layout for memory-minigame	16
implement functionality for memory-minigame	18
Design layout for sound-minigame	25
implement functionality 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 super classes with possibilities for inheritance	35
Testing	110
Total	512

tools and frameworks, thus making the group more fit to make estimations based on experience. See Appendix I.3 for an example of an activity plan used in the project.

4.4. Risk analysis

Table 4.3 shows the risks identified by the group, and the likelihood, impact and importance of the specified risks to happen. Likelihood reflects how likely the group thought the risk would be during the project period. Impact reflects how high the group considers the consequences of the risk. Importance is found by multiplying likelihood and impact. The higher number importance get, the more important is it to avoid the risk.

Table 4.3.: Risk analysis

Description	**Likelihood**	**Impact**	**Importance**	**Preventive action**	**Remedial action**
Misestimation of work/time	7	5	35	Read about time estimation	Continuous re-estimation of workload and open dialogue with the customer. Keep the possibility to simplify the project
Loss of work or data	7	5	35	Save and do backups. Use GitHub often.	Try to recover lost data and/or work.
Serious illness	3	8	24	Be safe	Speak with customer and try to simplify the project, as well as redistribute work task between remaining members

Continues on the next page.

Continued from the previous page.

Description	Likelihood	Impact	Importance	Preventive action	Remedial action
Problems working with new technology	6	4	24	Read documentation and prepare well.	Trial and error, discarding the technology. Use technology the group is familiar with.
Team members not contributing	3	8	24	Agree on a contract that the group members all have to sign.	contact regarding team member and the course leader.
Change in project requirements	3	7	21	Open dialogue with the customer and to make a good plan.	Adapt to changes
Problems with the customer	1	8	8	Keep in contact with the customer.	Contact our supervisor, and try to contact the customer.

4.4.1. Changes

As time goes by in a project, it is possible to see that other risks were at hand, and change the risk analysis accordingly. The changes that had to be done for this project involves team members being ill over time and thus not able to contribute to the project, team members going away for a while, making team meetings and some communication hard to perform. See Table 4.4 for the added risks to the analysis.

Table 4.4.: Added risks to the risk analysis

Description	**Likelihood**	**Impact**	**Importance**	**Preventive action**	**Remedial action**
Less serious illness	9	6	54	Try to stay healthy and avoid contagious scenarios.	Work from home. Focus on good communication. Help each other with the workload.
Change in the requirements	6	7	42	Have an active conversion with the customer.	Try to adapt to the changes. If time makes it impossible to implement the changes, tell the customer that it is not possible.

5. Requirements

Requirements are descriptions of features and functionality of a system. Requirements are divided into two categories, functional requirements and non-functional requirements. From the customer's point of view, requirements can be obvious, known, hidden, or unexpected, and it is the developers job to make sure that they are met.

5.1. Functional requirements

The functional requirements describes how the application should behave, as well as the operations and activities the system should be able to perform. They often detail what the output of an action should be, given the input (48). As previously mentioned, Vitensenteret did not have a lot of requirements for the application. This meant the group had to develop them. When the group had decided what the application should be, and contain, requirements were made to cover most aspects of the functionality.

Most important functional requirements for the application are mentioned in Table 5.1. A more detailed description of these, as well as an example of extensive functional requirements for the minigames, can be found in Appendix A

Table 5.1.: List of functional requirements

Name of requirement	Description of requirement
Hints in minigames	The minigames should provide the user with relevant feedback and hints when necessary.
Quiz	The application should have a quiz
Color lock minigame	The application should have a color lock minigame
Shortest path minigame	The application should have a shortest path minigame that corresponds to the traveling salesman problem in the exhibition.
Elements minigame	There should be a minigame that makes the user engage with the table of elements in the exhibition.
Pipes minigame	The application should have a minigame that illustrate pipes in a watersystem.
Simon says minigame	The application should have a minigame that tests the users cognitive skills, by making the user repeat a pattern.
Language	The user should be given the opportunity to select between Norwegian and English.
Map	There should be a map available with the location of where in the exhibition the different minigames should be completed
Collect robot parts	Whenever a minigame is completed, the user should be rewarded with a new robot part until the robot is completed.
Edit robot parts	The user should be able to edit their robot. Both the combination of different parts and their colors.

5.1.1. Use cases

A widespread practice for capturing functional requirements are "use cases". A use case defines goal oriented interactions between the users and the system. Use cases describe a scenario from a users perspective with a specific goal in mind. When the goal is reached, the use case is completed. A complete set of use cases specifies all the different ways to use the system(49). Tables 5.2 through 5.5 are textual use-cases for the most important parts of the application. An example of detailed use cases for the minigame can be found in Appendix B

Table 5.2.: Use-case for choosing language

ID	1
Name	Choose language screen
Goal	User selects language and proceeds to welcoming screen
Actors	User
Preconditions	User starts game for first time or user deletes his robot.
Prerequisite	Game is installed and running
Main Flow	1. User presses on the desired language. 2. System changes to welcome screen
Alternative Flow	None
Parent Use-case	None
Child Use-case	Welcoming Screen

Table 5.3.: Use-case for welcome screen

ID	2
Name	Welcome screen
Goal	User receives welcoming and proceeds to overview screen
Actors	User
Preconditions	Welcoming screen is on display
Prerequisite	Game is installed and running
Main Flow	1. User presses anywhere on the screen. 2. Message appears on screen with alternatives to proceed or cancel 3. User choses to proceed. 4. System changes display and controller to overview screen
Alternative Flow	1. User presses anywhere on the screen. 2. Message appears on screen with alternatives to proceed or cancel 3. User chooses to cancel. 4. Message disappears and welcome screen is displayed
Parent Use-case	Choose language screen
Child Use-case	Overview screen

Table 5.4.: Use-case for selecting minigame from the overview Screen

ID	3
Name	Overview Screen
Goal	User selects game and selected game is started
Actors	User
Preconditions	Overview screen is displayed
Prerequisite	Game is installed and started
Main Flow	1. User selects desired game from the list of games with touch. 2. System prompts user with story and options to start or cancel 3. User presses start. 4. System changes to game view and controller.
Alternative Flow	1. User selects desired game from the list of games with touch. 2. System prompts user with story and options to start or cancel 3. User presses cancel. 4. System exits the prompt and returns to overview screen.
Parent Use-case	Welcoming screen
Child Use-case	All minigames.

Table 5.5.: Use-case for editing look and composition of robot.

ID	4
Name	Customize robot
Goal	Customize the look and composition of the robot.
Actors	User
Preconditions	User has collected parts and app is displaying the build robot screen
Prerequisite	Game is installed and running
Main Flow	1. User presses desired (unlocked) part. 2. System shows two sliders below the image 3. User adjusts hue and light slider 4. User presses the arrows next to the image of the part 5. User changes skin on the part.
Alternative Flow	1. User presses delete your robot and start again 2. System provides alternatives to user: Delete or cancel 3. User chooses delete 4. Select language screen appears and game is totally reset
Parent Use-case	Part won screen, any minigame or the overview.
Child Use-case	None

5.2. Non-functional requirements

Non functional requirements describe the systems attributes, such as security, scalability and maintainability. They can also describe restraints, restrictions, and architectural requirements(50).

Because the application was developed to be stored on the device without access to personal information there were no special security measures. Since the beacons only send signals to bluetooth devices, and are unable to receive signals from other devices, they were not considered a point of threat to the security.

Appendix C contains the non-functional requirements for the application.

5.3. Changes to requirements

Throughout the project some of the functional requirements had to be changed. Following is a description of the change in requirements, and what they were changed from.

The functional requirement "Map"(Table A.10) was originally stated: "The application should have a map that shows which rooms are finished, and which are not.", but was changed to only contain the locations of the different minigames. This was changed because the group realized that the time spent implementing this could be better spent elsewhere. A simple map with the locations of the minigames would be sufficient information for the user.

The application originally had a functional requirement called "Change elements in minigames", which was formulated as follows: "The owner of the application should have the possibility to change elements of some of the minigames". This was a requirement that was formulated when the group asked Vitensenteret if they wanted the possibility to change the questions in the quiz after the application had been published. The group later decided, in collaboration with Vitensenteret, that this was not necessary. Implementing this functionality would require a database with a custom interface. It would also require the user to be connected to the Internet in order to get the correct quiz questions on their phone. This would require a lot of time to implement, which the group did not have.

6. Tools

Throughout the development process the group has used many tools to solve different tasks. This chapter will explain which tools have been used for what tasks and why.

6.1. Designing tools

In order to create some design drafts, and diagrams, the group needed some graphical outlining tools. This section covers the tools that were used.

6.1.1. Draw.io

Draw.io is a free online tool, which can be used to create flowcharts, process diagrams, organisation charts, UML(Unified Modeling Language), ER-diagrams(Entity–relationship diagrams) and network diagrams(51).

6.1.2. Ionic Creator

Ionic Creator is a drag and drop tool for making and designing mobile applications. It uses pre-built components to quickly build and link pages together(52).

6.2. Development tools

During the development process many tools and frameworks have been used to solve different kinds of challenges.

6.2.1. IDEs and Text Editors

The group used several IDEs and Text editors for different uses, like coding, resolving conflicts and building the application.

6.2.1.1. JetBrains IDE

A text editor and IDE which can refract, automatically find reference errors, and resolve merge conflicts(53).

6.2.1.2. Sublime Text

A cross-platform text editor with good support for markup, and scripting languages. (54).

6.2.1.3. Xcode

Xcode is an IDE for developing iOS apps, and is needed to build the iOS version of a Cordova application(55).

6.2.1.4. Terminal

The terminal is a text-based tool for exploring, running and modifying files, programs and systems. It is critical when using Git and Ionic projects.

6.2.1.5. Chrome Remote Debugger

A remote debugger for android devices which can inspect connected phones running chrome, showing errors and debug information on apps. It is an important tool for debugging errors not produced when testing the app on a computer(56).

6.2.2. 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(57)(58).

6.2.3. Ionic Lab

Ionic Lab is a tool which makes it easy to test ones' application directly in the browser, with live reloading. It supports application testing on multiple screen sizes and platform types(59).

6.2.3.1. Gulp

Gulp is a tool used by Ionic Lab to detect changes in code and "compile" SCSS, JS and other files into a functioning project. The group modified the default Ionic gulp script so it would concatenate our app files as well(60).

6.2.3.2. Bower

Ionic uses Bower to manage its dependencies. For instance getting the latest versions of Angular. This automates and simplifies updating and installs the correct dependencies(61).

6.2.4. GitHub

For collaborating on the source code the group used GitHub. This allows the group to have version control over the codebase, which means if something goes wrong, or if there are conflicting commits, it is easy to roll back to a previous version of code(62).

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.

6.2.5. 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(63).

6.3. Project management tools

To manage the project and have control over all the different files, documents, and information that have been created during the project. It was important to organize everything in a way that made it easy for the whole team to access at the same time.

6.3.1. Trello

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.(64) The reason for picking Trello over other project management tools was that it was simple and easy to use, and had the necessary functionality that the group needed to manage different aspects of the project using Scrum.

6.3.2. Google Drive

Google has a broad set of collaborating tools, which the group used throughout the project. Google Drive was used to store the documents, photos and other files that were shared within the group.

Google Docs is one of the collaboration tools provided by Google Drive. It is a text editor that enables real time editing from multiple users at the same time. It was very useful when planning, writing status reports, and editing the preliminary report.

Google Sheets is another service provided by Google Drive. Sheets was used to manage the groups time sheet, so each person in the group could write down how many hours they had worked on the project each day.

6.3.3. Google Calendar

The group created a calendar to keep track of the internal meetings as well as the meetings with Vitensenteret and our supervisor. This calendar contained information about when and where to meet, and what kind of meeting it was.

6.3.4. LATEX

LATEX is a document preparation system that uses markup tagging to stylize the written document. It is a good way to structure documents, have interactive bibliographies, and easily create tables of contents(65).

6.3.5. Sharelatex

Sharelatex is a real-time online collaboration tool for editing and storing LATEX documents. Sharelatex was used to write the final report. This made it easy to collaborate on the report, and give the report a professional, and neat look.

In addition to being a collaborative report writing tool, Sharelatex allows storage of 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 (66).

7. Graphical design

The graphical user interface is what most users of a system interacts with. Therefore, it is important that a user interface is well designed, responsive and intuitive. With a good design, the users can work effectively on the system, and not be confused by the layout. In order to make a great design, the designer have to really think about all the aspects with the design: Who is the user of the system? What should the user interface provide to the user? What should the user interface communicate? These are some of the questions that will be answered in this chapter.

7.1. Planning of design

Although the customer gave the group a design manual, which included suggestions to what colors to use, guidelines for using their logo, and other design guidelines related to Vitensenteret, they gave the group the possibility to go beyond these guidelines. In addition, Vitensenteret had a few inputs on the target audience, which was going to be children between six and ten years. From this information, these assumptions were be made:

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

The group spent the first weeks designing, and improving their initial ideas. After some discussion within the group, some clear points of design had been made, based on user interface design basics. (67)

- The interface should be consistent, and use common user interface 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 was 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 precisely enough for the user to know what the next step would be.

- The system should give intuitive feedback on what is happening. If a button was pressed, or a setting was switched, the system should give some kind of feedback, so the user would know that the program had responded.

7.2. Designing the application

The first few weeks were used to create a design proposal for the application. The group decided to follow the guidelines discussed in the planning phase, and the design manual handed out in the beginning.

7.2.1. Sketches

The group discussed the outline of the different parts of the application, and made some rough sketches. When these sketches were done, they were shown to the customer, who they gave their approval. A good example of the initial design outline was the overview screen, which can be seen in figure 7.1. The rest of the design drafts can be seen in figure J.1 - J.6 in the appendix.

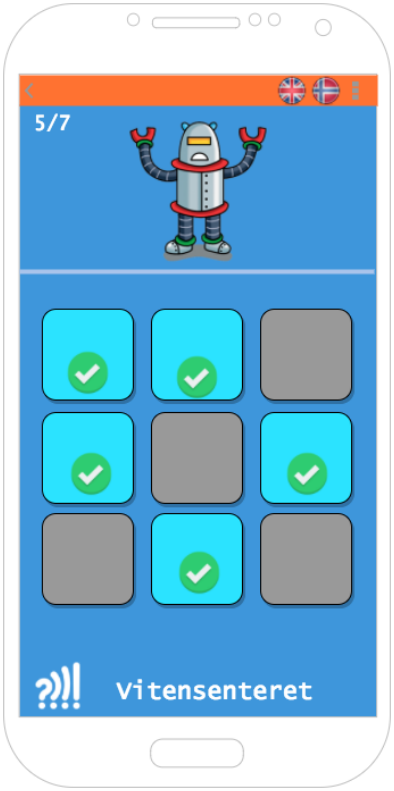


Figure 7.1.: Overview screen

7.2.2. Changes in design

Some changes were made in the final version to make the process more effective for the group. By using the built-in user interface elements from Ionic, the group could do the development of the graphical design faster than by designing all user interface elements, like pop-ups and information cards, from scratch. Some changes had to be made, but the idea, and principles remained the same as in the first drafts.

By going for a flat design, an increasingly popular (68) design approach, the application was following the evolution of popular design trends. This made the application up to date regarding design and seem more simplistic, while preserving the functionality.

Some changes were made when the design was implemented. The changes were based on removing shadows, and borders in order to make the application appear flatter. There was one major change to the overview design, as seen in Figure 7.1. Instead of having a view that combines robot parts, and all the minigames, this view was divided into two views. The first view was the overview-view, which gave an overview over all the minigames in the application, see Figure 7.2. The second view was the robot view where

one could customize the robot, as seen in figure 7.3.

The rest of the final design can be seen in figure J.7 - J.16 in the appendix.

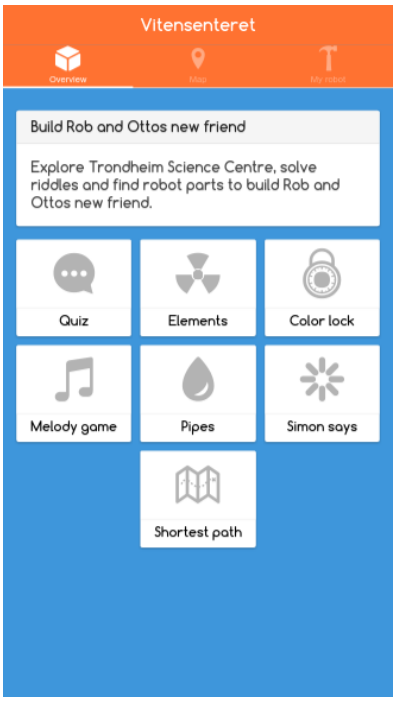


Figure 7.2.: Robot part main screen

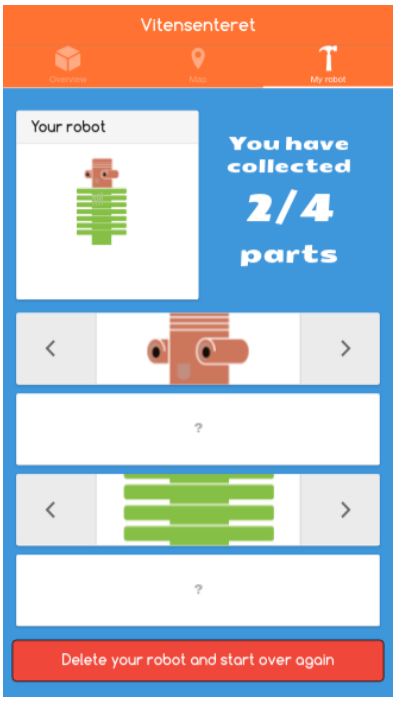


Figure 7.3.: Collected parts overview screen

7.3. Implementation of design

Design could be a challenge when creating mobile applications or other programs that needs an user interface. The design should engage the user and it should be intuitive(69). The group found some design principles, which could be used to create a clear, and tidy design.

7.3.1. Fonts

The application used two fonts: Comfortaa and Seymour One. These can be downloaded from Google Fonts (70), which was free to use. The font was childish, which aimed to engage the user base of the application.

The group strived to use font sizes in a way that made all messages given to the user seem clear. A large font size was used for headlines of messages, while other information

was written in a smaller font size. This created a clear hierarchy, which made it easy to read the information provided. Some information was written in bold text to emphasize its importance.

7.3.2. Colors

The colors used throughout the application were chosen from Vitensenterets color-palette. The main colors used were orange, blue, and green. The usage of colors were carefully thought through: Buttons with positive messages were green, while buttons negative messages were red. This was to give the buttons an intuitive color scheme. The color complements the text on the button, giving the user a visual representation of the action to be made. In order to give the user a visual feedback when pressing a button, the color of the button would turn darker darker when pressed.

7.3.3. Images

Some of the images used in the application were designed by Vitensenteret's designer. Those images not created by Vitensenteret's designer were downloaded from the internet, under the apache licence(71). This includes the flags in the language selection, the pipes used in the waterflow game, and the images in the periodic minigame.

7.3.4. Continuity

One of the most important design matters of a user interface, is continuity(69). Buttons, information boxes, pop-ups, and other commonly used elements were standardized across the application. If the user recognizes a pattern, it should be easy to get used to the application.

7.3.5. Logic

The group selected strong and complimentary colors from Vitensenteret's color pallette.An example of this are the primary light-blue and orange which are used as background and foreground colors. The buttons were made big and simple, so that the intended user group could easily press them.

8. System Design

When designing the system of the application the group had to adhere to the Ionic, and therefore Angular, way of system design. An application activity diagram was designed having the Ionic framework structure and quick access in mind. This can be seen in Figure I.1. This chapter describes the architectural model used for the application, as well as how the application and backend was implemented.

8.1. Architechtural model

Angular JS uses an architecture model known as Model View Whatever (MVW). This is a term used mainly by Google when describing the Angular JS architecture.(72) MVW is similar to Model View Controller (MVC) (73) and Model View ViewModel (MVVM) (74)but instead of having to use either a Controller or a ViewModel for application logic, one can use "Whatever works for you".

The system architecture the group chose to use was MVC. This is an architecture pattern found 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 seen in figure 8.1, The MVC architecture divides the system into three interconnected parts; Model, View and Controller. The user interacts with the view (HTML), which changes the data (Model). That action calls the controller, which in turn can modify the Model, and interact with Servers. AngularJS uses 2-way bindings to detect model changes and update the view.

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 interactive content is activated the signal and information goes directly to the controller, which then decides what happens.

The controller was the part of the application that communicated with databases and transmitted data between the modules of the application. It processed the data and sendt it to the model, which updated the view with the given data. The controller could communicate directly with the view as well. This was usually done when no information was changed, for instance in the case of scrolling; all the information was already in the view, but not on the screen.

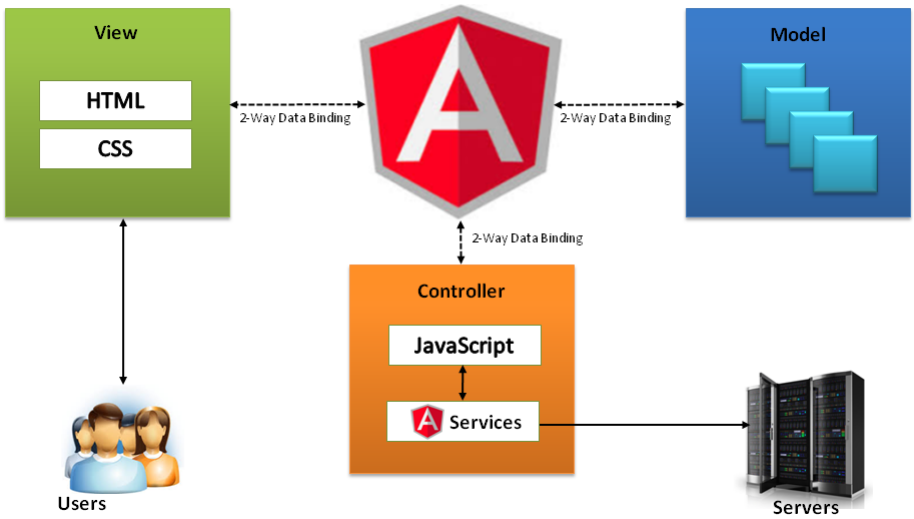


Figure 8.1.: The AngularJS MVC architecture.

The model was responsible for editing and updating the information in the view. It received data/information from the controller, stored it, and updated the data in the part of the view where it was supposed to go.

8.2. Implementation overview

The system was implemented as a web-application running with some native functionality on either iOS or Android through the Cordova framework (23) 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 used Angular JS, and Ionic Framework.

8.2.1. Organization

The project was organized first as a cordova application, with config files, plug-ins, and other application-building related files. The application's main project files are inside the "www" folder. Common files are organized by file-type, and files for the different games inside its own folder called "app".

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

The application has a central module, "routes.js", which decides which controller goes

with which view, and in what structure they are. It is here that the inheritance is defined.

When modified the application's JS files are concatenated into one big file, "apps.js", by Gulp. The Ionic SCSS files are compiled, and then linked in `index.html`, which is the first filed opened when starting the application.

8.2.2. User interface

The user interface views use inheritance to make reusable design possible. The design implementation 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, from where one can explore the sub views if they exist. This can be seen in fig 8.2.
When going to sub-pages, one's current position in the application is displayed at the top of the nav-bar, with a back button to take you back to the parent view.

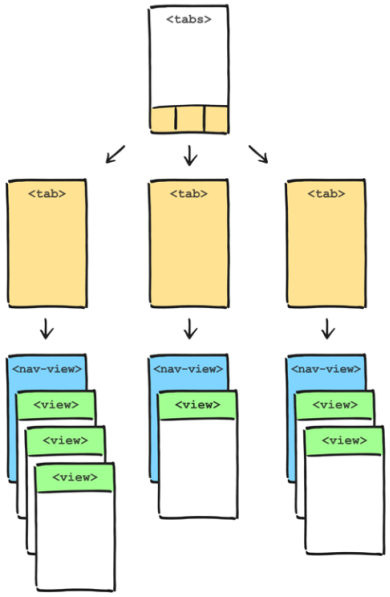


Figure 8.2.: View structure in the Ionic tabs layout

8.2.3. Frameworks

Cordova is the fundamental part of the system, in which Ionic is implemented on top of, as a facilitator of structure and with consistent design elements. Ionic uses AngularJS as a integral part of its workings and can almost be said to be a angular project written as a plug-in for Cordova (29). See Figure 8.3 for a visualization of this. When Ionic is run, Gulp compiles the SCSS code into normal CSS. The SCSS code gives control over the overall look of the app, without having to change values throughout the groups entire style code.

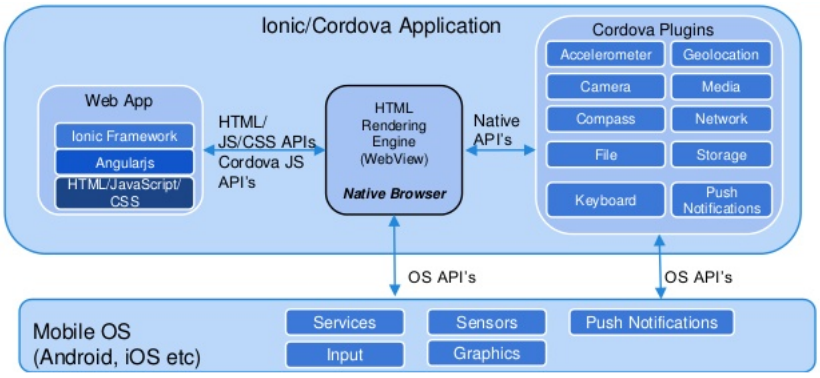


Figure 8.3.: Architecture of a normal Cordova application using Ionic

8.2.4. Controller

Each sub-app of the application, either a minigame or another part of the application, has its own controller. The controller in Angular is a JS file with a unique controller name defined, which is defined in the view. The group used different function and variable scopes for different needs. Variables needed in the view are defined on the $scope, and variables used in multiple controllers are defined on $rootScope.

8.2.5. Model

The group did not use a dedicated model in the traditional sense of the term, but used globally defined variables with data such as the game state and the robot part configuration. These global variables are stored and synced to the device's local storage, where they will be stored even after the user closes the application. Any changes to these variables will immediately be stored in the local storage to ensure that no data is lost. The implementation of this can be seen in "www/js/app.js".

The application also uses a separate file for storing the translations of the different views, located in "www/app/overview/translations.js". Due to the way Ionic concatenates files, the translations file had to be stored inside a sub-app's folder, not in a better suited location.

8.2.6. View

Each sub-app has its own view, a HTML document, which defines the structure and content which will be shown. Angular allowed the group to use dynamic elements in HTML, which usually can't be done in plain HTML. Examples of this include using "ng-repeat" which generates many similar elements from a scope variable.

8.2.7. Beacons

In order to get beacons to work with the application, the evothings-eddystone-plugin (24) was used. It is a simple framework for reading beacon data. The functions used to read beacon data is found in "overviewCtrl.js" located in "www/app/overview/OverviewCtrl.js". When the application starts, this file is initiated, and the application asks the user for permission to turn on Bluetooth. If permission is declined or bluetooth cannot be used, an error is thrown. This is handled by setting a boolean value "beaconsActive" to false, which makes the application use the "no beacons" mode. This means that the user can play all the minigames without them being unlocked by a beacon on a specific location. The hardware we used for beacons came pre-configured, so it just had to be turned on in order to start broadcasting.

8.3. Backend

When a user has collected all robot parts, the user has the possibility of naming and uploading the robot to Vitensenteret's servers, where it will be shown alongside other people's robots, like a hall of fame. See 8.4.

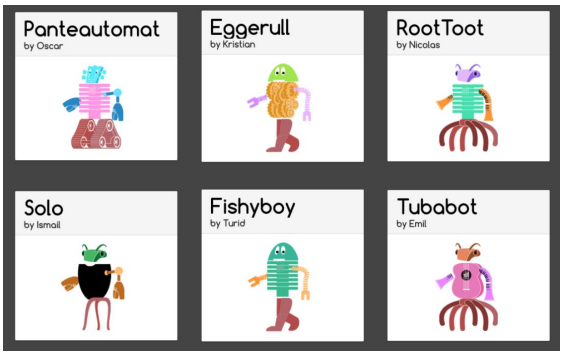


Figure 8.4.: Screenshot of the Robot Hall of Fame

The backend of the project is necessary to keep submitted robots in a central place. This was solved by having a simple script on Vitensenteret's servers, which take submitted robots, store them, and display them.

8.3.1. Implementation

The backend is a simple stand-alone Angular and PHP site, which can recieve new robot requesst from the mobile app, and display all existing robots.

When the app recieves a robot it gets the robot's name, creator's name and robot parts from the POST request the mobile app made. The robot parts are JSON encoded dictionaries, which the PHP script checks and stores.

When the server shows all robots, PHP is not used. The Angular app retrieves a JSON file with all the robots stored in it, imports it, and shows each robot using the exact same code (CSS, HTML and JS) that the mobile app uses, so that the robot is displayed consistently across different platforms. This also means that updates to the robot displaying code or design can be integrated into the backend quickly without having to adapt the code.

8.3.2. Storage and models

The robot model consists of the robot's name given by a user, the user's name and the robot parts selected and customized by a user. The parts object contains 5 dictionaries, one for each body part. Each body part contains the part's name, which variation on that part has been chosen and the hue and lightness of the part.

The data is stored in a JSON file by the PHP script running on the server. This storage method was chosen because the group had no guarantees of what the server environment would be like, or if the group had access to MYSQL or anything else. The format is sufficient for the group's simple use, and could be easily replaced with MySQL in the future, if there ever arises a need to supervise and manage the data.

9. Testing

Testing was a useful tool to ensure the quality of the final product. In an iterative project like this it was important to make sure the functional and non-functional requirements were being fulfilled throughout the entire project. Testing was used to show the customer that the product being created was the product they wanted, as well as getting rid of bugs and other issues.

The first section of this chapter describes the different kinds of testing which could be used. The second section describes how the tests have been implemented in the project. The third section presents the test results and the fourth section comes with a test conclusion.

9.1. Testing methods

When it comes to software testing, there are many ways this can be done. A common way is to run different kinds of test that checks if a line of code gives the expected output, and if different components work together the way they are supposed to and so on.

9.1.1. 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. One can then compare the results with the expected outcome (75)(76).

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

9.1.2. Graphical user interface testing

Graphical user interface testing is the process of testing if the system and interface meets its specifications and functional requirements. The GUI test cases are often developed by test designers. The designers should attempt to make the test cases cover the entirety of the functionality of the GUI (77).

9.1.3. User testing

User testing, also known as 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 (78)(79).

The user is given a scenario where the user has to complete a list of tasks that are part of the functionality of the application. The tasks are concrete, but not detailed on how to complete them. The tests are done to see if the user can navigate easily within the application and find what the user is looking for in order to complete the given task. After completing the tests the user is asked to report how they liked the product, any difficulties they might have encountered or any other comments about the experience. The observers time how long each individual task takes to finish, and other notable information (78).

9.1.4. 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 (80).

In this project integration testing was done by merging the code from the master branch in GitHub into a member's own code and test it thoroughly before pushing their own code up to the master branch. This was done in all the branches whenever something new had been added to the master branch, in order to make sure that the newly added code did not interfere with existing code.

9.1.5. System testing

System testing is the most extensive form of testing. This is where one would see whether or not all functional and non-functional requirements are met (81). System testing include stress-, and performance testing (82).

9.2. Test Plan

The following subsections describe how different tests were conducted throughout the project period.

9.2.1. Unit testing

As previously mentioned the application was being developed iteratively. This meant unit tests were being performed throughout the project. When writing code, unit tests were being implemented as well. The unit tests were small bits of code that wrote feedback to the console. By doing this, the developer could tell which parts of the code was

being run. This was also used as a form of debugging, to see where the code had crashed.

Figure 9.1 shows some of the unit tests implemented in the elements minigame. Figure 9.2 shows the console output for the tests in Figure 9.1. The first two lines in the console are from test completed earlier, when the user selected the language.

```
$scope.submitAnswer=function(answer) {
    console.log("Correct answer is: " +  $scope.buttons[$scope.nextElement.index].name);
    console.log("chosen Element: " + answer.name);
    var isCorrect = answer.correct;
    showPopup(isCorrect, answer);

    if (isCorrect) {
        $scope.buttons[$scope.nextElement.index].correct = false;
        urlAndArray.pop();
    }

    else {
        var oldElement = urlAndArray.pop();
        urlAndArray.unshift(oldElement);
        $scope.buttons[$scope.nextElement.index].correct = false;
    }
    console.log("remaining elements with status:")
    for (var i = 0; i < urlAndArray.length; i++) {
    console.log(urlAndArray[i].name + " " +  $scope.buttons[urlAndArray[i].index].correct + "\n");
    }
}
function initNextElement(){
    $scope.nextElement = urlAndArray[urlAndArray.length-1];
    $scope.buttons[$scope.nextElement.index].correct=true;
    console.log("New element initialized:" +  $scope.nextElement.name);
}
```

Figure 9.1.: Unit test for two functions

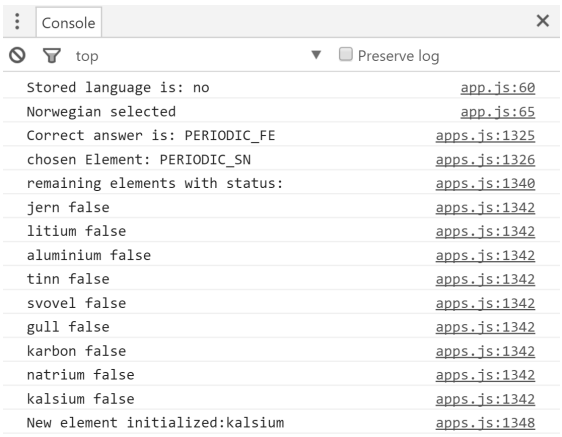


Figure 9.2.: Output from unit tests

9.2.2. Graphical user interface testing

In this project the GUI testing was done by the group. When a module was completed, it was tested by a member of the group who did not partake in the development of that module. When 5 of the minigames and the overview was finished these were uploaded to a couple of tablets owned by Vitensenteret, so they could perform their own tests and give the group feedback on the next meeting. The team was in contact with an educator employed at the Science Center with insight on necessities when it comes to communicating with children. This was a very useful resource for the group, considering that the main audience are families with children, ages six to ten.

Appendix D shows the detailed tests that were developed for the table of elements minigame. The same kinds of tests were developed and conducted on all modules of the application.

9.2.3. User testing

Initially the group was supposed to perform user tests towards the end of the project. This would not be in a laboratory, but in the exhibition, under close observation. The plan was to ask families if they were interested in testing our application. The group would follow them and take notes on what they said, how long the different games took to complete and ask them a couple of questions about the experience after they had finished. Time limitations prohibited this from happening. The product was not finished in time to complete the tests, and if it was completed, the group would not be able to change the application after the tests were completed.

9.2.4. Integration testing

The majority of the integration tests were 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 were done in order. One by one, the team members that had something to integrate merged the master branch into their own branch and tested if all of their functionality remained intact, before merging the changes back into the master branch. The reason for doing this and not merging everything to master directly was that this made it easier to figure out where and when an error occurred. The last step in this process was that the last person to integrate conducted a full GUI test in collaboration with the rest of the team.

9.2.5. System Testing

Because the application runs locally on the users device the application had no system to stress test. The only part of the system the group would consider necessary to stress

test was the database the application would send the robots to. This functionality was not implemented, so there was no stress testing.

In order to make sure the application could be run on different mobile devices it had to be tested on multiple devices and operating systems. The focus was on smart-phones, rather than tablets, because this was what Vitensenteret considered most important.

When running an application on different devices it is very important that it scales correctly with the screen size, resolution, and operating system. In the early phases of development the application was run in a web browser with a simulator called Ionic Lab, provided by the ionic framework. The simulator emulated how the application looked and behaved on both iOS and Android devices(9.3). This simulator was very useful in the beginning, but due to limitations, it was not used for long.

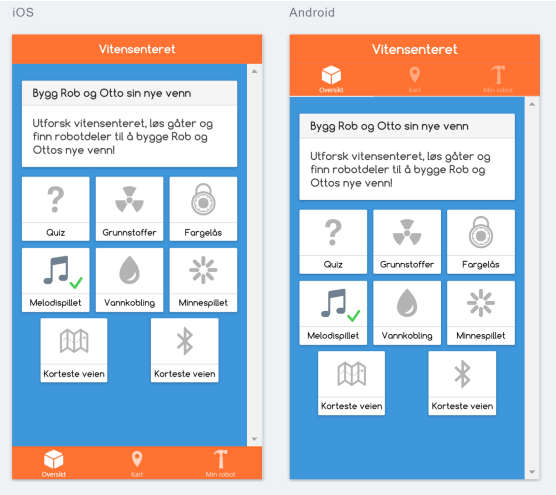


Figure 9.3.: iOS and Andoid simulation by Ionic Lab

Ionic Lab was replaced by running the application in “device mode” in Google Chrome. This provided the group with a simulator that could scale for a set of preset devices, as well as a responsive version, where one could enter the wanted screen resolution. This was used to test for the smallest and larges devices the application could be expected to run on. Figure 9.4 is a screenshot of two chrome browsers in device mode.

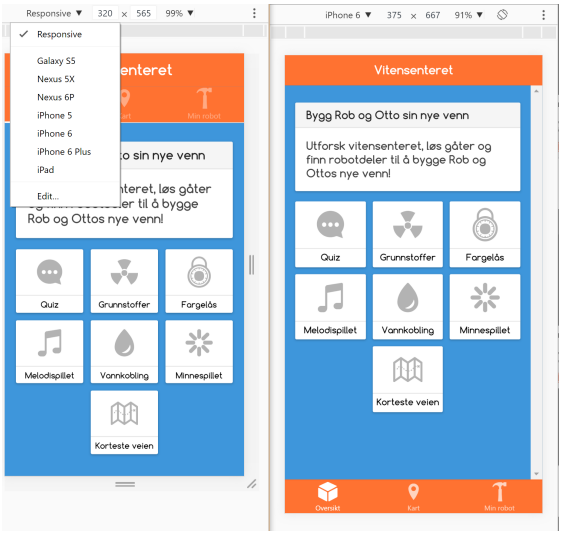


Figure 9.4.: Left: simulation showing the different preset resolutions of Google Chromes device mode. Right:Simulation for iPhone6

To test how the application performed on devices it was tested on different android smartphones from Samsung, LG and Sony. It was also run on an iPhone 4S. The early tests revealed some scaling issues, that were later fixed. Ideally it would be tested on a broader spectre of devices, but this was what the group had available.

9.3. Test Results

Not all test kinds of tests were relevant to our application. Below are a description of different test we conducted and what they resulted in.

9.3.1. Unit testing

Unit testing was a very useful tool to pinpoint errors and inconsistencies in the code. Every time a unit test failed the group could figure out what had gone wrong and fix it. By the time the application was finished, all the unit tests were running smoothly. The tests were later removed because the tests wrote to the console, something which the application should not do when it is run in production.

9.3.2. Integration testing

The group found that the integration method was working well. Considering the amount of changes that were made between the integrations there were not a considerable amount of problems. Occasionally the integration did not go as smooth as the group were hoping,

but this was because the merging branch had not been synchronized with the master branch in a long time.

9.3.3. System testing

We did not have the resources to test for all the different versions of Android. But the Android API and SDK that was used in the application (level 19) does not work for Android version lower than KitKat 4.4 (83). In may, 2016, more than 75% of the worldwide android users were on version 4.4 or higher (84). Because Norway is a highly technological country and earlier research indicate that Norwegians use the newer android verions (85) the group speculated that this percentage was higher in Norway, and that it would not be a problem.

9.4. Summary

The tests the group conducted on the application focused on making sure that all functional and non-functional requirements had been met, rather than focusing on usability and intuitiveness. The reason for this is the lack of time and resources. It was more important for the customer, and the group, that the delivered product was functioning properly than that it was super intuitive and fine tuned. Because of this some types of tests were given a lower priority in order for the group to spend time on getting the product to work properly.

The application now fulfills all the functional and non-functional requirements, but the group has concluded that there could have been a lot of feedback regarding intuitiveness and design from user testing and user acceptance testing if the group had the time to do them.

10. Further Development

Vitensenteret was very interested in having a completed and published application, and asked the group if they were interested in developing the application further during the summer. Several of the group's members were interested in seeing the project through, and were available during the summer. The persons in question would discuss terms and how it would be conducted further with Vitensenteret after the semester had ended.

This chapter describes the work which is left in the project and explains how it can be done. The sections described below are parts of the application the group would have focused more on, if given more time.

10.1. Eddystone beacons

In the final application, the group used Google's Eddystone standard for the beacons. It is a standard that works well the application, but it is not implemented to its full potential. Eddystone beacons can provide an URL, from which the user can download content. As long as the phone has the latest update of Google Chrome web browser installed, it is possible to get notifications for all Eddystone beacons nearby.

Since the application is written using web development languages, it could be ported to a website, and then be viewed on the phone, according to the beacon it connects to. It would then be easier to manage, and update the content of the application, which means that it does not have to be updated in every app-store it is released to.

10.2. Backend

The backend with the robot list was not ready for real-world deployment, as it did not have any moderation of submitted robots, or the necessary functionality for being shown on a large screen. It would need automatic scrolling and auto-refresh for showing the newest robots.

This was not continued as the system would need a moderator, or filter on the robot names, something Vitensenteret could not provide. It would be too complex to implement rules to filter out inappropriate content in the final days of the project.

10.3. Map screen

The map screen was not developed in conjunction with beacon technology. One of the ideas were to implement a sort of highlighting feature to add additional purpose to the map screen. A such feature would provide an overview over which rooms the user has and has not visited. This would be done by having a beacon in each room letting the application know if the user had entered, and for how long.

10.4. Rewards

Having a reward at the end of the game which gave the player something in return for playing the game, was a feature both the group and Vitensenteret wanted, but it was not part of the core functionality the group focused on. As the end of the project approached it became clear that it could not be done in time, and the idea was discarded.

If there had been more time for development, the group would have connected one of the beacons to a Tesla-coil Vitensenteret said could be used for a "robotic concert", so that a player who finished the game could activate it.

10.5. Realease

In order to release and launch the applications all the beacons have to be installed at the museum. Other small things needed before the launch is to create a more coherent storyline, create a nicer welcome-picture, and implement some reward. These were things Vitensenteret would work on, and come back to the group with during the summer. Then the application has to be published on Google Play and Apple's App Store.

Since Vitensenteret were given hardware in exchange for promoting Nordic Semiconductor, they have to get the Nordic logo visible in the exhibition, and it also have to be added to the application.

11. Evaluation

This chapter concludes the work on the final project and gives short summary of how the whole process has gone. In addition to explaining what kind of problems the group ran into on the way. Lastly it covers lessons the team has learned and what the learning outcome the course has been.

11.1. The assignment

The assignment the group initially given was as follows:

"Vitensenteret has made several pathways through the center where you can experience the exhibitions based on themes and interests. We would like to build these pathways into apps for smartphones and combine them with mini-games/ puzzles/ in-depth knowledge etc, that allows visitors to have new experiences in the science center and enrich the experience of the models om display. The pathways will be developed together with the student group."

With the little experience the group had with defining an assignment it was difficult to decide on a realistic and doable assignment, that both the customer and the group agreed upon. This gave the group a two weeks delayed start on planning and development.

11.2. Challenges

During the semester the group encountered many different challenges, other than what was first listed in the risk analysis in Table 4.3. Although our biggest problems were illness, injuries, absence and poor communication within the group and with the customer Were accounted for in the risk analysis, they were not accounted for to the correct extent.

Developing and testing applications for iOS turned out to be harder than expected. In order to test iOS applications Apple requires developers to use Xcode, which only runs on Apple's OS X. That fact that only a few of the team member owned an Apple computer made it hard. In order to publish an application on Apple's App Store a developers license is required, which cost approximately 100$

Due to the aforementioned difficulties with iOS, as well as having to have the application on the app store to be able to test it easily on iOS devices, the group made the

decision not to include a installation guide for iOS in the report. This does not mean that deploying the application for iOS is not possible, but that it required more time then the group had before the deadline.

The use and configuration, in addition to acquiring the beacons was also a complicated process. Initially Nordic Semiconductor provided the group with hardware compatible with the iBeacon standard. Unfortunately the group did not get iBeacons to work with the application. The group later acquired some hardware that was compatible with the Eddystone standard, which worked for the application. Unfortunately Nordic Semiconductor did not have enough beacons in stock for the group to get them up and running in the exhibition. By the time the group was given enough hardware, there was not enough time left to install them in the exhibition before the delivery deadline.

11.3. Decisions

During the project some decisions were made which benefited the group greatly. The first decision was to make use of the framework Ionic. At first the team encountered some problems setting it up and understanding how to use it. Looking back it is clear how it benefited us, and spared us for a lot of work. The second decision was to meet three more times a week, four times a week instead of one. The group quickly realized how much more, and better work was done when the whole group was working together. The last major decision that was made was to include beacons in the application, in order to make the application interact more with the exhibition.

11.4. Customer interaction

There were regular meetings with the customer every other Thursday. Each meeting represented the end of a sprint and was used by the group to present status and progress. These meetings were also used to discuss changes in the requirements and how to deal with the changes. In addition to biweekly meetings, communication with the customer was conducted by email.

Communicating by email was not always optimal, because it sometimes led to confusion and misunderstanding. The confusion would always be cleared on the meetings, but it would sometimes lead to delays. This could have been avoided by having customer meetings more often, but due to tight schedules on both sides, it proved hard to do.

The group was overall happy with the customer interaction, although there could have been some improvement. The customer was also happy with the interaction with the group, and how the project was carried out. See Appendix H for a written text from the customer regarding their experience with the group and the project.

11.5. Group interaction

The group started out with one mandatory group meeting per week, with the option to plan new meetings or work sessions to the week, if the group felt it was necessary. The mandatory meetings were orientation meetings where the group could have discussions about how to make progress and solve issues. This meeting was also used to plan sprints and what to be done. Although other meetings and work sessions could be, and were planned, the group realized early that it would be wise with additional mandatory meetings per week.This resulted in four mandatory meetings per week, where one meeting was the orientation meeting, and the three other meetings were workshop meetings.

Communication within the group proved to be harder than anticipated. Facebook was the chosen communication channel for the group, because it would give the members the opportunity to instantly get a hold of the other group members. This was not the case. There were some cases where Facebook did not work, because some team members did not answer when spoken to. To solve this, the group tried to contact the missing group members by phone. This did not always work either.

All members of the group were motivated to do well in this project, and it was agreed upon that each group member would use at least the 20 hours per week suggested by the course guidelines. The contract of cooperation that everyone signed in the very beginning made everyone commit fully to the project. The total workload conducted by the group was divided into sprints of two weeks. Table 11.1 shows the total workload per sprint conducted by the group.

Table 11.1.: Time spent working per two weeks

Weeks	Hours of work	Explanation
Week 4-5	76	Coming up with ideas for the application, and studying different technologies which could be used to realize the idea.
Week 6-7	83	Deciding on what technology to use and start designing the application. Learning the technologies that was decided to use.
Week 8-9	296	Finally starting development. Putting in some extra work because two thirds of the group were going to Japan for three weeks.
Week 10-11	132	Development in week 132, and two thirds of the group leaving for Japan in the end of week 10.
Week 12-13	37	Easter.
Week 14-15	302	Working more because of the Japan trip
Week 16-17	284	Last week of coding
Week 18-19	200	Writing report
Week 20-21	398	Writing report
Week 22	62	Finishing report.

11.6. Lessons learned

During a project like this there are many lessons the group has learned which the members can take with them into future projects. In retrospective one can see that assignment was too comprehensive, which made the workload bigger than expected. This combined with the communication troubles led to the fact that application was 100% complete before the deadline. The process of deciding on the idea and defining the requirement should have gone much faster.

In order to avoid misunderstandings and to speed up processes, all the people involved in the project form Vitensenteret's side should have been including in the customer meetings from the very beginning, not just the in the last few meetings.

When regards to testing, the group should have set a side more time for the testing phase. Even though different devices run the same operating system and same version does not mean that an application will behave the same way on the different devices. It would have been useful to test the application on broader specter of devices running both Android and iOS.

11.7. Conclusion

Initially, all requirements in the project were not met, but in cooperation with Vitensenteret, the group changed some of the requirement. This lead to all requirements in the project being met. The project management methods and software development methods that were applied made sure that the group maintained steady progress all the way towards the end. The group considers the project to be successfully complete and proud of the fact that Vitensenteret was both impressed and pleased with the end product.

List of Tables

List of Figures

Bibliography

[1] N.P., "It2901 - informatics project ii." http://www.ntnu.edu/studies/courses/IT2901#tab=omEmnet, 2016. [Online; accessed 28-May-2016].

[2] N.P., "Vitensenteret i trondheim." http://vitensenteret.com, 2016. [Online; accessed 28-May-2016].

[3] N.P., "Veiviser til vitensenteret." http://www.viten.ntnu.no/doc/veiviser.pdf, 2012. [Online; accessed 20-March-2016].

[4] N.P., "Science museum in london." http://www.sciencemuseum.org.uk, 2016. [Online; accessed 15-May-2016].

[5] N.P., "Science museum in london: Fiducial voice beacons." http://www.sciencemuseum.org.uk/online_science/apps/fiducial-voice-beacons, 2016. [Online; accessed 15-May-2016].

[6] N.P., "Science museum in london: Rugged rovers." http://www.sciencemuseum.org.uk/online_science/apps/rugged-rovers, 2016. [Online; accessed 15-May-2016].

[7] N.P., "American museum of natural history." http://www.amnh.org, 2016. [Online; accessed 15-May-2016].

[8] N.P., "American museum of natural history: Explorer." http://www.amnh.org/apps/explorer, 2016. [Online; accessed 15-May-2016].

[9] N.P., "What is virtual reality?." http://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html, 2016. [Online; accessed 28-May-2016].

[10] N.P., "How augmented reality works." http://computer.howstuffworks.com/augmented-reality.htm. [Online; accessed 16-May-2016].

[11] N.P., "Beacons: Everything you need to know.." http://www.pointrlabs.com/blog/beacons-everything-you-need-to-know/, 2016. [Online; accessed 16-May-2016].

[12] N.P., "Understanding the different types of ble beacons." https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/, 2015. [Online; accessed 16-May-2016].

[13] N.P., "About objective-c." https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html, 2014. [Online; accessed 16-May-2016].

[14] N.P., "About swift." https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/, 2016. [Online; accessed 16-May-2016].

[15] N.P., "Start developing ios apps." https://developer.apple.com/library/ios/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html?utm_source=statuscode&utm_medium=email, 2015. [Online; accessed 16-May-2016].

[16] N.P., "ios,android market share on mobile/tablet." https://www.netmarketshare.com/mobile-phones.aspx?qprid=9&qpcustomb=1&qpcustom=iOS,Android&qpcustomd=no&qpsp=2015&qpnp=2&qptimeframe=Y. [Statistics from between January 2015 to April 2016].

[17] G. Sims, "I want to develop android apps - what languages should i learn?." http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008/, 2016. [Online; accessed 16-May-2016].

[18] N.P., "Windows phone market share on mobile/tablet." https://www.netmarketshare.com/report.aspx?qprid=9&qpaf=&qpcustom=Windows+Phone&qpcustomb=1&qpcustomd=no. [Statistics from between January 2015 to April 2016].

[19] N.P., "Daydream." https://vr.google.com/daydream/, 2016. [Online; accessed 16-May-2016].

[20] N.P., "Welcome to vr at google." https://developers.google.com/vr/, 2016. [Online; accessed 16-May-2016].

[21] N.P., "Xamarin platform." https://www.xamarin.com/platform, 2016. [Online; accessed 16-May-2016].

[22] N.P., "Apache cordova overview." https://cordova.apache.org/docs/en/5.4.0/guide/overview/, 2016.

[23] T. A. S. Foundation, "Architectural overview of cordova platform." https://cordova.apache.org/docs/en/latest/guide/overview/, 2016. [Online; accessed 28-May-2016].

[24] Evothings, "Cordova/phonegap eddystone plugin." https://github.com/evothings/cordova-eddystone, 2016. [Online; accessed 29-April-2016].

[25] P. Metz, "cordova-plugin-ibeacon." https://github.com/petermetz/cordova-plugin-ibeacon, 2016. [Online; accessed 29-April-2016].

[26] N.P., "Cordova." https://en.wikipedia.org/wiki/Apache_Cordova, 2016. [Online; accessed 1-February-2016].

[27] N.P., "Ionic framework." http://ionicframework.com/, 2016. [Online; accessed 1-February-2016].

[28] N.P., "Ionic." http://ionicframework.com/docs/guide/preface.html, 2016. [Online; accessed 1-February-2016].

[29] N.P., "Ionic." http://ionicframework.com/docs/overview/, 2016. [Online; accessed 1-February-2016].

[30] N.P., "What is ibeacon? a guide to beacons." http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/, 2014. [Online; accessed 19-April-2016].

[31] N.P, "Mark up the world using beacons." https://developers.google.com/beacons/eddystone#full_support_for_eddystone, 2016. [Online; accessed 19-April-2016].

[32] A. L. Labs, "Scrum: A breathtakingly brief and agile introduction." http://www.agilelearninglabs.com/resources/scrum-introduction/, 2015. [Online; accessed 29-May-2016].

[33] M. Rouse, "sprint (software development)." http://searchsoftwarequality.techtarget.com/definition/Scrum-sprint, 2015. [Online; accessed 29-May-2016].

[34] I. E. CERTIFICATION, "What is waterfall model- advantages, disadvantages and when to use it?." http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/, 2016. [Online; accessed 13-May-2016].

[35] S. C. L. K. . S. Maquire, "information and communication tachnologies management in turbulent business environments." https://books.google.no/books?id=6ryRITfh-FMC&pg=PT134&lpg=PT134&dq=lean+7+principles+eliminate+waste+amplify+learning&source=bl&ots=PRbyoYwiZY&sig=B7NAI0Pw5yE2raQt71om_w8xQXc&hl=no&sa=X&ved=0ahUKEwjP0PTGxILNAhVIJpoKHW6rA9AQ6AEILjAC#v=onepage&q=lean%207%20principles%20eliminate%20waste%20amplify%20learning&f=false, 2009. [Online; accessed 28-May-2016].

[36] D. Wells, "Pair programming." http://www.extremeprogramming.org/rules/pair.html, 1997. [Online; accessed 10-May-2016].

[37] A. Cockburn, "The costs and benefits of pair programming." http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF, Unknown. [Online; accessed 10-May-2016].

[38] H. Jacob, "D3: Why what how." http://www.designdrivendevelopment.org/explore/whywhathow.html, 2007. [Online; accessed 16-May-2016].

[39] H. Jacob, "D3: Design games." http://www.designdrivendevelopment.org/explore/designgames.html, 2007. [Online; accessed 16-May-2016].

[40] H. Jacob, "D3 and agile." http://www.designdrivendevelopment.org/explore/d3agile.html, 2007. [Online; accessed 16-May-2016].

[41] D. Wells, "Extreme programming: A gentle introduction." http://www.extremeprogramming.org/, 2013. [Online; accessed 20-May-2016].

[42] D. Wells, "User stories." http://www.extremeprogramming.org/rules/userstories.html, 2013. [Online; accessed 25-May-2016].

[43] D. Wells, "Iterative development." http://www.extremeprogramming.org/rules/iterative.html/, 2013. [Online; accessed 20-May-2016].

[44] D. Wells, "The customer is always available." http://www.extremeprogramming.org/rules/customer.html, 2013. [Online; accessed 25-may-2016].

[45] N.P., "Nordic semiconductor." http://www.nordicsemi.com/, 2016. [Online; accessed 28-May-2016].

[46] N.P., "Work breakdown structure and gantt chart." http://www.me.umn.edu/courses/me4054/assignments/wbsgantt.html, 2011. [Online; accessed 16-May-2016].

[47] N.P., "Work breakdown structure (wbs)." http://www.workbreakdownstructure.com, 2014. [Online; accessed 16-May-2016].

[48] N.P., "Functional requirements." http://www.ofnisystems.com/services/validation/functional-requirements/. [Online; accessed 28-May-2016].

[49] R. Malan and D. Bredemeyer, "Functional requirements and use cases." http://www.bredemeyer.com/pdf_files/functreq.pdf, 2001. [Online; accessed 25-May-2016].

[50] U. Erikson, "Functional vs non functional requirements." http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/. [Online; accessed 15-March-2016].

[51] N.P., "Draw.io." https://www.draw.io, Unkown. [Online; accessed 9-March-2016].

[52] N.P., "The builder." http://ionic.io/products/creator/docs, Unkown. [Online; accessed 15-March-2016].

[53] JetBrains, "Webstorm - the smartest javascript ide." https://www.jetbrains.com/webstorm/, 2016. [Online; accessed 1-March-2016].

[54] J. Skinner, "Sublime text: The text editor you'll fall in love with." https://www.sublimetext.com, 2016. [Online; accessed 1-February-2016].

[55] Apple, "Xcode - what's new." https://developer.apple.com/xcode/, 2016. [Online; accessed 15-May-2016].

[56] Google, "Remote debugging android devices." https://developers.google.com/web/tools/chrome-devtools/debug/remote-debugging/remote-debugging, 2016. [Online; accessed 28-April-2016].

[57] N.P., "Angularjs." https://en.wikipedia.org/wiki/AngularJS, 2016. [Online; accessed 19-March-2016].

[58] N.P., "Angularjs." https://docs.angularjs.org/guide/introduction, 2016. [Online; accessed 19-March-2016].

[59] N.P., "Ionic lab." http://lab.ionic.io, 2016. [Online; accessed 10-March-2016].

[60] phated, "The streaming build system." https://github.com/gulpjs/gulp, 2016. [Online; accessed 30-March-2016].

[61] Bower, "Bower - a package manager for the web." http://bower.io, 2016. [Online; accessed 30-March-2016].

[62] N.P., "Github." https://github.com/, 2016. [Online; accessed 28-January-2016].

[63] N.P., "Nodejs." https://nodejs.org/en/, 2016. [Online; accessed 19-March-2016].

[64] N.P., "Trello." https://en.wikipedia.org/wiki/Trello, 2016. [Online; accessed 28-January-2016].

[65] N.P., "An introduction to latex." https://latex-project.org/intro.html, 2015.

[66] N.P., "Sharelatex." https://www.sharelatex.com/, 2016.

[67] N.P., "User interface design basics." http://www.usability.gov/what-and-why/user-interface-design.html, 2016.

[68] J. Cao, "6 web design trends you must know for 2015 & 2016." http://www.awwwards.com/6-web-design-trends-you-must-know-for-2015-2016.html, 2016. [Online; accessed 30-May-2016].

[69] W. O. Galitz, "The essential guide to user interface design: An introduction to gui design principles and techniques," 2007.

[70] N.P., "Google fonts." https://www.google.com/fonts, 2016. [Online; accessed 1-February-2016].

[71] N.P, "Apache license." http://www.apache.org/licenses/LICENSE-2.0, 2016. [Online; accessed 13-May-2016].

[72] I. Minar, "Mvc vs mvvm vs mvp." https://plus.google.com/+AngularJS/posts/aZNVhj355G2, 2016. [Online; accessed 28-May-2016].

[73] G. E. Krasner, S. T. Pope, *et al.*, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system." http://www.create.ucsb.edu/~stp/PostScript/mvc.pdf, 1988.

[74] Microsocft, "The mvvm pattern." https://msdn.microsoft.com/en-us/library/hh848246.aspx, 2012. [Online; accessed 28-May-2016].

[75] T. McFarlin, "The beginner's guide to unit testing: What is unit testing?." http://code.tutsplus.com/articles/the-beginners-guide-to-unit-testing-what-is-unit-testing--wp-25728, 2012. [Online; accessed 17-March-2016].

[76] N. P., "Unit testing." https://web.archive.org/web/20120429172731/http://guide.agilealliance.org/guide/unittest.html, 2011. [Online; accessed 20-May.2016].

[77] E. Horowitz and Z. Singhera, "A graphical user interface testing methodology." http://www.cs.usc.edu/assets/004/83254.pdf. [Online; accessed 25-March-2016].

[78] N. P., "Usability testing." http://www.usability.gov/how-to-and-tools/methods/usability-testing.html. [Online; accessed 20-March-2016].

[79] N. P., "Usability testing." http://www.usabilityfirst.com/usability-methods/usability-testing/. [Online; accessed 25-May-2016].

[80] N.P., "What is integration testing?." http://istqbexamcertification.com/what-is-integration-testing/. [Online; accessed 17-March-2016].

[81] N.P., "What is system testing." http://www.guru99.com/system-testing.html. [Online; accessed 15-March-2016].

[82] N.P., "System testing: What? why? & how?." http://www.softwaretestingclass.com/system-testing-what-why-how/. [Online; accessed 16-March-2016].

[83] N.P., "Android 4.4 apis." https://developer.android.com/about/versions/android-4.4.html. [Online; accessed 28-May-2016].

[84] N.P., "Dashboards." https://developer.android.com/about/dashboards/index.html?, 2016. [Online; accessed 28-May-2016].

[85] H. Otgard, "Plattformkrigen her er dataene - mobilstatistikk for mai." http://www.digi.no/kommentarer/2015/06/23/her-er-dataene---mobilstatistikk-for-mai-2015, 2015. [Online; accessed 28-May-2016].

Source of images

This section contains the sources for all the images not made by the group, but still used in the report.

Figure 2.1 in section 2.3:
N.P.. Web. May 18 2016
http://www.vanharen.net/blog/wp-content/uploads/2013/12/scrum.png

Figure 2.2 in section 2.3:
N.P.. Web. May 12 2016
https://upload.wikimedia.org/wikipedia/commons/thumb/e/e2/Waterfall_model.svg/350px-Waterfall_model.svg.png

Figure 2.3 in section 2.3:
N.P.. Web. May 16 2016
http://www.designdrivendevelopment.org/image/Agile_D3_small.png

Figure 8.1 in section 8.X
N.P.. Web. March 10 2016
https://avaldes.com/wp-content/uploads/2014/09/angularJS_MVC1.png?fcd5d2

Figure 8.2 in section 8.2:
N.P.. Web. March 10 2016
http://ionicframework.com/img/diagrams/tabs-nav-stack.png

Figure 8.3 in section 8.2:
N.P.. Web. March 10 2016
http://image.slidesharecdn.com/8dnenjjhq6iaqjxqfekm-signature-e311f9d5f5b8de7d3279f087d7
95/case-study-integrating-azure-with-google-app-engine-11-638.jpg?cb=1424309872

A. Functional requirements

A.1. Functional requirements

A.1.1. Overall functional requirements

Table A.1.: Functional requirement for hints in minigames

Name of functional requirement	Hints in minigames
Priority	High
Threat	If children get stuck without the possibility to get help or guidance they might give up finishing the game, and perhaps the other games as well.
Requirement	The game should provide guidance where necessary. If the user submits an answer that is partially correct, they should get some feedback about this. Or if the user get stuck in a minigame, they should have the possibility to get a hint that can help them finish the minigame.
Action	Feedback regarding the submitted answer, or hint-button that give the user hints.
Use case	1

Table A.2.: Functional requirement for quiz minigame

Name of functional requirement	Quiz
Priority	Medium
Threat	Visitors might not be motivated to read the questions already posted on the walls of the museum.
Requirement	There should be a quiz minigame. The questions in the quiz should be the same questions as the ones posted on the walls in Vitensenteret.
Action	Implement the quiz minigame
Use case	2

Table A.3.: Functional requirement for color lock minigame

Name of functional requirement	Color lock minigame
Priority	Medium
Requirement	There should be a minigame where the user is provided with a color and three sliders, representing red, green, and blue. When the sliders are moved the color changes accordingly. The user is supposed to replicate 5 color provided by the application.
Action	Implement Mimic the color lock minigame
Use case	2

Table A.4.: Functional requirement for shortest path minigame

Name of functional requirement	Shortest path minigame
Priority	Medium
Requirement	There should be a shortest path minigame. The game should contain the same cities as the installation in the exhibition. The order should be scrambled, and it is up to the user to drag an drop the cities into the right order
Action	Implement the shortest path minigame
Use case	2

Table A.5.: Functional requirement for elements minigame

Name of functional requirement	Elements minigame
Priority	Medium
Requirement	There should be a minigame with pictures of things that are mainly made from one element. The user should examine the table of elements in the exhibition to acquire information about what elements the thing on the picture is made of.
Action	Implement the elements minigame
Use case	2

Table A.6.: Functional requirement for pipes minigame

Name of functional requirement	Pipes minigame
Priority	Medium
Requirement	There should be minigame with a representation of a pipesystem. The user rotates the pipes 90 degrees by tapping them. The game is won by connecting a water source to a turbine.
Action	Implement the guide the water through the pipes minigame
Use case	2

Table A.7.: Functional requirement for melody minigame

Name of functional requirement	Melody minigame
Priority	Medium
Requirement	There should be a melody minigame. The game should provide the user with a sound sample, which the user is supposed to imitate by placing pipes on an installation in the exhibition in a pattern that recreates the sound.
Action	Implement the mimic the melody minigame
Use case	2

Table A.8.: Functional requirement for Simon says minigame

Name of functional requirement	Simon says minigame
Priority	Medium
Requirement	There should be a Simon says minigame that requires the user to repeat a series of patterns that are generated randomly.
Action	Implement the Simon says minigame
Use case	2

Table A.9.: Functional requirement for language in the application

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 vice versa.
Action	Implement a dictionary that contains both Norwegian and English texts.

Table A.10.: Functional requirement for the map

Name of functional requirement	Map
Priority	medium
Requirement	Because the minigames are somewhat connected to the exhibition there should be a map that tells the user where in the exhibition the different minigames should be completed
Action	Implement an easily available map with locations.
Use case	3

Table A.11.: Functional requirement for collecting parts

Name of functional requirement	Collect robot-parts
Priority	High
Requirement	For every completed minigame the user should be rewarded with a robot part.
Action	Ask Vitensenterets' designer to draw several modular parts that are easily interchangeable. Implement sliders to edit the hue and lightness of the parts.

Table A.12.: Functional requirement for editing robot parts

Name of functional requirement	Edit robot parts
Priority	Low
Requirement	In order for the user to be more invested in their robots, they should be able to edit their appearance by changing the combination of different parts, as well as the colors
Action	Implement rewarding "ceremony" after each completed minigame.

A.1.2. Detailed functional requirements for melody minigame

Following are the detailed functional requirements for the melody minigame. Similar requirements were developed for the other minigames, but have not been included in this report.

Table A.13.: Functional requirement for number of stages

Name of functional requirement	Melody number of stages
Priority	High
Requirement	Because the installation in the exhibit has 3 different shapes (triangle, rectangle and pentagon), the game should have the same number of stages

Table A.14.: Functional requirement sound

Name of functional requirement	Melody play sound
Priority	High
Requirement	There should be a button that can be pressed as many times as the user wishes that plays the sound to te corresponding stage. If the button is pressed again before the sound is finished playing, it should restart.

Table A.15.: Functional requirement selecting a pipe

Name of functional requirement	Melody select pipe
Priority	High
Requirement	The game should have correctly aligned number pickers. When the user selects one, they should be asked to select one of the pipes, represented by numbers.

Table A.16.: Functional requirement submitting wrong answer

Name of functional requirement	Melody submit wrong
Priority	Medium
Requirement	When submitting an answer the user should be prompted information regarding its correctness. If none of the pipes are placed correctly this should be communicated to the user. If some of the pipes are placed correctly, the user should be given information about how many and which.

Table A.17.: Functional requirement submitting correct answer

Name of functional requirement	Melody submit correct
Priority	Medium
Requirement	When submitting a correct answer the user should get feedback, and be able to move on to the next stage. If the game is completed they should be awarded their robot-part.

B. Use cases

B.1. Use cases

Below are detailed use-cases for the melody minigame. Similar use-cases were developed for all the minigames, but are not included in this report

Table B.1.: Use-case for playing sound

ID	Melody 1
Name	Play sound
Goal	User can hear the sound they are supposed to imitate
Actors	User
Preconditions	User has entered the minigame
Prerequisite	Game is installed and running
Main Flow	1. User presses button labeled “Play the melody!” 2. Sound plays until the end
Alternative Flow	1. User presses button labeled “Play the melody!” 2. Sound starts playing 3. User presses button again before the sound clip has finished 4. Sound restarts
Parent UC	Overview
Child UC	Next stage or part won screen.

Table B.2.: Use-case for playing sound

ID	Melody 2
Name	Select pipe
Goal	User inputs and submits their answer.
Actors	User
Preconditions	User has entered the minigame
Prerequisite	Game is installed and running
Main Flow	1. User presses a number picker surrounding the shape 2. The user is prompted to select a value and does so. 3. Repeat for as many number pickers as the user wishes to submit. 4. The user presses the button labeled “Test if it is correct!”, and submits the correct answer 5.1 If the user is on the final stage they are sent to the ”part won screen” 5.2 If the user is on stage on or two, the user is prompted to continue to the next stage.
Alternative Flow	1. User presses a number picker surrounding the shape 2. The user is prompted to select a value and does so. 3. Repeat for as many number pickers as the user wishes to submit. 4. The user presses the button labeled “Test if it is correct!”, and submits the wrong answer 5. The user is prompted with detailed information regarding the correctness of the answer. Including which values are correct, and which are not.
Parent UC	Overview
Child UC	Next stage or part won screen.

Table B.3.: Use-case exiting and reentering the melody minigame

ID	Melody 3
Name	Exit and reenter melody
Goal	User exits game and reenters the game to find it in the same state as it was left.
Actors	User
Preconditions	User has entered the melody minigame
Prerequisite	Game is installed and running
Main Flow	1. User presses a number picker surrounding the shape 2. The user is prompted to select a value and does so. 3. Without submitting an answer the user exits the minigame by pressing the overview button. 4. At a later point in time the user selects the melody game from the overview and reenters the game. Upon reentering the user find that the values they selected in the number pickers are still there.
Alternative Flow	None
Parent UC	Overview
Child UC	Next stage or part won screen.

C. Non-functional requirements

C.1. Non-functional Requirements

Below are the non-functional requirements for the application.

Table C.1.: Non-functional requirement for cross platform functionality

Name of non-functional requirement	Cross-platform
Priority	Medium
Threat	Excluding users if the group sticks to only one platform.
Requirement	The application should be developed for both Android and iOS.
Action	Cross-platform development with Cordova and Ionic.

Table C.2.: Non-functional requirement for functionality with beacons

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

Table C.3.: Non-functional requirement for functionality without beacons

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 work without beacons.

Table C.4.: Non-functional requirement for user group

Name of non-functional requirement	Age six to ten
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.

Table C.5.: Non-functional requirement for storage of information

Name of non-functional requirement	Storage
Priority	Medium
Requirement	The user should be able to use the application without an Internet connection. When the app is downloaded it should be able to function throughout the exhibition without Internet, except for submition of the created robot, which requires a connection.

D. Testing

D.1. Tests for "Elements"

Table D.1.: Table of Elements 01: Enter the application

Test ID	Table of Elements 01
Module	Table of Elements
Requirement	When entering the game from the "overview"-view the user should be provided a short story about the task, some information about how to win the game. The user should also be given thechoice to continue into the game, or leave it
Precondition	The user is in the overview-view
Approach	• Select the Table of Elements game from overview. • If a popup appears, review the information given. • Select the "cancel" option • If returned to overview - select Table of Elements again • Select the "start" option

Table D.2.: Table of Elements 02: Select an element

Test ID	Table of Elements 02
Module	Table of Elements
Requirement	When selecting an element, the user will get feedback regarding 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.

Table D.3.: Table of Elements 03: Recurring elements

Test ID	Table of Elements 03
Module	Table of Elements
Requirement	If the selected element is correct, it should not reappear in the game, but if an element is incorrect, it should be put back in the stack, and reappear when a user has been through all the elements.
Precondition	The user has entered the minigame
Approach	• Select an element that is correct. • Select the wrong element until the player 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.

Table D.4.: Table of Elements 04: Win the game

Test ID	Table of Elements 04
Module	Table of Elements
Requirement	If the user has selected all the correct elements to the corresponding pictures, the user should see a popup with information that they have won. The popup should have a button witch takes the user to the reward screen, where the user receives their robot part.
Precondition	The user has entered the minigame
Approach	• Select all the correct elements. • A popup should appear that gives the user the opportunity to go collect his new part. • When the button is pressed the user should be sent to the reward screen. • When the user presses the "accept part" button the should be sent to the "my robot" view. • If the corresponding robot part has appeared, the test has been passed.

E. Contract of collaboration

IT2901 – Prosjektarbeid 2
6. SEMESTER

SAMARBEIDSAVTALE

Mellom:

1. Didrik Pemmer Aalen
2. Nicolas Almagro Tonne
3. David Bakke
4. Kristian Svoren
5. Oscar Conrad
6. Bjor Byrkjedal

1) Målet for samarbeidet
a) Målet for samarbeidet i vår gruppe er skriving og innlevering av en felles prosjektoppgave. Foreløpig tittel for oppgaven lyder «Vitensenteret».
I gruppen ønsker vi å fremme læring, samhold og trivsel mellom de enkelte medlemmene og det skal legges et grunnlag for gjensidig faglig utveksling og samarbeid også i den senere studietiden.

b) Vi er enige om at rettferdig arbeidsfordeling i gruppen er svært viktig for å oppnå dette målet. Rettferdig arbeidsfordeling er en forutsetning for gruppens arbeide fordi gruppen skal få en felles karakter på prosjektoppgaven. Gjennom denne avtalen ønsker vi å trekke opp en ytre ramme for samarbeidet slik at det kan fungere etter hensikten.

c) Målet til denne gruppen er å strebe etter karakteren A. Forventet deltagelse fra hvert gruppemedlem er minst 20 timer pr. uke for å oppnå dette.

2) Møtetidspunkt - og sted
Vi er enige om å møtes kontinuerlig i prosjektperioden. Vi forplikter oss til å møtes som det framgår av møteplanen. Møter finner sted minst 1 dager i uken. Medlemmer er pliktet til å følge Prosjektplan, angående møtetidspunkt, varighet og sted.

3) Oppmøte
a) Oppmøte skal være presist. Gruppedeltagere som kommer for sent noteres i møteprotokollen. (Akademisk kvarter gjelder)

b) Vi er imidlertid enige om følgende, frem til 15 minutters forsinkelse fra møtestart tilsvarer 50 kr i matpott, etter dette tidspunkt er straffen 100 kr.
Gruppen setter et tak for antall ganger det tillates å komme for sent tilsvarende X antall ganger i forhold til X antall møter. Gruppemedlemmer som nærmer seg denne grensen, skal gjøres oppmerksom på dette av fungerende gruppeleder.

c) Dersom man kommer for sent pga. omstendigheter man ikke kan gjøres ansvarlig for (sykdom, dødsfall etc.), aksepteres dette. Bortfall og forsinkelser pga. forelesning er også akseptert.

d) Alt blir vurdert ut ifra skjønn, unntatt fastslått matpott-straff.

1

4) Fravær
a) Fravær skal varsles minst en dag i forveien til fungerende gruppeleder og det skal noteres i møteprotokollen.

b) Dersom resten av gruppen Vi ser på fravær på mer enn 20% som en alvorlig hendelse.
(obs!) alt fravær p.g.a. forelesning skal ikke telles med.
Fravær på mer enn 20% skal bli tatt opp med foreleser.
Om noen nærmer seg denne grensen, skal han/hun gjøres oppmerksom på dette av fungerende gruppeleder.

c) Fravær pga. omstendigheter man ikke kan gjøres ansvarlig for (sykdom, dødsfall etc.), aksepteres.

d) Alt blir vurdert ut ifra skjønn.

5) Forberedelse
Alle gruppemedlemmene forplikter seg til å møte forberedt til gruppemøtene slik det blir avtalt under møtene. Uforberedt oppmøte pga. omstendigheter man ikke kan gjøres ansvarlig for (sykdom, dødsfall etc.), kan aksepteres.

En bot på 50 kr i matpott ved en liten forglemmelse. En liten forglemmelse forstås slik at den ikke forhindrer videre arbeid med gruppen. Ved en stor forglemmelse straffes vedkommende med en bot på 100 kr i matpott. En stor forglemmelse forstås å være noe som hemmer gruppens mulighet til å utføre de oppgaver den hadde satt for seg.

7) Gruppens ledelse
a) Leder:
Vi vil ha en leder som velges demokratisk etter prosjektstart. Leder sitter ut prosjektet med mindre det godkjennes nytt valg.

Lederens ansvar.

- Ordstyrer. (Åpne/avslutte møter, lede diskusjoner.)
- Kundeansvarlig.
- Ansvarlig for kontakt med veileder.
- Oversikt av fravær. Og ta det opp hvis nødvendig.
- Leder har ikke noen mer stemmerett ved uenigheter o.l.

b) Sekretær:

Sekretær går på ukentlig rundgang etter følgende liste:
Didrik Pemmer Aalen, Kristian Svoren, Oscar Conrad, David Bakke, Nicolas Almagro Tonne, Bjor Byrkjedal.
Ved sykdom overtar nestemann på lista.

- Skrive logg i forhold til pkt. 8.
- Føre dagens pott inntekter.

c) Nestleder:

- Ved fravær av lederen skal ukens sekretær fungere som leder til leder er tilbake.

8) Møteprotokoll
Vi er enige om at det skal skrives en kort protokoll i løpet av hvert gruppemøte. Denne inneholder a) møtetidspunkt og sted b) GruppeID, c) hvem som innkalte til møtet, d) type møte e) for sent fremmøte, fravær, matstraff, f) kort beskrivelse av møtets innhold, g) hvilke arbeidsoppgaver vi er blitt enige om til neste gang og hvordan disse skal fordeles, h) evt. problemer og i) annet.

Protokollen skal gjøres tilgjengelig for alle gruppemedlemmer (Dropbox, Git, mail, eller lignende). Ingen tilbakemelding på protokoll i inntil 7 dager etter tilgjengeliggjørelse blir vurdert som godkjennelse av protokoll.
Klager sendes til gruppeleder og blir tatt opp på neste gruppemøte. Dette kan føre til endring av protokoll.
Protokollen kan kun forandres opptil en uke etter utsendelse og da kun pga. klager på innhold

I tilfelle tap av protokoll skal det derfor skrives ny protokoll fra hukommelsen. Også denne protokollen skal gjøres tilgjengelig og godkjennes som en vanlig protokoll.

9) Mulige komplikasjoner i samarbeidet
Vi er enige om at det bør ryddes opp i evt. komplikasjoner i gruppesamarbeidet så tidlig som mulig. Eventuell misnøye kan anonymt noteres i møteprotokollen under «annet». Fungerende gruppeleder avgjør i samråd med de andre om og når slike problemstillinger skal tas opp. Han/hun kan da kalle sammen til særskilt møte, og da gjelder møteplikt for alle innkalte gruppemedlemmer. Gruppelederen kan i samråd med gruppen invitere en ekstern samtaleleder til et slikt møte.

10) Eventuelle tiltak ved mislighold
Vi er enige om at hyppig vesentlig forsinket, hyppig uforberedt oppmøte eller gjentatt ubegrunnet fravær slik det er konkretisert under punktene 4, 5 og 6, er svært uheldig for samarbeidet. Ved mislighold av disse punktene er vi derfor enige om følgende tiltak:

- Et gruppemedlem som ikke overholder disse punktene skal først advares av fungerende gruppeleder. Er det gruppelederen selv som ikke følger opp disse punktene, kan advarselen komme fra et vanlig gruppemedlem. Advarselen skal skje skriftlig, den skal underskrives av fungerende gruppeleder og det aktuelle gruppemedlem, og den skal legges ved møteprotokollene. Er vedkommende ikke til å få tak i eller nekter han/hun å underskrive på advarselen, bevitnes dette skriftlig av to gruppemedlemmer. Uttalt og dokumentert advarsel er en forutsetning for at gruppen kan bestemme seg for ytterligere skritt.

- Som neste skritt kan fungerende gruppeleder innkalle til krisemøte der restgruppen og veileder møtes for å diskutere saken nærmere. Innkalling til krisemøte betraktes uansett utfall som forsterket advarsel. Krisemøtet skal få egen protokoll og det skal diskuteres ulike løsninger på problemet.

- Overholder et gruppemedlem heller ikke avtalen etter å ha blitt advart blir dette medlemmet rapportert inn til faglærer med ønske om individuell vurdering.

11) Varighet, gyldighet og reformulering

Denne avtalen er gyldig fra underskriftsdato inntil 30.01.16 eller så lenge en ønsker å være med i gruppen. Hvis flertallet av gruppen kan bestemme at avtalen skal reformuleres. Det er ikke adgang til å be om reformulering etter at et gruppemedlem er blitt advart iht. pkt.10. Dersom det blir oppdaget språkfeil i kontrakten kan disse fikses på et møte der alle medlemmer er til stede og godkjenner endringen og dersom tekstens mening ikke blir forandret.

Trondheim, den 30.01.16

Underskrift

Didrik Pemmer Aalen

Oscar Conrad

David Bakke

Nicolas Almagro Tonne

Kristian Svoren

Bjor Byrkjedal

F. User Manual

Contents

F.1. Introduction

This is the user manual for the Vitensenteret mobile app developed in 2016. The mobile application consists of a series of minigames, each of which have corresponding room or element in Vitensenteret's exhibition. It's meant to be played like an exploring game, where bluetooth beacons notify the player when he/she finds a game in the exhibition, but can also be played without the beacons.

This manual will explain how the application works and how to play the different games.

F.2. Requirements

Device with Android 4.4 KitKat or newer.
Bluetooth Low Energy (Optional, for beacons)

An internet connection (Optional, for submitting robot at the end)

F.3. Installation

See Appendix G, Installation Guide

F.4. Get to know the application

F.4.1. First run

When first running the application you will be asked to select your preferred language, as seen in Figure F.1. Simply click one of the flags and the rest of the application will be in that language. There is also a welcome screen with a introduction to the app and game story. See fig. F.2.

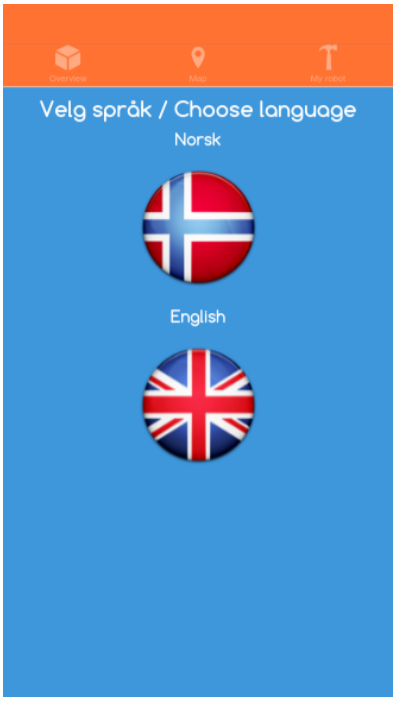


Figure F.1.: The "choose language" app view

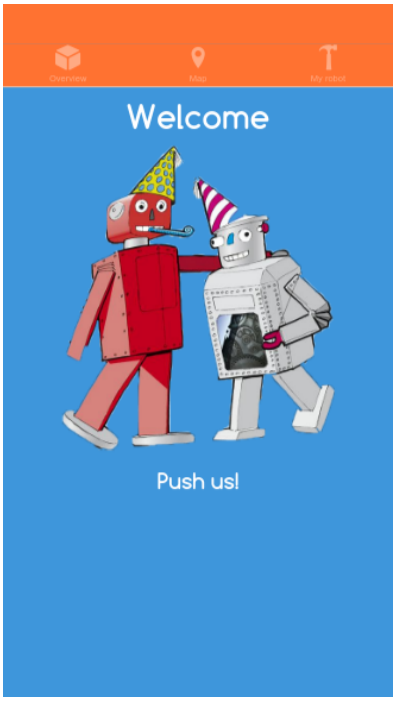


Figure F.2.: The "Welcome screen" app view

F.4.2. Game Overview

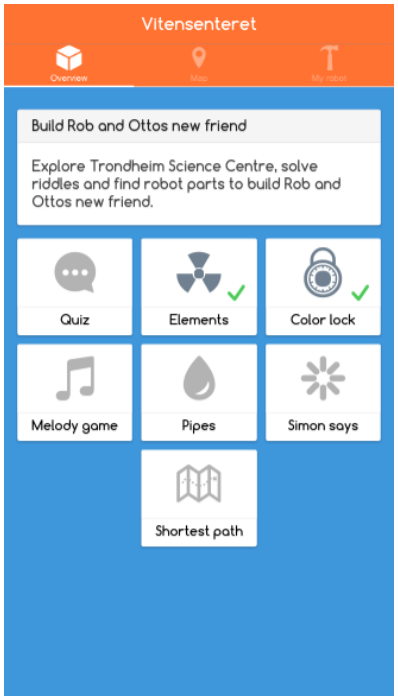


Figure F.3.: The "Game overview" app view, with some games finsihed.

The game overview, as seen in figure F.3, is the home screen of the application, where one can find the minigames and their status. They can be either semi-transparent, which means they have not been found yet (with beacons activated, as seen in figure F.4), colored with a checkmark, which means they have been completed, or colorless, which means they have not yet been completed.

By clicking a game card a popup will be shown, telling a short story and instructing the player how to play the game. From there the player can start the game.

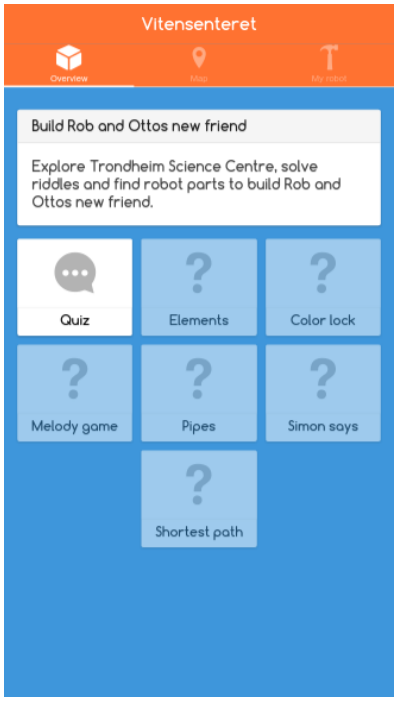


Figure F.4.: The "Game overview" app view, with beacons enabled.

F.4.3. Tabs

The three non-game views can be navigated to by using tabs located at the bottom or top of the screen, depending on what OS you are using. These can take you to the minigame overview, map or robot builder.

F.4.4. Map

The map shows a floor-plan of Vitensenteret's exhibition, with the minigames's locations plotted, as seen in figure F.5. One can zoom and pan the map.

Figure F.5.: The “Map” app view.

F.4.5. Robot builder

In the robot builder view one can see the different robot parts one has collected by winning games, as well as switch which part and color one wants for the robot, as seen in figure F.6.

- By clicking the robot at the top left one can zoom in on it.
- By clicking the arrows one can change the selected robot part.
- By clicking the robot part in the list a color and lightness selector is opened, where one can change the color of the robot. See figure F.7.

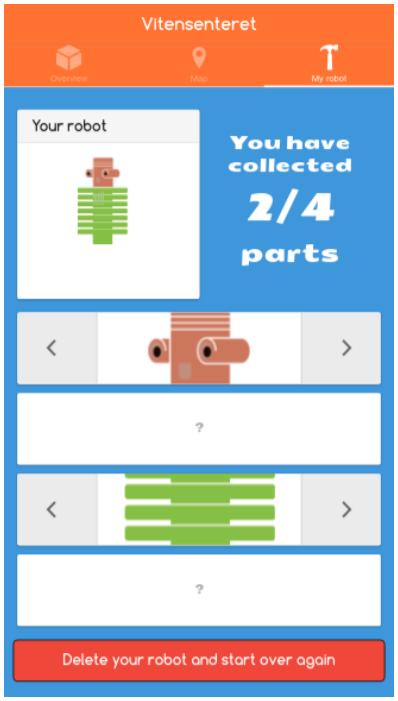


Figure F.6.: The “Robot parts” app view.

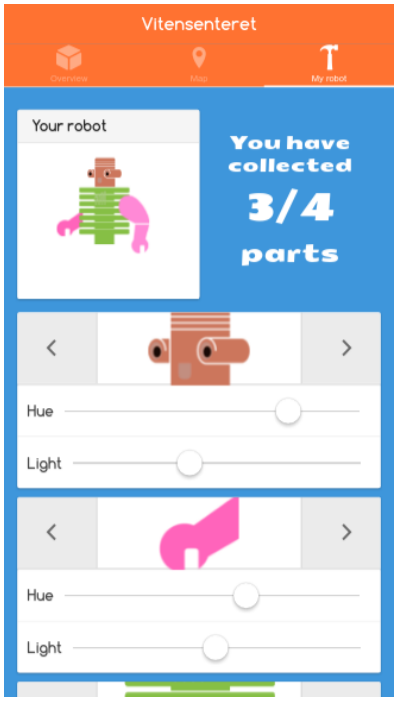


Figure F.7.: The “Robot parts” app view, with the color customization sliders open.

F.5. Games

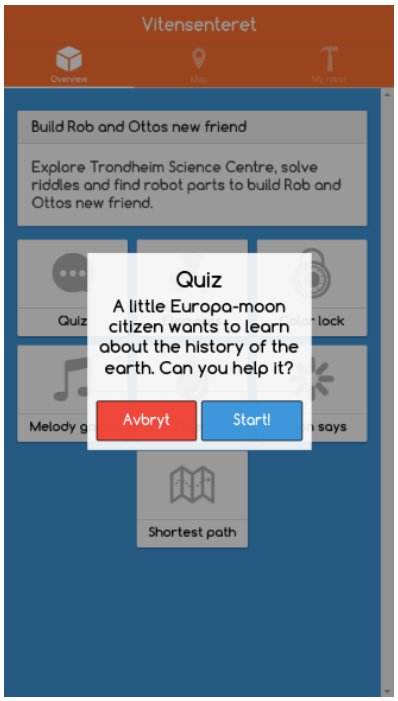


Figure F.8.: An example of the game story.

The minigames have a common story where the player is helping two robots find robot parts for a new robot. They don't need to be done in a certain order, but as there are only four types of robot parts, and seven minigames, one will win all the robot part types before finishing all minigames. See Fig. F.8

Each game has different stages with varying difficulty, all of which have to be completed to win the game. When one wins a game one is shown the won robot part, and then one is taken to the robot builder view to modify the robot. Fig. F.9

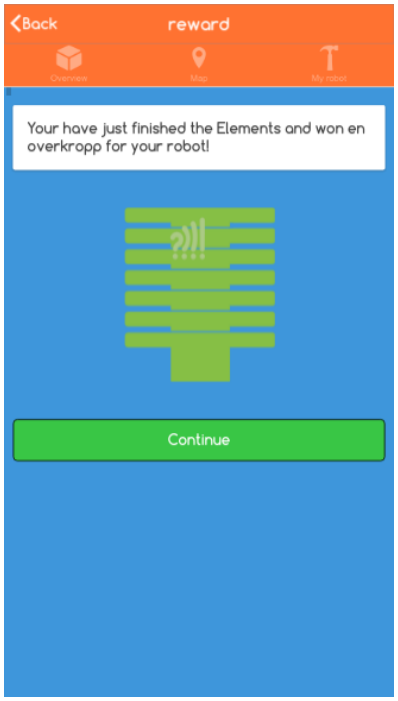


Figure F.9.: Example of part won when player wins a minigame.

When all parts are collected one can submit the robot to Vitensenteret, where it can be seen on a screen with all submitted robots. This is done by pressing the green button at the top of the robot builder page when all parts are collected. One will be sent to a page where one can fill out the robot's name and your own, with a final preview of the robot. By clicking the Submit button at the bottom the robot will be submitted, and one will be taken to the robot overview. Fig. F.10 Fig. F.11

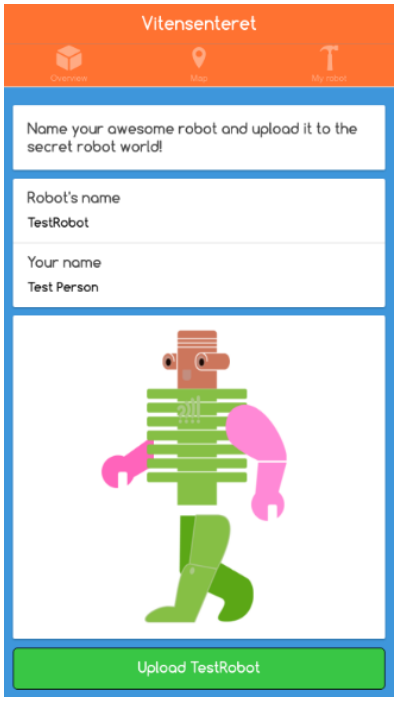


Figure F.10.: View of when a player has finished the game and is about to submit the Robot.

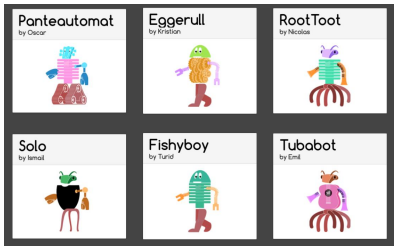


Figure F.11.: The webpage with all the submitted robots.

F.5.1. Quiz

The quiz game uses questions found throughout the exhibition to quiz the player on different themes. Press the correct alternative to progress through the 11 questions. If one answers wrong the question will be asked again at the end. Game depicted in figure F.12

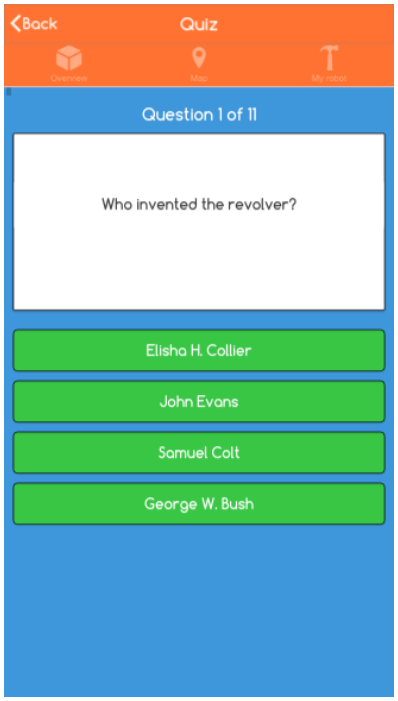


Figure F.12.: The “Quiz” app view.

F.5.2. Elements

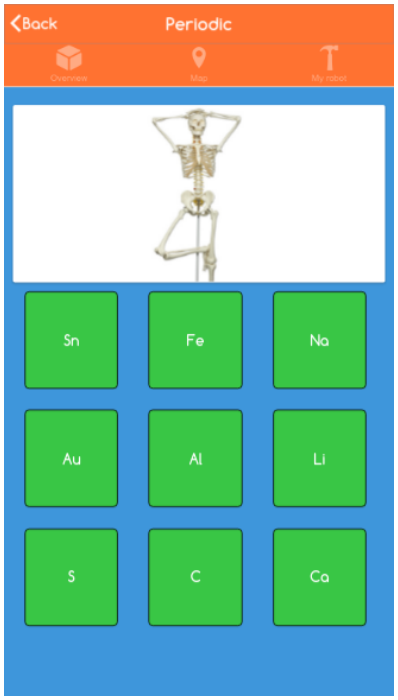


Figure F.13.: Periodic minigame

The elements game is a game where one is to find what chemical element the depicted item is made of, and is supposed to be played in front of the "Table of elements" exhibit. Search the exhibit for the displayed item, find what the scientific name of that element is, and press the corresponding button. A descriptive text of that element will then be shown, letting the player know if he was right or wrong. Game depicted in figure F.13

F.5.3. Color Lock

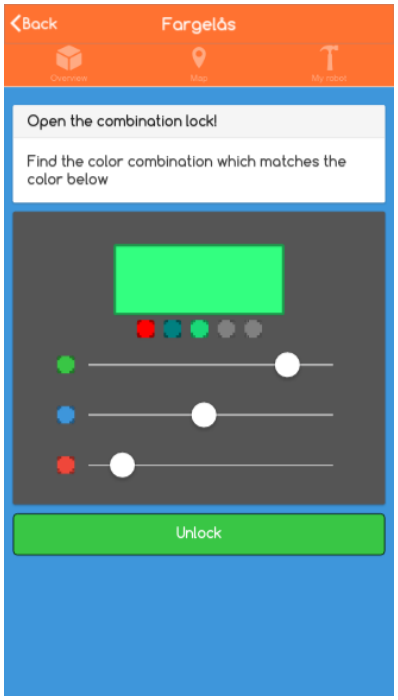


Figure F.14.: Color lock minigame

The color lock game is a game where one is supposed to unlock a combination lock locked with different color combinations, and is linked to the color combination exhibit.

To play the game, drag the three sliders, each of which represents green, blue and red, so that they generate the same color that is displayed at the top. One is supposed to use the exhibit to find the combination, but a small preview of the current color combination can also be found under the large color rectangle. The "Unlock" button will become active when the correct combination is entered, and will make the next color active. Game depicted in figure F.14.

F.5.4. Melody Game

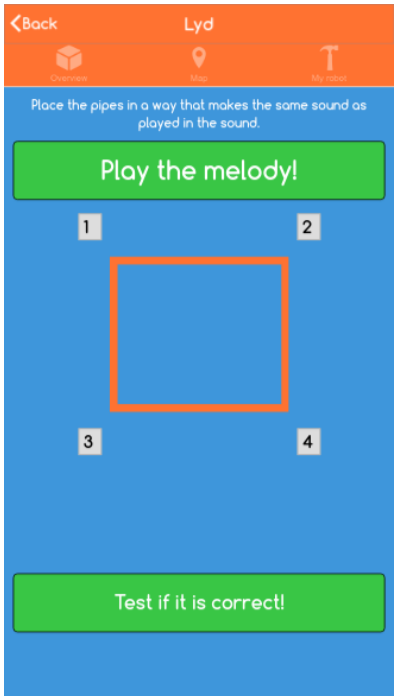


Figure F.15.: Melody minigame

The melody game is a game where one is to rearrange metallic tubes in the rythm-exhibit to make the same sound as the app and input the position of the tubes into the app.

To play the game click “Play the sound”, listen to the sound, and re-create that sound with the tubes. You can then input the position of the tubes into the white dropdown boxes in the app and click the “Test” button. The game is depicted in F.15.

F.5.5. Pipes

Figure F.16.: Waterflow minigame

The Pipes game is a game where one is to rotate the different pipes so that water can flow from the water container at the top left, to the turbine in the bottom right. This is done by simply pressing the pipe one wants to rotate until it is in the correct position. There are three levels, all of which have to be completed to win the game. The game can be played anywhere, but is supposed to be played in the water power room. The game is depicted in figure F.16.

F.5.6. Simon Says

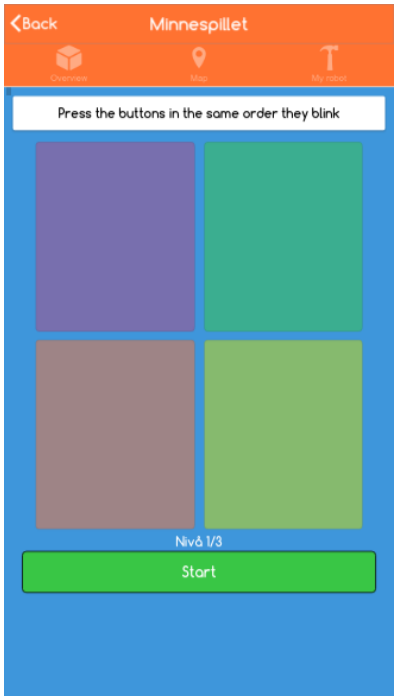


Figure F.17.: Simon says minigame

The Simon Says game is a game where one has to remember in which order a series of blinking lights blink, to test the memory of the player. It can be played anywhere, is connected to the memory room.

To play it press the "Start" button, watch the lights blink, and re-create the same sequence by pressing the correct colored squares. There are three levels, with increasing difficulty, all of which have to be completed to win the game. The game is depicted in figure F.17.

F.5.7. Shortest Path

Figure F.18.: Shortesst path minigame

The Shortest Path game is a game where one has to find the shortest single path between multiple cities in Europe, where Amsterdam is the start and "Nidarosia" or Trondheim is the end. A map and measuring tools can be found in the mathematics room, and must be used to measure the distances.

To play the game drag the red boxes into their correct positions and click the "Check" button. If some of the cities are incorrectly placed they will remain red, but turn green when placed in the correct position. The game is depicted in Fig. F.18.

F.6. Restarting your game

To restart your game and delete all saved data go to the "My Robot" tab, scroll to the bottom, press the red "Delete my robot and restart" button and confirm the action. The application will then delete all saved data and reload the application.

This can be useful if one wants to replay the games, test out something, if there are any bugs, or if one has installed a new version of the app and some old data is stored.

F.7. Troubleshooting

- If beacons don't work, try enabling Bluetooth on the phone, go into application settings and enable the Location permission for the app, and restart the game, as described in the "Restarting your game" section above.
- If you chose the wrong language when the app started, restart the game as described in the "Restarting your game" section above.
- If you can't install the app, try enabling "Unknown sources" in settings.

F.8. Known Issues

- Some views do not fit into a mobile screen, and the user has to scroll to see all elements.
- The application can not be installed on android phones with a version below 4.4 KitKat.
- On android 6.0.1 and above one has to manually allow location services to be used by the app (in the application settings) so that it can connect to Beacons.

G. Installation Guide

G.1. Android installation

This guide will show how one can install the Vitensenteret application on a android device.

G.1.1. Prerequisites

- A phone with Android 4.4 (KitKat) or newer.
- The APK file from our project.

When building the application from source:

- Node.js installed
- Android SDK manager installed.

G.1.2. Building the application from source

Note: This step is not necessary; one can install the application from the pre-built APK.

1. Download Android SDK 19 from the Android SDK Manager.
2. In the project's root folder run "npm install .", which will install all dependencies.
3. Run "ionic serve" to check that ionic and the project is properly installed, as well as compiling some necesary files. A browser window should open with the application running.
4. Run "ionic platform add android" to add android as a build platform.
5. Now one can either build an APK file, or install/run the application directly on ones device. To build the app run "ionic build android". To run the application on ones device connect the phone in developer mode and run "ionic run android".
6. One should now have either the APK placed in "platforms/android/ant-build/CordovaApp-debug.apk" or running on ones phone.

G.1.3. Preparations

To install an app from outside the Play Store one has to enable installation from unknown sources on the android device. This is done by going into Settings, then under Security enable "Unknown Sources".

One also needs the APK file to be on the android device. This is most easily done by downloading it from the internet, by sending it to yourself or something similar. It can also be done by connecting the device to a computer and transfering the APK to the Downloads folder after enabling file transfer mode on the android device.

G.1.4. Installation

To install the app, open either the "Downloads" app or another file browser, find, and open the APK file that should be there.
Install the app, and open it.
If all went well one should now have the application running on the device.

G.1.5. Troubleshooting

If the app won't install, make sure your android device is version 4.4 or above, which can be checked in Settings, About, then Android Version. Also make sure Unknown Sources is enabled and that the APK was not corrupted during the transfer to the device.

If one does not have a Android device with version 4.4 or higher one can use a Android emulator to test the app. See http://developer.android.com/tools/devices/emulator.html

H. Customer Feedback

"Vitensenteret i Trondheim (ViT) has had the privilege to host a student project for IT 2901 in spring 2016. The scope of the project was to produce an app for smart phones designed to be used in the exhibitions at ViT. The product was designed in collaboration with the student group, and the aim was to make a playful experience in the center and explore the possibilities of science dissemination through gamification and new media technology.

ViT is very satisfied with the process and outcome of the project. The students were given an open task where they defined the product in a joint process with ViT. We have had meetings every fourteen day to ensure progress, and that the group worked in line with the scope of the project. In our experiences the collaboration has worked very good. The student group have showed an excellent ability to explore ideas and at the same time limit their workload to the time schedule. The group has showed good progress, delivered on time and showed good communications skills towards their assigned employer.

Planning and implementation of student projects like these are new to the ViT, so we have experienced some minor challenges in the internal organization of the project, but this is not related to the student groups, but to ViT not assigning our graphic designer early enough in the process.

ViT is very satisfied with both the process and the product, and we are looking forward to present the app to our audience.

Arnfinn"

I. Figures

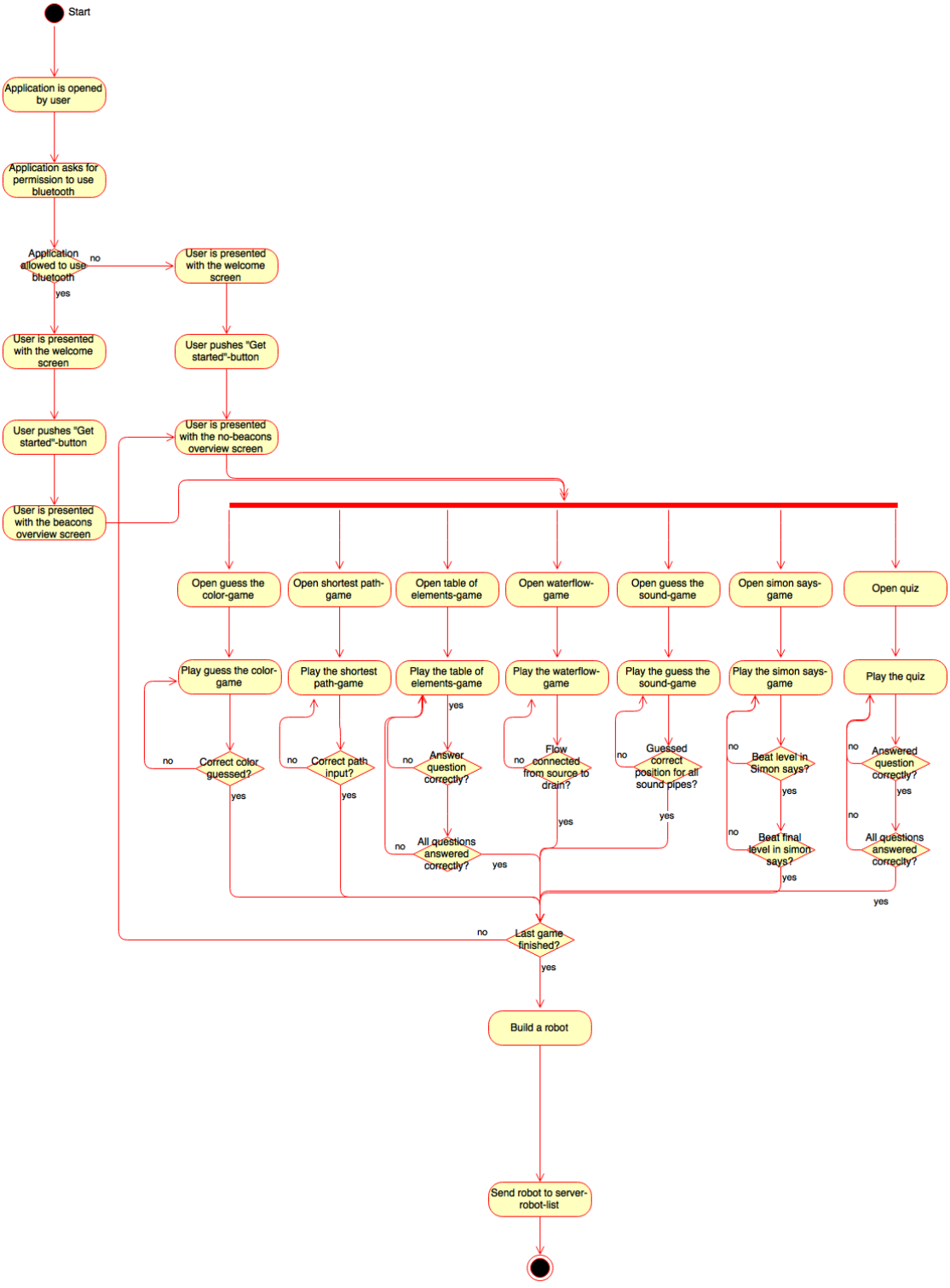


Figure I.1.: Activity diagram for the application

Figure I.2.: GANTT diagram

		Planning				Follow-up	
Work Package	Activity	Resource	Planned work (hrs)	Start	Finish	Actual work (hrs) per person	Status (% complete)
Pre study	Requirements engineering	All		1	1	5	100%
	Development research	All		1	1	10	100%
	Brainstorming for ideas	All		1	1	10	?
Design (draft)	Design welcome screen	All		2	2	1	100%
	Design Water flow	All		2	2	1	100%
	Design Table of elements	All		2	2	1	100%
	Design color-game	All		2	2	1	100%
Project Management	Planning and discussions	All	5	1	2	5	100%

Figure I.3.: Activity plan for the first week

J. Graphical design

J.1. Sketches

Figure J.1.: Quiz minigame

Figure J.2.: Shortest path minigame

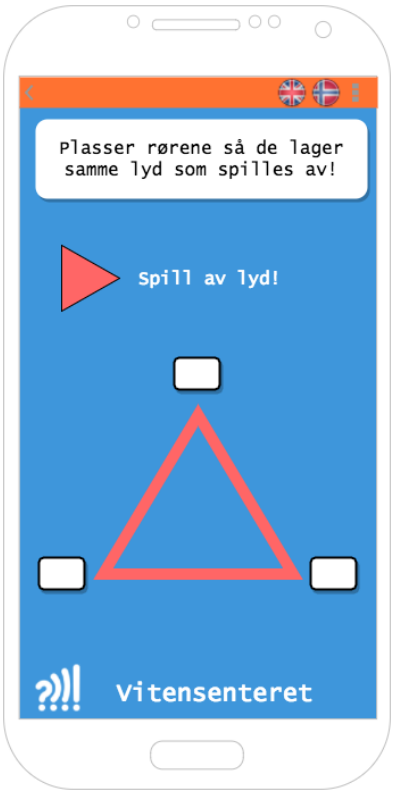


Figure J.3.: Sound minigame

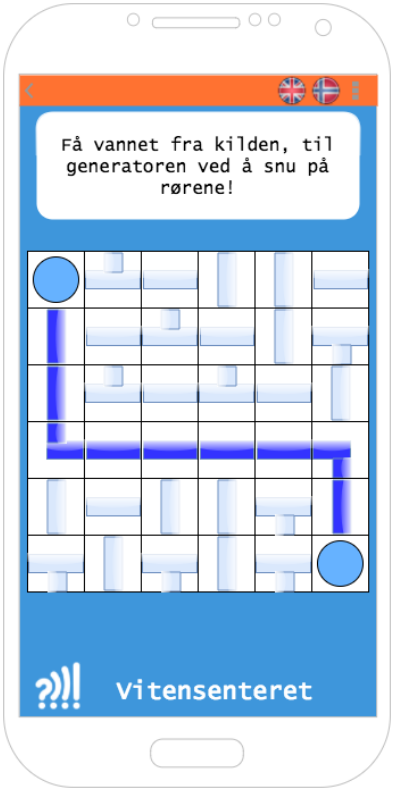


Figure J.4.: Waterflow minigame

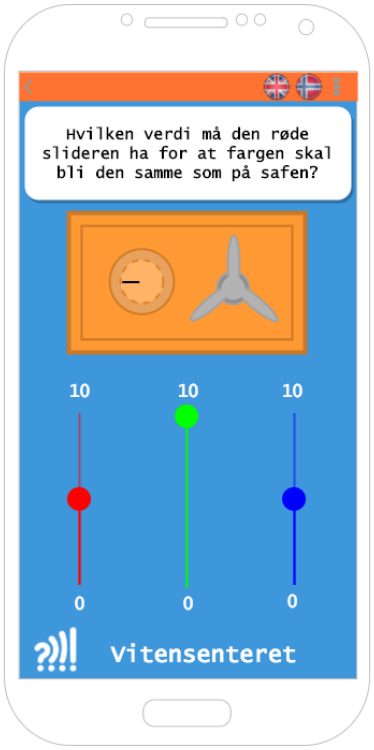


Figure J.5.: Color slider minigame

Figure J.6.: Memory minigame

J.2. Final design

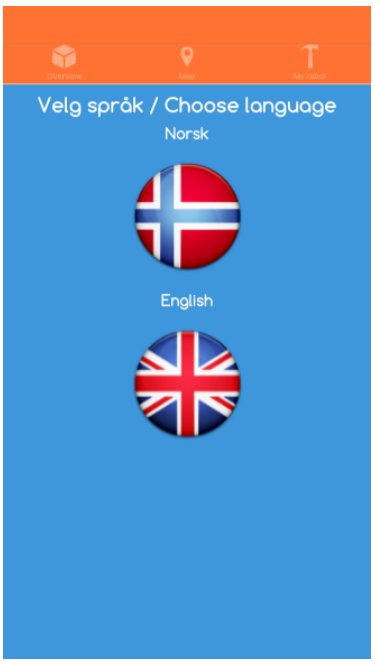


Figure J.7.: Choose language screen

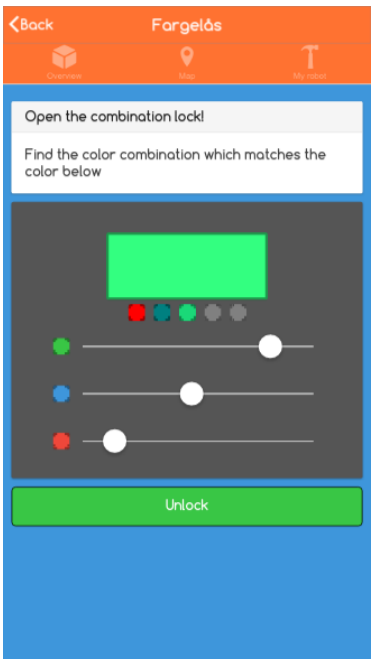


Figure J.8.: Color lock minigame

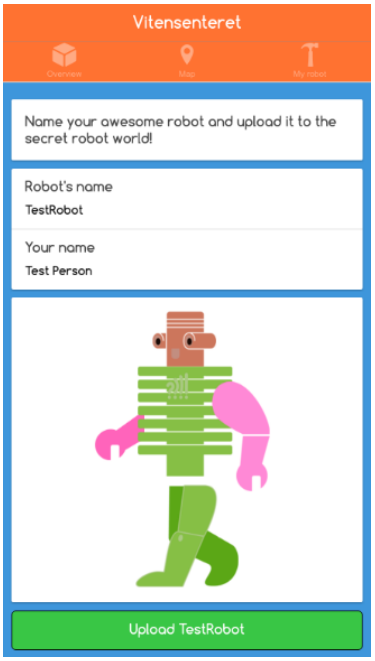


Figure J.9.: The finished screen showing your robot

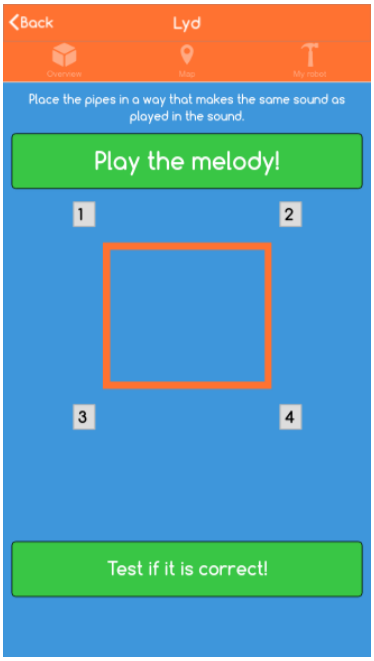


Figure J.10.: The sound game

Figure J.11.: The map of the exhibition

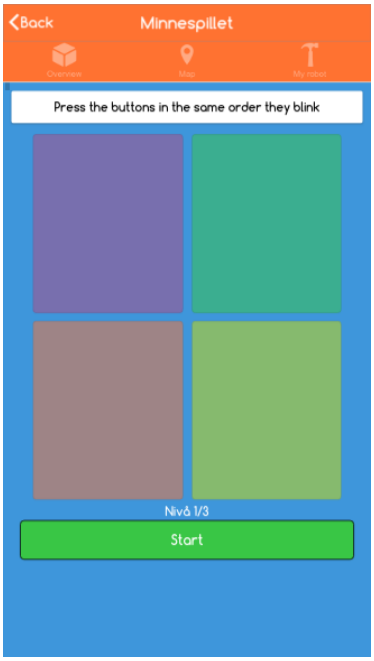
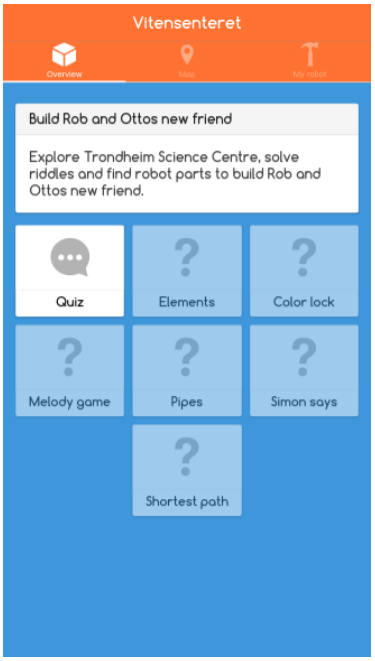
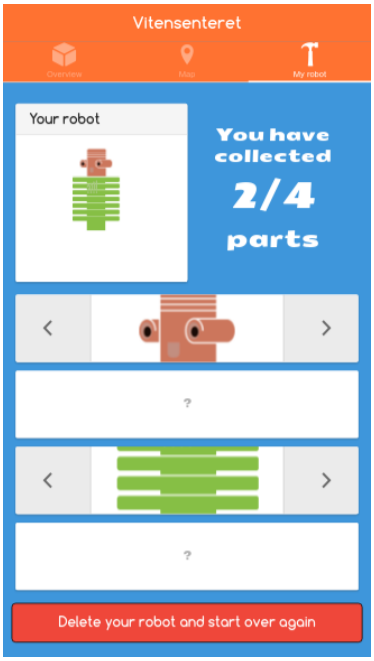


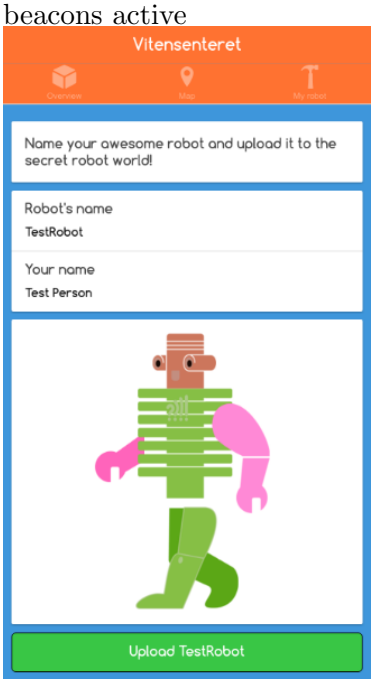
Figure J.12.: Memory game



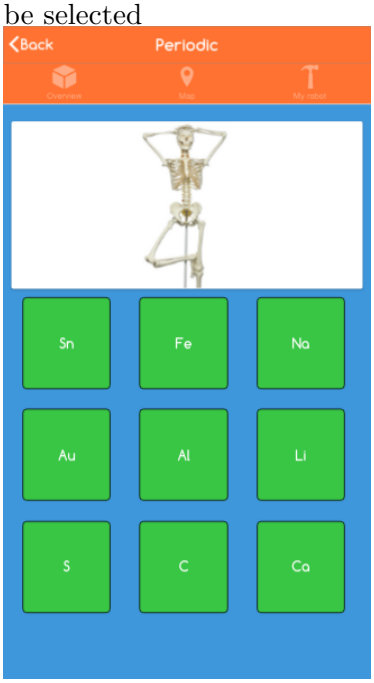
The overview screen with beacons active



The screen where parts can be selected



The finished screen



Periodic table game

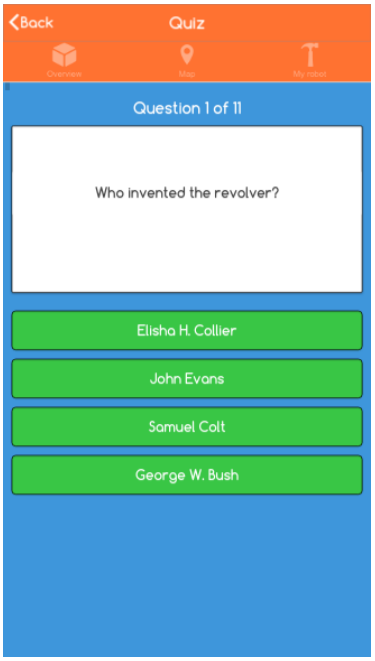


Figure J.13.: The Quiz minigame

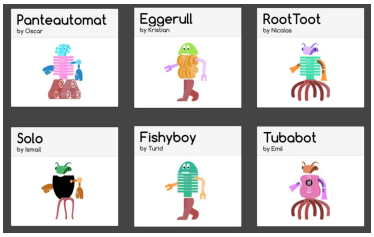


Figure J.14.: The screen that shows the robots after submission

Figure J.15.: The shortest path minigame

Figure J.16.: The waterflow minigame