NTNU
Norwegian University of
Science and Technology

Porting mobile iOS app to Android - Klimb AS

Author(s)

Magnus W. Enggrav
Adam A. Jammary

Bachelor in Game Programming
20 ECTS
Department of Computer Science
Norwegian University of Science and Technology,

16.05.2018

Supervisor Mariusz Nowostawski

Sammendrag av Bacheloroppgaven

Tittel:	**Porting av mobil iOS app til Android - Klimb AS**
Dato:	16.05.2018
Deltakere:	Magnus W. Enggrav Adam A. Jammary
Veiledere:	Mariusz Nowostawski
Oppdragsgiver:	Klimb AS
Kontaktperson:	Sindre Helelborgås
Nøkkelord:	Bachelor, Porting, iOS, Android, React, Native, Mobil
Antall sider:	89
Antall vedlegg:	6
Tilgjengelighet:	Åpen

Sammendrag

Hovedmålet med prosjektet var å få den eksisterende Apple iOS-spesifikke appen til å fungere på Google Android-plattformen for vår arbeidsgiver Klimb AS. Den eksisterende kodebasen ble allerede skrevet ved hjelp av React Native rammeverket, som skulle gi funksjonalitet på tvers av plattformene iOS og Android. Appen ble utviklet på, og hadde bare blitt testet på iOS-plattformen.

Under arbeidet med prosjektet, fant arbeidsgiveren vår ut at det var bedre å endre mye av den nåværende arkitekturen, og flytte mer av funksjonaliteten ut i tjenester. Dette var noe vi trengte å tilpasse oss til.

Vårt hovedfokus mens vi jobbet med prosjektet var å fikse feil som dukket opp på Android-versjonen, og få programmet til å fungere på Android uten å krasje, og å få funksjonaliteten til å virke som forventet, sånn som det gjør på iOS-versjonen. Og i tillegg å få Android-versjonen til å se og føles så lik iOS-versjonen som mulig.

Summary of Graduate Project

Title:	**Porting mobile iOS app to Android - Klimb AS**
Date:	16.05.2018
Authors:	Magnus W. Enggrav Adam A. Jammary
Supervisor:	Mariusz Nowostawski
Employer:	Klimb AS
Contact Person:	Sindre Helelborgås
Keywords:	Bachelor, Porting, iOS, Android, React, Native, Mobile
Pages:	89
Attachments:	6
Availability:	Open

Abstract

The main goal of the project was to port the existing Apple iOS-specific app to the Google Android platform for our employer Klimb AS. The existing codebase was already written using the React Native framework which was supposed to provide cross-platform functionality between iOS and Android. The app was developed on, and had only been tested on the iOS platform.

While working on the project, our employer decided that it would be better to change a lot of the current architecture, and move more of the functionality out into services. This was something that we needed to adapt to.

Our main focus while working on the project was to fix bugs that appeared on the Android version, and get the application to work on Android without crashing, and make the functionality work as expected, like it does on the iOS version. Also to make the Android version look and feel as similar to the iOS version as possible.

Preface

We want to start by thanking everyone that has been a part of this project.

We would especially like to thank Elin Årseth, Sindre Helelborgås and everyone else at Klimb AS for providing us with this Bachelor project, and for all their help and cooperation throughout the project.

We also want to thank Mariusz Nowostawski for supervising our project and providing us with great practical feedback on how to setup ShareLatex, and how to get us started in general.

Contents

List of Figures

1 Introduction

1.1 Background

Klimb is a health and fitness mobile game, where the objective is to collect points by walking up hills using the GPS coordinates of the mobile device. You get one height point for every meter you climb up the hill. The goal of the game is to move up levels, where each level represents a known mountain you can climb up.

The game displays a list of every top nearby, and you collect points by selecting a top and start moving towards it, or you can collect "free-points" if you are not moving towards any specific top. If you are close to a top you can choose to capture the top and it will be displayed in your collection of mountain tops.

You can hike up your favourite mountain and collect points at the top. If the mountain does not exist in the game, you can add the top and become the owner of that mountain top. The application will automatically measure your height meters every time you walk or run uphill. Your profile page will show your progress towards the next available level, and how many height points you have collected.

You can share your progress and activity on the app-specific social feed and on Facebook to inspire others. Klimb uses the Facebook SDK for authentication which makes it easier and faster to manage user logins.

1.2 Project Description

Klimb AS used external consultants "Snowball Digital AS" to create the first version of the application and the back-end server. After some time, Klimb AS decided to hire a new programmer to take over the development and move the development internally. This all happened right before we started our bachelor project so the new developer already had a lot to do. Since Klimb AS already had received user requests about an Android version of the application, we saw this as an opportunity to help them create a functional Android version as our bachelor project.

The main goal of the project is to port the existing Apple iOS-specific app to the Google Android platform. Even though the existing codebase is already written using the React Native framework which allows for cross-platform functionality between iOS and Android, the app has been developed and tested only on the iOS platform. The result is that it does not currently work on the Android platform.

The issues of what is and what is not working is not clearly defined this early in the process, and it is mainly our task to find all the issues and resolve them, so that the app can work on Android devices as well. This means that it will be difficult to plan in much detail, as the details are not yet known. The React Native framework uses JavaScript as the programming language for both platforms, but allows the use of the native language of each specific platform if needed.

We will try to use as little platform-specific code as possible, and have the two platforms share as much of the codebase as possible. This will require clear communication, and a well-defined work process as we will describe later. The goal is to have the UI (User Interface) look and feel almost the same on both platforms with less than 10% differences between the two.

Initially the plan is for the employer to focus on the back-end server communication, and for us to focus on the Android front-end, but this may change during the project as needed.

The employer has a huge list of features that they wish to implement, they are not officially part of the Bachelor project, but they are available if the current workload turns out to be too small.

1.3 Scope

Field of Study

- Cross-platform (Android and iOS) mobile development
- React Native Framework
- Internationalization (I18N)
- Geolocation (GPS-data, Google Maps)
- Login services (Facebook SDK)

Project Restrictions

- We will not develop anything for the iOS platform
- We will not work on the codebase in the master branch of the repository
- The app will not support older devices than Android 4.1 (API 16 - JELLY_BEAN)

1.4 Purpose

One of the members in the group (Magnus W. Enggrav) knew the owner of Klimb AS and had been in contact with Elin Årseth. They had discussed the possibility of doing a Bachelor project, but they did not specifically know what the objective of the project would be. After discussing the project with the other group member (Adam A. Jammary), he also liked the general idea of the game, where the user has to be active in real life to gather points. After some discussions about what Klimb AS needed, they asked us if we wanted to make the application work on Android.

We talked to our supervisor Mariusz Nowostawski, who also thought that this would be a great experience. With this, we could learn more about mobile development using a cross-platform JavaScript framework like React Native, and at the same time help Klimb AS reach a new user base on the Android OS.

We were also told that the design of the application may change while working on the Bachelor project, so a close collaboration with Sindre Helelborgås would be necessary. This would also give us more experience on how it is to work in a professional setting. Klimb AS told us that if we are able to complete the task, they will have more then enough tasks for us in their backlog. This was great since we did not yet know how much work would be needed to get the application working on Android.

We were also interested to learn about the differences between Android and iOS, since React Native is supposed to be a cross-platform framework between Android and iOS, and why the Android version of the application did not work.

1.5 Target Audience

The company Klimb AS already has a user base for their existing application released on iOS, and are now looking to expand to users with Android devices. Their target audience is anyone who actively hikes a lot, or spends time walking up mountain tops, and wants to get feedback on how well they are doing and share this with friends or other hiking enthusiasts.

Originally all of the text labels in the application were only written in Norwegian, since their target audience were only users based in Norway. But as they are planning to expand, possible users in the future may come from any country. So one of our tasks will be to implement a framework to more easily provide translated text labels in multiple languages, at least initially in both Norwegian and English.

1.6 Roles

Mariusz Nowostawski	Supervisor
Elin Årseth	Employer: Owner/CEO
Mette Kristensen	Employer: Part Owner
Asetto Capital	Employer: Part Owner
Sindre Helelborgås	Employer: Project Manager
Magnus W. Enggrav	Developer
Adam A. Jammary	Developer

1.7 Academic Background

Both of the team members have taken the same courses during our Bachelor program "Bachelor in Game Programming". These courses have provided us some practical experience with general software programming principles in Object-Oriented languages like C++ and Java. Courses in Networking, Operating Systems, Databases and Software Engineering etc. have mostly provided theoretical knowledge in these fields which helps us understand how most technology fundamentally works.

Much of our prior knowledge before starting the project has been focused on developing desktop applications, including games or more service-oriented back-end processes. We had no prior experience in developing mobile applications, neither natively for Android or iOS, nor using frameworks like React Native which uses the more web-oriented JavaScript language.

This means that much of our time will initially be spent on learning a new way of thinking. Luckily we have a mobile development course running in parallel with the Bachelor project, which may provide us some insights in to mobile development, and hopefully we will be able to apply this insight to this project.

We are both looking forward to face new technical challenges, and gain valuable practical experience on working with technology that is actually used in the industry.

1.8 Glossary

Klimb AS

Name of the company we are working for, where Elin Årseth is the owner.

Klimb

Name of the application we are working on.

Codebase

The existing source code used to create the Klimb application.

UI

User Interface, the interface that the user interacts with.

JavaScript

One of the core programming languages used for web development, supported by most modern web browsers. JavaScript is dynamically typed, which means that it can change the data type of a variable throughout the program.[1]

TypeScript

TypeScript is based on JavaScript, but it is statically typed, which means that when a variable has been assigned a data type, it can never be changed later.[2]

Android

An Operating System developed by Google, and is mostly used for touch-based mobile devices.[3]

iOS

An Operating System developed by Apple, is only used on Apple-specific hardware. Like Android it is mostly used on Apple mobile devices like mobile phones and tablets.[4]

1.9 Document Structure

Chapter 1 - Introduction

Provides a basic understanding of what this project is about and who is involved.

Chapter 2 - Project Management

Explains how we worked during the project, and how we used different tools to keep everything organized.

Chapter 3 - Alternative Technologies

Looks at some alternative technologies that could have been used instead of React Native.

Chapter 4 - Technologies

Provides a deeper explanation of the technologies used to create this application, how they work and how they are used.

Chapter 5 - System Architecture

Explains how the different parts of the technology used in this project is organized.

Chapter 6 - Implementation

Goes in to details on how we contributed to the project by implementing features and functionality.

Chapter 7 - QA - Testing

Explains various testing and debugging alternatives for React Native, and which of them we found the most useful.

Chapter 8 - Summary

A summary of the project as a whole, the results we achieved, some evaluations and a conclusion.

2 Project Management

2.1 Software Development Methodology - Scrum with XP

In our project plan (see page 55) we chose to go with Scrum [5] as a software methodology because it has "more transparency and project visibility" [6]. This was needed because the task we were given told us that we had to work closely with Sindre Helelborgås, the programmer at Klimb AS, and would give both the employer and us a better understanding of what everyone was working on.

We also wanted to include some parts of XP (Extreme Programming) [7], like pair-programming. Our plan was to use pair-programming in the beginning of the project when learning about React Native and how the Klimb codebase is structured. And then later on split up, and do separate tasks when we are more comfortable with how the code works. We also wanted to use pair-programming for bigger implementations like "TutorialManager" 35 or other more complicated issues, to make sure both members were on the same page when it comes to the logic and reasons for why a specific solution was the best solution. Working with pair-programming worked out really well for us.

We originally planned to focus more on testing by using "Jest" [8], which is a JavaScript [1] testing tool. However much of the codebase is using "TypeScript" [2] files, or ".ts" files, and these are not supported by "Jest" as they cause the tests to fail. It is possible to import external packages to solve this problem, however this could also cause dependency issues. And due to our time constraints we chose to prioritize fixing existing bugs on Android instead of possibly introducing new ones.

2.2 Status Meetings and Decision Points

2.2.1 Meetings with employer Sindre Helelborgås

Our plan was to have regular meetings with our employer once a week, on Mondays at 10 AM on Discord [9]. This was something that we followed the first couple of weeks of the project, when we still did not have a good understanding of the code. After this we decided to have meetings only when we had questions, needed feedback on something we had implemented or if there had been some big changes to the application. For smaller questions we used Discord to communicate and Sindre Helelborgås would answer when he had time.

2.2.2 Meetings with Supervisor Mariusz Nowostawski

We had our first meeting with our supervisor right after our initial meeting with our employer. We discussed our project and the report, and he suggested writing more about the process instead of planning too much in detail, as we will not know about the problems that will arise until later on in the project. During our project we would usually send an email if we had any questions, this was much faster and more agile than scheduling a meeting for every little question we had.

2.3 Time Managment

2.3.1 Toggl

We used Toggl [10], an online time management tool, to track the amount of time we used on the various activities during our Bachelor project. This helped us manage our time more efficiently, and we got a better overview of our time distribution. This also helped us improve our estimations on how much time we would spend on an issue. Our Toggl time is shown in the "Hour Log" 70.

2.4 Task Board

2.4.1 Trello

We decided to use Trello [11] as our Scrum board so that every member of the team, including Sindre Helelborgås, could keep track of what everyone else was working on. To be able to track the tasks properly, we labeled all of our commits, as well as the Trello cards, with the issue number, ex. "9 Internationalization - i18n". We categorized each issue as either a "task" or a "bug". We labeled the issue as a "bug" for already implemented features which did not work as expected and needed to be resolved. Otherwise we labeled it as a "task" for features that needed to be implemented.

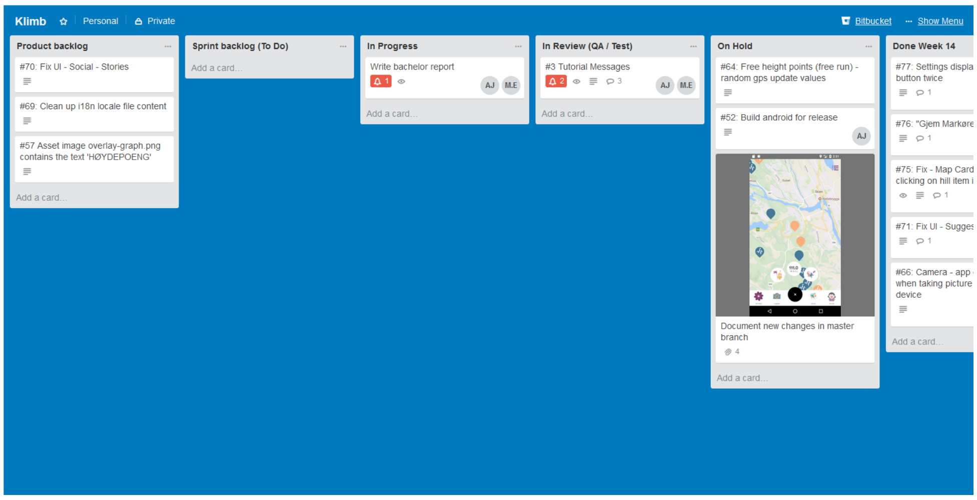

Klimb
Personal
Private
Bitbucket
Show Menu
Product backlog
#70: Fix UI - Social - Stories
#69: Clean up i18n locale file content
#57 Asset image overlay-graph.png contains the text 'HØYDEPOENG'
Add a card...
Sprint backlog (To Do)
Add a card...
In Progress
Write bachelor report
AJ
M.E
Add a card...
In Review (QA / Test)
#3 Tutorial Messages
AJ
M.E
Add a card...
On Hold
#64: Free height points (free run) - random gps update values
#52: Build android for release
AJ
Document new changes in master branch
Add a card...
Done Week 14
#77: Settings displa button twice
#76: "Gjem Markøre
#75: Fix - Map Card clicking on hill item i
#71: Fix UI - Sugges
#66: Camera - app when taking picture device
Add a card...

Figure 1: Trello Board

- **Product backlog**
  Shows the features that Klimb AS wanted to have implemented in the app.

- **Sprint backlog**
  Shows the issues that we will work on this week.

- **In Progress**
  Shows the issues that we are currently working on right now.

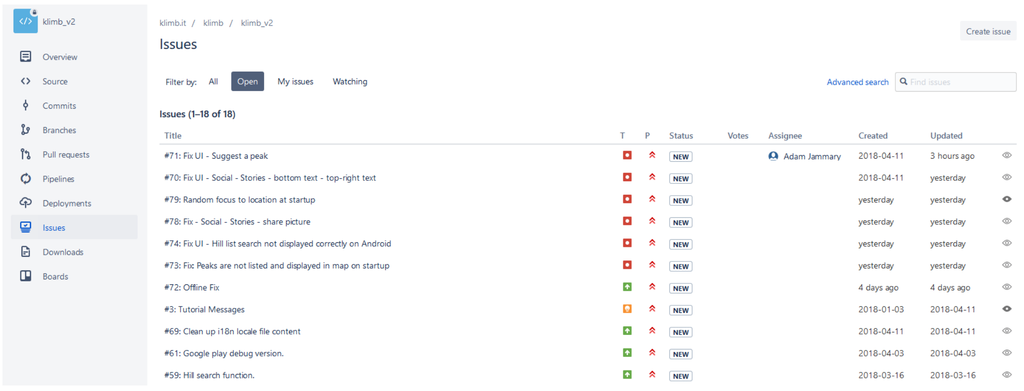
- **In Review (QA / Test)**
  Shows the issues that are done and waiting for somebody to review it.

- **On Hold**
  Shows issues that are currently "on hold" meaning to they will be fixed later. One reason for this category can be that somebody is currently working on another issue that may fix the "On Hold" issue. Another reason may be that some issues are located in the same source file, and this is a way to avoid problematic merge conflicts.

- **Done Week**
  Shows what we have accomplished during that week.

2.4.2 Bitbucket Issues

Whenever a member of the team came across a bug, or the project manager wanted a new feature, they would create a new issue in Bitbucket Issues with a good description of what the bug or task was, and a screenshot of what was shown on the screen of the emulator or mobile phone. When a member wanted to work on a specific issue they would assign themselves to the issue.



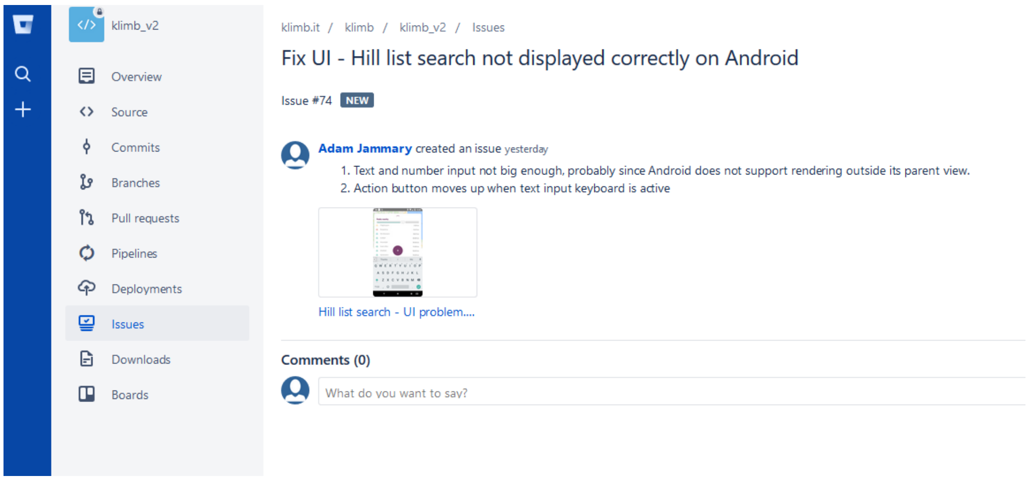
Issues

Issues (1–18 of 18)

Title	T	P	Status	Votes	Assignee	Created	Updated
#71: Fix UI - Suggest a peak	🟥	⌃	NEW		Adam Jammary	2018-04-11	3 hours ago
#70: Fix UI - Social - Stories - bottom text - top-right text	🟥	⌃	NEW			2018-04-11	yesterday
#79: Random focus to location at startup	🟥	⌃	NEW			yesterday	yesterday
#78: Fix - Social - Stories - share picture	🟥	⌃	NEW			yesterday	yesterday
#74: Fix UI - Hill list search not displayed correctly on Android	🟥	⌃	NEW			yesterday	yesterday
#73: Fix Peaks are not listed and displayed in map on startup	🟥	⌃	NEW			yesterday	yesterday
#72: Offline Fix	🟩	⌃	NEW			4 days ago	4 days ago
#3: Tutorial Messages	🟧	⌃	NEW			2018-01-03	2018-04-11
#69: Clean up i18n locale file content	🟩	⌃	NEW			2018-04-11	2018-04-11
#61: Google play debug version.	🟩	⌃	NEW			2018-04-03	2018-04-03
#59: Hill search function.	🟩	⌃	NEW			2018-03-16	2018-03-16

Figure 2: List of issues

Figure 2 shows how the Issues list looks like in Bitbucket. As you can see, it provides a great and clean overview of what the other members are working on, and what priority they assigned the issue.



Fix UI - Hill list search not displayed correctly on Android

Issue #74 NEW

Adam Jammary created an issue yesterday

1. Text and number input not big enough, probably since Android does not support rendering outside its parent view.
2. Action button moves up when text input keyboard is active

Hill list search - UI problem....

Comments (0)

What do you want to say?

Figure 3: An example of an issue

Figure 3 shows an example of one specific issue. We tried as much as possible to use screenshots after making major progress in the issue. We also tried to be as detailed as possible when making the description, and keeping the issue updated with comments as we progressed towards a solution.

2.4.3 Bitbucket GIT Repository

Klimb AS used Bitbucket as a GIT repository, and we were already familiar with how to use Bitbucket from other school projects. We also decided to use a GIT managament tool called Sourcetree [12] which provides a nice UI, and makes it easier to keep track of changes, commits, merges and branches. Bitbucket is a great tool for GIT management, but it could have a better project management integration with Trello, like Github has. On Github the issues are automatically linked with Scrum cards, that users can move to show the progression of what they are working on. Github also links commits if you mark them with "#issueNumber" and they will show up directly in the issue you are working on. On Bitbucket on the other hand, we had to manually add Trello cards with the issue, and also copy the link to a commit related to an issue, and paste it in the Bitbucket issue comment section. Not a big problem, but it was some extra work every time we created a new issue or commit.

We only worked in a separate repository branch called "android", while our employer would merge our branch into the master branch after testing it and verifying that it had no negative effects on the master branch. And when new features were added in the master branch, that worked on iOS, we would merge it into the android branch, debug it and get it to work on the Android platform.

3 Alternative Technologies

3.1 Responsive Web Design (RWD)

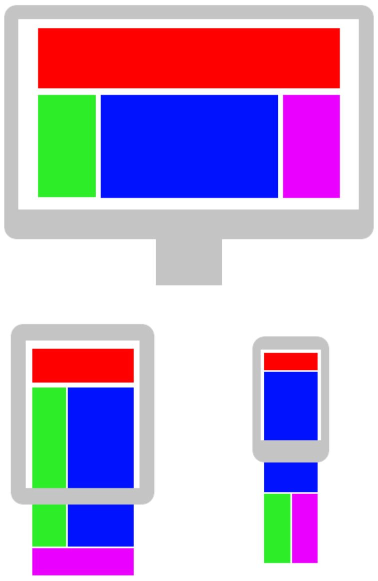


Figure 4: Responsive Web Design

When mobile devices got web browser support, developers could start developing web-based apps that could be used by any mobile device, independent of the underlying operating system and CPU architecture. Since physical screens on mobile devices vary, the web apps could look perfectly nice on some devices, but not perfectly scaled on others.

So instead of making static or fixed layouts, the responsive web design principle, tries to make dynamic layouts that adapts to the physical screen size, resolution or orientation. This is accomplished by using containers that layout their child components proportionally by using relative sizes like percentages instead of hard-coding sizes in static pixel or point units.

More and more web sites started moving over to responsive web design starting in early 2000, and Ethan Marcotte coined the term Responsive Web Design (RWD) in 2010 "and defined it to mean fluid grid / flexible images / media queries". [13]

3.2 Adaptive Web Design (AWD)

Adaptive web design is an additional design principle, that can be used by itself, but also in combination with responsive design. What makes adaptive design different, is that instead of creating one flexible page that handles all the different devices out there, you create multiple pages that each are optimized for a different mobile device family or specification. For example, one page for horizontal orientation and one for vertical, or one for high-resolution and a different one for low-resolution screens, etc.

Responsive design only, or at least mostly, utilizes client-side technology, like modern CSS and HTML, to detect the features of the current device. While adaptive design uses server-side technology to query for the most optimal layout and design to use for the specific device. Each page can use any web design principle available, even responsive design as mentioned earlier. [14]

3.2.1 Standard Layout

The standard layout consists of making separate pages for different features. For example for the most common resolutions, screen sizes or device types, like one for desktop and one for mobile etc. This means that the list of pages may become huge as more devices come out with a newer feature set.

3.2.2 Responsive Layout

The responsive layout tries to use fewer pages than the standard layout by combining adaptive and responsive web design. The responsive layout for each page can flexibly handle orientation changes, while the different adaptive pages can handle the various device families by choosing the best page layout on the server-side.

3.2.3 Scaled Layout

The scaled layout allows the use of absolute sizes in pixel or point units, because it will scale the sizes relatively to the size of the viewport automatically behind the scene.

3.2.4 Flux and Zoom Layout

The flux and zoom layout requires the use of absolute pixel or point sizes. It then scales everything up relatively using scaled layout, and usually only requires two different page layouts, one for desktop and one for mobile devices.

3.3 Progressive Web Apps

The progressive web design approach is to initially design only a simple core feature-set, that can be used by any device with simple web browser support. Then progressively, add layers of more complexity for more modern and feature-rich browsers. This allows everyone to be able to enjoy the most basic but important features of the web app.

A basic markup document is used to define the core feature-set, and web technologies like CSS and JavaScript are used to add more advanced feature layers. The download process is also optimized, since extra features are externally linked from the core features, so the device only downloads the layers that they can support. [15]

3.4 Web vs. Native

With a responsive web app you only need to develop one "version" of your app, and it will be available to anyone via a modern web browser without having to go through the process and restrictions of publishing to an app store for each native platform. This makes the process of development, testing, building and publishing easier and cheaper than developing a native app.

Even though reaching such a wide audience in such a simple way is a huge benefit, web apps also have their own drawbacks. The most obvious one is that they depend on an internet connection. Of course parts of the app can be cached offline, but the app will not be fully functional without an internet connection.

Another drawback is performance. Native apps can utilize OS-specific (Operating System) optimizations, while web apps fully depend on the performance of the web browser and on the speed and stability of the internet connection. Both options are improving and becoming more widely available at a very fast rate.

The UI (User Interface) varies on different Operating Systems, but the UI on web browsers is (almost) the same across platforms. This means that the user can be disappointed when the app does not behave and/or look as they expect in comparison with other native apps on that specific platform.

In addition to the UI being different, many OS-specific features will not be available via a web browser, or at least they will be much more restricted. Some examples are push notifications, access to contacts, the calendar or hardware such as GPS, camera etc. Although many of these features are becoming more available in modern web browsers.

Lastly, security is a major issue, although security is a concern no matter what type of app you develop and how it is distributed, being available directly on the web means that the web server needs to be secured. With a native app store, all security concerns are handled by the company behind the app store, but when distributing directly on the web, you need to manually keep the web server secured yourself. This means an additional cost or resource that needs to be taken into account when making a decision between web and native apps. [16]

3.5 Native vs. Hybrid

Developing a native app means that you have to develop a separate codebase for each native platform, since each platform uses their own programming language, framework, API, library dependencies etc. But as we previously described, a web app only needs a single codebase that works with most modern web browsers.

This is where hybrid apps come to the rescue, they allow you to write the codebase in one common language, like JavaScript or C#, and build the app for several platforms, while hiding the OS-specific code from the developers.

One of the biggest problem they all face, is how to provide a consistent look-and-feel, and also how to access all available platform- or device-specific features, because different platforms and devices vary a lot. They all use different approaches to try and solve this, and we will now have a look at the most popular hybrid app frameworks. [17]

3.5.1 Ionic

Ionic uses standard web technologies like HTML, CSS and JavaScript, and builds a cross-platform mobile app for both iOS and Android. What makes it hybrid is that it builds the app for the native platform, which means the user opens an app installed on the device, but it is still mostly a web app in regards to functionality and performance.

The programming framework is based on Angular, which is "a JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications". [18]

But it can also provide native features such as geolocation, push notifications, camera etc. via Cordova, which is "a mobile application development framework originally created by Nitobi". [19]

What makes Ionic, and most of these hybrid frameworks so powerful and popular, is the ecosystem around them, consisting of so many developers contributing additional features, as needed by the community and as new features become available on the native platforms.

Ionic provides a visual editor called "Ionic Creator" to easily design the UI, and "Ionic View" to share the app with other users in the community via invitations. [20]

3.5.2 Apache Cordova (Adobe PhoneGap)

- "Apache Cordova (formerly PhoneGap) is a mobile application development framework originally created by Nitobi. Adobe Systems purchased Nitobi in 2011, rebranded it as PhoneGap, and later released an open source version of the software called Apache Cordova." [19]
- "The software was previously called just 'PhoneGap', then 'Apache Callback'." [19]
- "PhoneGap is Adobe's commercial version of Cordova along with its associated ecosystem." [19]

As you can see above, it's easy to get confused by the difference between Apache Cordova and Adobe PhoneGap, but John M. Wargo provides a nice summary:

"Apache Cordova is the current name for the open source project formerly known as PhoneGap. Adobe PhoneGap is Adobe's distribution (flavor) of Apache Cordova, with some extra capabilities added by Adobe.". [21]

PhoneGap also uses web technologies like HTML, CSS and JavaScript, but instead of locking JavaScript to one framework such as Ionic and Angular, PhoneGap allows you to use any JavaScript-based framework. [20]

3.5.3 Xamarin

Microsoft's Xamarin differs from the other frameworks by using C# as the programming language instead of JavaScript, which in turn is compiled in to native code for the specific platform. This cross-compilation is provided by the Mono platform. [22] While the previous frameworks could best be described as web apps running on a native platform, Xamarin could better be described as actually creating native apps running on a native platform with all the benefits of performance and native look and feel.

Although Xamarin can take advantage of many native features, they cannot use all of them. It will take some time from when the native platform releases a new feature in their API, until Xamarin implements that feature in to their API, which in turn developers can use in their Xamarin C# codebase. Users can also override shared API, and write OS-specific code for features that are just too different and too specific for that OS, which gives developers a lot of flexibility. [20]

4 Technologies

4.1 React Native

Klimb AS had already chosen React Native as their hybrid framework, but they had only developed for the iOS platform and tested on iPhone devices. And as we explained in the introduction our job was to make the app functional on Android as well. In theory this should have been pretty straightforward, but this turned out to be more cumbersome than we expected. We will go more in detail on the challenges later on when we discuss the implementation details. For now we will talk about the chosen React Native framework, and explain how it works in general.

4.1.1 React (JavaScript Library)

React Native, like most of the previously mentioned hybrid frameworks, is based on the JavaScript language, or more specifically the React JavaScript library, also called "React.js" or "ReactJS" for short. React is used to build large web-applications where data can change without having to reload the entire page every time something changes. It is maintained by huge organizations such as Facebook and Instagram, but also by a large community of developers around the world. [23]

4.1.2 ReactJS vs. React Native

A standard web application uses HTML as the markup language to define the layout and content of the UI, CSS to style or format it, and JavaScript to dynamically manage it. The web browser contains the engine that manages the interaction between these elements and produces the final rendered output, as well other I/O (Input/Output) functionality between the user and the app.

React uses something they call JSX (JavaScript XML) [24] which is basically a combination of HTML, JavaScript and CSS and accomplishes the same.

The main difference between ReactJS and React Native, is that ReactJS is a general JavaScript library which builds a web app that can be viewed in a web browser, while React Native builds an app that runs natively on an Android or iOS device.

4.1.3 Bridging

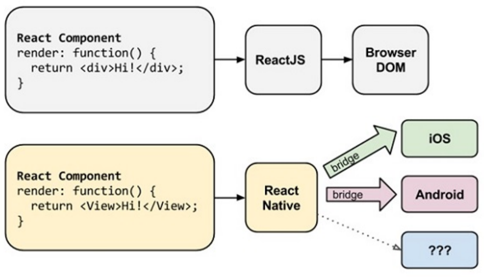


Figure 5: JavaScript Bridge
[25]

The React Native code is run natively on the device thanks to a feature called "Bridging", which allows the JavaScript code to communicate with the device processor during runtime. The React Native project contains a small default project for each platform, one for iOS and one for Android, which contains the platform-specific code, and the code which manages the communication between JavaScript and the CPU (processor), as these cannot communicate directly.

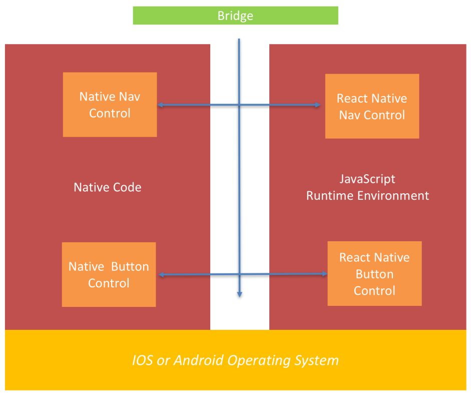

Bridge
Native Nav Control
React Native Nav Control
Native Code
JavaScript Runtime Environment
Native Button Control
React Native Button Control
IOS or Android Operating System

Figure 6: React Native Controls
[26]

Each React Native control maps to a native control. For example, a control which handles button presses will be implemented differently on different platforms. React Native will for example provide a "react-native-button" control written in JavaScript, and this control will do all the necessary mapping behind the scenes and communicate with the correct OS-specific I/O-control. The OS-specific control could be a Java (on Android) or Swift/Objective-C (on iOS) API function call. Button handling may perhaps be very similar across platforms, but other functionality like geolocation or networking may be implemented very differently. To summarize, React Native bridging provides a very nice abstraction layer for developers.

4.1.4 Building

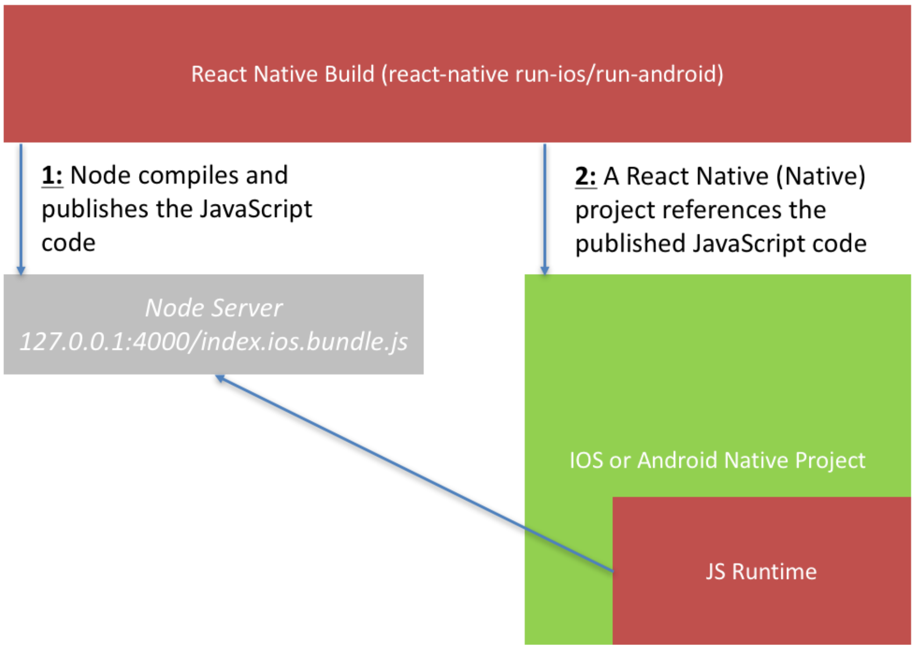

React Native Build (react-native run-ios/run-android)
1: Node compiles and publishes the JavaScript code
2: A React Native (Native) project references the published JavaScript code
Node Server 127.0.0.1:4000/index.ios.bundle.js
IOS or Android Native Project
JS Runtime

Figure 7: React Native Build Process
[26]

After setting up all the necessary dependencies and build environment, the developer can build the project by running a simple command "react-native run-ios" or "react-native run-android". This command will start the NodeJS web server [27] which will handle the JavaScript code and provide debugging information, then it will build the native Android or iOS project, and finally deploy it to the device using the Gradle build system. [28]

When running this command, it will not deploy an APK (for Android) or IPA (for iOS) package to the device, it deploys a bundle. Which means the device needs to retain a connection between the device and the NodeJS web server while debugging. When the connection drops, the command needs to be run again to re-deploy the bundle and re-establish the connection. When the app is ready for deployment, the packages can be manually created by running the Gradle tool with various options.

4.1.5 Reloading

React Native also provides a really nice debugging feature which, in theory, will save developers a lot of time, but in practice may cause a lot of strange caching issues that can waste a developers time. This feature is called "Reloading", which will only change the JavaScript code that has been changed after the last build, and since JavaScript does not need to be compiled, the app should, in theory, just refresh and run the newly updated code. [26]

4.2 Project Dependencies

Dependency	Version	Description
babel-core	6.26.0	Babel compiler core
babel-jest	22.2.2	Simple testing configuration for React Native with Jest
babel-preset-react-native	4.0.0	Babel presets for React Native applications
create-react-class	15.6.3	A drop-in replacement for React.createClass
jest	22.3.0	JavaScript testing framework
prop-types	15.6.0	Runtime type checking for React props and similar objects
react	16.2.0	An npm package to get immediate access to React, without also requiring the JSX transformer.
react-native	0.53.0	React Native lets you build mobile apps using only JavaScript
react-native-background-geolocation	2.11.0	Background location-tracking and geofencing module with battery-conscious motion-detection intelligence
react-native-camera	0.12.0	The comprehensive camera module for React Native
react-native-fbsdk	0.7.0	React Native FBSDK is a wrapper around the iOS Facebook SDK and Android Facebook SDK
react-native-i18n	2.0.11	Integrates i18n (internationalization) with React Native
react-native-image-gallery	2.1.5	A pure JavaScript image gallery component for React Native apps
react-native-keyboard-aware-scroll-view	0.4.3	A ScrollView component that handles keyboard appearance and automatically scrolls to focused TextInput
react-native-maps	0.20.1	React Native Map components
react-native-md5	1.0.0	Native JS function for hashing messages with MD5
react-native-svg	6.1.4	Built to provide a SVG interface to react native
react-native-swipe-gestures	1.0.2	React Native component for handling swipe gestures
react-native-swiper	1.5.13	Swiper component for React Native
react-navigation	1.0.3	Extensible yet easy-to-use navigation solution based on Javascript
react-native-typescript-transformer	1.2.3	Seamlessly use TypeScript with react-native
react-test-renderer	16.2.0	Experimental React renderer that can be used to render React components to pure JavaScript objects
rxjs	5.5.6	Reactive Extensions Library for JavaScript
typescript	2.7.1	Language for application-scale JavaScript, adds optional types, classes, and modules to JavaScript.
util	0.10.3	node.js util module as a module
whatwg-fetch	2.0.3	Promise-based mechanism for programmatically making web requests in the browser

[29] NPM Package Search (npmjs.com)

4.3 Package Managers and Build Tools

4.3.1 Node.js 7.10.1 med NPM 4.2.0

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world. [30]

4.3.2 Yarn 1.3.2

Yarn is a new package manager that replaces the existing workflow for the npm client or other package managers while remaining compatible with the npm registry. [31]

4.4 IDE (Integrated Development Environment)

4.4.1 Visual Studio Code 1.19.2

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.[32]

4.5 Emulators and device testing

4.5.1 Android Studio 3.0.1 for Windows

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.[33]

4.5.2 Android Emulators (run through Android Studio)

- Nexus 5X x86 - Android 8.1 - API 27 - 1080x1920
- Nexus 6 x86_64 - Android 6.0 - API 23 - 1440x2560

4.5.3 Android Devices (our own Android devices)

- Samsung Galaxy S6 edge (SM-G925F) - Android 7.0 - API 24 - 1440 x 2560
- Samsung Galaxy S7 (SM-G930F) - Android 7.0 - API 24 - 1440x2560

5 System Architecture

5.1 Authentication using Facebook SDK

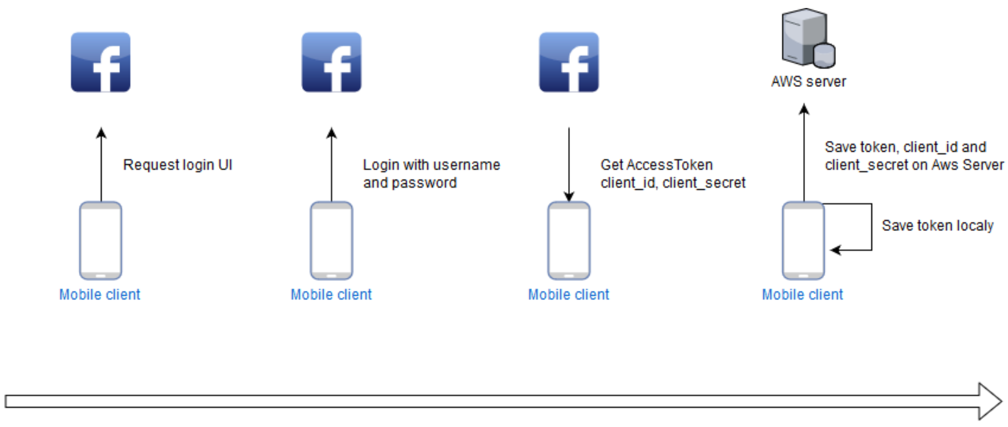

AWS server
Request login UI
Login with username and password
Get AccessToken client_id, client_secret
Save token, client_id and client_secret on Aws Server
Save token localy
Mobile client
Mobile client
Mobile client
Mobile client

Figure 8: First time authentication

Figure 8 shows how the application does first time authentication using Facebook SDK. First, the phone will check if any tokens [34] are stored locally on the phone and if they are still valid, meaning the timestamp has not expired. If no tokens exist, it means that this is the first time starting the application, and it will request a Facebook authentication page using the Login Manager from the Facebook SDK.

The user will then log in with their Facebook account (email and password) and receive an access token, which consists of a client ID and a client secret key. These are then stored locally on the phone and on the AWS (Amazon Web Services) back-end server, and the user is now authenticated and will receive user data from the AWS server.

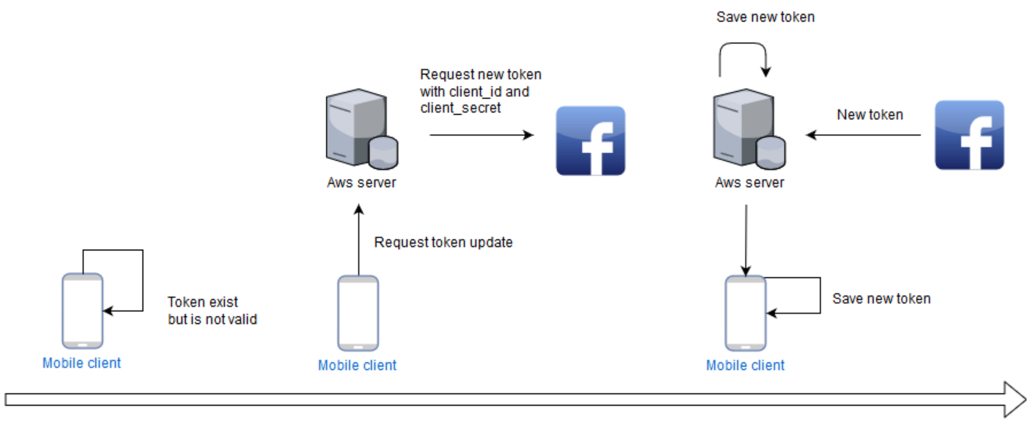


Figure 9: Authentication when token is expired

Figure 9 shows how the authentication is done when the application starts, and the phone already has a token saved locally but the timestamp has expired. The phone will start by requesting a token update from the AWS server, and the server will then request a new token from Facebook using the previously stored client ID and secret key. Facebook will then send the new access token to the AWS server, the server will then update its existing local token and pass the token on to the phone, which saves it locally. This happens in the background so the user does not have to log back in again with their email and password after the first initial login.

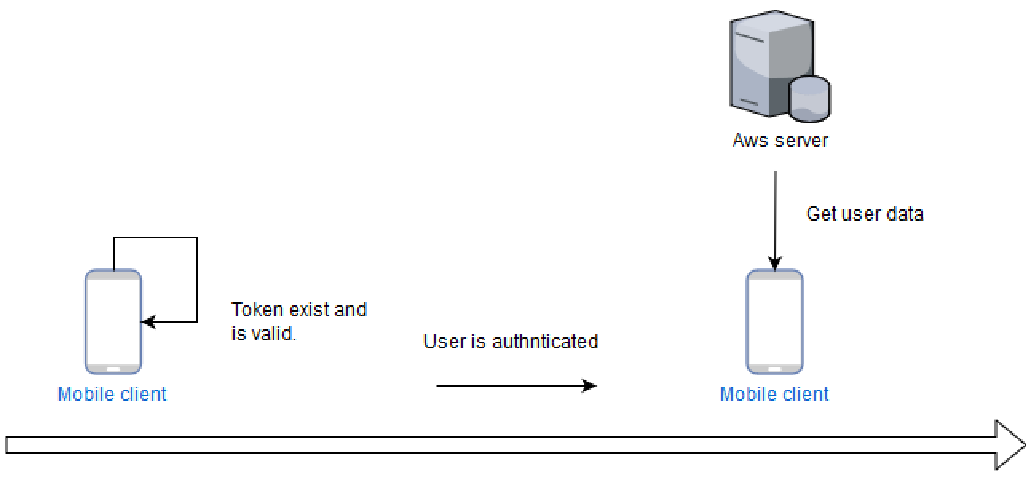


Figure 10: Authentication when token is valid

In figure 10, the phone will check if it has a token stored locally, and if the timestamp is still valid. If the token exists and is valid, the user is authenticated and will receive user data from the AWS server.

5.2 Services

Services are background processes that the user never sees, they handle the server interactions, the local data storage and general data processing, before passing them on to the components to be displayed.

Observer Pattern

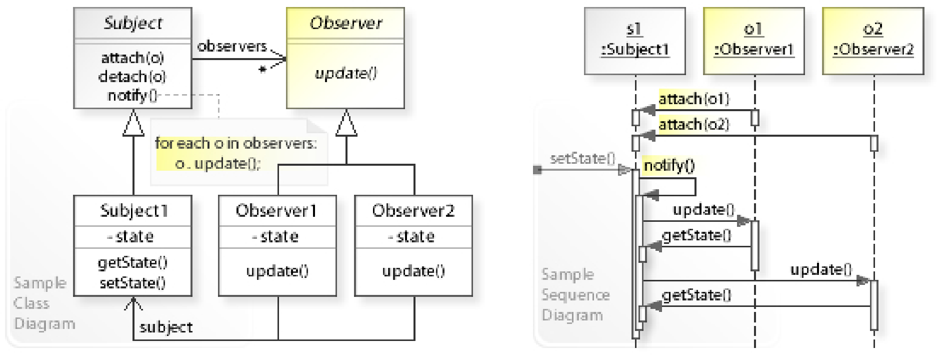


Figure 11: Observer Pattern
[35]

The observer pattern is a well-known software design pattern mostly used when developing services or other back-end logic. The pattern involves one object called the subject which manages a list of other objects called observers. The observers start by subscribing or registering with the subject.

In a normal client-server pattern, the observers (clients) would have to query the subject (server) every time they wanted something. In the observer pattern however, it is the job of the subject to notify all of its observers whenever a change in the state has happened, and the observers need to update their local states.

It is the most used pattern in web applications based on the MVC (Model-View-Controller) design pattern [36], and in event management systems used in various UI and Window-based frameworks.

The event management system for example runs separately from UI rendering, and whenever it detects a button press or some other event, it notifies registered listeners about the event, and the listeners can in turn take an action based on the event. [35]

src/services/TokenService.ts

This example will describe step by step how the Token service works. And how the "MainPage" component (src/components/MainPage.js) subscribes to this service.

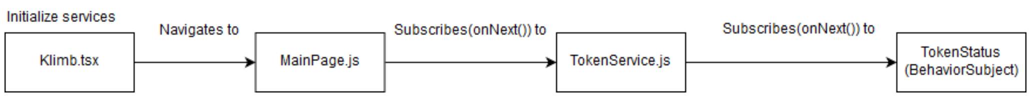


Figure 12: TokenService initialization

As figure 12 shows, all the services are initialized in the "Klimb.tsx" file.

```
this.token = new TokenService();
```

"src/Klimb.tsx" then sends the service as a prop to the "KlimbNav" component, which will send the services further down the chain to the components that need them. "MainPage" now subscribes to the "TokenService" in its own "componentWillMount()" function, and passes it on to the "onNext" callback function. The callback function in this case is the "onTokenStatus()" function in "MainPage". "MainPage" then pushes the service to a local subscriptions array like so:

```
subscriptions = [];
componentWillMount() {
    this.subscriptions.push(
        this.props.screenProps.service.token
        .subscribeToTokenStatus(this.onTokenStatus)
    );
}
```

In "TokenService.ts", the Token Service subscribes to the Behavior Subject named "tokenStatus", and passes on the callback function (onNext/onTokenStatus) it got from MainPage.

```
this.tokenStatus = new BehaviorSubject(TokenStatus.Loading);

subscribeToTokenStatus(onNext: (value: TokenStatus) => void) :
Subscription {
    return this.tokenStatus.subscribe(onNext);
}
```

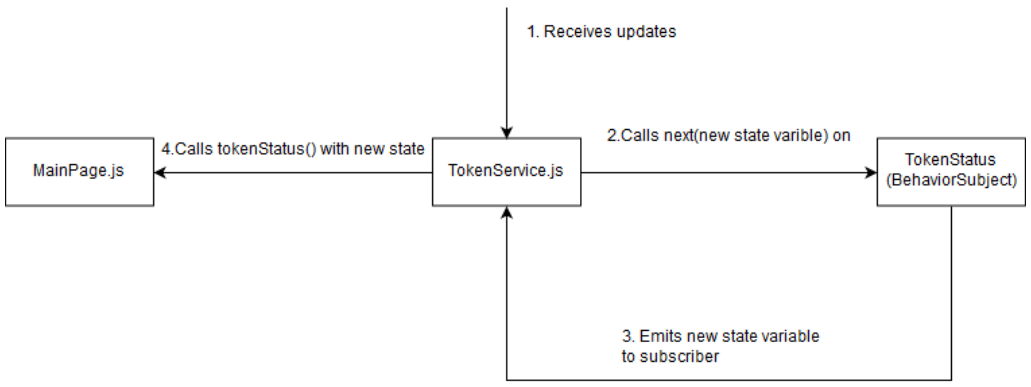

1. Receives updates
4.Calls tokenStatus() with new state
2.Calls next(new state varible) on
MainPage.js
TokenService.js
TokenStatus (BehaviorSubject)
3. Emits new state variable to subscriber

Figure 13: TokenService when receiving updates

Figure 13 shows what is happening when "TokenService" receives updates.

1. Receives updates from the AWS server.
2. "TokenService" calls the "next()" function of "BehaviorSubject" with a new state variable. ex:

```
this.tokenStatus.next(TokenStatus.Loading);
this.tokenStatus.next(TokenStatus.Valid);
this.tokenStatus.next(TokenStatus.None);
```

3. "BehvaiorSubject" then emits the updated state to all its subscribers, which in this example is "TokenService".
4. "TokenService" will then update all of its subscribers, which in this example is "MainPage". "MainPage" will then call its own registered "onNext" function, which in this case is the "onTokenStatus()" function, which takes in the new token state as a parameter. ex:

```
inLogin = false;

onTokenStatus(tokenStatus) {
    if (tokenStatus === TokenStatus.None && !this.inLogin) {
        this.inLogin = true;
        this.props.navigation.navigate('LoginPage');
    }
}
```

5.3 Components

A component is the view that is displayed on the screen for the user to see or interact with.

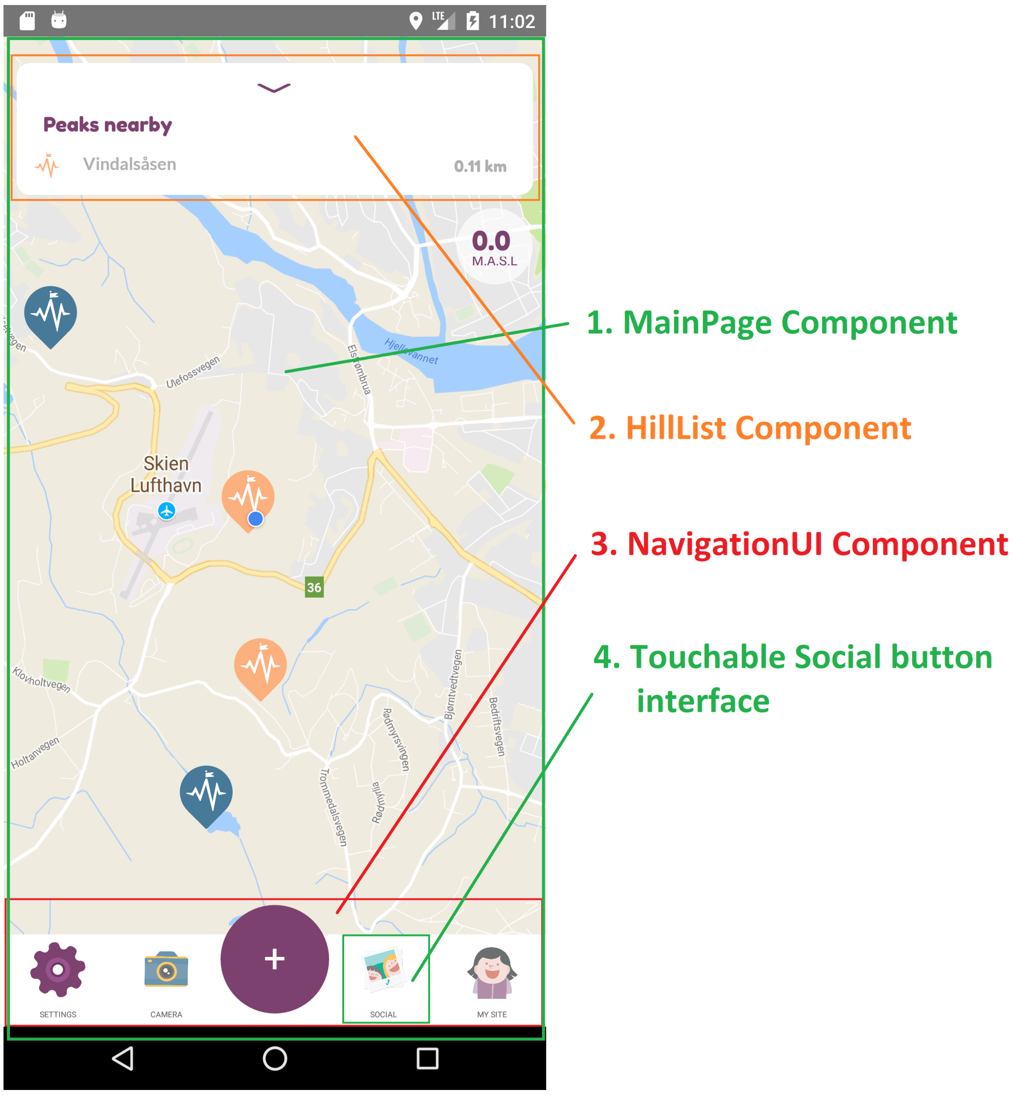


Figure 14: Component example for the Klimb application

5.4 Internationalization

5.4.1 Localization (i10n)

"Localization refers to the adaptation of a product, application or document content to meet the language, cultural and other requirements of a specific target market (a locale)." [37]

As we can see from the W3C definition, localization means much more than just translating text from one language to another, it includes customizing the entire application to the target culture.

This can include well-defined, but not trivial, things like how decimal numbers, dates, times, currencies etc. are written or separated, like commas, dots/periods, spaces, hyphens, forward/back-slashes etc.

But it can also include less defined things that are specific to each culture, like symbols, icons or even colors. In some cultures, certain colors can refer to things like death or happiness, and using the wrong colors in the wrong context may not provide the expected result. Certain countries even have legal prohibitions against certain images, words or topics.

As we can see, localizing an application requires much more work than just direct translation, even direct translation may not work if the culture or hidden implications are not taken in to account. Localization would require hiring experts in the culture or cultures you are trying to target, including linguists, lawyers etc. [37]

5.4.2 Internationalization (i18n)

"Internationalization is the design and development of a product, application or document content that enables easy localization for target audiences that vary in culture, region, or language." [37]

Again, as we can see from the W3C definition above, even the word internationalization entails more than just translating the text to multiple languages. One could say that internationalization is the actual process of localizing the application.

In this report we will refer to internationalization as having a framework in place to easily being able to add multiple text translations by non-programmers, without having to access the source code.

5.4.3 react-native-i18n

We had the choice between developing an i18n framework from the grounds-up, or see if there already existed a package that handled internationalization for the React Native framework. To avoid re-inventing the wheel, we started by searching for existing i18n packages, and found the package "react-native-i18n" by "AlexanderZaytsev". [38]

We started by following the installation instructions, but unfortunately the automatic setup did not work on our existing solution. The "link" command which is supposed to link all the dependant libraries to the native iOS and Android projects failed to do so. We had to instead manually link the libraries, and fortunately enough, the documentation was very detailed and gave clear steps on how to achieve this for both native platforms. Our employer who is responsible for the existing iOS platform performed the manual linking on iOS, and we linked the library on the Android project.

src/i18n/i18n.js

We created the main i18n entry-point (src/i18n/i18n.js) which manages which language should be used based on the regional locale settings of the device. Norwegian contains two variations, "Bokmål" and "Nynorsk", and in addition there is a general abbreviated name called "no". So we added a fallback for "nb", "nn" and "no" to point to the same "no" file. But if the company decides to create separate translations for "nb" and "nn", they can easily just link to each file sparately.

```
import I18n from 'react-native-i18n';
import nb from './locales/no';
import nn from './locales/no';
import no from './locales/no';
import en from './locales/en';

I18n.fallbacks = true;

I18n.translations = {
    nb,
    nn,
    no,
    en
};

export default I18n;
```

src/i18n/locales/*.js

We then created separate locale files for each language we currently support, for now these are English and Norwegian, so we added "en.js" and "no.js" to the "locales" sub-directory. Even though the files are JavaScript files, and contain "programming" syntax, they are separate from the logical source code. It should be easy enough for a non-programmer to later on add another language by copying an existing file, and adding more translations by using the existing content as a template.

```
import I18n from "../i18n";

export default {
    ...
    component: {
        header: 'Header text in local language',
        content: 'Content text in local language ...',
    },
    ...
};
```

src/components/component/*.js

The next and final step was to replace existing static text strings with references to the i18n localized content. This means that each component that wants to use i18n must first import the package:

```
import I18n from '../../i18n/i18n';
```

then, reference the content in the localized i18n files (en.js, no.js etc.) by replacing the existing code:

```
<Text>Header text - one specific language</Text>
```

with new code:

```
<Text>{I18n.t('component.header')}</Text>
```

It was nice to find an existing package that worked and provided the results we were expecting. Now, the application is ready to be translated to as many languages as our employer wants, which also means they can target a more international audience instead of being limited to only Norwegian-speaking users as they were before.

We have to mention that the i18n package does not handle any of the more advanced parts of localization as described previously such as cultural customization, it only handles direct translation of text, which is exactly what we were looking for.

6 Imlementation

6.1 Fixing UI Problems - Android vs. iOS

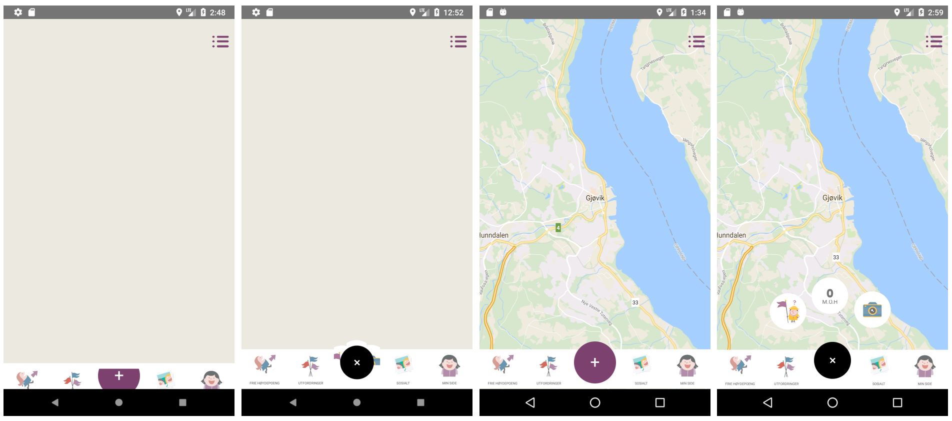


Figure 15: Left: UI problem on Android. Right: UI problem resolved.

One of the first issues we came across, was that the UI rendering is handled completely differently by Android compared to iOS. For example, much of the existing UI design was based on some components overlapping other background components. As shown in Figure 15, the purple and black buttons with the plus sign on the left figures are supposed to look like the buttons on the right figures.

This is easily accomplished on iOS by using the "overflow" style property as described by the official React Native documentation: "overflow controls how children are measured and displayed. "`overflow: 'hidden'`" causes views to be clipped while "`overflow: 'scroll'`" causes views to be measured independently of their parents main axis." [39]

The above quote is from the documentation as per version 0.46, and it did not mention any problems or differences with how the "overflow" style property worked on the Android platform compared to the iOS platform. The documentation implied that it should work on both platforms, since React Native is per definition a cross-platform framework. This means that we spent a lot of time trying to get it to work as expected.

But after looking through dozens of forums with frustrated developers telling the same story: "it works perfectly fine on iOS but not on Android", we concluded that it was time to find an alternative solution for Android, and hope it would not break the UI on iOS.

The problem is that Android cannot draw components outside of its parent component, so the solution is to wrap all the involved components inside a new parent component with a much bigger size that will fit all the components.

By making the background color of the parent transparent, it will not be visible, and if positioning becomes a problem, we can force the position by using absolute instead of relative positioning. Another issue that comes up is deciding which component should be rendered above other components, this is solved by setting the "zIndex" style property. [40]

The official React Native documentation was updated after version 0.47, and now it specifically mentions the problem between Android and iOS: "overflow: visible only works on iOS. On Android, all views will clip their children." [40]

Most of the issues we had in the first part of the project were UI issues like the example above, where the existing UI design relied on overflowing the parent component since it worked perfectly fine on the existing platform (iOS).

We used a combination of the above mentioned techniques, wrap in a bigger and transparent parent component, test absolute vs. relative positioning and setting a higher z-index to force it to render over other components with a lower z-index.

6.2 Navigation UI

6.2.1 Fixing the navigation UI

The navigation UI was one of the first issues that we worked on before we had gained any experience in React Native. Just as a side-note, the map is not displayed on some of the images below due to emulator issues, the map was displayed correctly on a physical mobile device.

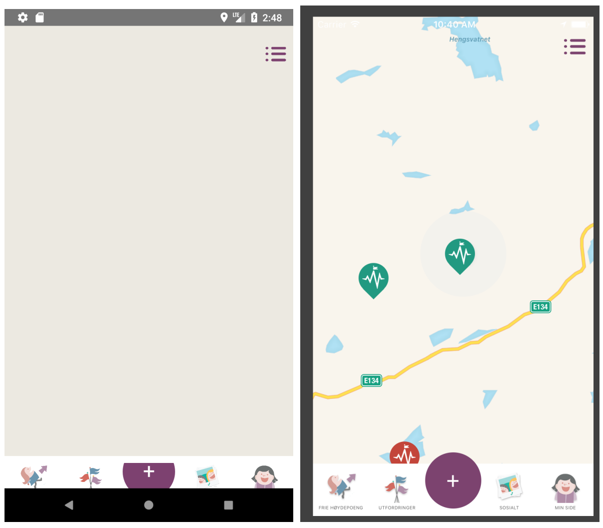


Figure 16: Right to left: Android navigation UI before we started, compared to the iOS.

The first problem was that the navigation UI did not look like it did on iOS, see Figure 16. This was caused by the fact that Android does not support the 'overflow' style property, and that the parent view on Android clips the view of its children components if they are outside the scope of the parent view. However, this was not yet in the official React Native documentation when we started working on this issue. So it took us some time to figure out that this was the case.

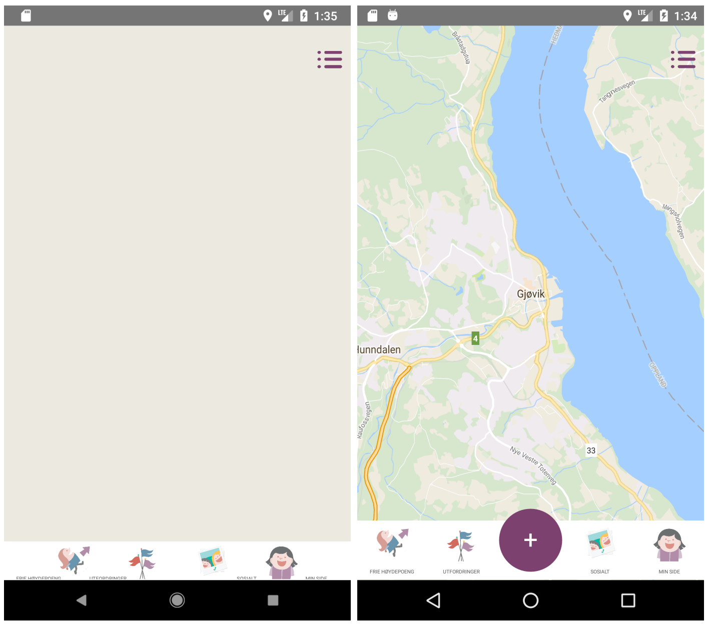


Figure 17: Left: Navigation UI when moving action-menu out of the navigation bar. Right: Displaying action-menu above navigation UI.

The first thing we tried was changing the z-index and setting the parent with the "`position: 'absolute'`" property in hope that the button would then render above the map. But sadly this did not work either. We then changed the size of the navivigation UI to cover the whole screen, this caused everything to look good, but we were not able to interact with anything except for the navigation UI, since we were not able to click through a view. The solution we found later was using this view component property "`pointerEvents = 'box-none'`", this made us able to click through the view and interact with the underlying views.

The next thing we tried was moving the entire action-menu (the purple button in the middle shown on the right in Figure 17) out into "MainPage", so that it would render as a sibling of the "KlimbMap" component (src/components/map/Map.js), and would be displayed on top of the map instead of underneath it. However doing this caused the rest of the buttons in the navigation UI to not display as they should, see Figure 17. The solution was to add an empty view with the same size as the action-menu we removed, so that it filled the void space when rendered.

After this was fixed, the navigation UI looked like it should. There were still some smaller style properties that we needed to change to make the buttons and text look good, but we always made sure to use platform checks if we thought our new values would cause problems on iOS.

```
    ...Platform.select({
          ios: {
            zIndex: 100
          },
          android: {
            zIndex: 200
          },
      });
```

Now the navigation UI and the action-menu looked good, but the action-menu items (the buttons that show up on the screen when pressing the action-menu, shown on the right in Figure 18) did not display as they should.

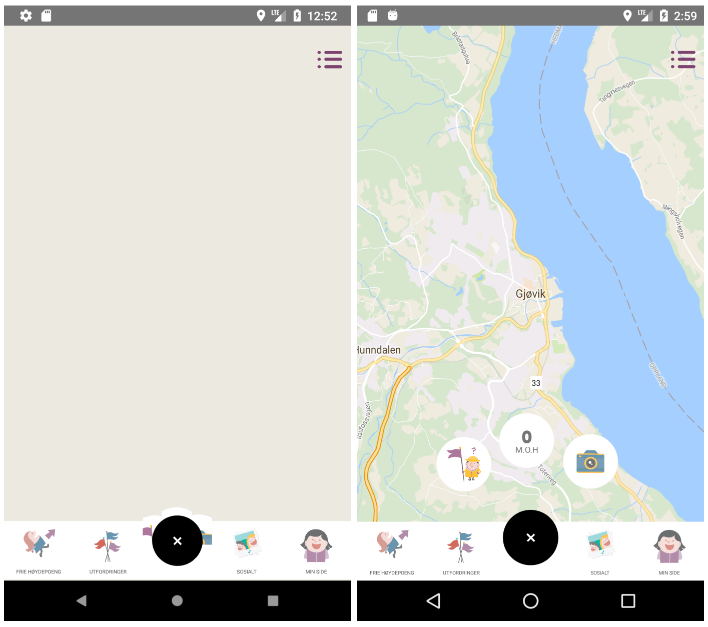


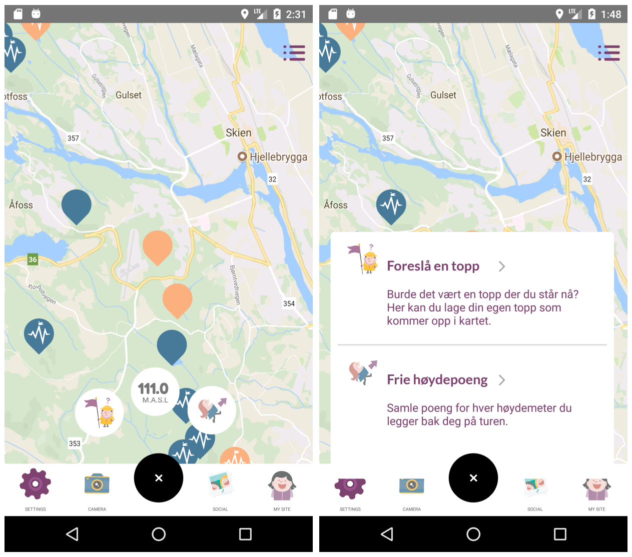
Figure 18: Left: Navigation UI unable to display items when action-menu is pressed. Right: Displaying items correctly.

Displaying the action-menu items was solved by increasing the size of the view. But we still had some minor problems that the action-menu items view was overlapping other buttons in the navigation UI, this is when we discovered what we could use "`pointerEvent='box-none'`", which made us able to click on the buttons underneath.

If we had known about this earlier we could have spared ourselves a lot of time and not moved the entire action-menu outside the navigation UI. This was fixed later by Sindre when Klimb AS wanted to change the action-menu items to be more descriptive using the solution of giving the navigation UI a bigger part of the screen as view size, and adding "`pointerEvent = 'box-none'`".

6.2.2 UI change

Halfway through our project, Klimb AS decided that they wanted to do some changes to the UI, one of them was making the action-menu items more descriptive. We got some images from Klimb AS with the new design that showed us what they wanted the new action-menu items to look like.


Foreslå en topp

Burde det vært en topp der du står nå? Her kan du lage din egen topp som kommer opp i kartet.

Frie høydepoeng

Samle poeng for hver høydemeter du legger bak deg på turen.

Figure 19: Left: UI before changes.
Right: After changes.

We started by creating a new "ActionMenuItem" class (src/components/nav/ActionMenu-Item.js) that would display the new buttons as shown in Figure 19. We also thought it would look good if we added an animation when pressing the action-menu button, so that if the button was pressed, a view containing the new menu item buttons would appear from the bottom of the screen, animating together with the rotating action-menu animation. Since some of the animation values and structure was already set up from the previous "ActionMenuItem" class, it was not that hard to achieve.

We also had to implement which buttons would be displayed where, based on the activity the user was in, or what screen the user was currently in. The button functionality was already implemented from the previous action-menu items, so we just reused the same methods by mapping them to the new buttons.

6.3 Independent Task - Tutorial Popups

6.3.1 Task description

The app had an existing introduction tutorial that started the first time a user logged in, the tutorial had some pages describing the general use of the app. Our employer wanted in addition a system that could show a small popup with information describing a specific feature on a page.

This popup should only be displayed the first time a user visits a page, and then hidden the next time they visit that same page. They should also have the possibility of resetting the popup visit-state in Settings to make the popups appear again in case they forget the instructions, or just want a refresher.

6.3.2 Discussion

We started by trying to get a better understanding of the existing system architecture, like how all the components and services worked together. After gathering some information, we started discussing some alternatives for where to start.

- How should the popups be managed?
- By each component?
- By a service, a manager?

Option 1

```
<Popup pos="x, y">
    <Button id="id1" ... />
</Popup>
```

Option 2

```
<Popup pos="x, y", ...>
    <View><Text /></View>
</Popup>
```

Option 3

```
<Popup id="pop1" pos="x, y">
    <PopupButton id="popItem1_1" ... />
</Popup>
<Popup id=pop2 pos="x, y">
    <PopupButton id="popItem2_1" ... />
    <PopupButton id="popItem2_2" ... />
</Popup>
```

Option 4

```
PopupService popupService = new PopupService();
<Popup ...>
```

We decided to try out option 3 first, as this option made the most intuitive sense. And maybe try to combine with a service or a manager later as in option 4, so we can store the state of the popups so they stay hidden the next time the page is visited.

After researching for existing React Native solutions that could provide the features we wanted, we thought we could use React Navigation [41] for navigating between child popups, so that we have a main popup bubble which has multiple popup items (children or pages) that could appear in sequential order. We could also use React Native Async-Storage [42] to store the visited state of the pages and popups locally on the device.

We also didn't know how the popups should appear.

- Should they appear automatically when the page loads?
- Have a tutorial icon somewhere on the page?
  - The user has to click the icon to see the popup tutorial?
- Override the click event trigger of the target?
  - Make it show the popup on the first click?
  - And then do what it's actually supposed to do on the next click?

Another unknown was how to link the tutorial popup to the part of the page it was trying to describe.

- Have an arrow pointing in the direction of the target component?
- Gray out the background with a transparent color?
- Highlight the target component with a circle or some visual queue?

6.3.3 Solution

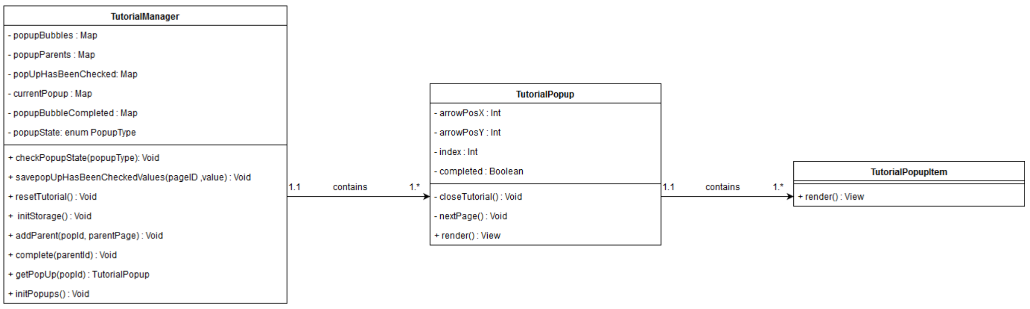


Figure 20: Tutorial Manager class diagram

src/components/popup/TutorialManager,js

The Tutorial Manager manages all the popups for the entire app. We started by letting each component do some of the management, but started moving more and more over to the manager to avoid code duplication in various places in the code.

To add a new popup, one starts by adding it to "popupBubbles" (a map data structure) in the "initPopups()" method, "popupBubbles" then maps a unique identifier to a "<TutorialPopup>" UI component.

```
    this.propTypes.popupBubbles.set(
        TutorialKeys.MAIN_PAGE,
        [
            <TutorialPopup>
                <TutorialPopupItem />
                <TutorialPopupItem />
            </TutorialPopup>,
            <TutorialPopup>
                <TutorialPopupItem />
            </TutorialPopup>
        ]
    );
```

src/config/TutorialKeys.js

Every page that will show a tutorial popup needs to have a uniquely identifiable key registered in "TutorialKeys", we decided to use the filename to make management easier.

```
    export const enum TutorialKeys {
        MAIN_PAGE = "MainPage",
        SOCIAL_PAGE = "SocialPage",
        RUN_PAGE = "RunPage",
        CAMERA_EDIT_FRAME = "CameraEditFrame",
        MAP_CARD = "MapCard"
    }
```

TutorialPopup.js and TutorialPopupItem.js (src/components/popup/)

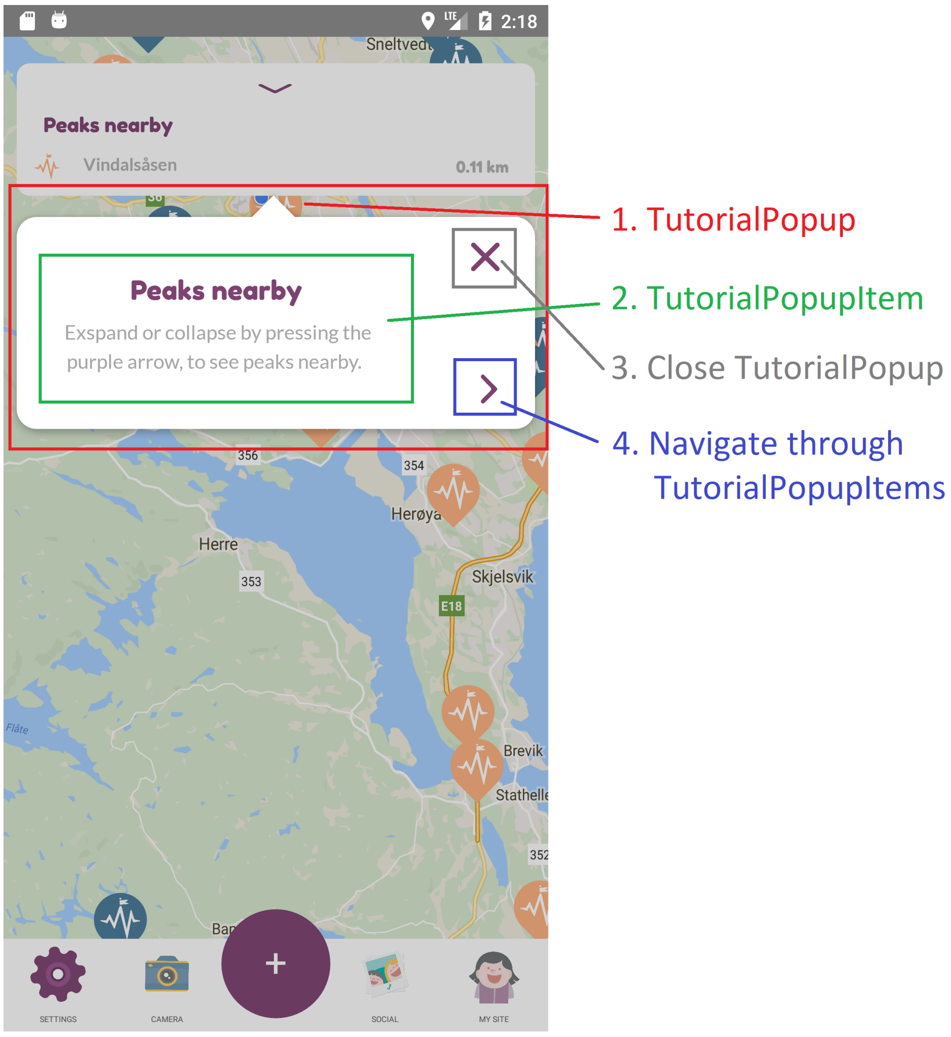


Figure 21: TutorialPopup view

"TutorialPopup" represents the popup bubble which can contain one or more popup items. The items are the content within the surrounding bubble, each item also represents a page within the bubble, and one can navigate to the next page, if it exists, by clicking on the next arrow.

When there are no more items or popup bubbles left, "TutorialPopup" will let "Tutorial-Manager" know, and "TutorialManager" will update the state of the popup. When the component page tries to request a new popup after this, "TutorialManager" will return a "NULL" object which means it will not be rendered.

We chose to gray out the background to make the popup bubble more visible, and we chose to display an arrow towards the target component to highlight which feature of the page we are trying to describe. The position and size of the popup, as well as the arrow, are handled by "TutorialPopup".

Component Page - For example src/components/MainPage.js

The component page that will show a popup needs to start by importing "TutorialManager" and "TutorialKeys".

```
    import TutorialManager from './popup/TutorialManager';
    import {TutorialKeys} from '../config/TutorialKeys';
```

Then we need to link the page to the popup, this would not be necessary if the popups were managed by each page, but since we decided to move management to "TutorialManager", we need to manually link the page instance to the unique page ID in "TutorialKeys" when the component mounts.

```
    componentWillMount() {
        TutorialManager.addParent(TutorialKeys.MAIN_PAGE, this);
    }
```

The popups are now ready to be rendered to the screen, and to achieve this we need to ask "TutorialManager" for the popup belonging to this page.

```
    render() {
        <View>
            {TutorialManager.getPopUp(TutorialKeys.MAIN_PAGE)}
        </View>
    }
```

7 QA - Testing

7.1 Debug Developer Menu

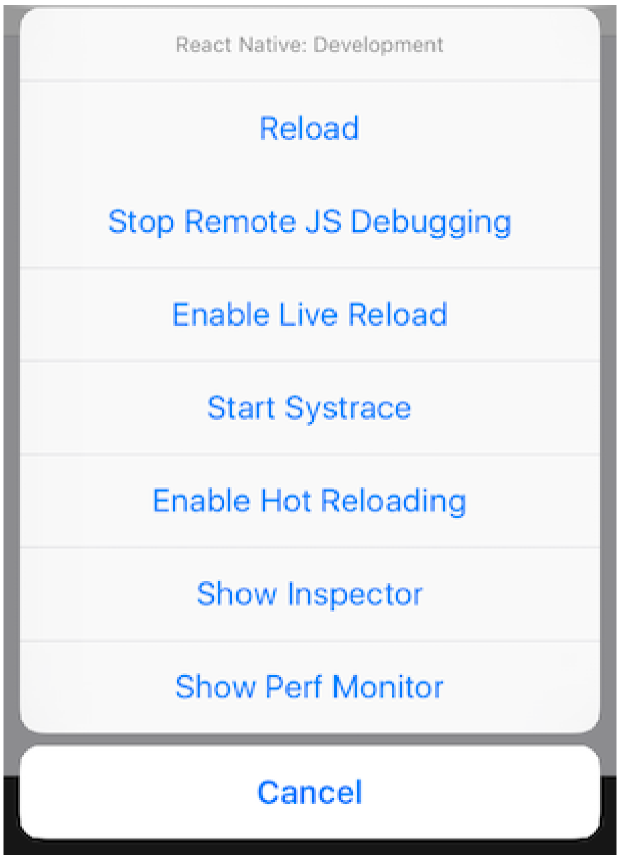


Figure 22: Debug Developer Menu

React Native provides a nice debugging tool while developing, and the debug developer menu can be accessed by:

1. **Shake Gesture** - Either physically shake the device, or send a "Shake Gesture" to the emulator.
2. **Keyboard Shortcut** - Send the CTRL+D (Command+D) to the iOS emulator or CTRL+M (Command+M) to the Android emulator.
3. **Key Event** - Only on the Android emulator, run the command "adb shell input keyevent 82".

[43]

7.1.1 Reload

Manually reloads only the JavaScript code, which can be replaced during run-time without having to be re-compiled. This can also be called directly by double-tapping the R key on Android emulators, or sending the "Command+R" keyboard shortcut to an iOS emulator. [43]

7.1.2 Enable Live Reload

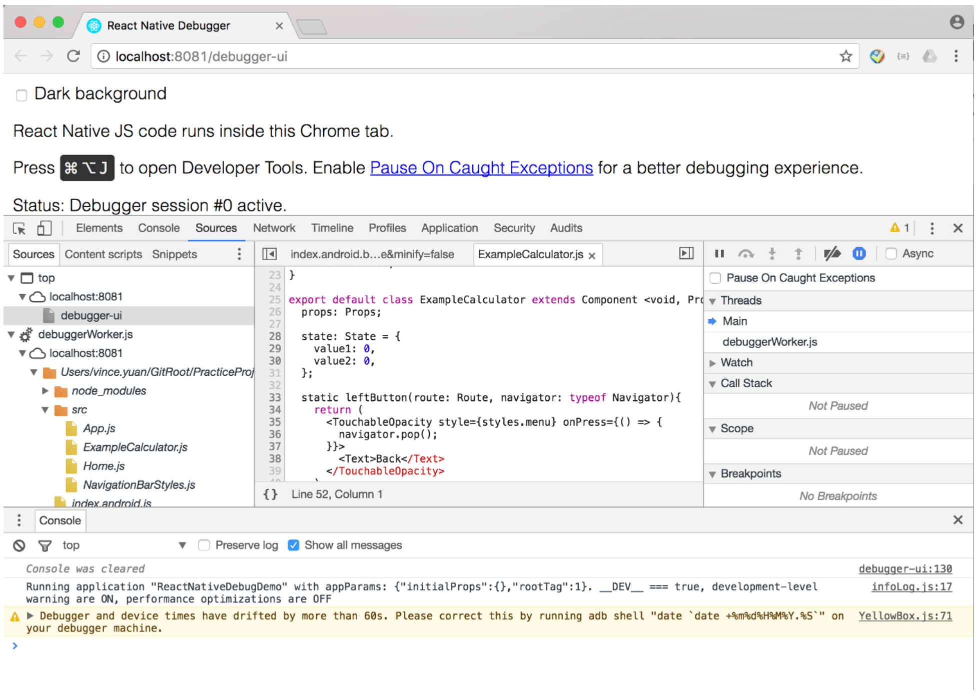
If Live Reload is enabled, it will automatically try to reload the JavaScript code as described above whenever it detects that the code has changed. [43]

7.1.3 Hot Reload

When reloading the JavaScript code, the app may reset the state if the code change affects too many files. For example, if you have navigated in to a menu, and maybe further in to a sub-menu, then reloading may send you back to the home page and you have to manually navigate back to the sub-menu.

Hot Reloading tries to solve this by keeping the app state as it was before the reload operation. Reloading only works for JavaScript code of course, so if resources have been added or native code has been changed, it will make a full reload by resetting the state of the app. [43]

7.1.4 Remote JS Debugging



Dark background

React Native JS code runs inside this Chrome tab.

Press ⌘⌥J to open Developer Tools. Enable Pause On Caught Exceptions for a better debugging experience.

Status: Debugger session #0 active.

Pause On Caught Exceptions

```
}

export default class ExampleCalculator extends Component <void, Pro
  props: Props;

  state: State = {
    value1: 0,
    value2: 0,
  };

  static leftButton(route: Route, navigator: typeof Navigator){
    return (
      <TouchableOpacity style={styles.menu} onPress={() => {
        navigator.pop();
      }}>
        <Text>Back</Text>
      </TouchableOpacity>
```

```
Console was cleared                                                                                       debugger-ui:130
Running application "ReactNativeDebugDemo" with appParams: {"initialProps":{},"rootTag":1}. __DEV__ === true, development-level   infoLog.js:17
warning are ON, performance optimizations are OFF
▶ Debugger and device times have drifted by more than 60s. Please correct this by running adb shell "date `date +%m%d%H%M%Y.%S`" on   YellowBox.js:71
your debugger machine.
```

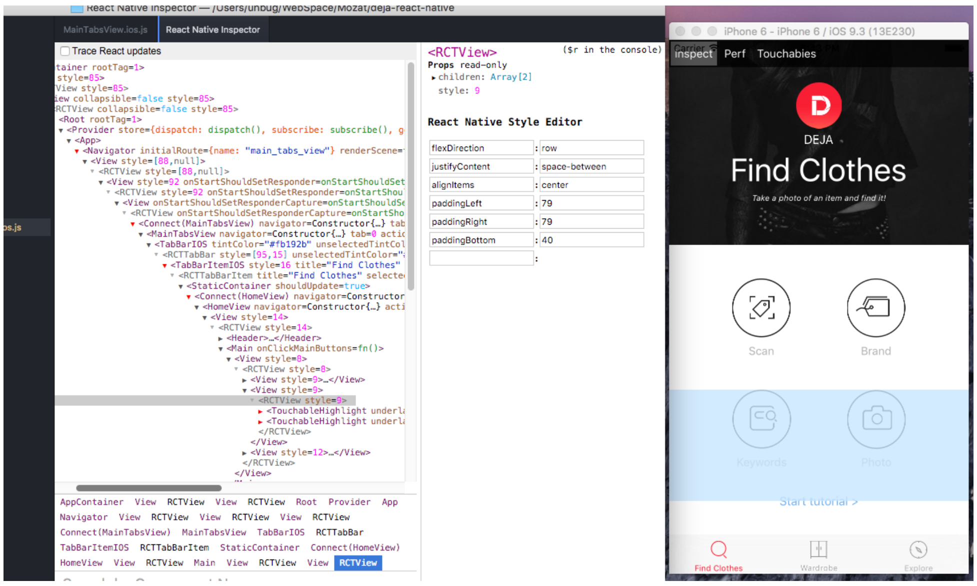
Figure 23: Remote JS Debugging

You can also attach remote JavaScript debuggers, one of the most used one is the web browser Developer Tools. Although most tutorials will mention the Chrome browser specifically, this will work with most web browsers that have some sort of Developer Tools built in.

The React Native debugger is accessed via `http://localhost:8081/debugger-ui/` locally on the development host, the Developer Tools are usually accessed with the F12 keyboard key or via the Tools menu (if the browser menu is hidden, you press the ALT key in most browsers).

The Sources tab will provide a hierarchical list of all the source files on the left side. And on the right side you can, as in most debugging environments, watch the variable values, the call stack, the scope, the breakpoints etc. [44]

7.1.5 Inspector



```
tainer rootTag=1>
 style=85>
View style=85>
iew collapsible=false style=85>
RCTView collapsible=false style=85>
 <Root rootTag=1>
 ▾ <Provider store={dispatch: dispatch(), subscribe: subscribe(), g
   ▾ <App>
     ▾ <Navigator initialRoute={name: "main_tabs_view"} renderScene=
       ▾ <View style=[88,null]>
         ▾ <RCTView style=[88,null]>
           ▾ <View style=92 onStartShouldSetResponder=onStartShouldSet
             ▾ <RCTView style=92 onStartShouldSetResponder=onStartShou
               ▾ <View onStartShouldSetResponderCapture=onStartShouldSe
                 ▾ <RCTView onStartShouldSetResponderCapture=onStartSho
                   ▾ <Connect(MainTabsView) navigator=Constructor{…} tab
                     ▾ <MainTabsView navigator=Constructor{…} tab=0 actic
                       ▾ <TabBarIOS tintColor="#fb192b" unselectedTintCol
                         ▾ <RCTTabBar style=[95,15] unselectedTintColor="
                           ▾ <TabBarItemIOS style=16 title="Find Clothes"
                             ▾ <RCTTabBarItem title="Find Clothes" selecte
                               ▾ <StaticContainer shouldUpdate=true>
                                 ▾ <Connect(HomeView) navigator=Constructor
                                   ▾ <HomeView navigator=Constructor{…} acti
                                     ▾ <View style=14>
                                       ▾ <RCTView style=14>
                                         ▸ <Header>…</Header>
                                         ▾ <Main onClickMainButtons=fn()>
                                           ▾ <View style=8>
                                             ▾ <RCTView style=8>
                                               ▸ <View style=9>…</View>
                                               ▾ <View style=9>
                                                 ▾ <RCTView style=9>
                                                   ▸ <TouchableHighlight underla
                                                   ▸ <TouchableHighlight underla
                                                   </RCTView>
                                                 </View>
                                               ▸ <View style=12>…</View>
                                             </RCTView>
                                           </View>

AppContainer View RCTView View RCTView Root Provider App
Navigator View RCTView View RCTView View RCTView
Connect(MainTabsView) MainTabsView TabBarIOS RCTTabBar
TabBarItemIOS RCTTabBarItem StaticContainer Connect(HomeView)
HomeView View RCTView Main View RCTView View RCTView
```

```
<RCTView>                    ($r in the console)
Props read-only
 ▸ children: Array[2]
   style: 9

React Native Style Editor
```

flexDirection	row
justifyContent	space-between
alignItems	center
paddingLeft	79
paddingRight	79
paddingBottom	40

Figure 24: Debug Inspector
[45]

The Inspector will run on the device or emulator as a transparent overlay over the app UI, and provides a nice way to inspect different UI components in real-time on the actual device. When tapping on different UI components, the Inspector will provide detailed information on the state of that component.

It can also show information about network traffic, which is useful when trying to access online resources over a network such as the internet. The Inspector works the same way in the web browser Developer Tools, but shows the details in the browser instead of on the device. [43]

7.1.6 Perf Monitor (Performance Monitor)

The Performance Monitor will display a transparent UI layer in the top-right corner with details about the UI performance on the device, like for example the frame rate in FPS (Frames Per Second). It even separates between the frame rate of what the JavaScript code requests and what the UI actually displays, with additional details about frame drops. This information is especially useful when developing graphically intensive apps.

7.2 Console Log

In addition to React Native specific debugging tools, most modern web browsers provide a way to log some information to the web browser console. [46]

7.2.1 Console.log()

Prints out information without any specific severity level, usually displayed with the default text color without any special formatting.

7.2.2 Console.warn()

This will use the Warning severity level, which may sometimes be formatted with a yellow or orange color. Even when no formatting is used, the warning messages can be filtered by severity level to filter out non-warning messages. With React Native it will in addition to printing to the console, also display a yellow non-transparent full screen with the warning details, React Native calls this a YellowBox. [43]

7.2.3 Console.error()

This will use the Error severity level, which may sometimes be formatted with a red color. Even when no formatting is used, the error messages can be filtered by severity level to filter out non-error messages. With React Native it will in addition to printing to the console, also display a red non-transparent full screen with the error details, React Native calls this a RedBox. [43]

7.3 Debugging

Our most used method for debugging was "console.log()" for locating the problem when the react-native debugger did not work. We also used it to see the content of variables, and to figure out how the code worked and when states where changed. It took us some time, but after a while we found out that we could use "`console.log(JSON.stringify(variable))`" to see the content of objects. React Native debugger was also very helpful to locate problems and errors, when it worked correctly.

For the UI, we used style properties like "borderWidth" and "borderColor" to see how much space a specific view occupied on the screen, or if it was outside its scope (parent view). It also came in handy when placing and setting sizes of new UI components.

Positives

- When the debugger was working correctly and displayed the actual stack-trace of which file and line number the error was located.
- "console.log()" helped us a lot to locate the problems we were looking for when the debugging tools did not work properly.
- Double-tap R to refresh changes in the code so we did not have to build the entire project every time.
- When Live Reloading worked as it should, and displayed the changes without having to reload the project every time.
- The option to wipe the emulator to run a fresh installation of the application.

The usage of "borderWidth" and "borderColor" in styles, helped us to see the outer lines of the view we were currently working on. They made it easier to see why the rendering was wrong and where they happened in the code, ex:

```
    const styles = StyleSheet.create({
      container: {
        borderColor:'red',
        borderWidth: 2,
      },
    },
```

Negatives

The React Native debugging tools gave us more problems than solutions, and did not help us locate various problems. This was especially frustrating if random crashes occurred.

Reloading did not always work because of various caching issues, so it was not always clear if the changes we had made got displayed on the screen. This took up a lot of time when we wanted to change a position or a size on a view, and we were not sure if the new changes had been applied. Because of this, we often had to rebuild the project, or worse, wipe the emulator clean and install the application again.

When using "`console.log(object)`" to print out an unknown variable, it would only display the memory address of the object if it was not "NULL", and not the actual content of the object, like the type, member variables, member methods etc. However we did find a nice solution to this, as mentioned before, we could use "`console.log(JSON.stringify(object))`" to display the entire content and all of its content.

8 Summary

8.1 Results

8.1.1 Porting to Android

The main goal of the project was to port the existing Apple iOS-specific app to the Google Android platform for our employer Klimb AS. The existing codebase was already written using the React Native framework which was supposed to provide cross-platform functionality between iOS and Android, but the app was developed on, and had only been tested on the iOS platform.

While working on the project, our employer decided that it would be better to change a lot of the current architecture, and move more of the functionality out into services. This was something that we needed to adapt to.

Our main focus while working on the project was to fix bugs that appeared on the Android version, and get the application to work on Android without crashing, and make the functionality work as expected, like it does on the iOS version. Also to make the Android version look and feel as similar to the iOS version as possible.

Architectural Changes and Issues

After big architectural changes were made, Klimb AS decided that they wanted to do more changes to the UI and some of the general functionality of the application. These changes were also needed on the Android version, and gave us an opportunity to add new features to the app that was not already implemented, instead of just debugging existing code.

The consequences of porting the application from iOS to Android, while the main iOS application was still under constant development, resulted in us constantly having problems and issues to resolve after merging between branches.

Some of the issues we were working on turned out be partly, or even sometimes completely irrelevant after having resolved them. This is because while we were trying to fix some issues on Android, at the same time, our employer had internal meetings without us where they discussed major changes to both the UI and general functionality which sometimes overlapped with what we were fixing.

Most of the times these changes did not overlap, but sometimes they did have a direct connection and overlapped with what we were working on. This could have been avoided if we were part of their internal meetings, or at least if we were informed of the decisions made during those meetings, at least the one that had an impact on our work.

Another issue is that the iOS app was continually being developed while we tried to make the Android app function based on the original version of the iOS app. Since we had no iOS device to test on, we had to make our own assumptions based on screenshots of the original iOS app, and ask our employer if certain functionality still existed in the new versions of the iOS app. If the development on the iOS app had been paused until both platforms were ready, many of the issues could possible have been avoided.

8.1.2 Internationalization

The original app was not available to an international market, and all the text labels were statically written in Norwegian. As part of our project, we decided to implement a framework for internationalization, or at least to easily and dynamically provide text labels in different languages.

We implemented an i18n package, and translated all the existing text labels to English. This would at least provide a default fallback to English for international users, and a Norwegian interface for Norwegians. It is now also very easy to add other languages because of the i18n package, all translations are done in separate files instead of being tightly integrated with the source code.

8.1.3 Platform-Specific Code

One of our goals was to use as little platform-specific code as possible, and rather make both platforms (iOS and Android) share most of the codebase. We achieved this to an extent, but some things just did not work the same for both platforms. These differences were not properly documented when we started, but have been added to the official documentation later on.

React Native provides a nice interface for making platform-specific code by just writing a simple if-check in the JavaScript code for if the platform is "android" or "ios". We wrote platform-specific code for differences where the platforms basically behaved differently, and there was no way to write shared code for both platforms. Even though we made platform-specific code, it was kept to a minimum, most often it was just a couple of lines of code.

Some examples include how to define a font name, by title or by file name, how the height of the display is calculated, where one platform includes the top status bar as part of the height, and the other does not, or if UI components can be drawn outside of their parent component or not etc.

8.1.4 Learning React Native

Another goal was to learn how to use the React Native framework for mobile development, as it is widely used in the industry. We had no prior experience with neither front-end development using a JavaScript-based framework, nor with native development using Java for Android or Swing/Objective-C for iOS.

Although the learning curve was quite steep by going from no experience, it was at the same time very exciting to face this challenge which has prepared us for employment after our studies. This has also provided us with a basic and general foundation which will make it much easier to learn new relevant frameworks like Agile and React for web development without having to learn everything from scratch.

8.2 Further Development

We have resolved all the specific issues we found while testing on both Android emulators and physical mobile devices. We have no way of verifying that our solutions have not broken the iOS release. We have always made sure to ask our employer if everything is still working on iOS after making our changes and merging our android branch with the master and/or iOS branch, and the reply has always been that everything is still working.

The remaining issues we have in the product backlog (Appendix D) are more vague and only occur randomly, so it is difficult to know if they are actual issues, or if they are related to the debug environment. We often had issues that were resolved after restarting the development machines or wiping the emulators etc. Since the debug environment of React Native and the nodeJS server is pretty complicated, there will always be caching issues that resolve themselves after a restart of the system.

Our employer has full access to all the remaining issues on BitBucket and Trello, and we have informed them that they need to fully test with a release version on real mobile devices. We have not heard about any final plans to release the product on Google Play Store, but they seemed optimistic to release an Android version as soon as possible.

Releasing on Google Play Store requires a paid developer account, creating a project on the Store and linking a release encryption key between the local developer machine and the online project. They are currently in the process of transferring ownership of the iOS app on the Apple Store from the external developers that originally developed the app to their in-house developer. We assume that they will handle release preparations for the Android platform after completing the transfer of ownership for the iOS platform.

They were also in the process of coming up with ideas for how to make money on the app, and one of the ideas was to have sponsors pay them money for putting their company name or logo in the app. For example when getting details about a hill or mountain peak, a possible sponsor could be a travel agency or something similar who provides services or arranges activities near that hill. We are not sure if this idea will be in the next release, or if it will be available in a future release instead. But it definitely sounds like a great idea.

8.3 Group Evaluation

The group dynamic was great this entire project, we already had a lot of experience working together on other projects from other classes. All discussions were constructive, and we did not experience any major conflicts that we could not resolve ourselves. We also never experienced any problems of data loss or technical problems that could have sabotaged our project.

We had fixed routines to meet at school, and work together on Mondays, Tuesdays and Wednesdays at 10 AM, this made it a lot easier to communicate and decide how we would delegate new issues that appeared. Since our group only consisted of two members we never really had any specific role for each member. So the roles where split between us, the developers, and the employer, Sindre, who was the project manager and also the developer.

The use of pair-programming from the XP methodology was really helpful at the start of the project. This helped a lot during the initial research stage to get a better understanding of how React Native worked, and also to get a faster understanding of how the existing codebase worked. Pair-programming was also used on bigger problems or implementations to make sure that both group members understood what was happening in the code, and come up with different alternative solutions much quicker than if we had worked more individually. However for smaller bug fixes and issues we did work more individually.

We used Trello and Bitbucket Issues to let everybody know what the different team members were working on, and which new issues that had been added. This helped us organize and delegate tasks, and plan for the future. It also made it easy to log the work that we had done, so we could look it up later when writing the report. We also made sure to include screenshots and a good explanation for every issue that we made.

Since we mostly worked on independent problems by ourselves, we did not experience too many discussions or conflicts between us. But when implementing the Tutorial Manager 35 we got the chance to have a lot of discussions and test many different solutions.

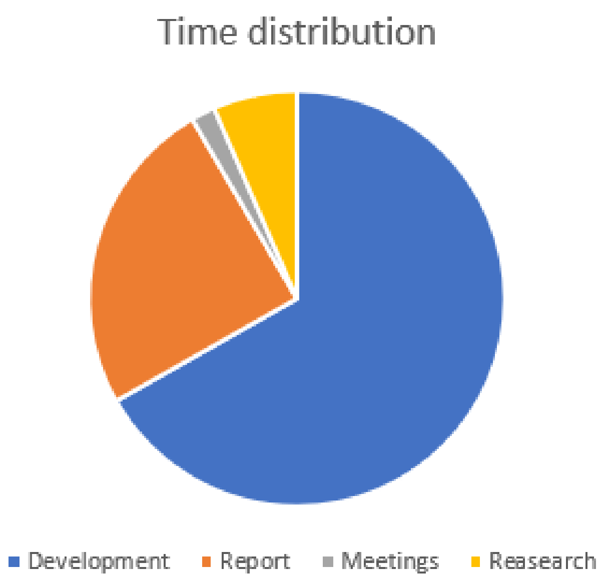


Figure 25: Time distribution

Most of our time were spent on development, and around April we started to move over to writing the report, where one member was writing on the report and the other was still developing. And after most of the specific issues were fixed both members were writing the report. We also included an hour log to show what we did when, and where most of our time was spent 70.

8.4 Conclusion

The practical aspects of the Bachelor project has provided us with a much more detailed understanding of how it can be to work in a professional setting. Especially how it is to enter a company as a new employee, with little or no existing knowledge of the technology used by the company, and start working on an existing code base.

We also learned about the challenges of wanting to write code based on best practices, and having to let go of our own preferred way of coding from the past. We really learned about the importance of commenting code, and making the code structured and easier for others to read.

It is easy sometimes, when working alone, to assume everyone understands what you are coding, and feel that comments and structured code will just make everything take longer. But after working on a group project, you quickly realize that if the code is not commented and not intuitively structured, you will spend a lot of unnecessary time trying to figure out what the code does, and even make unnecessary mistakes because you have to make assumptions that could turn out be wrong.

We also learned that working closely in the same physical location helps with communication tremendously. Since we worked in a different city from our employer, we could not just walk down to the office to have a quick meeting or discussion. We relied on remote voice communication tools such as Discord, which helps a lot, but is definitely not a replacement for meeting face to face. Many misunderstandings could have easily been resolved immediately if we were in the same office. At least it prepared us for remote cooperation, which is very common in the IT-industry.

We had hoped to learn more about how to use GIT and Trello in a more professional setting, especially learn more about more complicated and advanced features and best practices. But since our employer was a very small and young company, they did not have a very established development practices.

Using a framework that is widely used in the industry was probably the biggest takeaway from the project. We also had a mobile development course running in parallel to the Bachelor project, which provided us with great insight in to the differences between developing with a cross-platform framework compared to native development for Android using Java.

Many, if not most of our courses throughout the last three years have helped us greatly in this project. The first year gave us a solid foundation for general programming and debugging. The second year helped us with more advanced programming concepts, especially working in groups with project management tools. And finally having the mobile course in the third and final year, helped us greatly understand the challenges of developing for mobile devices.

In conclusion, we feel this Bachelor project has prepared us greatly for any future work life and employment.

Bibliography

[1] Wikipedia contributors. 2018. Javascript — Wikipedia, the free encyclopedia. [Online; accessed 8-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=840176949.

[2] Wikipedia contributors. 2018. Typescript — Wikipedia, the free encyclopedia. [Online; accessed 8-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=TypeScript&oldid=838415153.

[3] Wikipedia contributors. 2018. Android (operating system) — Wikipedia, the free encyclopedia. [Online; accessed 8-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=840107449.

[4] Wikipedia contributors. 2018. Ios — Wikipedia, the free encyclopedia. [Online; accessed 8-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=IOS&oldid=839447982.

[5] Wikipedia contributors. 2018. Scrum (software development) — Wikipedia, the free encyclopedia. [Online; accessed 18-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=836974539.

[6] Patents, A. R. R. S. I. & Pending., P. 2018. Advantages of scrum. URL: https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban#advantages-of-scrum.

[7] Wikipedia contributors. 2018. Extreme programming — Wikipedia, the free encyclopedia. [Online; accessed 18-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Extreme_programming&oldid=830770686.

[8] Inc., C. . F. 2018. Jest-delightful javascript testing. URL: https://facebook.github.io/jest/.

[9] Discord. 2018. It's time to ditch skype and teamspeak. URL: https://discordapp.com/.

[10] OÜ, T. 2018. Everything works much better with toggl. URL: https://toggl.com/.

[11] rettigheter reservert, A. O. . A. 2018. Trello. URL: https://trello.com/.

[12] Atlassian. 2018. A free git client for windows and mac. URL: https://www.sourcetreeapp.com/.

[13] contributors, W. 2018. Responsive web design — wikipedia, the free encyclopedia. [Online; accessed 28-March-2018]. URL: https://en.wikipedia.org/w/index.php?title=Responsive_web_design&oldid=832660943.

[14] contributors, W. 2017. Adaptive web design — wikipedia, the free encyclopedia. [Online; accessed 28-March-2018]. URL: https://en.wikipedia.org/w/index.php?title=Adaptive_web_design&oldid=814876811.

[15] contributors, W. 2017. Progressive enhancement — wikipedia, the free encyclopedia. [Online; accessed 28-March-2018]. URL: https://en.wikipedia.org/w/index.php?title=Progressive_enhancement&oldid=800247276.

[16] ThinkApps. 2018. Responsive web vs native apps. URL: http://thinkapps.com/blog/development/responsive-web-vs-native-apps/.

[17] Netkow, M. 2018. Hybrid mobile apps are overtaking native. URL: https://blog.phonegap.com/hybrid-mobile-apps-are-overtaking-native-951a3aacacd1.

[18] contributors, W. 2018. Angularjs — wikipedia, the free encyclopedia. [Online; accessed 2-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=AngularJS&oldid=833199461.

[19] contributors, W. 2018. Apache cordova — wikipedia, the free encyclopedia. [Online; accessed 2-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Apache_Cordova&oldid=825682522.

[20] Kukic, A. 2018. Alternatives to native mobile app development. URL: https://auth0.com/blog/alternatives-to-native-mobile-app-development/.

[21] Wargo, J. M. 2018. Which to use: Cordova or phonegap? URL: http://www.informit.com/articles/article.aspx?p=2478076.

[22] contributors, W. 2018. Mono (software) — wikipedia, the free encyclopedia. [Online; accessed 2-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Mono_(software)&oldid=831241182.

[23] contributors, W. 2018. React (javascript library) — wikipedia, the free encyclopedia. [Online; accessed 3-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=React_(JavaScript_library)&oldid=833378551.

[24] Wikipedia contributors. 2018. React (javascript library) — Wikipedia, the free encyclopedia. [Online; accessed 14-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=React_(JavaScript_library)&oldid=841145185.

[25] @unbug (https://twitter.com/unbug). 2018. React native training - 1.2 how it works. URL: https://unbug.gitbooks.io/react-native-training/content/12_how_it_works.html.

[26] Heard, P. 2018. React native architecture : Explained! URL: https://www.logicroom.co/react-native-architecture-explained/.

[27] contributors, W. 2018. Node.js — wikipedia, the free encyclopedia. [Online; accessed 3-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Node.js&oldid=833229609.

[28] contributors, W. 2018. Gradle — wikipedia, the free encyclopedia. [Online; accessed 3-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Gradle&oldid=829263035.

[29] npm, I. 2018. Npm js website. URL: https://www.npmjs.com/package/package.

[30] Foundation, N. 2018. Node.js. URL: https://nodejs.org/en/.

[31] Sebastian McKenzie, Christoph Nakazawa, J. K. 2018. Yarn: A new package manager for javascript. URL: https://code.facebook.com/posts/1840075619545360.

[32] contributors, W. 2018. Visual studio code — wikipedia, the free encyclopedia. [Online; accessed 3-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=833282639.

[33] contributors, W. 2018. Android studio — wikipedia, the free encyclopedia. [Online; accessed 3-April-2018]. URL: https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=833126826.

[34] Inc., A. 2018. Token based authentication made easy. URL: https://auth0.com/learn/token-based-authentication-made-easy/.

[35] Wikipedia contributors. 2018. Observer pattern — Wikipedia, the free encyclopedia. [Online; accessed 14-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=Observer_pattern&oldid=840234511.

[36] Wikipedia contributors. 2018. Model–view–controller — Wikipedia, the free encyclopedia. [Online; accessed 14-May-2018]. URL: https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93controller&oldid=839034827.

[37] Richard Ishida, W3C, S. K. M. B. 2018. Localization vs. internationalization. URL: https://www.w3.org/International/questions/qa-i18n.

[38] AlexanderZaytsev. 2018. react-native-i18n. URL: https://github.com/AlexanderZaytsev/react-native-i18n.

[39] Inc., F. 2018. React native docs v. 0.46. URL: https://facebook.github.io/react-native/docs/0.46/layout-props.html#overflow.

[40] Inc., F. 2018. React native docs v. 0.55. URL: https://facebook.github.io/react-native/docs/layout-props.html#overflow.

[41] Navigation, R. 2018. React navigation docs. URL: https://reactnavigation.org/docs/navigation-prop.html.

[42] Inc., F. 2018. React native docs - asyncstorage. URL: https://facebook.github.io/react-native/docs/asyncstorage.html.

[43] Inc., F. 2018. React native - debugging. URL: https://facebook.github.io/react-native/docs/debugging.html.

[44] Yuan, V. 2018. React native debugging tools. URL: https://codeburst.io/react-native-debugging-tools-3a24e4e40e4.

[45] @unbug (https://twitter.com/unbug). 2018. React native training - 1.3 debug tools. URL: https://unbug.gitbooks.io/react-native-training/content/13_debug_tools.html.

[46] Mozilla & individual contributors. 2018. Mdn web docs - console. URL: https://developer.mozilla.org/en-US/docs/Web/API/Console.

A Project Plan

BSP3900 Bachelor Project Plan - 24.1.2018

App and Gamification - Klimb AS

Magnus W. Enggrav, Adam A. Jammary

1. Introduction

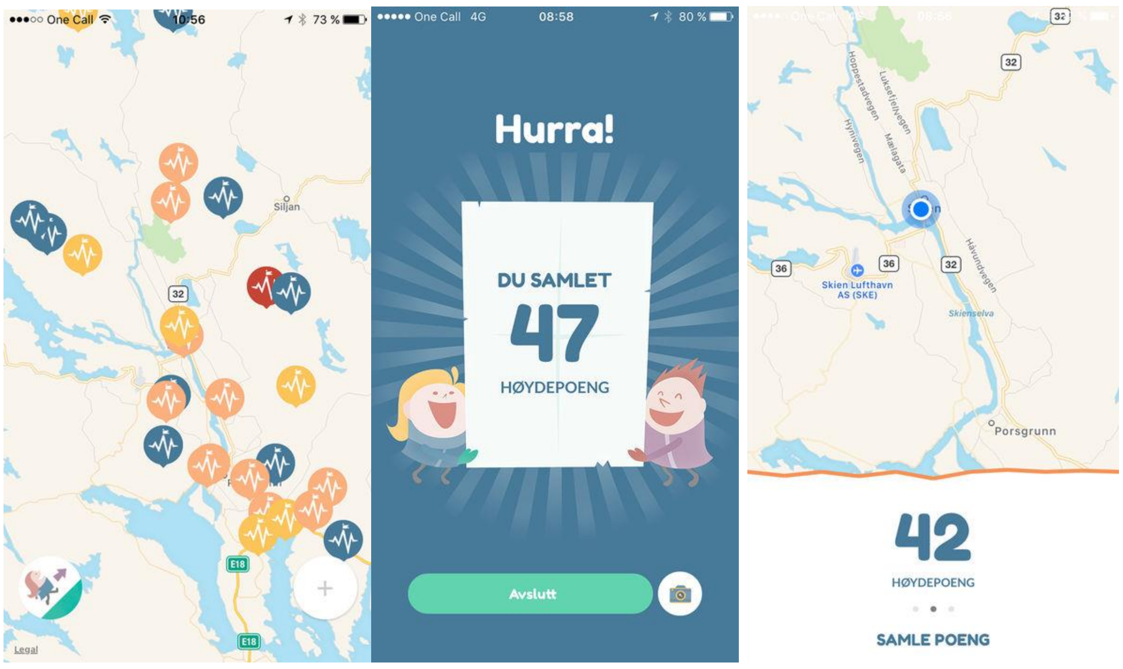
1.1. Background

Klimb is a health and fitness game where you collect points when walking uphills using GPS coordinates. One height meter equals one height point. The goal of the game is to move up levels and share activity with your friends.

You can hike up your favourite mountain and collect points at the top, if the mountain does not exist in the app, you can add the top and become the owner of that mountaintop. You also have the ability to measure height meter everytime you walk or run uphill, the application will automatically measure this for you. Your profile page will show you the number of height point you have collected and you can see how every level is a known mountain you can move towards.

You can share your progress on the stream and on facebook to inspire others. In “Challenges” you can find fun challenges that you can do together with your friends or alone. Klimb uses Facebook login system to make it easy and fast to login.

Klimb is currently running on iOS and our project will be to port this over to Android.



1.2. Learning Objectives

- How mobile development is done in the industry.
- What frameworks and technologies are used.
- How the practicing agile methodology affects the project.
- What factors are important when developing for multiple platforms.

1.3. Impact Objectives

- The ported Android version of the Klimb app will be available in the Google Play Store.
- The UI will be improved by being more intuitive to use for customers.

1.4. Performance Objectives

- The customer will have more users using their app since it will be available for the Google Android platform as well as the existing Apple iOS platform.
- The customer will get more positive reviews because of the improved UI design.
- Hopefully the customer will have increased earnings because of the above factors.

1.3. Constraints

We will work under the following constraints:

- We have a maximum of three persons working on development.
- We must complete the work by the time frame of the bachelor project which is by the time of writing this plan 16th May 2018.
- We have no physical iOS device to develop or test on, but must rely on screenshots and having the employer test our contribution on iOS devices.
- The employer is not located in the same city as us, so all communication is remote.

2. Scope

2.1. Field of study

- Cross-platform (Android and iOS) mobile development
- React-Native Framework
- Internationalization (I18N)
- Geolocation (GPS-data, Google Maps)
- Login services (Facebook SDK)

2.2. Project Restrictions

- We will not develop anything for the iOS platform.
- We will not work on the codebase in the master branch of the repository.
- The app will only support Android 4.1 (API 16 - JELLY_BEAN) and newer.

2.3. Project description

The main goal of the project is to port the existing Apple iOS-specific app to the Google Android platform. Even though the existing codebase is already written using the React Native platform which allows for cross-platform functionality between iOS and Android, the app has been developed and tested only on the iOS platform. The result is that it does not currently work on the Android platform.

The issues of what is and what is not working is not clearly defined this early in the process, and it is mainly our task to find the issues and resolve them so that the app can work on Android devices. This means that it will be difficult to plan in too much details as the details are not yet known. The React Native framework uses JavaScript as the programming language for both platforms, but allows to use native languages for the specific platforms if needed.

We will try to use as little platform-specific code as possible, and have the two platforms share as much of the codebase as possible. This will require clear communication, and a well-defined work process as we will describe later. The goal is to have the UI look and feel almost the same on both platforms with less than 10% differences.

Initially the plan is for the employer to focus on the backend server communication, and for us to focus on the Android frontend, but this may change during the project as needed. The employer has a huge list of features they wish to implement that are not officially part of the bachelor project, but they are available if the current workload is not enough.

3. Project Organization

3.1. Roles and Responsibility

Mariusz Nowostawski	Supervisor
Elin Årseth	Owner/CEO
Sindre Helelborg	Project Manager
Magnus W. Enggrav	Developer
Adam A. Jammary	Developer

3.2. Project rules and Routines

3.2.1. Working hours

Monday to Wednesday we will work together at campus while Thursday to Friday will be a little more flexible depending on what we are working on individually.

Core hours:

Monday - Wednesday:	10:00 - 18:00
Thursday:	13:00 - 16:00
Friday:	09:00 - 12:00

3.2.2. Conflict management

- Will try to solve conflict with each other first.
- If conflict is not solved employer will have the last word.

3.2.3. Communication

- We will work and communicate together at campus under set working hours.
- Other communication with each other will happen over Discord or Slack.
- Communication with employer will also happen over Discord and Slack.

4. Planning, Monitoring and Reporting

4.1. Software Development Methodology - Scrum with XP

There is currently only one developer working on this application, so there is no software development methodology currently in place. And since we do not yet know what problems exists on the Android version, alot of our job will be to find out what problems exists and what our tasks will be. This makes it hard to plan ahead, which is why we have chosen the agile methodology framework **Scrum** [2] in a combination with aspects of **XP** [3] since agile methodologies focus more on an "incremental and iterative approach instead of in-depth planning at the beginning of the project" [1]. And "agile methodologies are more open to changing requirements over time and encourages constant feedback" [1] from our employer.

We chose to use a Scrum because it has "more transparency and project visibility" [1], and this is important since we will be working very closely with our employer. This helps us know what our employer is working on, and vice-versa. And since we are mostly working together at campus we are able to have more face-to-face discussions when we need it, and our employer is always available on Discord and Slack if we need to discuss something.

We will use the general framework of Scrum to organize the project, and the simplicity and flexibility of XP for our daily development. The existing codebase does not contain any unit testing, so part of our job will be to implement unit tests for new functions that we implement in accordance with one of the four core activities of XP, “Extreme programming's approach is that if a little testing can eliminate a few flaws, a lot of testing can eliminate many more flaws.” [3].

In the beginning phase we will try to pair-program as much as possible, this will benefit us if one of us gets sick because if the other will know everything that the other person has worked on. Later on when we get more familiar with the project and get more specific tasks, we can start dividing them between the us (the development team).

4.2. Roles

Project Manager:	Sindre Helelborg
Development Team:	Magnus Enggrav and Adam Jammary

We have chosen not use the official Scrum roles since their definitions [1] do not fit with the size of our group which consists only of two developers and our employer. Our employer can be more accurately described as the general project manager since he makes overall project decisions and delegates the tasks.

4.3. Workflow

Scrum allows to easily accommodate changes, “With short sprints and constant feedback, it’s easier to cope with and accommodate changes.” [1]. Constant feedback from the employer is necessary since we don't yet know what problems need to be solved, and using Scrum makes it easier to handle changes or problems that may unexpectedly arise. We have chosen a Sprint period of one week since the tasks and priorities may unexpectedly change, so it’s better to meet once a week to more easily adapt to changes.

We have two types of meetings, one is with our employer once every monday morning to discuss what we did last week (Sprint review), what went well and what went wrong (Sprint retrospective), and plan for what we will for the rest of that week (Sprint planning). The other type is between us without the employer (Daily scrum), where we meet daily and use the first 15 minutes to plan what to work on during that day, or discuss problems we might have.

4.4. Artifacts

We will use Trello as our Scrum board to keep track of the various tasks and what the members of the team are working on. We will use Bitbucket as the code repository and Issue Tracker, and GIT as the version control system. Trello is already integrated in to the Bitbucket project.

1) We start by creating the task in the Bitbucket Issue Tracker which will assign it an auto-incremented issue number, which we in turn use to create a Trello card in the "Product backlog".
2) After we have decided what to work on for that week (sprint) we move the task over to the "Sprint backlog".
3) Each one of us will then select a task to work on by moving it over to "In Progress" and assign ourselves to the task.
4) After we have completed a task, we move it over to "In Review (QA / Test)", where we review and test that the code works as expected.
5) Finally we move it over to "Done" marking the task as complete.

To be able to track the task properly, we label all our commits as well as the Trello cards with the issue number [4], ex. "#9 Internationalization - i18n". Issues can be of the type "task" or "bug". We use "bug" for already implemented features which do not work as expected and need to be resolved, and "task" for a feature that needs to be implemented.

We will only work in a separate repository branch called "android", while our employer will merge this branch into the master branch after testing it and verifying that it has no negative effects on the master branch.

Toggl will be used to track the amount of time we use on the various types of activities during our Bachelor project. This will help us manage our time more efficiently and get a better overview of how we have distributed our time, and hopefully help us make better time estimates for future tasks.

4.5. Plan for status meetings and decision points during the period

- Regular meetings with employer once a week, Monday at 10:00 on Discord.
- Meetings with supervisor every two weeks if necessary .
- We can easily contact employer on Discord, Slack or mail outside of meetings.

5. Quality Assurance

5.1. Commenting and style guide

Airbnb React/JSX Style Guide

- https://github.com/airbnb/javascript/tree/master/react

We cannot guarantee that existing code, or code made by anyone other than us will follow the above style guide from Airbnb, but we will try to follow it as closely as possible.

5.2. Risk analysis

#	Issue	Probability	Impact
1	Group member(s) get sick	Low	Medium
2	Group member leaves	Low	High
3	Can't complete the project, but can write the report	Low	Medium
4	Loss of data locally, due to malware or loss of devices	Low	High
5	Bitbucket is down	Low	Low
6	Inacurate time estimation	High	Low

#	How to prevent the issue	How to deal with the issue
1	Try to be active, get enough sleep, eat health and be generally hygienic.	Update the sick person on the status of the project.
2	Use pair-programming as much as possible (at least with critical code).	Notify supervisor and employer, and ask for advice.
3	Clear and frequent communication, and make sure each sprint delivers a small but a functional product.	Deliver what has been completed at the deadline.
4	Keep systems updated and secure, and store critical data online.	Restore data from online sources. Use software emulators to replace mobile
5	Commit to GIT locally, and once in a while store a zipped backup on Dropbox.	Keep working locally until it's up again, if someone else needs the changes immediately, they can copy over the zipped backup, but this is only as a last resort.
6	Clear and frequent communication.	Use experience to plan the time better the next time.

The documents are written in Google Docs so everything is constantly saved online which provides a backup. Every document is also saved in Dropbox just in case.

5.3. Tools

5.3.1. Package Managers and Build Tools

Node.js 7.10.1 med NPM 4.2.0
“Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.” [5]

Yarn 1.3.2
“Yarn is a new package manager that replaces the existing workflow for the npm client or other package managers while remaining compatible with the npm registry.” [6]

5.3.2. IDE (Integrated Development IDE)

Visual Studio Code 1.19.2
“Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring.” [7]

5.3.3. Emulators and device testing

Android Studio 3.0.1 for Windows
“Android Studio is the official[6] integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.” [8]

Android Emulators (run through Android Studio)
- Nexus 5X x86 - Android 8.1 - API 27 - 1080x1920 420 dpi
- Nexus 6 x86_64 - Android 6.0 - API 23 - 1440x2560: 530 dpi

Android Devices (our own Android devices)
- Samsung Galaxy S7 (SM-G930F) - Android 7.0 - API 24 - 2560x1440
- Samsung Galaxy S6 edge (SM-G925F) - Android 7.0 - API 24 - 1440 x 2560

5.3.4. Project Management

Trello
“Trello is a web-based project management application” [9]

We use Trello as a Scrum board to keep track of each sprint and what each team member is working on.

Discord
“Discord is a proprietary freeware VoIP application designed for gaming communities.” [10]

We use Discord to have meetings and chat with our employer and the team.

Slack
“Slack is a cloud-based set of proprietary team collaboration tools and services” [11]

We use Slack for more “official” discussion with our employer.

Toggl (web and desktop app)
“Toggl is a time tracking app operated by Toggl OÜ, headquartered in Tallinn, Estonia, that offers online time tracking and reporting services through their website along with mobile and desktop applications.” [12]

We use Toggl for time tracking to get a sense of where most of our time is used.

Bitbucket (GIT)
“Bitbucket is a web-based version control repository hosting service owned by Atlassian” [13]

We use Bitbucket for version control and issue tracking

SourceTree
SourceTree is a GIT Gui that makes GIT easier.
https://www.sourcetreeapp.com/

Google Docs
Google docs is an online word processing application created by google. It has version control and can be shared between all team members which make is a very useful tool when collaborating. We will use this to write the project plan.

Microsoft Excel
“Microsoft Excel is a spreadsheet developed by Microsoft for Windows, macOS, Android and iOS.” [14]

We used Microsoft Excel to create the tables in “5.3. Risk analysis”.

Microsoft Project

“Microsoft Project is a project management software product, developed and sold by Microsoft.” [15]

We used Microsoft Projects to create the gantt diagram in “6. Project Plan”.

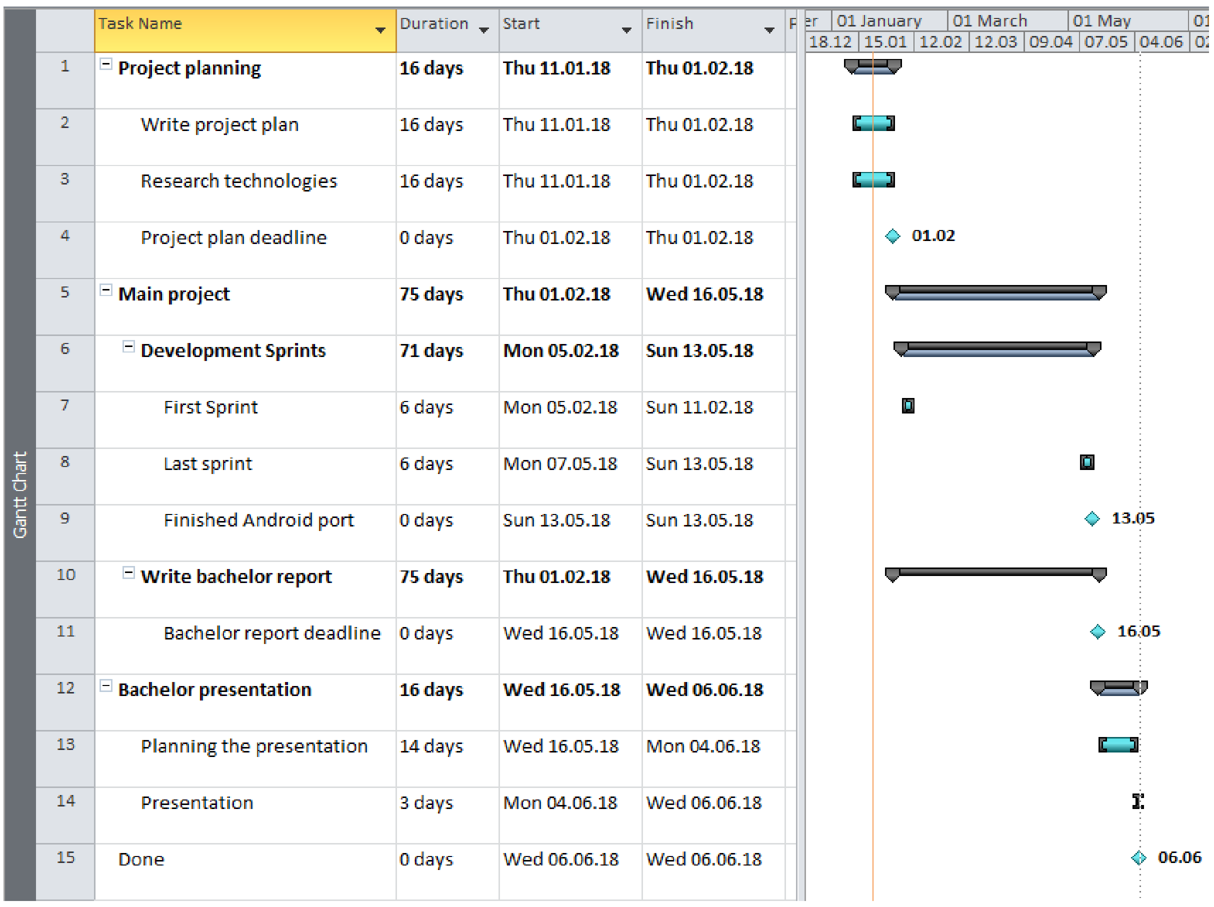
Latex

“ShareLaTeX is an online LaTeX editor that allows real-time collaboration and online compiling of projects to PDF format.” [16]
https://www.sharelatex.com/project

Bachelor thesis template forked from GitHub:
https://github.com/COPCSE-NTNU/bachelor-thesis-NTNU

We will use ShareLaTeX to write the Bachelor report.

6. Project Plan



	Task Name	Duration	Start	Finish
1	**Project planning**	**16 days**	**Thu 11.01.18**	**Thu 01.02.18**
2	Write project plan	16 days	Thu 11.01.18	Thu 01.02.18
3	Research technologies	16 days	Thu 11.01.18	Thu 01.02.18
4	Project plan deadline	0 days	Thu 01.02.18	Thu 01.02.18
5	**Main project**	**75 days**	**Thu 01.02.18**	**Wed 16.05.18**
6	**Development Sprints**	**71 days**	**Mon 05.02.18**	**Sun 13.05.18**
7	First Sprint	6 days	Mon 05.02.18	Sun 11.02.18
8	Last sprint	6 days	Mon 07.05.18	Sun 13.05.18
9	Finished Android port	0 days	Sun 13.05.18	Sun 13.05.18
10	**Write bachelor report**	**75 days**	**Thu 01.02.18**	**Wed 16.05.18**
11	Bachelor report deadline	0 days	Wed 16.05.18	Wed 16.05.18
12	**Bachelor presentation**	**16 days**	**Wed 16.05.18**	**Wed 06.06.18**
13	Planning the presentation	14 days	Wed 16.05.18	Mon 04.06.18
14	Presentation	3 days	Mon 04.06.18	Wed 06.06.18
15	Done	0 days	Wed 06.06.18	Wed 06.06.18

The initial part of the project will consist mostly of writing the project plan and researching the current technologies in use, but also possible technologies that can be added or maybe even replace existing ones.

We have decided to work on the bachelor report in parallel with the development sprints instead of spending the first half or two thirds for development and saving the report writing for the end. This will help us keep the report up-to-date from the beginning to end, instead of rushing it at the end. The report will consist mostly of notes and short sentences in the first part, and will be more and more organized and revised the further in to the project we get.

We have reserved the last period between the report deadline and the presentation for preparation. This does not mean that we will deny the employer to work on their code, but we will make it clear that we will prioritize preparing for the presentation. We also have an exam in the same period in another course, so we will be quite busy during this period.

7. References

[1] What's the Difference? Agile vs Scrum vs Waterfall vs Kanban
Retrieved January 23, 2018, from
https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban

[2] Scrum (software development). (2018, January 20). In Wikipedia, The Free Encyclopedia. Retrieved 13:41, January 23, 2018, from
https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=821429866

[3] Extreme programming. (2018, January 21). In Wikipedia, The Free Encyclopedia. Retrieved 13:57, January 23, 2018, from
https://en.wikipedia.org/w/index.php?title=Extreme_programming&oldid=821561346

[4] Mark up comments, issues, and commit messages
Retrieved January 23, 2018, from
https://confluence.atlassian.com/bitbucket/mark-up-comments-issues-and-commit-messages-321859781.html

[5] nodeJS
Retrieved January 24, 2018, from
https://nodejs.org/en/

[6] Yarn: A new package manager for JavaScript. Introducing Yarn.
Retrieved January 24, 2018, from
https://code.facebook.com/posts/1840075619545360

[7] Visual Studio Code. (2018, January 24). In Wikipedia, The Free Encyclopedia.
Retrieved 10:55, January 24, 2018, from
https://en.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=822060320

[8] Android Studio. (2018, January 23). In Wikipedia, The Free Encyclopedia.
Retrieved 10:58, January 24, 2018, from
https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=821856626

[9] Trello. (2018, January 2). In Wikipedia, The Free Encyclopedia.
Retrieved 11:03, January 24, 2018, from
https://en.wikipedia.org/w/index.php?title=Trello&oldid=818184684

[10] Discord (software). (2018, January 23). In Wikipedia, The Free Encyclopedia. Retrieved 11:04, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=Discord_(software)&oldid=821860404

[11] Slack (software). (2018, January 19). In *Wikipedia, The Free Encyclopedia*. Retrieved 11:06, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=Slack_(software)&oldid=821267478

[12] Toggl. (2017, October 5). In *Wikipedia, The Free Encyclopedia*. Retrieved 11:07, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=Toggl&oldid=803897264

[13] Bitbucket. (2018, January 10). In *Wikipedia, The Free Encyclopedia*. Retrieved 11:08, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=Bitbucket&oldid=819655939

[14] Microsoft Excel. (2017, December 30). In Wikipedia, The Free Encyclopedia. Retrieved 11:14, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=Microsoft_Excel&oldid=817705353

[15] Microsoft Project. (2018, January 15). In Wikipedia, The Free Encyclopedia. Retrieved 11:16, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=Microsoft_Project&oldid=820510761

[16] ShareLaTeX. (2017, September 24). In *Wikipedia, The Free Encyclopedia*. Retrieved 10:45, January 24, 2018, from https://en.wikipedia.org/w/index.php?title=ShareLaTeX&oldid=802100411

B Hour Log

Applies	**Date**	**Start**	**End**	**Category**	**Description**
All	11.01.2018	10:00	10:30	Meeting	First meeting with Sindre
All	11.01.2018	10:30	14:00	Report	Setting up Project plan
All	11.01.2018	14:00	17:00	Research	React native tutorials
All	12.01.2018	12:30	13:30	Research	React native tutorials
All	15.01.2018	10:00	10:30	Meeting	Meeting with Sindre
All	15.01.2018	11:30	16:30	Development	Setting up environment
All	16.01.2018	10:30	11:30	Project Agreement	Signing and upload Project Agreement
Magnus	16.01.2018	11:30	16:30	Research	React native tutorials
Adam	16.01.2018	11:30	15:00	Research	Look through earlier bachelor assignments
Adam	16.01.2018	15:00	16:30	Research	React native tutorials
All	17.01.2018	10:30	11:30	Meeting	Meeting with Sindre
All	17.01.2018	11:30	12:30	Development	#7 Make LoginScreen look the same on Android as it does on existing iOS
Magnus	17.01.2018	12:30	16:00	Research	Reading about react native
Adam	17.01.2018	12:30	13:00	Development	#8 Rotation - Lock to portrait mode only on Android
Adam	17.01.2018	13:00	16:00	Development	#9 Internationalization - i18n
All	22.01.2018	11:00	11:30	Report	Write project plan document
All	22.01.2018	11:30	12:00	Meeting	Meeting with Sindre
Magnus	22.01.2018	12:00	17:00	Report	Working on project plan
Adam	22.01.2018	12:00	12:30	Development	Make sure project builds after new commits from employer
Adam	22.01.2018	12:30	17:00	Report	Write project plan document
Adam	23.01.2018	10:00	12:30	Development	Can't login on my mobile device
Magnus	23.01.2018	10:00	17:00	Report	Working on Project plan
Adam	23.01.2018	12:30	17:00	Report	Write project plan document
All	24.01.2018	10:00	12:30	Report	Write project plan document
Magnus	24.01.2018	12:30	16:30	Development	Started to look at MapView and button positioning
Adam	24.01.2018	12:30	13:00	Report	Write project plan document
Adam	24.01.2018	13:00	16:00	Development	Can't log back in using Facebook SDK after logging out
Adam	24.01.2018	16:00	16:30	Development	Mobile is stuck on loading map
Magnus	29.01.2018	10:00	17:00	Development	NavigationBar: Rendering action button above map
Adam	29.01.2018	10:00	17:00	Development	#9 Internationalization - i18n
Adam	30.01.2018	10:00	13:00	Development	#9 Internationalization - i18n
Magnus	30.01.2018	10:00	15:30	Development	Moving actionButtons from NavigationBar to MainView
Adam	30.01.2018	13:00	15:30	Development	#14 Internationalization - i18n - Only English is selected

Adam	31.01.2018	10:00	13:30	Development	#12: NavigationBar
Magnus	31.01.2018	10:00	15:00	Development	ActionButtonItems rendering
Adam	31.01.2018	13:30	15:00	Report	Setting up ShareLatex
Adam	31.01.2018	15:00	16:00	Research	Looking in to rxJS observables/services
Magnus	31.01.2018	15:00	16:00	Research	Learning LaTeX
All	05.02.2018	10:00	11:00	Meeting	Meeting with Sindre
Adam	05.02.2018	11:00	15:00	Development	#12: NavigationBar
Magnus	05.02.2018	11:00	15:30	Development	Fixing ActionButton
All	05.02.2018	15:00	16:30	Development	#16 #18 #19
Adam	06.02.2018	11:30	12:30	Development	Looking for bugs
Magnus	06.02.2018	11:30	14:15	Development	Finding Issues and solving #21
Adam	06.02.2018	12:30	14:30	Development	#17 Fix Free Run button in the navigation bar
Magnus	06.02.2018	14:15	17:00	Development	#22 HillActivityPage onClick
Adam	06.02.2018	14:30	17:00	Development	#24 Make UI look better - Camera - Take picture
Adam	07.02.2018	10:30	16:00	Development	#24 Make UI look better - Camera - Take picture
Magnus	07.02.2018	10:30	14:30	Development	#27 HillActivityPage showHillCard UI
Magnus	07.02.2018	14:30	16:00	Development	#28 Specify fonts manually for Android vs. iOS
Adam	12.02.2018	10:00	15:30	Development	#24 Make UI look better - Camera - Take picture
Magnus	12.02.2018	10:00	15:00	Development	#26 Fix Free Run UI - looks weird
Magnus	12.02.2018	15:00	16:00	Research	Research
Adam	12.02.2018	15:30	16:00	Research	Research
Magnus	13.02.2018	10:30	17:00	Development	Debug #30
Adam	13.02.2018	10:30	14:00	Development	#29 Camera - SharePictureFrame - keyboard overlaps text input
Adam	13.02.2018	14:00	17:15	Development	#31 Camera - navigation state not consistent
Adam	14.02.2018	10:00	13:30	Development	#31 Camera - navigation state not consistent
Magnus	14.02.2018	10:00	13:00	Development	#30 fixed M.O.H debug distance
Adam	14.02.2018	13:30	16:40	Development	#25 Camera - can’t share picture - loads forever
Magnus	14.02.2018	13:30	16:30	Development	#30 fixed M.O.H debug distance
Magnus	19.02.2018	10:00	11:00	Development	#32 Specify Lato fonts manually for Android vs. iOS and Debug this.circle.update
Adam	19.02.2018	10:15	11:00	Development	#25 Camera - can’t share picture - loads forever
All	19.02.2018	11:00	11:30	Meeting	Meeting with Sindre
Adam	19.02.2018	11:30	15:00	Development	#33 Camera - Crashes if switching camera type to front when the device has no front camera
Magnus	19.02.2018	14:45	17:10	Development	#13 I18n

Adam	19.02.2018	15:00	17:30	Development	#13 Internationalization - i18n - translate classes
Adam	20.02.2018	10:30	16:45	Development	#13 Internationalization - i18n - translate classes
Magnus	20.02.2018	10:30	15:30	Development	#13 Internationalization - i18n - translate classes
Magnus	20.02.2018	15:30	17:00	Development	#23 Social View button sometimes crashes when clicked
All	21.02.2018	10:00	10:30	Development	#23 Social View button sometimes crashes when clicked
Adam	21.02.2018	10:30	18:10	Development	#42 Fix crash when logging out - Social => Settings => Log out
Magnus	21.02.2018	10:30	16:00	Development	#41 Fix Instructions UI - Social => Settings => Instructions
Magnus	21.02.2018	16:00	17:00	Development	#40 Fix Collect Points UI - Free Climb => Collect Points =>YES
Adam	26.02.2018	10:20	11:40	Development	#42 Fix crash when logging out - Social => Settings => Log out
Magnus	26.02.2018	10:20	11:40	Development	Looking for issues
Magnus	26.02.2018	11:40	13:00	Development	#44: Make "Min Side" look nice
Adam	26.02.2018	11:40	12:40	Development	#25 Camera - can't share picture - loads forever
Adam	26.02.2018	12:40	14:30	Development	#43: No response from Free run => Hider markers.
Magnus	26.02.2018	13:00	13:15	Development	#45: Fix top statusbar on android.
Magnus	26.02.2018	13:15	16:30	Development	Fixing Decimals in top list , testing after merging master to android
Adam	26.02.2018	14:30	16:30	Development	#42 Fix crash when logging out - Social => Settings => Log out
Adam	27.02.2018	13:30	16:00	Development	Tested new changes in master branch
Magnus	27.02.2018	13:30	16:20	Development	Finding new Issues after merging with master
Adam	27.02.2018	16:00	17:15	Report	Working on report
Magnus	27.02.2018	16:20	17:15	Report	Looking at LaTeX and start planing
Magnus	28.02.2018	10:00	17:00	Report	Setting up report and learning more latex
Adam	28.02.2018	10:00	10:15	Development	Tested new changes in master branch
Adam	28.02.2018	10:15	17:00	Report	Working on report
Magnus	05.03.2018	10:20	11:00	Development	Looking for issues merging master into android
Adam	05.03.2018	10:20	11:00	Development	Tested new changes in master branch on Android emulator
All	05.03.2018	11:00	12:00	Meeting	Meeting with Sindre
Magnus	05.03.2018	11:00	16:00	Meeting	#55 Peaks nearby list, maybe smaller, overlaying map.
Adam	05.03.2018	12:00	12:15	Report	Adding meeting summary

Adam	05.03.2018	12:15	12:30	Development	#48 Camera - text not displaying after style changes
Adam	05.03.2018	12:30	14:00	Development	#25 Camera - can't share picture - loads forever
Adam	05.03.2018	14:00	16:00	Development	#52: Build android for release
Adam	06.03.2018	10:00	16:30	Development	#54 Merge EditPictureFrame and SharePictureFrame
Magnus	06.03.2018	10:00	16:00	Development	#55 Peaks nearby list, maybe smaller, overlaying map.
Magnus	06.03.2018	19:00	20:10	Report	Update hour log in report
Magnus	07.03.2018	13:00	17:00	Development	#55 Peaks nearby list, maybe smaller, overlaying map
Adam	07.03.2018	13:00	15:00	Development	#54 Merge EditPictureFrame and SharePictureFrame
Adam	07.03.2018	15:00	17:00	Development	#53: Make react-native-background-geolocation work on android
Magnus	12.03.2018	10:00	16:30	Development	#56: Make more descriptive ActionButton items
Adam	12.01.2018	10:00	16:30	Development	#54 Merge EditPictureFrame and SharePictureFrame
Magnus	13.03.2018	10:00	10:30	Development	#55 Peaks nearby list, maybe smaller, overlaying map
Magnus	13.03.2018	10:30	16:30	Development	#56: Make more descriptive ActionButton items
Adam	13.01.2018	10:00	11:30	Development	#54 Merge EditPictureFrame and SharePictureFrame
Adam	13.03.2018	11:30	16:30	Development	#53: Make react-native-background-geolocation work on android
Magnus	14.03.2018	12:00	15:00	Development	#56: Make more descriptive ActionButton items
Magnus	14.03.2018	15:00	15:30	Meeting	Meeting with Sindre items
Magnus	14.03.2018	15:30	17:00	Development	#56: Make more descriptive ActionButton items
Adam	14.01.2018	12:00	17:00	Development	#53: Make react-native-background-geolocation work on android
Magnus	19.03.2018	10:00	11:00	Development	Looking at changes
All	19.03.2018	11:00	11:30	Meeting	Meeting with Sindre
Magnus	19.03.2018	11:30	16:30	Development	3 Tutorial Messages
Adam	19.03.2018	10:00	11:00	General	General
Adam	19.03.2018	11:30	15:00	Development	#53: Make react-native-background-geolocation work on android
Adam	19.03.2018	15:00	16:30	Development	#3 Tutorial Messages
All	20.03.2018	10:00	17:00	Development	#3 Tutorial Messages
All	22.03.2018	14:00	18:00	Development	#3 Tutorial Messages
All	26.03.2018	10:00	17:00	Development	#3 Tutorial Messages
All	26.03.2018	10:00	17:00	Report	Writing report
Magnus	27.03.2018	10:00	16:00	Development	#3 Tutorial Messages

Adam	27.03.2018	10:00	16:00	Report	Writing report
Magnus	28.03.2018	10:00	16:00	Development	#3 Tutorial Messages
Adam	28.03.2018	10:00	16:00	Report	Writing report
Magnus	02.04.2018	11:00	17:00	Development	#3 Tutorial Messages
Adam	02.04.2018	11:00	17:00	Report	Writing report
Magnus	03.04.2018	10:00	16:00	Development	#3 Tutorial Messages
Adam	03.04.2018	10:00	16:00	Report	Writing report
Magnus	04.04.2018	10:00	16:00	Development	#3 Tutorial Messages
Adam	04.04.2018	10:00	16:00	Report	Writing report
All	09.04.2018	10:00	17:00	Development	#3 Tutorial Messages - Reset viewed pages in async storage
All	10.04.2018	10:00	16:00	Development	#3, #62, #63, #65
Magnus	11.04.2018	10:00	17:00	Development	#3 Tutorial Messages i18n, Feedback fixes, CameraEditTutorial
Adam	11.04.2018	10:00	17:00	Development	#68 - InstructionService, #67 - Social - Follow friends
Magnus	16.04.2018	10:00	17:00	Report	#75, Writing Report
Adam	16.04.2018	10:00	17:00	Development	#66, #76, #77, Writing Report
Magnus	17.04.2018	10:00	17:00	Report	Writing Report
Adam	17.04.2018	10:00	17:00	Development, Report	#71: Fix UI - Suggest a peak, Writing Report
Magnus	18.04.2018	10:00	18:00	Report	Writing Report
Adam	18.04.2018	10:00	18:00	Development	Issue #74 #70
Magnus	23.04.2018	10:00	17:00	Report	Writing Report
Adam	23.04.2018	10:00	17:00	Development	Issue #80 #85, #86
All	24.04.2018	10:00	17:00	Report	Writing Report
All	25.04.2018	10:00	17:00	Report	Writing Report
All	30.04.2018	10:00	17:00	Report	Writing Report
All	01.05.2018	10:00	17:00	Report	Writing Report
All	02.05.2018	10:00	17:00	Report	Writing Report
All	07.05.2018	10:00	17:00	Report	Writing Report
All	08.05.2018	10:00	18:00	Report	Writing Report
All	09.05.2018	10:00	17:00	Report	Writing Report
All	14.05.2018	10:00	14:00	Report	Finalizing report for submission

C Meeting Summaries

C.1 Meeting with employer Sindre Helelborgås

Thursday 11. January 2018, 10:00 - 10:30

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

General introduction.

Discussion

Introduced ourselves and mainly discussed some of Sindres plans and ideas.

Conclusions

Decided to learn more about React Native by following some online tutorials.

C.2 Meeting with supervisor Mariusz Nowostawski

Thursday 18. January 2018, 15:30 - 16:30

Present

Magnus W. Enggrav, Adam A. Jammary, Mariusz Nowostawski

Agenda for the meeting

Initial discussion on where and how to start, try to get some tips and pointers on how to write the project plan.

Discussion

Explained to Mariusz about our situation, that all the tasks for the project have not yet been clearly defined, and that they will unfold along the way.

Mariusz was very understanding, and guided us with practical tips for what to focus on in the plan and report.

Conclusions

He suggested writing more about the process instead of planning too much in detail, as we we will not know the details until later in the project.

Mariusz also suggested looking into using Latex for the Bachelor report.

C.3 Meeting with employer Sindre Helelborgås

Monday 22. January 2018, 11:30 - 12:00

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

Discuss how we should work on the repository.

Discussion

Sindre planned to change the login process.

Discussed linking dependencies in to the Android and iOS projects using 'react-native link'.

Conclusions

We will work on the android branch while employer merges our changes in to the master branch after testing.

We need to manually link new packages in to the Android projects since the 'react-native link' tool fails, Sindre will do the manual linking on the iOS projects.

C.4 Meeting with employer Sindre Helelborgås

Monday 5. February 2018, 11:30 - 12:00

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

Find out how we should handle the rendering difference between Android and iOS.

Discussion

Android needs to render things in the correct order and does not work like iOS that can use Overflow:’hidden’ to make children render above their parent. What should we do about this?

Conclusions

Maybe create some new classes specifically for Android, or maybe redesign some stuff on Android. Sindre suggested that we start experimenting ourselves to find the best solution. He suggested either making a new class/component hierarchy by refactoring the existing code, or keep the existing hierarchy for iOS and make a separate one for Android.

C.5 Meeting with employer Sindre Helelborgås

Monday 19. February 2018, 11:00 - 11:30

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

- Discuss the latest changes in the android branch
- Discuss the latest changes in the master branch
- Discuss the issues found after merging branches
- Discuss what to work on next

Discussion

Sindre talked about meeting with product owner, and how there may be possible changes to the UI.

Discussed the issues we were working on, and the latest merge with changes from the master branch.

#20: App randomly crashes on startup

Sindre will take a look, seems the circle is not currently drawn, can be ignored if possible for now.

#25: Camera - can’t share picture - loads forever

Seems the problem lies on the backend server, will leave the issue open for now and continue on other issues in the meanwhile.

Conclusions

Concluded that we will continue working on current issues in Bitbucket, and let Sindre know when we are finished.

C.6 Meeting with employer Sindre Helelborgås

Monday 5. March 2018, 12:00 - 13:00

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

Discuss issues found last week, and what to work on this and next week.

Discussion

Issue #46: Social => Retrieving friend info . . .

Sindre refactored social services and other stuff in the core architecture that may have affected all service-based features.

Issue #47: My Site => error

Sindre is not working on it, we can fix it.

Issue #25 Camera - can't share picture - loads forever

Skal jeg legge til en liten alert melding som sier at den feila å dele/laste opp bildet?

Yes, add alert.

Sindre discussed caching, offline picture management, but this was only an idea for the future if time is available.

New issue on share page:

Text on bottom, difficult to click on message input field switch is displayed on share page, does nothing, should be hidden.

Will redo UI, maybe single edit/share pic, see issue #54 Merge EditPictureFrame and SharePictureFrame.

Issue #48 Camera - text not displaying after style changes

Skal jeg legge tilbake button, eller planlegger du noen UI endringer på kamera sidene?

Seems to be fixed?

Yes, is fixed.

Should map centering icon be displayed on top-right corner?

Probably specific to Android when displaying Google Maps.

Conclusions

What should we work on?

- Merge iOS branch in to android branch
- Merge camera single edit/share pic?
- Peaks nearby list, maybe smaller, overlaying map.
- https://projects.invisionapp.com/share/WEEPGYGGN/screens/266597100
- LocationServices:
- - android vs iOS, geolocation
- - https://github.com/transistorsoft/react-native-background-geolocation

C.7 Meeting with employer Sindre Helelborgås

Monday 19. March 2018, 11:00 - 11:30

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

- Discuss current issues
- Discuss what to work on next

Discussion

#47: My Site => error

Sindre said he had fixed the issue, but it still loads data forever on Android.

Free (collect points): UI overlaps

Sindre was working on a new iOS release, and will look at the issue after the release is complete.

New tasks

#3: Tutorial Messages

- Create a popup service that can called from any component
- Keep track of which popups/instructions that have already been seen, and only display unseen popups.
- Can be disabled/reset in settings
- White box with drop shadow
- Arrow ?
- Make individual instruction popups based on existing instructions slides

- LoginPage subscribes to instructions service

#59 Hill search function

Conclusions

Sindre said that the master branch could safely be merged in to the android branch now.

C.8 Meeting with employer Sindre Helelborgås

Wednesday 11. April 2018, 10:30 - 11:00

Present

Magnus W. Enggrav, Adam A. Jammary, Sindre Helelborgås

Agenda for the meeting

- Discuss current issues
- Discuss what to work on next

Discussion

#3: Tutorial Messages

- Show Popups first time
- Saves viewed popups in async storage
- Popup bubble linked to page
- Page can have multiple popup bubbles
- Popup bubble can have multiple pages
- Reset viewed pages in async storage
- Missing: i18n

Conclusions

- Q: How does it look?
- A: Add darker background to separate popup from background
- A: Suppress popup when another popup is visible
- A: Change color on next button
- A: Make buttons smaller and move them more away from edges
- Q: How does services work and how do we use them?
- A: Haven't looked at it yet
- Q: Other issues, bugs, new tasks?
- A: Everything works on iOS
- A: We will have another look at what's different on Android
- Q: Merge status? Master => android

- A: OK to merge

D Backlog

#	Type	Title
86	Bug	Start Climb - MapCard - SideHillList - Crash on android branch
84	Bug	Random crash when logging in on mobile device
83	Bug	Search/Filter Hill List - Number filter button is not scaled or center-aligned vertically
82	Bug	Search/Filter Hill List - Number does not work as expected
81	Bug	Failed to build ios branch with updated camera dependency
79	Bug	Random focus to location at startup
78	Bug	Fix - Social - Stories - share picture
73	Bug	Fix: Peaks are not listed and displayed in map on startup
72	Enhancement	Offline Fix
69	Enhancement	Clean up i18n locale file content
61	Enhancement	Google play debug version
58	Enhancement	Create hills automatically capture hill
57	Proposal	Asset image overlay-graph.png contains the text 'HØYDEPOENG'
52	Task	Build android for release
34	Bug	Facebook friends permissions
6	Task	Fresh Data
5	Task	Groups - Plan
4	Bug	Fix offline
3	Proposal	Tutorial Messages

E Email Communication

E.1 Wednesday, 24. January 2018

Fra: Adam Abdellah Jammary
Sendt: 24. januar 2018 14:50
Til: Mariusz Nowostawski
Kopi: Magnus Wirgeness Enggrav
Emne: SV: BSP3900 Bachelor Project - Project Plan
Thanks Mariusz, we'll go with how it is now.

Fra: Mariusz Nowostawski <mariusz.nowostawski@ntnu.no>
Sendt: 24. januar 2018 14:34
Til: Adam Abdellah Jammary
Emne: Re: BSP3900 Bachelor Project - Project Plan
Hi Adam,

It can be separate, but it can be like you have it now. It looks much better, and the context is clear. Go ahead with that.

Cheers
Mariusz

——Original Message——
From: Adam Abdellah Jammary <adamaj@stud.ntnu.no>
Date: Wednesday, 24 January 2018 at 14:30
To: Mariusz Nowostawski <mariusz.nowostawski@ntnu.no>
Cc: Magnus Wirgeness Enggrav <magneng@stud.ntnu.no>
Subject: SV: BSP3900 Bachelor Project - Project Plan
Something like this?

Or is it important that the front cover page is separate from the content of details?

———————————————-
Fra: Mariusz Nowostawski <mariusz.nowostawski@ntnu.no>
Sendt: 24. januar 2018 14:05:43
Til: Adam Abdellah Jammary
Emne: Re: BSP3900 Bachelor Project - Project Plan
Hi ,

Make sure that the project has at the very start of the document:

Title of the project

Your names (both)
Date (current, date of submitting the doc)
The rest of the document is looking OK.

More answers inline:

—–Original Message—–
Also, regarding the project agreement (between us and the company), Blackboard says only one member of the group needs to upload the documement. Magnus uploaded it, but since there is no group upload available, my blackboard doesn't register it as uploaded for me. Should I just ignore it?

Yup, ignore it. If one of you submitted it, then it is fine.

kind regards,
Adam and Magnus

E.2 Tuesday, 27. February 2018

Fra: Mariusz Nowostawski <mariusz.nowostawski@ntnu.no>
Sendt: 27. februar 2018 17:18
Til: Magnus Wirgeness Enggrav
Kopi: Adam Abdellah Jammary
Emne: Re: Bachelor project question
Hi,

It is OK for you to change and give a slightly different title in your final submitted report, so, dropping "gamification" from the title is fine. Also, "App and gamification" as a title feels a bit odd - rather as a placeholder, so, you probably should think and design a better title anyway, for you final report, such that it conveys the actual nature of your project. Do not worry about it at this stage - it can be done later, once you have the actual shape of the final delivery for the report.

Hope that helps,

Cheers
Mariusz

—–Original Message—–
From: Magnus Wirgeness Enggrav <magneng@stud.ntnu.no>
Date: Tuesday, 27 February 2018 at 16:58
To: Mariusz Nowostawski <mariusz.nowostawski@ntnu.no>
Cc: Adam Abdellah Jammary <adamaj@stud.ntnu.no>
Subject: Bachelor project question
Hi Mariusz

We have a question for you:

Our offical title of the bachelor project is "App and gamification". However our task is make an app work on android using react native. and possible do other tasks from their backlog. So i feel like the title is a bit off. is this a problem?

Or can we use our own title in the report since it is possible that there will be no gameification, depending on their backlog.

Kind regards,
Magnus Enggrav
Adam Jammay

F Project Agreement

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Klimb AS

(oppdragsgiver), og

MAGNUS ENGGRAV

ADAM JAMMARY (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 15.1.18 til 16.5.18.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): MARIUSZ NOWOSTAWSKI

Oppdragsgivers kontaktperson (navn): Elin Årseth

Student(er) (signatur): Adam Jammary dato 16.1.18

Magnus Enggrav dato 16.1.18

dato

dato

Oppdragsgiver (signatur): Elin Årseth dato 15/1-18

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): dato