Expertise
and insight
for the future

metropolia.fi/en

Metropolia
University of Applied Sciences

Muhammad Monis Amjad

Designing and developing a smart commuting application to support healthy lifestyles

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

31st May 2021

PREFACE

I would like to thank my colleague Johannes Bohren for suggesting this excellent topic for my thesis. When I heard about this concept for the first time, I didn’t know that it was even possible to develop such an application which can be beneficial for so many areas like climate change, reducing co2 footprints, employee health etc. I have faced a lot of challenges. Mostly, it was trying to find the peace of mind to write and study the different areas required to complete this paper, especially after office hours with 2 energetic kids.

I would like to give credit to my wife, Hafiza Mariam Khan, for her help in managing my routine and taking all the burdens on her shoulders so that I could concentrate on my studies. I would also like to thank Ville Jääskeläinen for encouraging and pushing me so that I could complete my studies as soon as possible, and also for providing loads of ideas on how to write a good paper, along with time management. I would also like to thank Zinaida Grabovskaia and Vesa Ollikainen for designing the course in a way that helped me to write the thesis paper easily.

I would also like to thank my friends and university teachers, especially Dr. Humera Tariq and Dr. Nadeem Mahmood, who suggested the idea of completing my Masters as soon as possible: finally, after 10 years, I have managed to follow their suggestion.

Last but not least, I would like to thank my parents Shair Muhammad and Rubina Shair for keeping ne on the path of studies during my early years.

Nurmijärvi, 31.05.2021
Muhammad Monis Amjad

Author	Muhammad Monis Amjad
Title	Designing and developing a smart commuting application to support healthy lifestyles
Number of Pages	77 pages
Date	31 May 2021
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Head of IT Master's program

Obesity is one of the great challenges of modern civilization. Heart disease, diabetes and other conditions related to obesity are already a major problem. People become lazy when encouraged to use their own car instead of public transport or walking to work: this leads to more problems, such as traffic congestion, pollution and parking issues. The aim of this thesis is to design and develop the basic required characteristics, functionalities, options and features of an application which can be used by companies to provide incentive to their employees and provide analytics about the wellbeing of these employees. It is all about incentivizing people and organizations to make choices that have a positive impact on the environment. The mobile application Cleancentive was designed, developed and tested for companies so they can support their employees with their physical health activities.

The study shows that while testing the application, the daily routine and lifestyle of the author was benefitted, bringing more energy for the day and also increasing productivity. It was also found that incentive models may need to be customised on a company-by-company basis to find out which model works best for them, always bearing in mind that people tend to get motivated and excited when they’re getting some kind of financial reward.

Keywords	Smart Commuting, Wellbeing, Health

Contents

List of Abbreviations

AJAX	Asynchronous JavaScript And XML
API	Application programming interface, provides an interface so that two or more applications can communicate with one another.
CSS	Cascading style sheet
HTTP	Hypertext transfer protocol
IDE	Integrated development environment
MPA	Multi-page application
MVC	Model view controller
MVP	Minimum viable product
MVVM	Model view viewmodel
NPM	Node package manager
REST	Representational state transfer
SOAP	Simple object access protocol
SSR	Server side rendering
UI	User interface
UX	User experience
VCS	Version control system

1. Introduction

Obesity is one of the great challenges of modern civilization. Heart disease, diabetes and other conditions related to obesity are already very costly. Lower productivity, an increase in taxes to help fund an overworked healthcare system and, on a personal level, lowered resistance to other health threats such as Covid-19 are just the most obvious outcomes. Obesity is caused by many different factors. Besides eating too many sweets and processed food, a major factor is a lack of basic physical movement. Although workers typically commute further than ever before, many move their bodies too infrequently. Many people are spending their work time in front of a computer screen with little physical movement during their daily routine. Most spend a full day sitting down, not really moving their bodies. Nowadays each and every company offers health care to their employees precisely because inactivity in their daily routines creates numerous health issues, and the companies are simply covering their own backs.

Most employees use their car to get to the office which leads to pollution, traffic congestion and parking issues. It is a commonly accepted fact that human beings are overstretching the planet's capacities with their lifestyles and resource usage. A major factor in this usage is the phenomenon of commuting to work by car. It is crucial that we research new technologies and find innovative solutions to address this. At the same time, people also have to learn to value their mobility more, and discover smarter and healthier ways of moving around.

Many companies are willing to invest considerable sums in corporate responsibility and sustainability programs, partly because of customer demand and existing regulations. As a result, many platforms and projects have emerged where emission rights can be traded or compensated. The transparency for many of these is limited, and critics talk of greenwashing. It is crucial to first improve one’s own operations and then support projects which have a positive impact, in line with the views and intentions of the company itself. Think globally and act locally, also when it comes to compensation for emissions.

The visionary design of a new application called Cleancentive is all about incentivizing people and organizations to make choices that have a positive impact on the environment. By incentivizing healthy mobility, it can improve public health issues. Giving positive and negative externalities a price, the "healthy" becomes cheaper, and the "dirty" can't freeride. By offering cities analytics and data about commuting behaviour, behavioural changes and infrastructure bottlenecks, it can help to establish a genuinely "smart city".

The aim of this thesis is to design and develop the basic required characteristics, functionalities, options and features of an application which can be used by companies to provide incentive to their employees and provide analytics about the wellbeing of these employees. This means developing a mobile application to answer questions such as:

- What technologies could be used to develop such a tracking application?
- What benefits does it bring to the employee and employer?
- What kind of incentive model works best?

Codistan Oy, as a sustainable solution provider company, is focusing on helping both companies and consumers to adopt efficient and convenient methods for everyday tasks with the help of local communities.

Codistan's goal, with the help of this thesis and the innovative product, is to fight climate change and other environmental issues related to transport with the help of technology, data and a market approach. The study is important in many ways and it will help to solve the following challenges:

- Unhealthy lifestyle choices
- Weight management issues
- Joint and skeletal problems

- Cardiovascular diseases
- CO_2 emission levels

The study includes the development of mind maps of the concept, a visual design of the application and the development and deployment of the application. It also includes analyses and testing of the application model to find health benefits for the employees, along with their experience of the app, a compensation model and health analysis along with analyzing the commuting habits of employees and their impact on pollution and CO_2 offsetting. The study doesn’t cover major problems such as infrastructure, public transport, traffic congestion and parking improvements.

This thesis has been divided into seven sections. The first section introduces the topic of the thesis, the second sections covers the background and history of the topic, the third section explains the development of the mind map and feature selection for the application, the fourth section explains the design process of the application, the fifth section describes the developmental journey, the sixth section shows the testing methods and the final section defines outcomes, results and the further development needs of the application.

2. Method and Material

During the development of any web and mobile application, the theory involves some key processes of software development, user activity monitoring, human resources and compensation. It also needs to take into account how to attract potential users to the application in a better way, by providing a proper user interface and considering the user's needs. The application design and interface should be the most important things in a new software system. This involves going through several iterations with users to understand the usability, reliability, trust, extensibility and easiness in using the application, all important aspects which shouldn't be ignored. This section covers the development background of the application.

2.1. Background of this Thesis

The idea of the application originates from the Chief executive officer (CEO) of Codistan Oy, Johannes Bohren, who came up with an idea to compensate users for not using cars to avoid polluting the environment and to reduce traffic congestion. This idea became more interesting to the author because of its focus on compensating company employees for using bicycles, or walking/running or using public transport to get to the office, from a health benefits perspective. Enjoying a healthy lifestyle and reducing carbon emissions are very important to the author, and Codistan Oy wanted to reduce the amount of money paid for health facilities. Instead, the money could be spent on people's well-being so that they remain healthy and don't have to use health care services so often.

The application created in this thesis was originally the kind which is able to track user activity automatically as soon as the user starts moving. However, the scope of the application was narrowed down to a more traditional way of recording user activity by providing an option to the user to start, pause and end the trip and activities. The application is able to track the activity of the user and the mode of transport. On the basis of the type and distance of an activity, benefits will be calculated and transferred to the user. There is no actual money transfer integration being done as part of the scope of this thesis.

The suggestions for future development added to the application include determining actual user physical activity and the compensation of users based on the agreed/proposed model. Multi-language support is added as part of the solution so it will be possible to target more users. The mobile application will be tested by Codistan employees, who will receive compensation for being healthy.

2.2. Research Design and Solution Development

This thesis is based on knowledge which was collected and organized by reading various articles on the internet, technical solutions and their documentation and interviewing clients and their employees to generate a feasible solution for them. Designing the application was thoroughly based on collected data. As the application and its concept is unique in nature and requires an understanding of different systems, there was no existing reference solution available related to the concept itself. However, there is much information available about different technical solutions, concepts and designs separately in their domains which were later merged to form the design of the application. Youtube, Stackoverflow, Github, Wikipedia and technical documentation of the technology being used were good sources to search for help, but the biggest help was provided by the CEO of Codistan in testing and formalizing the concept of this thesis, and the thesis instructor who has helped a great deal in reviewing the content of the thesis thoroughly and frequently. Once the required information had been collected it was comparatively easy to develop the solution. Figure 1 below describes the design process of this thesis.

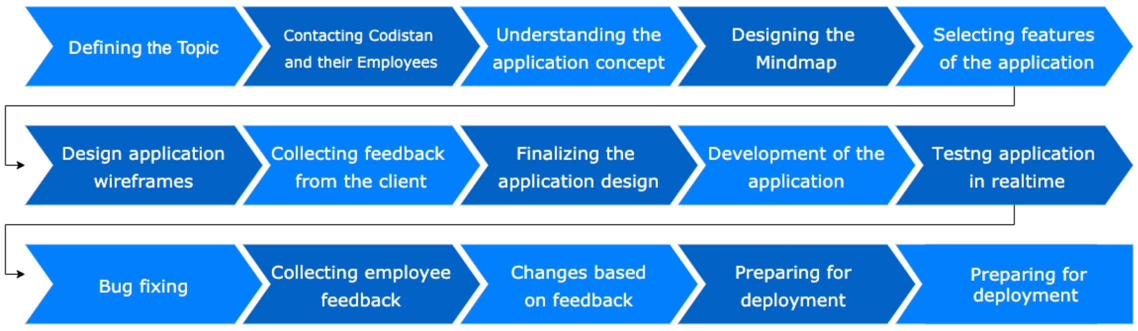

Defining the Topic
Contacting Codistan and their Employees
Understanding the application concept
Designing the Mindmap
Selecting features of the application
Design application wireframes
Collecting feedback from the client
Finalizing the application design
Development of the application
Testng application in realtime
Bug fixing
Collecting employee feedback
Changes based on feedback
Preparing for deployment
Preparing for deployment

Figure 1. Application design process.

The application design process for creating a minimum viable product began by having a meeting with Codistan and their employees. The target was to understand the application concept by identifying the core elements of the system, designing a mind map of the application and selecting a feature set from the mind map. After several meetings over the course of 2-3 months, a wireframe of the application was finalized. During the application UI design, the client’s colour preferences were kept in mind and wireframes were transformed to a functional user interface using Adobe XD to try out the look and feel of the real application.

2.3. Solution Evaluation, Reliability and Validity

The evaluation of the solution was done throughout the whole writing and coding process. It was done by testing the application in a normal work day’s routine by the CEO of the company and the author himself, along with a few other volunteers and friends. The reliability of the solution was tested through constantly using the application in different weather conditions both in winter and spring. It was tested by trying out different modes of transportation such as car, bicycle, public transport and walking. Only one compensation model was tested with Codistan, as agreed with them.

The model worked for their employees, so it wasn’t necessary to try out different compensation models. Codistan will continue developing and customizing the application to include new features and try different compensation models with different companies. The application demo was presented at the end of the thesis to Codistan.

3. Technologies Used for Tracking Application

The word “Cleancentive'' was a combination of two words “Clean” and ''Incentives''. It literally means: get an incentive for keeping the environment clean. This chapter explains different ways that have been used to counter climate change, the history and theory in the background, the basics of creating web and mobile applications, technical challenges, and a short comparison of different programming solutions.

3.1. Developing Tracking Application

The process of creating a mobile or web application is widely known in the software industry. Developing web applications requires knowledge of HTML and CSS to make it look attractive. It also requires knowledge of API to provide an interface for the database and JSON/XML knowledge to work as a transfer medium. On the other hand, developing mobile applications requires understanding of mobile platforms such as iOS or Android to build frontend of the mobile and API. On top of that, it is also necessary to have knowledge of the options which cloud providers offer for deploying the web application, along with understanding the application publishing process to Google Play and the Apple store. This chapter covers everything needed to develop the web and mobile application.

3.1.1. Technology Decision for Web Application

There are many web technologies present today to help develop the web application, some of which have existed for more than 20 years in the market and have proven to be strong enough to be the default choice for many developers. To develop Cleancentive, there were two choices: either to develop everything integrated in one application by using some php based frameworks such YII2 or Laravel or ASP.NET or Java servlet application, or to have it created frontend of the web application independent of backend. Both have their pros and cons.

Developing integrated applications means that there are several pages with static or dynamic information (text, images, etc) and links to the other pages with the same content. During a jump to another page, a browser reloads the content of a page completely and downloads all resources again, even the components which are repeated throughout all pages (e.g., header, footer) [1]. HTML, CSS and Javascript along with any backend technologies such as PHP, JSP, ASP etc. can be used for building multi-page web applications. The Multi Page Application (MPA) life cycle usually looks as show in the picture below:

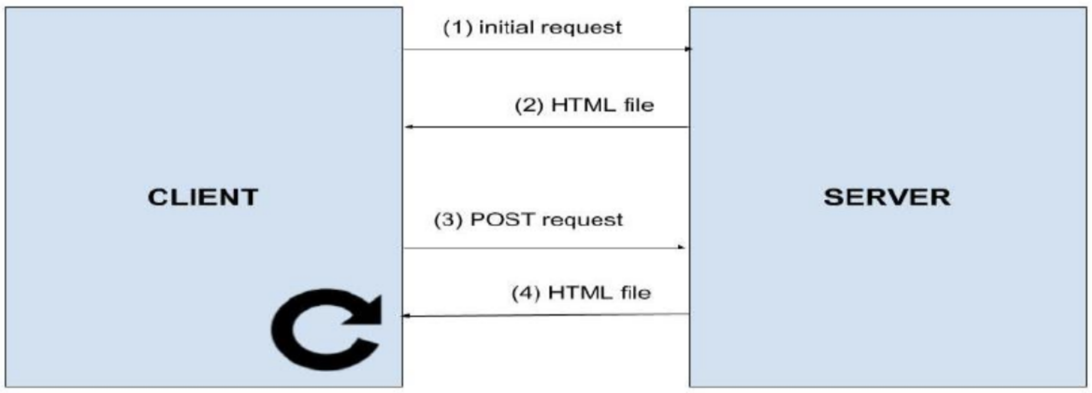


Figure 2. Multi page web applications lifecycle [2]

On the other hand, Single Page Application (SPA) is a web application hosted on a single web page that downloads all the necessary code for the job, along with the page itself [2]. It usually loads a single piece of content and updates the page as the user interacts with the app. They use AJAX and HTML5 to create fluid and responsive Web apps, without constant page reloads. This means that much of the processing happens on the client side, mainly using JavaScript. The picture below explains the lifecycle of a single page application:

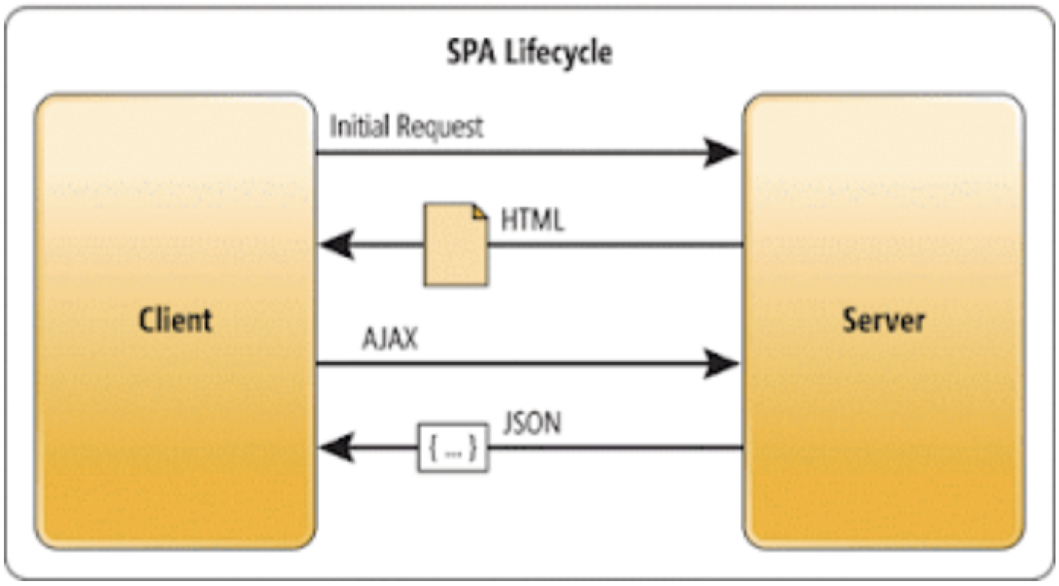


Figure 3. Single page web applications lifecycle [3]

The development of MPA is usually fast, so the system could be quickly got ready within a few months but it would also require developing some sort of interface for mobile applications to be connected with the backend. In the normal world it would require double development by creating integrated applications and creating some sort of API to connect mobile devices. This also means that it requires business logic to be applied in two different places, which is not ideal when the products become huge and also not good in terms of maintenance and the scalability perspective. Sometimes in integrated applications, when updating certain models it is also required to update the frontend logic of the application, which leads to different problems.

Table 1. Difference between SPA and MPA [4]

SPA	MPA
Responsiveness and High Speed	SEO
Offline Opportunities (Caching)	Scalability
Easier to make mobile responsive	Existing frameworks and solutions
Decoupled frontend	

Cleancentive requires an API to provide an interface for the mobile application and for the web application. SPA is famous for its decoupled frontend logic, now with modern framework scalability, and Search Engine Optimization (SEO) is not a big issue so it was decided to develop the web application as an SPA so it could use the same API. It would also provide a possibility to change the API or frontend technology separately in future, if needed, without affecting another part of the application.

Now that it had been decided how the web application would be built, it required a few more choices for frontend application and API.

3.1.2. Technology Decision for Frontend

This section covers what was chosen for the frontend application. There are many frontend technologies available in the market: the most famous of these are AngularJs, ReactJs or VueJS.

AngularJs

AngularJS is a JavaScript-based open-source front-end web 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 [5]. It aims to simplify both the development and the testing of such applications by providing a framework for client-side Model View Controller (MVC) and Model–View–Viewmodel (MVVM) architectures, along with components commonly used in web applications and progressive web applications. AngularJS is used as the frontend of the Mongo Angular Express NodeJs (MEAN) stack, consisting of MongoDB database, Express.js web application server framework, AngularJS itself, and Node.js server runtime environment. AngularJS's design goals includes:

- Decoupling Document Object Model (DOM) manipulation from application logic. The difficulty of this is dramatically affected by the way the code is structured.

- Decoupling the client side of an application from the server-side. This allows development work to progress in parallel and allows for reuse of both sides and to provide structure for the journey of building an application: from designing the UI, through writing the business logic, to testing.

ReactJs

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality [6].

React makes it painless to create interactive UIs and design simple views for each state in the application. It will efficiently update and render just the right components when data changes occur. Declarative views make the code more predictable, simpler to understand, and easier to debug. It makes it possible to build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, it can easily pass rich data through the app and keep state out of the DOM.

VueJs

Vue.js is an open-source model–view–viewmodel front end JavaScript framework for building user interfaces and single-page applications. It was created by Evan You, and is maintained by him and the rest of the active core team members [7].

Vue.js features an incrementally adaptable architecture that focuses on declarative rendering and component composition. The core library is focused on the view layer only. Advanced features required for complex applications such as routing, state management and build tooling are offered via officially-maintained supporting libraries and packages, with Nuxt.js as one of the most popular solutions. Vue.js lets the user extend HTML with HTML attributes called directives. The directives offer functionality to HTML applications, and come as either built-in or user defined directives.

Table 2. Comparison between AngularJs, ReactJs and VueJs [8]

React	Angular	Vue
It is a library of JavaScript	It is a framework of JavaScript	It is a framework of JavaScript
Small-size, fast bundles are provided.	Medium-sized fast bundles are provided	Small, fast bundles are provided
Provides only a core set of instructions.	Provides a huge set of features.	Provides a medium size set of features.

Some additional improvements are provided.	Lots of additional improvements are provided.	Some additional improvements are provided
Popular and relatively mature library.	Popular and relatively more mature.	Extremely popular and relatively mature.
It is well established and is being used, also easy to learn and implement.	Developed and maintained by Google.	Open source team effort. Establishing Framework and needs more time to capture the market

Since ReactJs is a library, it doesn't come with all sorts of unnecessary code and whatever is needed for the application can just be added separately. It also possesses a small bundle size and is widely used in the community. It was decided to use ReactJs for Cleancentive because the author has previous working experience with ReactJs. It was also the choice of development by the client.

The client also wanted Typescript to be used to develop the system along with ReactJs because Javascript is a statically-typed language, and it is very difficult to manage the code with it. Typescript is the layer built on top of Javascript to provide the possibility of using types, which makes it easier to manage the code and scale well in future. For these reasons, it was decided to use Typescript. Below is the comparison between Typescript with Javascript.

Table 3. Comparison of typescript vs javascript [9]

Typescript	Javascript
Superset of JavaScript developed to overcome code complexity for large projects	A scripting language that helps create dynamic web page content
Errors can be found and corrected during compile time	Errors can be found only during run-time as it is an interpreted language
Strongly typed, supports both static and dynamic typing	Weakly typed, no option for static typing
Converted into JavaScript code to be understandable for browsers	Can be directly used in browsers
Since it is a superset, all the JavaScript libraries, and other JavaScript code works without any changes	JS libraries work by default
There is support for ES3, ES4, ES5 and ES6	No support for compiling additional ES3, ES4, ES5 or ES6 features
Supports modules, generics and interfaces to define data	No support for modules, generics or interface
Functions can have optional parameters	This feature is not possible in JavaScript
Numbers, strings are considered as interfaces.	Numbers, strings are objects.
Powerful and intuitive language	Neat and clean, most suitable for simple web applications
The community support is still growing and not so huge	Huge community support with lots of documentation and support for solving issues

Prototyping is possible	Prototyping support is not present
Takes time to learn and code, scripting knowledge is a must.	Can be learned on the go, no prior scripting experience is needed.
Proper build setup (npm package) is required for static type definitions	No build setup is required

The biggest problem with the single page applications is that they are rendered on the client side. It was required by the client for the page to be parsed by search engines. It is necessary for the application to be rendered on the server side and send the HTML from the server, so that search engines will be able to understand the content of the pages. This means it requires Server Side Rendering (SSR). There is a mechanism available in React which could be used to render the application on the server side, but that would mean that it has to build and maintain the logic, which is not ideal. It also needs routers which means that the application has to make sure that SSR works along with routes, and also in different languages. For this purpose, it was decided to use NextJs, which is a framework built using ReactJs, and provides an easy routing interface as well as server side rendering. Also, it causes the development bundle to be loaded when it is required. React-i8n-next-js module was added separately to provide the support of internationalization in the application.

Now that the development technologies for the frontend had been decided, it was time to select the styles technology to make the application look good. There are many alternatives, starting from a plain Cascading Style Sheet (CSS) or using some framework such as Bootstrap or MaterialUI or using preprocessors similar to LESS, SASS, POSTCSS etc or StyleJsx or StyledElements. CSS is very difficult to manage for large scale applications, and requires more time to write code and add unnecessary duplications. It also adds to the complexity that if some class name or id were used in one place then there is a chance that it will affect another place with the same class name. LESS, SASS and POSTCSS help in avoiding the problem of code duplication and makes it easy to write and maintain css code so technically it could use any of these, but it doesn’t solve the problem of duplicate naming, so it needs to combine some technique such as BEM styles to make the naming specific to the component. StyleJsx and StyledElement solve this problem automatically by randomizing the styles but there is a limitation to that, as well as the fact that randomizing makes it easier to target child elements from the parent component. After considering all these permutations it was decided to choose NextJs SASS module along with StyleJsx.

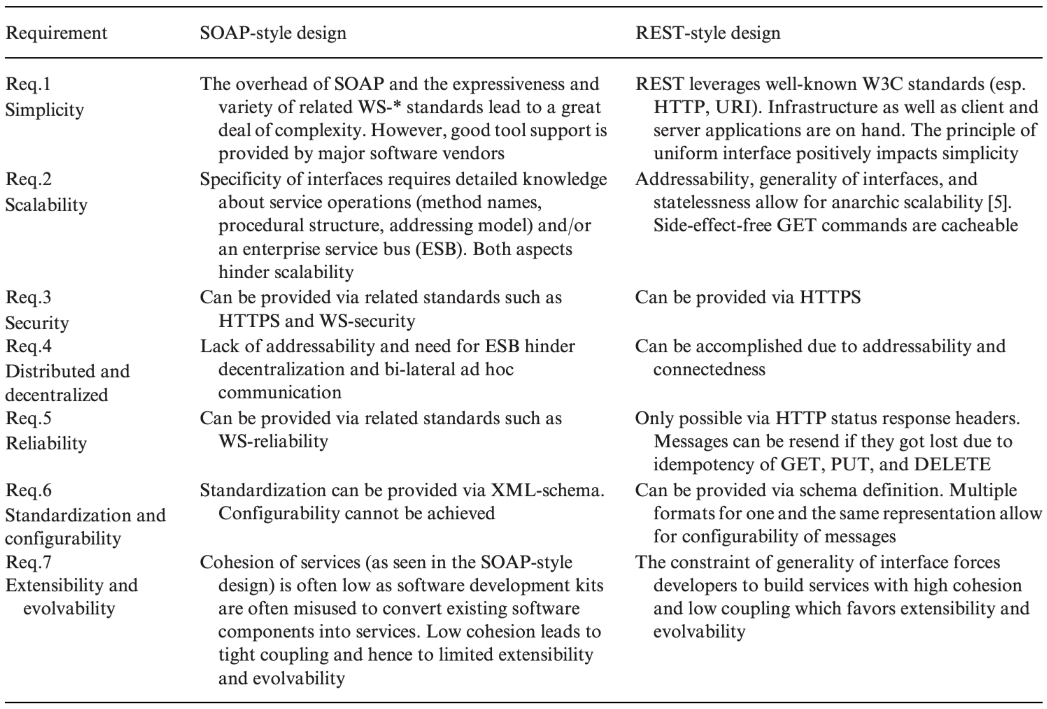
3.1.3. Technology Decision for API

API stands for Application Programmable Interface, and is a software intermediary that allows two applications to talk to each other [10]. It was discussed in 3.2.3 that Cleancentive uses a single page application which requires an API to provide an interface to the database. It will be used to store and retrieve the trip, activity and other data to Web and Mobile applications.

Currently there are two different architectural styles mainly used in Service Oriented Architectures (SOA). One of them is Rest and the other one is Simple Object Access Protocol (SOAP). This chapter covers the comparison of the two styles – the SOAP-style with procedural designs and the REST-style with loosely coupled service.

Rest is commonly used to create interactive applications that use Web services. A Web service that follows these guidelines is called RESTful [11]. SOAP on the other hand uses XML for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP) for message negotiation and transmission [12].

The table presented below was taken from ***Comparing Architectural Styles for Service-Oriented Architectures – a REST vs. SOAP Case Study*** and explains the head-to-head comparison between SOAP and REST. As per the author, The REST style comprises many services (or better still, resources) which are explicitly linked to each other. This might look complicated and overwhelming in the beginning, but due to the uniform and fixed interfaces of each service it leads to a much looser coupling in the end. [13]



Requirement	SOAP-style design	REST-style design
Req.1 Simplicity	The overhead of SOAP and the expressiveness and variety of related WS-* standards lead to a great deal of complexity. However, good tool support is provided by major software vendors	REST leverages well-known W3C standards (esp. HTTP, URI). Infrastructure as well as client and server applications are on hand. The principle of uniform interface positively impacts simplicity
Req.2 Scalability	Specificity of interfaces requires detailed knowledge about service operations (method names, procedural structure, addressing model) and/or an enterprise service bus (ESB). Both aspects hinder scalability	Addressability, generality of interfaces, and statelessness allow for anarchic scalability [5]. Side-effect-free GET commands are cacheable
Req.3 Security	Can be provided via related standards such as HTTPS and WS-security	Can be provided via HTTPS
Req.4 Distributed and decentralized	Lack of addressability and need for ESB hinder decentralization and bi-lateral ad hoc communication	Can be accomplished due to addressability and connectedness
Req.5 Reliability	Can be provided via related standards such as WS-reliability	Only possible via HTTP status response headers. Messages can be resend if they got lost due to idempotency of GET, PUT, and DELETE
Req.6 Standardization and configurability	Standardization can be provided via XML-schema. Configurability cannot be achieved	Can be provided via schema definition. Multiple formats for one and the same representation allow for configurability of messages
Req.7 Extensibility and evolvability	Cohesion of services (as seen in the SOAP-style design) is often low as software development kits are often misused to convert existing software components into services. Low cohesion leads to tight coupling and hence to limited extensibility and evolvability	The constraint of generality of interface forces developers to build services with high cohesion and low coupling which favors extensibility and evolvability

Figure 4. Evaluation of SOAP-style and REST-style

Based on the comparison above, it was decided to use REST API for Cleancentive because REST is easy to implement and many web applications nowadays use REST-based architecture for API. It was also one of the suggestions by the client to use REST API to connect Cleancentive mobile and web applications.

Now that it was decided that the backend of the application would be developed using REST API, It was time to select what technology would be suitable for developing Cleancentive API. Similar to other technologies there are many options available to develop Rest API. One possibility is to use NextJs, a technology which is used for developing the frontend, to develop the API as it provides support for API as well. Another alternative is to use standalone NodeJs along with ExpressJs, Spring Boot or some PHP frameworks etc. The author has worked on Spring Boot and NodeJs before, they are very well established frameworks and provide good support for building rest-based API. Kotlin is getting popular these days because of its interchangeability with Java, support of writing web, mobile and desktop applications.

Table 4. Comparison of typescript vs javascript [14]

Features and functionalities	NodeJs	Java
Rock-solid foundation		🟩
Ubiquity	🟩	
Better support of IDEs		🟩
Database queries		
Types	🟩 with TS	🟩
Syntactic flexibility	🟩	
Simple build process		🟩
JSON	🟩	
Remote debugging		🟩
Handhelds		🟩
Desktop	🟩	
Libraries	🟩	🟩
Solid engineering		🟩
Speed	🟩	
Threads		🟩

Momentum	🟩	

It was suggested to use Kotlin (Ktor) framework to develop a Cleancentive API as the author also wanted to learn something new while working on Cleancentive but due to time constraints and based on the above comparison it was agreed to use NodeJs to create quick MVP of the application.

3.1.4. Technology Decision for Database

For databases there are many popular alternatives such as SQL or NoSQL. In relational databases there are many choices such as Postgres and MySql (which are the most popular ones) and also NoSql or MongoDb could be used. But the application can’t use any of these, as Cleancentive will have a relational database structure as well as a time series kind of structure where it requires the need to store user location from time to time and relational databases are not ideal for this.

Because of the kind of structure needed in the application, it was decided to use a combination of Relational database along with NoSql document based structure to store user trip and activity data in a column of the relational database. It was also decided to use SqlLite on mobile applications to store active trip data, to avoid unnecessary calls to API and once the trip is completed it will send all the trip data at once to the server.

3.1.5. Technology Decision for Mobile Application

Since the client wanted to develop an application which runs on both Android and iOS, there were only two options for developing the application.

Native Technology

Native technology means that applications will be running in the native language such as Swift or Objective C for iOS and Kotlin or Java for Android. This would mean maintaining two separate developments of the same application which requires two separate code bases and two separate maintenance and bug fixing processes. It would double the development time, so if an application has certain features which would require at least 2 months of work, developing in the native language using this approach means that it will take at least 4 months. This also increases the testing time and makes operations tasks such as release to the app store, fixing bugs etc. difficult to manage in the future. It would also require that the author has enough knowledge of at least one iOS and one Android programming language to complete the application. The advantages of developing native applications is that the app will only be focused on one device, so changes will not disturb another platform. It will also be smooth in terms of usage and UI, as the technology is focused specifically for that platform.

Hybrid technology

Hybrid technology means that there will be one code base, and that technology will be able to convert the application to run on both Android and iOS platforms. This doesn’t necessarily mean that 100% of the code will be written only once. In most types of application, hybrid technology works fine and one could keep 90-95% of the code the same for both platforms. It just requires maintaining configuration, build and release-related code separately in most cases, provided an application isn’t required to use native functionality such as camera, sensors, hardware etc. More native functionality means less common code, but there are modules available which help in this regard. One disadvantage for choosing hybrid technology is that changes affect both platforms, so one has to be careful and test properly. Another disadvantage is that the more external modules are used, the less visibility there is about what is happening inside, and some minor update could just break everything on both platforms.

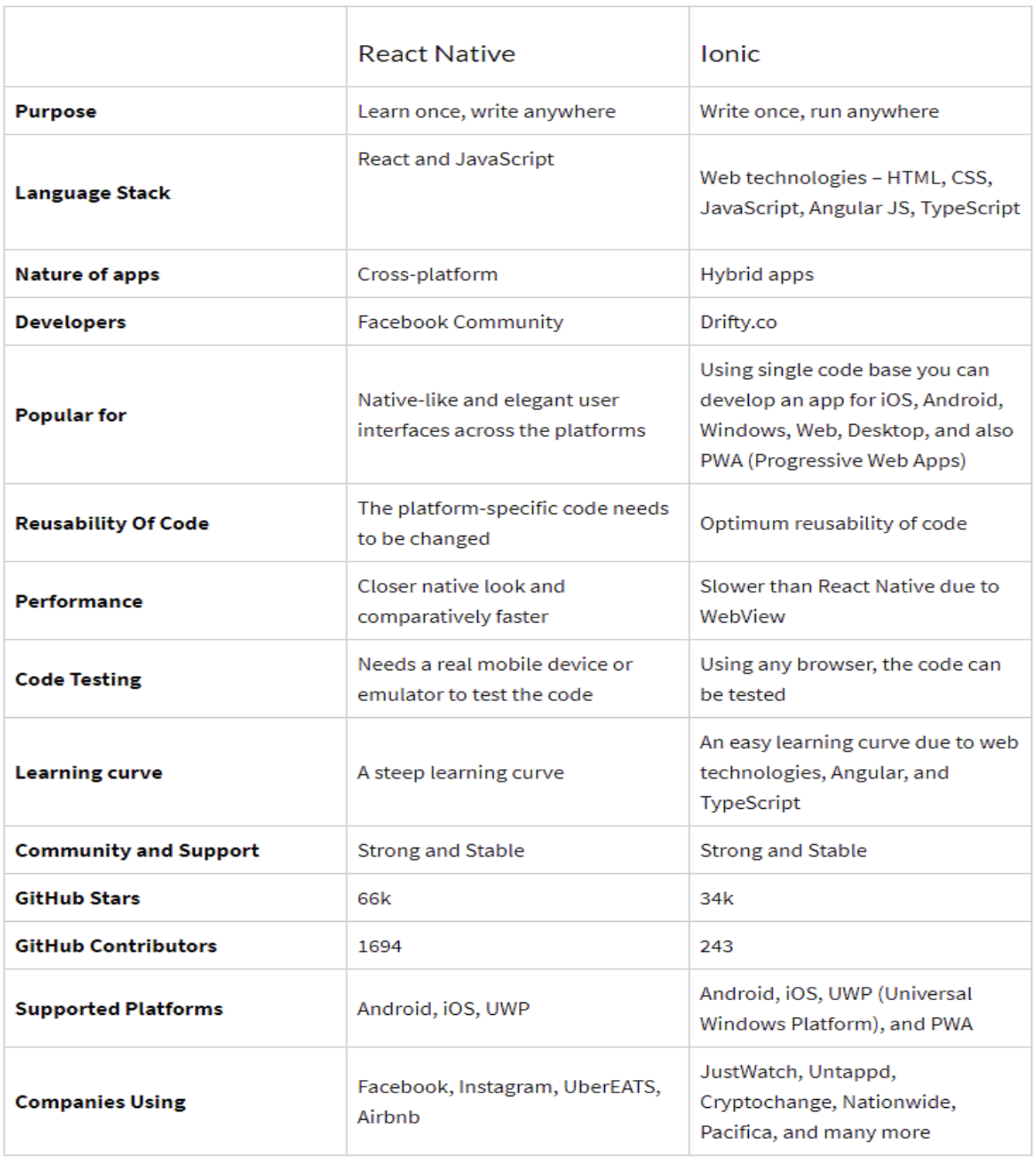
There are many options available currently in terms of hybrid application: Phonegap, Ionic, React Native, Flutter, Zamrin etc. The most commonly used of these are Ionic and React Native.

React Native:

Javascript technology can be used to write the code and in the end it converts the code into the native platform. If React is a familiar web application programming framework, then it could easily be used to develop the mobile application by using more or less the same sort of pattern. The result will be as smooth as the native application, if developed properly. In cases where there is previous experience with React, this should be the first choice.

Ionic:

Ionic is a Javascript technology which can be used to develop the application for both the platforms, but it runs the application inside the browser embedded in the native application. It has been observed that when using Javascript technology, some small issue is capable of damaging the whole application. The feeling of the application is also more like using a web application customised for mobile UI embedded in native browsers. Angular or Vue or React could also be used to write the code by using Ionic.



	React Native	Ionic
Purpose	Learn once, write anywhere	Write once, run anywhere
Language Stack	React and JavaScript	Web technologies – HTML, CSS, JavaScript, Angular JS, TypeScript
Nature of apps	Cross-platform	Hybrid apps
Developers	Facebook Community	Drifty.co
Popular for	Native-like and elegant user interfaces across the platforms	Using single code base you can develop an app for iOS, Android, Windows, Web, Desktop, and also PWA (Progressive Web Apps)
Reusability Of Code	The platform-specific code needs to be changed	Optimum reusability of code
Performance	Closer native look and comparatively faster	Slower than React Native due to WebView
Code Testing	Needs a real mobile device or emulator to test the code	Using any browser, the code can be tested
Learning curve	A steep learning curve	An easy learning curve due to web technologies, Angular, and TypeScript
Community and Support	Strong and Stable	Strong and Stable
GitHub Stars	66k	34k
GitHub Contributors	1694	243
Supported Platforms	Android, iOS, UWP	Android, iOS, UWP (Universal Windows Platform), and PWA
Companies Using	Facebook, Instagram, UberEATS, Airbnb	JustWatch, Untappd, Cryptochange, Nationwide, Pacifica, and many more

Figure 5. Comparison of React Native vs Ionic [15]

For Cleancentive, it was decided to use React Native because React Native generates native cross-platform apps and also has better performance compared to Ionic, as well as better community support. On top of that, the author has previously worked on React and the web application of Cleancentive will be deployed using ReactJs. Also, the client team suggested that React Native was a more suitable fit for Cleancentive, as the application doesn’t require any such advanced level hardware interaction features and the client wanted the application to give a native feeling. Since there is no co-author of this thesis, it was decided that it doesn’t make sense to develop a native application and work twice on the same stuff. Maybe there are other better alternatives, but as per current understanding it was decided to choose React Native to develop the Cleancentive application, a line of approach also supported by Codistan.

3.2. Usability, Testing and Releasing

User experience and usability are two of the most important elements of any design: they should both be considered thoroughly when designing mobile applications. Usability is important throughout the product development cycle. It needs to be discussed thoroughly when the project starts, and during any feature planning. Why is usability important? Because it helps meet user needs and also helps save the cost of development. How does it save the cost of development? It helps us avoid implementing something which is not user-friendly and which probably needs to be changed. This also brings quality to the product: if the product is developed considering the user's needs, this is clearly a big plus. Usability describes the extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency and satisfaction in a specified context of use [16].

Involving people who are going to use the product during the mind mapping and design process, and considering their opinion, is the optimal way to find the best design. It is always very time-consuming and expensive to involve people throughout the design process, so the best one can do is to try to have one meeting to check the wireframes, or then try to give users the product to test before launching, and consider their feedback.

Usability testing hooks into top level terms such as 25 Information Architecture (IA) and Iterative Design. The first is the process of organizing information including the structure, design, layout and navigation in a way that is easy for people to find, understand and manage. The second is a design methodology involving repeated cycles of design, evaluation, and analysis. Refinements are made for the next cycle based on current analysis and feedback.

In Cleancentive, several meetings were conducted with the client and employees to finalise the usability of the product. It included the onboarding/registration process, the compensation model, tracking user activity, gathering user locations, the payment process and several other items to make sure usability and the UX of the product were not compromised. These processes have been explained thoroughly in chapter 4.2-4.4

Testing the Cleancentive application can be done in a similar way to any other application. Features such as onboarding, maintaining user profiles etc. can be tested by adding unit tests along with automated tests, but only manual testing was considered for the purposes of this thesis because the most important functionality of the application is tracking user activity, which anyway requires manual testing. Employees have promised to use and test the application throughout the process by bringing the application into their daily routine so it can produce more and more data, and it will be easy to test further in the future. Real-time testing of the application will be covered in more detail in upcoming chapters.

Releasing the Cleancentive application is straightforward, as it has already been determined from the thesis that it will not be published to Google Play or the Apple App store. Although applications will be built using hybrid technology, testing and releasing on iOS was also disregarded because of the complexity of the application and the platform itself. The Android bundle will be generated and provided to the employees who can install the application on their mobile, and that is how the release of the app will be handled.

4. Results and Analysis

In the start of this thesis process, the main objective was to build a proper functional MVP for the Cleancentive application. It was realised during the process that making an actual MVP would be more suitable as a doctoral thesis because it would require definitions and design work (requirements, technical solutions, user experience and interface etc.) and the development itself would have taken many months just for the mobile application, while the web application would also require several months, following which would come deployment and the release of the application to app stores. Considering all this, it was decided to disregard many of those things during this thesis and focus only on the development of the simple MVP, which tracks user activity and presents users with the benefits they have earned.

This MVP includes designing the Cleancentive application which includes: collecting requirements, assessing different technical platforms, designing the Mindmap, selecting the features of the application and wireframing the user interface of the application. As far as development is concerned, it includes: designing simple mobile applications where users can login and register, tracking user activity and presenting the incentive. For deployment, only Android and APK will be used to install directly within the mobile application instead of from an actual app store. This approach will be enough to test the application and its model. Everything else has been put aside for future development.

This chapter explains the finalization of the target audience, the design of the Mind Map, the design of the wireframe/UI, development of the application, usability and testing of the application.

4.1. Understanding Requirements

After several meetings with the clients, the requirements of the application were identified. The client expected us to create a mobile application which employees can install on their phone which tracks their location and activity.

Based on their activity, it will give them health and environmental benefits. The target audience for Cleancentive was always employers and employees.

One of the important challenges in the development for Cleancentive was to decide the scope of the application and what to include and what to skip. This section covers the task of defining the application's requirements.

To define the requirement properly, one needs to understand the whole concept behind the application. Looking at the Mind Map can help us with that.

The diagram below shows the Mind Map of the application, covering almost everything required in this mobile and web application. This thesis only looks at the home screen, activity recording, the display of different activities, the active trip and incentive calculation.

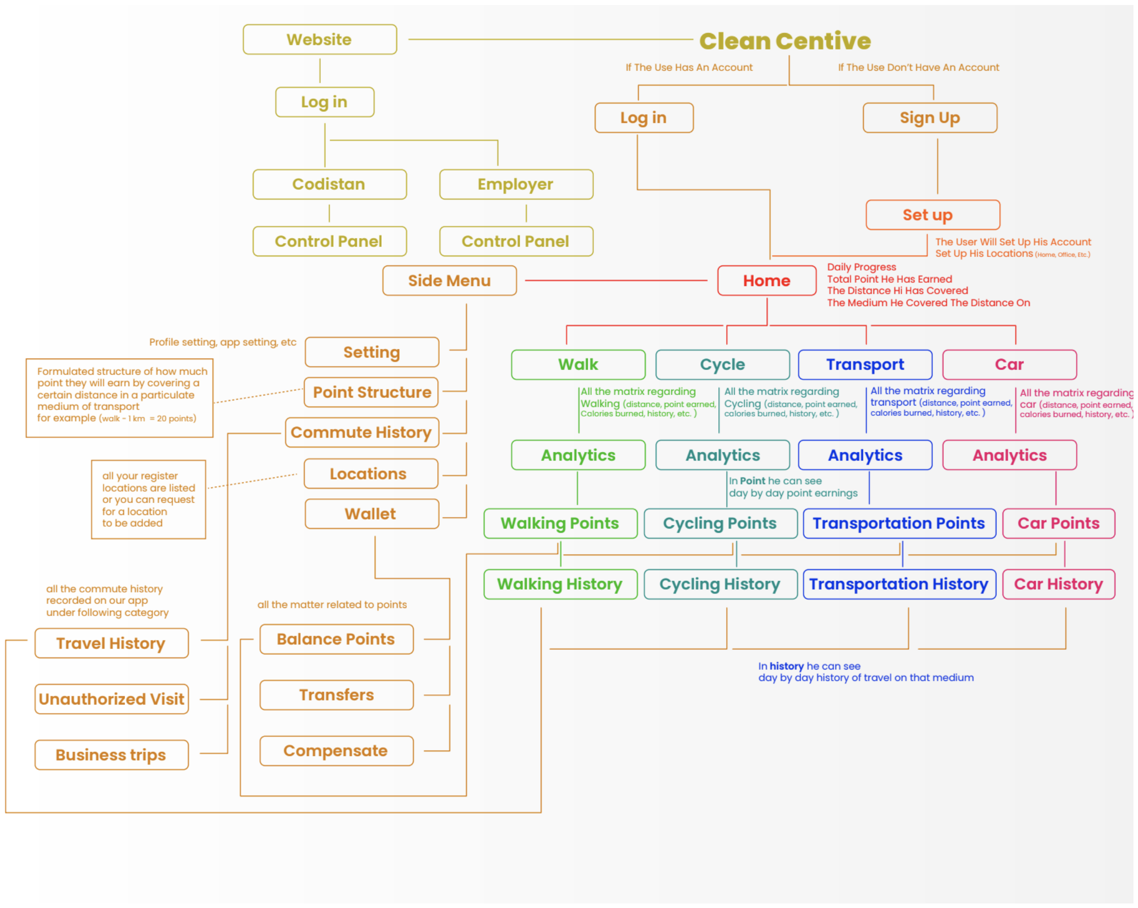

Website
Clean Centive
If The Use Has An Account
If The Use Don't Have An Account
Log in
Log in
Sign Up
Codistan
Employer
Control Panel
Control Panel
Set up
The User Will Set Up His Account
Set Up His Locations (Home, Office, Etc.)
Side Menu
Home
Daily Progress
Total Point He Has Earned
The Distance Hi Has Covered
The Medium He Covered The Distance On
Profile setting, app setting, etc
Setting
Formulated structure of how much point they will earn by covering a certain distance in a particulate medium of transport for example (walk - 1 km = 20 points)
Point Structure
Commute History
all your register locations are listed or you can request for a location to be added
Locations
Wallet
Walk
Cycle
Transport
Car
All the matrix regarding Walking (distance, point earned, Calories burned, history, etc.)
All the matrix regarding Cycling (distance, point earned, calories burned, history, etc.)
All the matrix regarding transport (distance, point earned, calories burned, history, etc.)
All the matrix regarding car (distance, point earned, calories burned, history, etc.)
Analytics
Analytics
Analytics
Analytics
In Point he can see day by day point earnings
Walking Points
Cycling Points
Transportation Points
Car Points
Walking History
Cycling History
Transportation History
Car History
all the commute history recorded on our app under following category
all the matter related to points
Travel History
Unauthorized Visit
Business trips
Balance Points
Transfers
Compensate
In history he can see day by day history of travel on that medium

Figure 6. Mind Map of the Application

The above Mind Map defines the possible options and features in the web and mobile application. This thesis only covers the mobile application, which is explained on the right section of the diagram.

When users start using the mobile application, they will have the option to register to the service. In case they are already registered they will use the login option. Users will go to the home page, where they will see their current environment and health benefits. They will also be able to see how many kilometers they have covered in each activity along with mobility records, meaning all their activities and their benefits.

4.2. Compensation Model

The idea behind Cleancentive was to develop an application which will track user activity and compensate them. How the user gets compensated needs to be decided. By having several meetings with clients and employees, it was agreed that there will be two types of compensation or benefits. One will be called health benefit, and the other will be called environmental benefit.

4.2.1. Environmental Benefit

This will be awarded to users for not using the car or not producing Co2. This benefit will be calculated at the start of every month, and will be paid to the user by the end of the month if the user hasn’t used a car for their trips to the office. The application will show the possible environmental benefit to the user as soon as they open the application, and on the basis of the user’s car activity this possible incentive will be deducted. Benefit will be calculated based on the user’s distance from home to work, how many trips are allowed to the user (in normal cases, 2) and the number of working days in the month. This factor then will be a multiplied base unit of per kilometre compensation which is currently set at 0.11 cents, based on client suggestions.

environment_incentive = distance_from_home_to_work x no_of_times_trip_allowed x no_of_working_days_in_month * 0.11

So if the user's work to home distance is 10 km, and they are allowed to have 2 trips everyday, and there are 22 working days in a month then the calculation will look like this:

environment_incentive = 10 * 2 * 22 * 0.11 = 48.5 Eur

This would mean that if the user uses public transport, they will be able to cover the cost of the seasonal public transport ticket. If the user uses the car more often than permitted, then a sum will be deducted from the environmental incentive. Let say someone uses the car on 2 days for two trips: then an amount of 4.4 euros will be deducted from environmental incentive.

lost benefit = 2 * 2 * 10 * 0.11 = 4.4 Eur

Where the first 2 is the number of days the individual used a car, the second 2 is the number of trips the user has made, and 10 is the distance from home to work.

4.2.2. <u>Health Benefit</u>

Health benefits are given when the user travels to work by using a bike or walking or running. The benefit amounts have been set at 0.48 per km for walking or running and 0.30 per km for Bicycles. These values were suggested by the customer.

Based on this incentive model, if users don't use cars and at the same time use bikes to go to work, they will get more money, because they are not producing any Co2 so they automatically get environmental benefits and at the same time they are getting health benefits for keeping themselves healthy.

4.3. Design of the application

The design of the application was carried out by having a meeting with the client. Although full application design was created during this process, this thesis only explains the few screens required for the implementation.

The first wireframe of the application was designed using paper and pencil which at a later stage converted to Adobe XD wireframes. The client wanted it so that on the first day of every month the application would calculate the possible benefit users could get, based on the calculation defined in the 4.1 section. It would display the active trip on the home screen along with the total mobility incentive. Depending on how many times the user uses the car, the application will update the benefits and show only the remaining benefits. The application was also required to display the active trip on the map and the total distance covered in each activity, along with the total time spent.

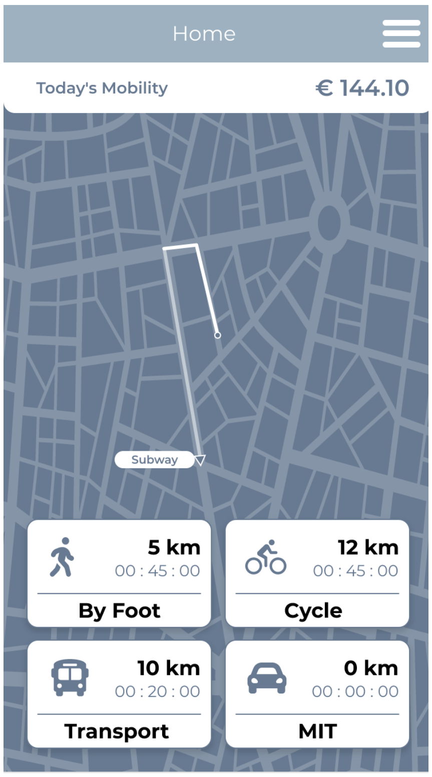


Figure 7. Wireframe of the application

Figure 7, shown above, presents the wireframe of the application's home page. It includes the benefits (shown in the header) and the total number of kilometers the user has covered in each category.
After designing the wireframe the colour of the application was finalised, along with some other minor changes. The current activity boxes consume a lot of space on the map, so it was decided to change them to a tab bar format, such that clicking on each item will highlight the activity part of that trip. After these changes and filling up the colours in the wireframe application, the home page now appears as shown below in figure 8:

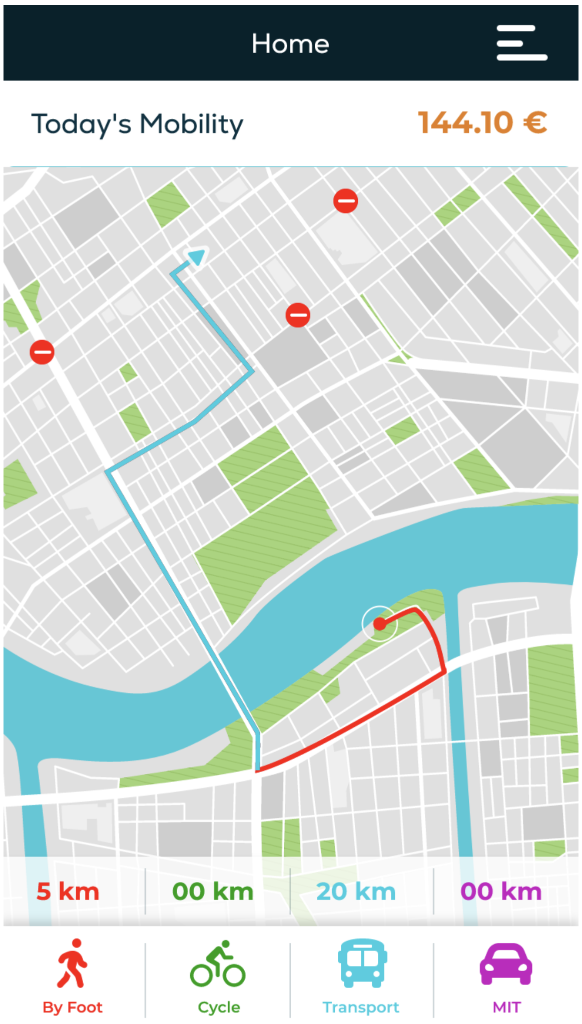


Figure 8. Design of the application

4.4. <u>Development of the application</u>

As with any other web and mobile application, the development process required knowledge of programming languages, methods and environment. This section covers everything that was needed to develop the Cleancentive application.

4.4.1. Development environment setup

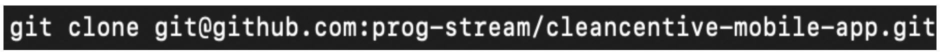
Setting up the development environment is one of the basic steps when starting any project. It requires creating code repositories on some cloud services, the installation of language and necessary modules and setting up IDE. This section covers everything which is required to set up the development environment.

Version Control System (VCS) setup

VCS is used to maintain changes in the code and keep track of the differences between them. It creates a copy of the code on the cloud every time new changes are published. This is a very handy tool for keeping everything in one place, and also if something needs to be changed back in the future it would be easy to accomplish.

In this thesis Git, which is a very popular VCS, has been used to code and maintain the history on Github cloud. The client has created the repository and provided the read and write access to the author so that code can be maintained on the client's own supervised environment. Setting up Github is very easy: it requires installation of the Git binary installer which was downloaded and installed with few clicks. After the repository was created by the client and named `**Cleancentive-mobile-app**`, everything was set up by running a few commands.

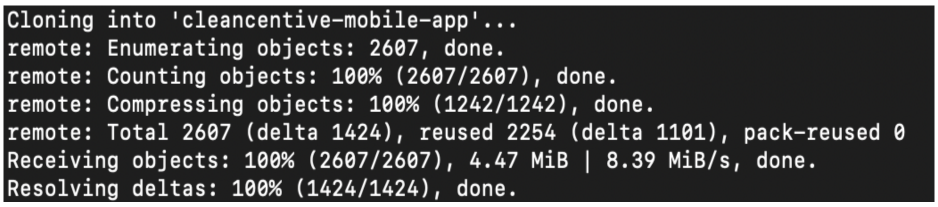
Listing 1 shows the commands which were used to clone the repository on the local machine.



```
git clone git@github.com:prog-stream/cleancentive-mobile-app.git
```

Listing 1. Git clone command

After running the above command, code was copied to the local machine as shown in Listing 2.



```
Cloning into 'cleancentive-mobile-app'...
remote: Enumerating objects: 2607, done.
remote: Counting objects: 100% (2607/2607), done.
remote: Compressing objects: 100% (1242/1242), done.
remote: Total 2607 (delta 1424), reused 2254 (delta 1101), pack-reused 0
Receiving objects: 100% (2607/2607), 4.47 MiB | 8.39 MiB/s, done.
Resolving deltas: 100% (1424/1424), done.
```

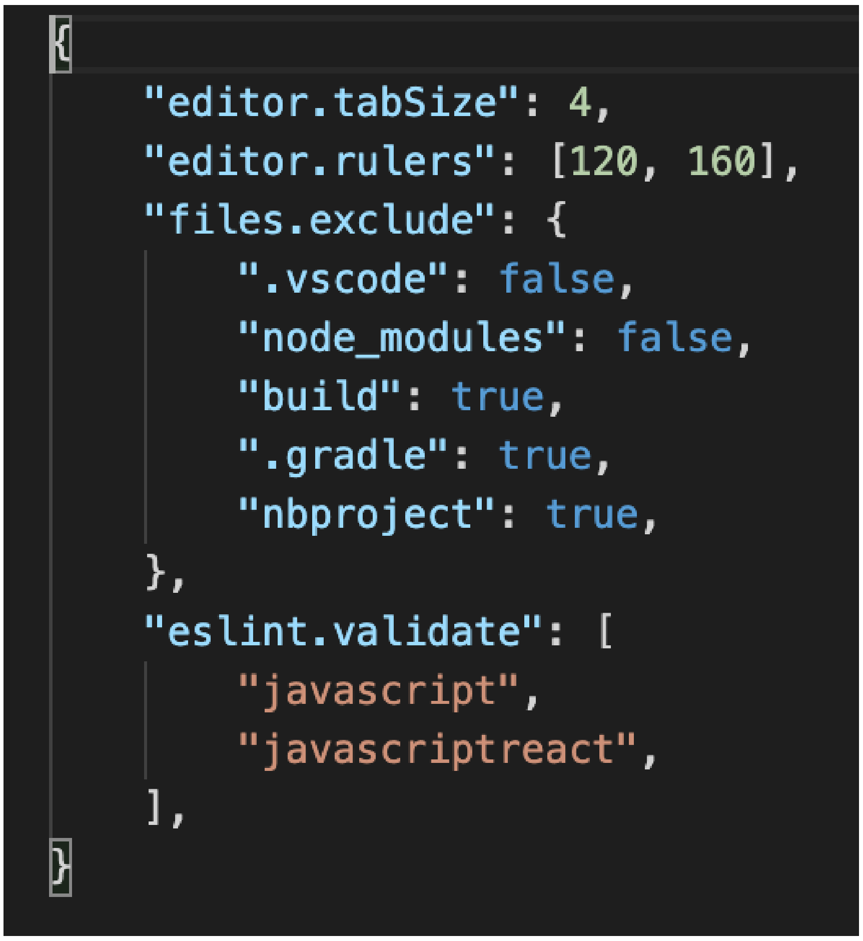
Listing 2. Cloning the repo

<u>**Setting up IDE**</u>

An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools and a debugger. Some IDEs also contain the necessary compiler, interpreter, or both [17].

For the development of Cleancentive, it was decided to use Visual Studio code because it is a common tool used by the client’s technical team. The code repository was opened in Visual Studio code and then the basic parameters required for this project were initiated.

The following settings were used to maintain code spacing and general consistency, as shown in Listing 3.



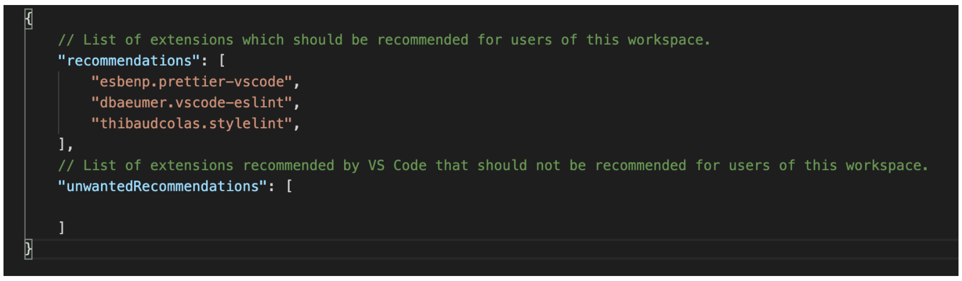
```
{
    "editor.tabSize": 4,
    "editor.rulers": [120, 160],
    "files.exclude": {
        ".vscode": false,
        "node_modules": false,
        "build": true,
        ".gradle": true,
        "nbproject": true,
    },
    "eslint.validate": [
        "javascript",
        "javascriptreact",
    ],
}
```

Listing 3. Prettier setting

Prettier Setup

Prettier is an opinionated code formatter. It enforces a consistent style by parsing the code and reprinting it with its own rules that take the maximum line length into account, wrapping code when necessary [18].

The following plugins have been installed to ensure that the format of the code remains constant throughout the thesis.



```
{
    // List of extensions which should be recommended for users of this workspace.
    "recommendations": [
        "esbenp.prettier-vscode",
        "dbaeumer.vscode-eslint",
        "thibaudcolas.stylelint",
    ],
    // List of extensions recommended by VS Code that should not be recommended for users of this workspace.
    "unwantedRecommendations": [

    ]
}
```

Listing 4. Visual Studio code extensions settings

React Native Installation

This section covers the installation process for the Cleancentive mobile application. There are two ways React Native can be installed.

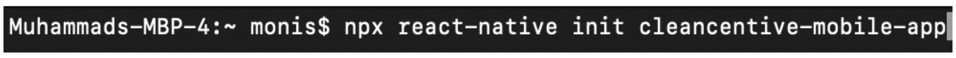
Expo CLI

The easiest way to install React Native is to use Expo CLI. It comes with a variety of tools built around React Native, which helps one to get started with the application quickly. It only requires NodeJs and an emulator/phone to get started.

React Native CLI

React Native CLI is one option for setting up mobile application development, but it is a little more complex compared with Expo because it requires XCode in case of developing iOS apps, or Android Studio when developing an Android app. It also requires the user to have CocoaPods.

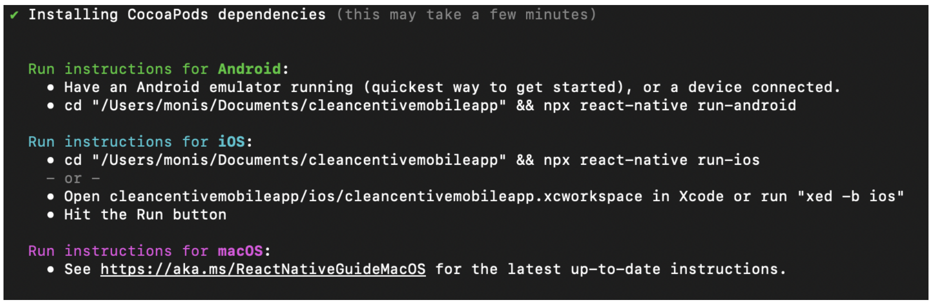
The author was already familiar with developing mobile applications so it was easy to get started with React Native CLI. It was decided to use the name "Cleancentive-mobile-app" so by running the following command things were arranged quite quickly.



```
Muhammads-MBP-4:~ monis$ npx react-native init cleancentive-mobile-app
```

Listing 5. Initialise React Native

After running the command, it begins to download the template and install Cocoapods dependencies. Once the installation is complete, it will present the following screen as shown in Listing 6, and can start running the application.



```
✔ Installing CocoaPods dependencies (this may take a few minutes)

  Run instructions for Android:
    • Have an Android emulator running (quickest way to get started), or a device connected.
    • cd "/Users/monis/Documents/cleancentivemobileapp" && npx react-native run-android

  Run instructions for iOS:
    • cd "/Users/monis/Documents/cleancentivemobileapp" && npx react-native run-ios
    - or -
    • Open cleancentivemobileapp/ios/cleancentivemobileapp.xcworkspace in Xcode or run "xed -b ios"
    • Hit the Run button

  Run instructions for macOS:
    • See https://aka.ms/ReactNativeGuideMacOS for the latest up-to-date instructions.
```

Listing 6. React Native installation

Now that React Native is installed the following commands, presented in Listing 7, can be used to start the application which will start Metro Builder.

```
npx react-native start
```

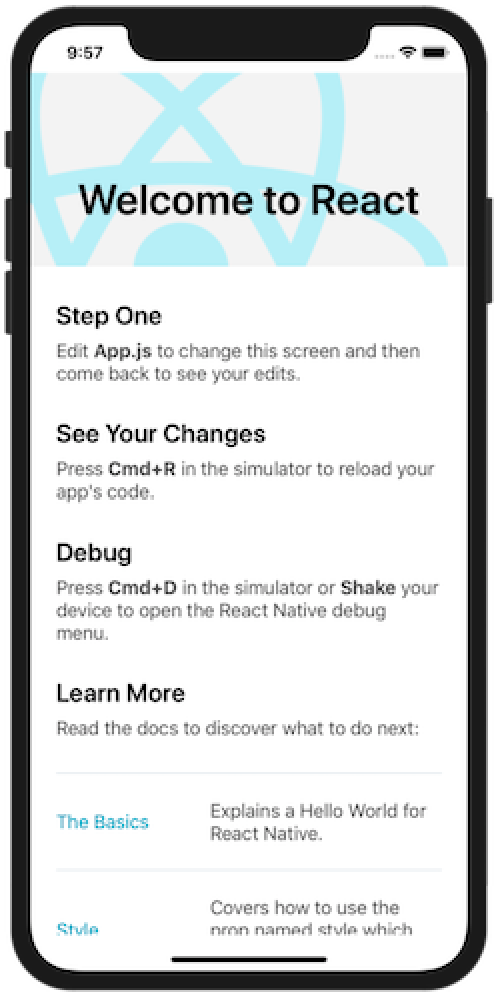
Listing 7. Start React Native application

Listing 8 displays the command to run the application on iOS. It should be run in the new terminal.

```
npx react-native run-ios
```

Listing 8. Running react native application

Once the application is started, it will present the following screen as shown in Figure 9.


9:57

Welcome to React

Step One

Edit **App.js** to change this screen and then come back to see your edits.

See Your Changes

Press **Cmd+R** in the simulator to reload your app's code.

Debug

Press **Cmd+D** in the simulator or **Shake** your device to open the React Native debug menu.

Learn More

Read the docs to discover what to do next:

The Basics	Explains a Hello World for React Native.
Style	Covers how to use the prop named style which

Figure 9. Default view of React Native application

Installation of necessary modules

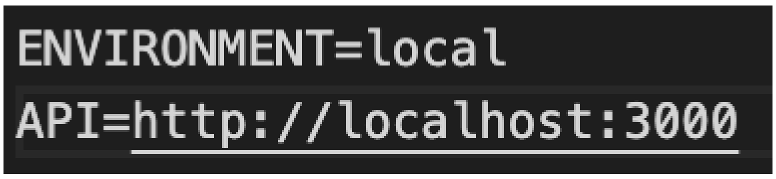
This section covers the extra plugins or modules required to build up the application. The Cleancentive application needs the support of different languages. i18next module is installed to manage multiple languages in the application. It allows language translation to be stored in a JSON file and based on the selected language to fetch the translated text.

It also requires state management tools to be able to maintain the state of the user. For that purpose, Redux and Redux Persist were installed so that the state can be managed in one place and stored somewhere in the file system. Redux-persist, as the name suggests, persists the state into application storage.

Along with the above module, Axio is required for making an HTTP connection with the API, and for React-navigation to have different types of navigation such as drawer, tab and stack navigation in the application.

Build management

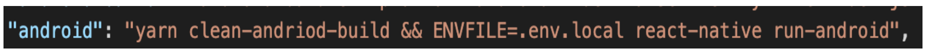
For build management, React Native already provides support where applications just need to manage different environment files containing different configuration values. At the time of building the application, these files can be provided and the app loads that particular file for the release. Parameters such as API URL, Google Analytics ID etc. can be stored in that file. The file has an extension of .env. Here is how the Cleancentive local environment file looks, as shown in Listing 9 below:



```
ENVIRONMENT=local
API=http://localhost:3000
```

Listing 9. Setting application environment

The command to create the build for Android and iOS on react native includes loads of parameters and is difficult to remember, so the command which is displayed in Listing 10 was added to package.json to create a release with local configuration. Users just need to run the **yarn android** command to create the build.

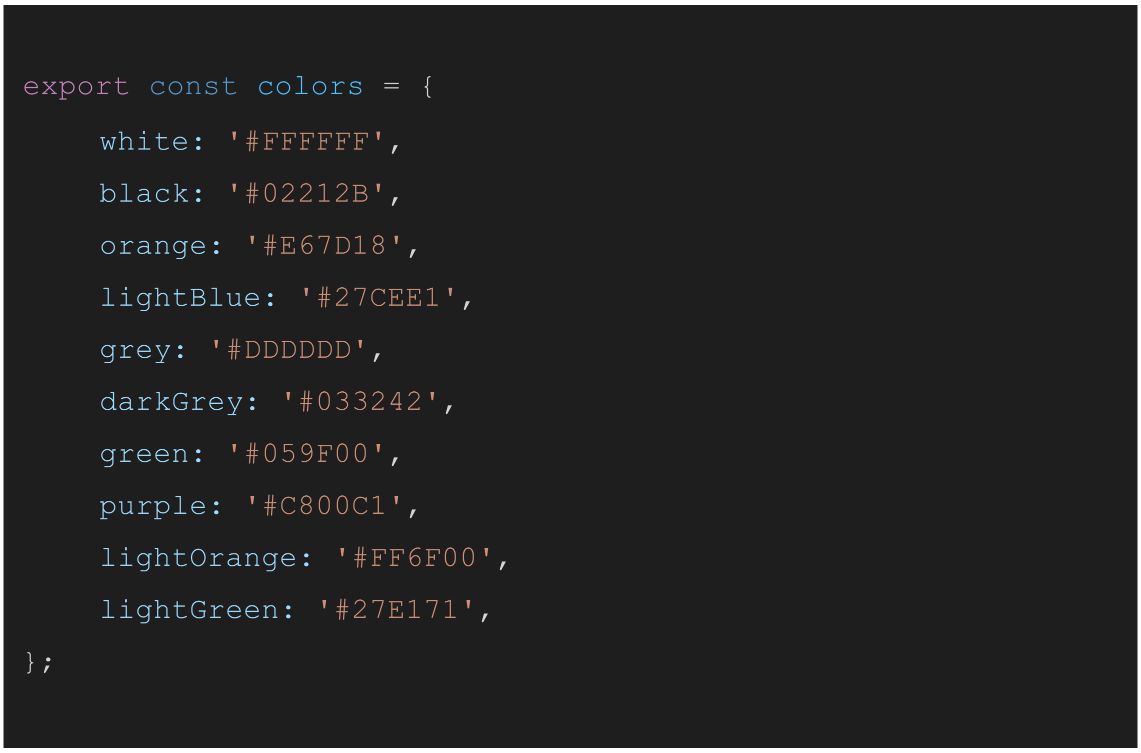


```
"android": "yarn clean-andriod-build && ENVFILE=.env.local react-native run-android",
```

Listing 10. Running application on Android

4.4.2. <u>Integrating Design</u>

Integration designing was a little challenging as initial thinking had to focus on how the different screens and navigation will appear. For this purpose, the colours required by the theme were declared in a separate file which is displayed in Listing 11. This also defines the font sizes which will be used in the app. To keep spacing consistent in the app, it also defines spacing between different sections on the screen by a multiple of 4 as shown in Listing 12.



```
export const colors = {
    white: '#FFFFFF',
    black: '#02212B',
    orange: '#E67D18',
    lightBlue: '#27CEE1',
    grey: '#DDDDDD',
    darkGrey: '#033242',
    green: '#059F00',
    purple: '#C800C1',
    lightOrange: '#FF6F00',
    lightGreen: '#27E171',
};
```

Listing 11. Application theme

```
export const fontSizes = {
    xs: 10,
    small: 12,
    medium: 14,
    large: 16,
    extraLarge: 18,
    xxl: 20,
    xxxl: 24,
    xxxxl: 28,
};
```

Listing 12. Define application font sizes

The view was created with basic styles and padding around the edges to keep the screen padding consistent. A separate component for header was created to match the design which includes the title of the screen, the back button and burger menu. In order to display icons, React Native vector icons have been used which provide the ability to add different icons to the application.

There are three types of navigators available in React Native. These are defined below.

Tab Navigator

The tab navigator is the most common navigation style used in mobile applications. It means the user can press the tab and it switches the screen. It is used in many mobile applications, usually on the bottom of the screen or sometimes on the top of the screen, even in the header.

It is used in the Cleancentive application on the bottom of the screen to bring up the total number of kilometers the user has traveled in each activity type. It is also used to display activities in each and every trip and to highlight the activity which was selected. For example, a user has covered 15 km in total, of which 3 km is by bicycle, 2 km by walking, 7 km by public transport and 8 km by car. In this case, the application will display all these activities on the map with different colors for the tabs and the user can get a highlighted route by clicking on the activity tab. Figure 10 below shows the design of the trip when the user has travelled 5 km by walking and 20 km by public transport.

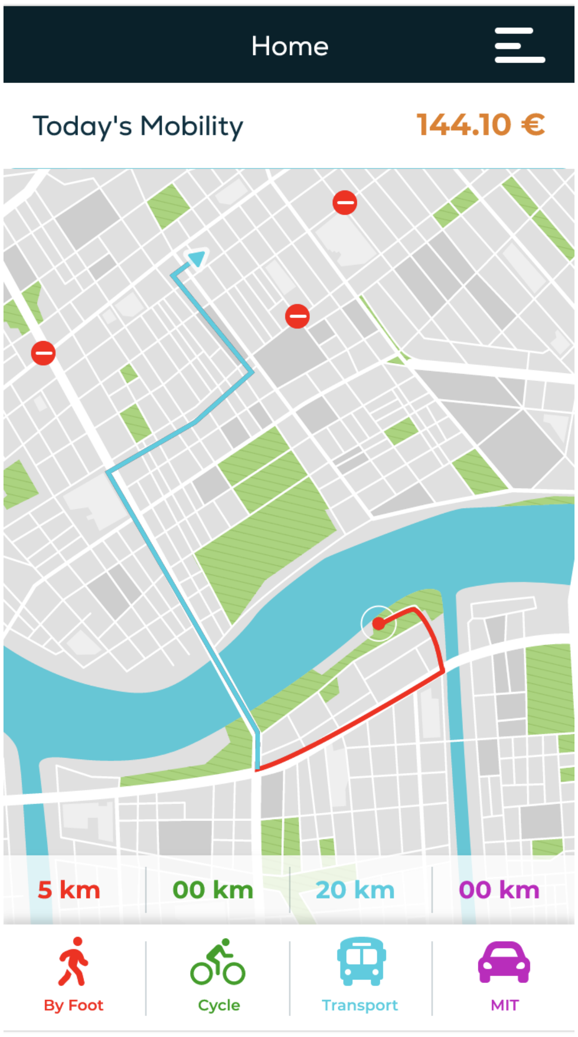


Figure 10. Home screen of the application

The following command has been used to install tab navigator in the application, as shown in Listing 13:

```
npm install @react-navigation/bottom-tabs
```

Listing 13. Tab bar installation

After installing the tab navigation, a reusable component is created in the app with the following properties as displayed in listing 14:

```
export interface ITabScreenProps {
  icon?: IconType;
  component: React.ReactNode;
  name: ScreenType;
  translation: string;
  color: string;
}

interface ITabNavigatorProps {
 tabList: ITabScreenProps[];
 tabBar: (props: BottomTabBarProps<BottomTabBarOptions>) =>
React.ReactNode;
}
```

Listing 14. Tab bar properties

This tab component will take an icon shown in the tab, a translation of the text shown, colour, name and component of the screen. Tab navigation will take a list of tabs to be shown in the tab bar and return the navigator.

Stack Navigator

This provides a way for the app to transition between screens where every new screen is placed on top of a stack. By default, the stack navigator is configured to have the familiar iOS and Android look & feel: new screens slide in from the right on iOS, and fade in from the bottom on Android. On iOS the stack navigator can also be configured to a modal style where screens slide in from the bottom [19].

In Cleancentive, the purpose of the stack navigator is the same as usual, i.e. to stack the screens on top of one another. This means that when a user clicks on the mobility record, it will add and display mobility record details on top of that screen. By using the back button, the user will return to the mobility record screen once again.
The following command, shown in Listing 15, has been used to install the stack navigator in the application:

```
 npm install @react-navigation/stack
```

Listing 15. Installation of stack navigation

All the related sections in the application are stacked to form different stack navigators which will be used in the drawer navigator presented in the next section. As per design, the following stack navigator categories have been identified:

- Home
- Compensation Model
- Location
- Mobility Records
- Wallet
- Projects
- Setting
- Safety

Drawer navigator

Nowadays, mobile applications use a navigation pattern shown by swiping from left or right. It will appear on the top of the screen where users can choose the options. In Cleancentive, swiping from right to left displays the navigation drawer as shown in design section 4.6. The command displayed in Listing 16 was used to install the drawer navigation which is shown below.

```
npm install @react-navigation/drawer
```

Listing 16. Installation of drawer navigation

After installation, a component is created in a way that drawer navigation items can be added in future easily, and the navigator can be reused for further development in the future. As per design section 4.6, it will have a header section showing information of the login user, menu items and footer links for terms, privacy and data policy screens.

A drawer navigator component shown in Figure 11 was created, which will accept a list of different stack navigators containing name, icon and screen. It is divided into 4 parts: Header, ProfileMenu, MenuItems and Footer, where Header components contains the heading and close button, ProfileMenu contains the login user information, DrawerContentItem contains the navigation menu and Footer contains the links of terms, privacy and data policy screens. This whole process includes pure coding, so it was not included in this section.

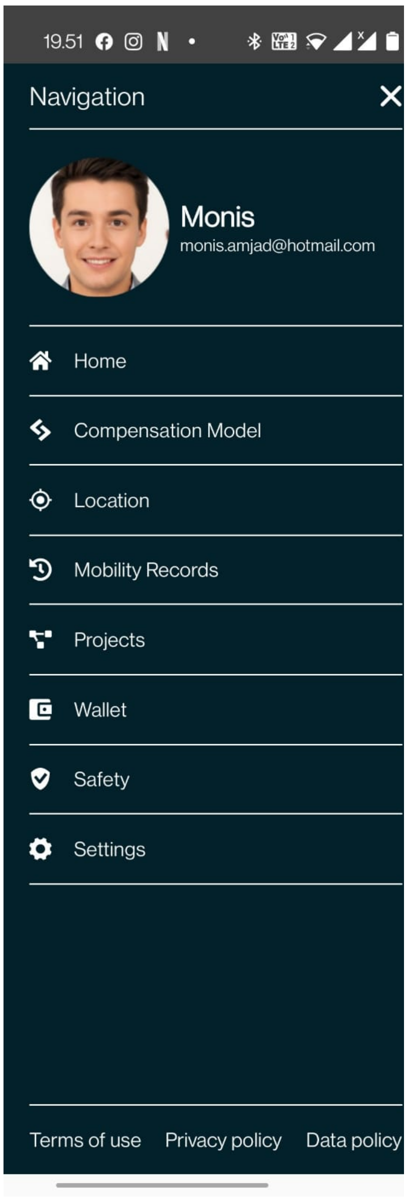


Figure 11. Drawer navigation in the application

4.4.3. User Authentication

Originally the plan was to implement a proper authentication system where users were able to register and login to the application using their desired credentials. As this is not the core functionality for developing the application, it was decided that user options will be presented in the application by choices which bind the data with the user profile.

This option will be presented to the user as soon as the application is launched and upon selection it will save the selection for later use. When a user trip or activity is started, it will automatically fetch the selected user and connect the trip data with the user. This will also present the option to the user to select the desired language as displayed in Figure 12 below.

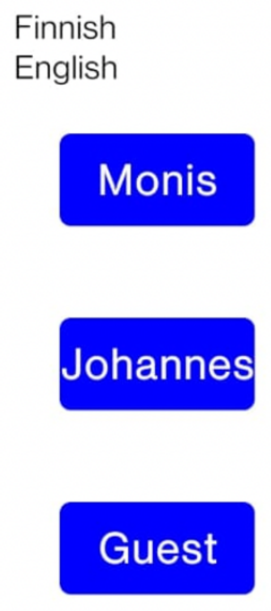


Figure 12. User selection screen

4.4.4. Application Permission

As with any other mobile application, Cleancentive requires the internet to work properly: along with that, the application also requires user storage, activity recognition and location permission.

React Native provides a PermissionsAndroid model to access the permission in Android. Normal permissions can be granted by default at the time of installation of the application if they are added in AndroidManifest.xml. However, permission which requires access to sensitive information of the user requires a dialog prompt. Cleancentive uses the PermissionsAndroid module for asking permissions. Permission in iOS works differently than Android. Usually it requires permission at the time of using such sensitive information which can be requested with a separate popup, but they are not covered in this thesis because of their complexity.

If a user has previously turned off a permission that you prompt for, the OS will advise your app to show a rationale for needing the permission. The optional rationale argument will show a dialog prompt only if necessary - otherwise the normal permission prompt will appear [20].

The following permissions listed in Listing 17 were added in **AndroidManifest.xml** considering the requirement of the application:

```
<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission
android:name="android.permission.FOREGROUND_SERVICE"/>

<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>

<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

```
<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION"
/>

<uses-permission android:name="android.permission.WAKE_LOCK"/>

<uses-permission
android:name="android.permission.ACTIVITY_RECOGNITION" />
```

Listing 17. Application permission setting

- INTERNET permission is required for the application to function properly.
- FOREGROUND_SERVICE service permission is needed to run a process to track user activity.
- BACKGROUND_LOCATION, COARSE_LOCATION and FINE_LOCATION is used to fetch user location at the time the activity is performed.
- ACTIVITY_RECOGNITION is to determine whether a user is stationary or moving.
- WAKE_LOCK permission indicates that the application requires the device to be kept on.

Now it is clear what kind of permissions are needed. Listing 18 presents the code which was used to request the permission.

```
const {request, PERMISSIONS}  = PermissionsAndroid;
const permissionResult = await request(PERMISSIONS.ACCESS_FINE_LOCATION, {
      title: t('PERMISSIONS.LOCATION.TITLE'),
      message: t('PERMISSIONS.LOCATION.MESSAGE'),
      buttonPositive: t('PERMISSIONS.LOCATION.BUTTON.ALLOW'),
      buttonNegative: t('PERMISSIONS.LOCATION.BUTTON.CANCEL'),
      buttonNeutral: t('PERMISSIONS.LOCATION.BUTTON.IGNORE'),
});
```

Listing 18. Requisition Application permission

Once this code runs it will request location permission. Similarly, all other permissions can be requested. It will display a location permission popup and it can be checked by using the code mentioned in Listing 19, whether the user has granted the permission or not.

```
if (permissionResult === PermissionsAndroid.RESULTS.GRANTED) {
     return true;
}
```

Listing 19. Code to grant permission

4.4.5. Tracking User Location

This chapter covers everything related to the tracking of user location. Now that the application has already covered the part to request user location, it needs to integrate that part to a process which will record the location continuously.

It seems React Native provides a way to run background tasks using Headless JS. Headless JS is a way to run tasks in JavaScript while your app is in the background. It can be used, for example, to sync fresh data, handle push notifications, or play music [21]. Integrating Headless JS was not so difficult, as all that was needed was to copy paste the following code:

```
import { AppRegistry } from 'react-native';

AppRegistry.registerHeadlessTask('SomeTaskName', () =>
  require('SomeTaskName')
);

module.exports = async (taskData) => {
 // do stuff
};
```

Listing 20. Background task listener

List 20 explains the code required to run the different tasks in the background. After looking into Headless JS, it was soon realised that it doesn’t really work for iOS. It was decided to start looking for other options on both Android and iOS considering that Cleancentive will work on iOS and Android.

After this Headless JS situation, many React Native plugins and their solutions have been tried, such as **react-native-queue**, **react-native-background-job** and **react-native-background-task** but it was very difficult to integrate them into the application. After so many hits and misses, it was discovered that **react-native-background-timer** seems to work fine on both Android and iOS. It was integrated with the separate custom native module which was created to show the notification bar when the background tasks are running.

Native Module was only created for Android and later on it could also be done for iOS. NativeModule exposes instances of native classes to JavaScript through JavaScript objects which will allow applications to execute native code. This was the same reason React Native was selected initially. If a native API is not exported from React Native which was required by the application then there should be a possibility to write a custom module!

There are two ways to create a custom native module in React Native. The first method is to directly create an iOS/Android project in the code and the second method is by creating a NPM package that can be installed as a dependency by other React Native applications. In Cleancentive it was decided to implement using the first option, which is to create an Android project. It requires defining the module name in native code which will be used in the JavaScript code by implementing the getName method as shown in Listing 21.

```
public class ForegroundServiceModule extends ReactContextBaseJavaModule {

  private final ReactApplicationContext reactContext;

  public ForegroundServiceModule(ReactApplicationContext reactContext) {
      super(reactContext);
      this.reactContext = reactContext;
  }*
```

```
  @NonNull
  @Override
  public String getName() {
      return "ForegroundService";
  }
```

Listing 21. Foreground Service module

After exposing the methods in native code, it is then required to create the bridge in a JavaScript file by using the code in Listing 22. It also requires creating a package by implementing a ReactPackage shown in listing 23

```
import BackgroundTimer from 'react-native-background-timer';
import {NativeModules, AppRegistry} from 'react-native';

const ForegroundServiceModule = NativeModules.ForegroundService;
```

Listing 22. Foreground Service Bridge

```
public class ForegroundServicePackage implements ReactPackage {
  @NonNull
  @Override
  public List<NativeModule> createNativeModules(@NonNull ReactApplicationContext
reactContext) {
      return Collections.singletonList(new ForegroundServiceModule(reactContext));
```

```
   }

   @NonNull
   @Override
   public List<ViewManager> createViewManagers(@NonNull ReactApplicationContext
reactContext) {
       return Collections.emptyList();
   }
 }
```

Listing 23. Foreground package

Using the command in Listing 24 will register the package in the main Android application so it is available in Javascript.

```
packages.add(new ForegroundServicePackage());
```

Listing 24. Foreground package registration

This thesis doesn’t cover the whole volume of code written for integration, just the basic necessary steps. Now the bridge between the native and React Native application is ready, the logic behind adding the notification is required. Since the application has to show different text based on the different notification, it was decided that Notification config will be provided by JavaScript. Listing 24 explains the notification config, Listing 25 explains how notification is passed from JavaScript to Native code and Listing 26 explains how it actually starts the service with the notification.

```
export type NotificationConfig = {
      id: number;
      title: string;
      message: string;
      vibration?: boolean,
      visibility: 'public' | 'private' | 'secret',
      largeicon?: string,
      icon?: string,
      ongoing?: boolean,
      importance: 'default' | 'max' | 'high' | 'low' |
      'min' | 'none' | 'unspecified'
      number: number,
      button?: boolean,
      buttonText?: string,
      buttonOnPress?: string,
      mainOnPress: string,
};
```

Listing 24. Notification configuration

```
  if (!this.started) {
    await ForegroundServiceModule.startService(config);
    this.started = true;
  }
  return this.started;
}
```

Listing 25. Starting foreground service

```
private boolean startService(Bundle notificationConfig) {
  int id = (int) notificationConfig.getDouble("id");
  Notification notification = NotificationHelper
    .getInstance(getApplicationContext())
    .buildNotification(getApplicationContext(), notificationConfig);
  startForeground(id, notification);
  return true;
}
```

Listing 26. Displaying notification when activity starts

Automated starting of the user activity was difficult to implement in the scope of this thesis because of the limitations imposed by iOS and Android, so a simple process for recording the activity was selected. This section shows simple play, pause and stop buttons which were added to record user location and to draw that on Google Maps as shown in Figure 13 and Figure 14.

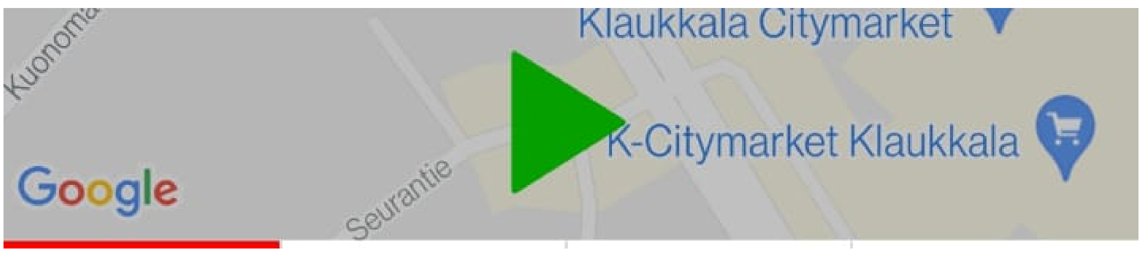


Figure 13. Start trip action

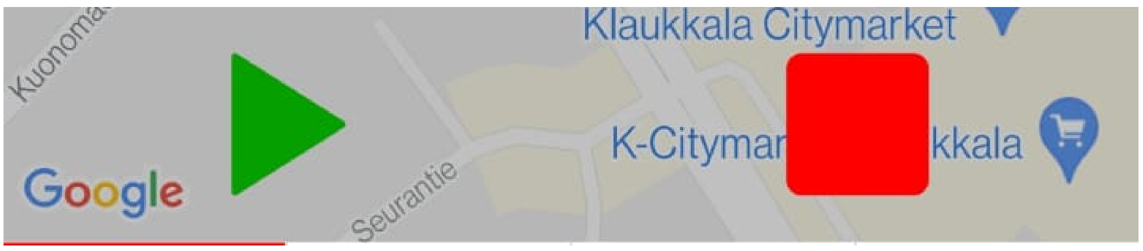


Figure 14. Restart and stop trip actions

Based on the above information it is clear how the recording of the location should be carried out, but the application needs some sort of map to display that information on, so it can be seen that recording actually took place.

It was discussed and finalised with the client's technical team that **React-native-maps** would be used to install Google Maps within the application. There is an option to show the map on both Android and on iOS. This requires the Google Map key, which is generated using Google Console. On iOS, one can choose between Google Maps or the native Apple Maps implementation (22). The command shown in Listing 27 was used to install Google Maps within the application.

```
npm install react-native-maps --save-exact
```

Listing 27. Installation of Google Maps

After installation, the Google Maps API key should be added to AndroidManifest.xml along with the installation of the play service in the emulator as shown in Listing 28.

```
<application>
<meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="GOOGLE_MAP_API_KEY"/>

<uses-library
      android:name="org.apache.http.legacy"
      android:required="false"/>
      </application>
```

Listing 28. Google Maps configuration

After installation was successful, a new component was created for “Map”, as it will be used in many places. The code of the reusable map component is shown below in Listing 29.

```
import React from 'react';
import MapView, {MapViewProps} from 'react-native-maps';
import {styles} from './Map.styles';

interface IProps extends MapViewProps {
   width?: number;
   height?: number;
}

const Map: React.FC<IProps> = ({height = 1920, width = 1080, children,
...props}) => {
   const ASPECT_RATIO = width / height;
   const LATITUDE = 60.3803784;
   const LONGITUDE = 24.7555976;
   const LATITUDE_DELTA = 0.005;
   const LONGITUDE_DELTA = LATITUDE_DELTA * ASPECT_RATIO;
   return (
       <MapView
           style={styles.map}
           initialRegion={{
               latitude: LATITUDE,
               longitude: LONGITUDE,
               latitudeDelta: LATITUDE_DELTA,
               longitudeDelta: LONGITUDE_DELTA,
           }}
           {...props}>
           {children}
       </MapView>
   );
};

export default Map;
```

Listing 29. Google Maps reusable component

Now that we have a reusable map component, it can be added to the home page where the application displays the user location. Applications are now required to record the user location by implementing the actions of play, pause and stop.

```
 <ActivityActions onEndTrip={onEndTrip} onStartActivity={onStartActivity}
onStopActivity={onStopActivity} />
```

Listing 30. Activity action component

ActivityActions was created, with buttons to start and stop the activity or the trip. The original idea was that as soon as the user moves out of the apartment, the activity will start automatically. This sort of logic will be implemented in the next section. Currently, when the user presses the start button it will begin recording the location, while the stop button will stop the recording. When there is no active trip in the record, pressing the start button for the first time will start the trip and at the same time will start the activity as well. When the trip is ongoing, pressing the stop button will end the activity but it will not end the trip. Pressing the stop button for a second time will end the trip, but if the user has stopped in the middle of the activity for some reason, then pressing the start button again will start another activity. Later on, these activities will be adjusted to conform with the actual activity.

The following pseudocode explained in Listing 31 is the implementation of activity recording when the start button is pressed. Task configuration defines the name of the task and the delay in which this task will be re-executed. It checks if the service is already running: if not, it displays the activity recording notification.

```
const taskConfig: TaskConfig = {
  taskName: 'RecordingLocation',
  repeat: true,
  delay: secondsToMillis(5),
  taskId: 'activityRecording',
  onError: (e: Error) => console.log('on error:', e),
};

const onStartActivity = async () => {
      if (!await foregroundService.isRunning()) {
          await foregroundService.startService(notificationConfig);
      }
      await setActiveTripStatus('started');
      await foregroundService.addTask({
         ...taskConfig,
         onExecution: trackLocation,
      });

      setStopRecording(false);
  };
```

Listing 31. Starting activity with the start button

When the activity is started, it adds a task which runs trackLocation method with the defined time interval. Track location fetches the user’s current location, adds that to the paths which are used to draw the activity on the map and adds at the same time the new coordinates to the active trip as shown in Listing 32.

```
const trackLocation = async () => {
  if (!fetchingLocation) {
      const coordinates = await getCurrentLocation(t, defaultGeoOptions);
      const prevPath = await getActiveTrip();
      if (coordinates) {
         updatePath(prev => [...prev, coordinates]);
         await setActiveTrip([...prevPath, coordinates]);
      }
   }
};
```

Listing 32. Location tracking

Stopping the activity is a simple process: the application simply has to mark the trip as paused, remove the notification and recording of the topic as shown in Listing 33:

```
const onStopActivity = async () => {
      await setActiveTripStatus('paused');
      if (await foregroundService.isRunning()) {
          await foregroundService.removeTask(taskConfig.taskId);
          await foregroundService.stopService();
          setStopRecording(true);
      }
};
```

Listing 33. Stop activity on stop button

The following pseudocode, shown in Listing 34, explains the logic behind ending the trip. It removes the active trip from the record, and sets the status to end. If there are any tasks running in the background it also stops those tasks and clears the activity notification.

```
const onEndTrip = async () => {
      await removeActiveTrip();
      await setActiveTripStatus('ended');
      if (await foregroundService.isRunning() || stopRecording) {
          const stopped = await foregroundService.stopService();
          setServiceStopped(true);
          setStopRecording(false);
          return stopped;
      }
  };
```

Listing 34. End trip

The recording of the activity and the location is completed, and now drawing that to the map requires Google Maps Polyline feature. Polyline is the line drawn between two points, tracking the whole activity to the map application with all coordinates. By using the code in Listing 35, Polyline can be drawn on the maps with the given coordinates where the coordinates are extracted from the active trip. How Polyline is displayed on the map is also show below in Figure 16:

```
<Map showsUserLocation={true}>
  <Polyline coordinates={trip.map((item) => item.coords)} strokeWidth={5} />
</Map>
```

Listing 35. Draw trip on the map

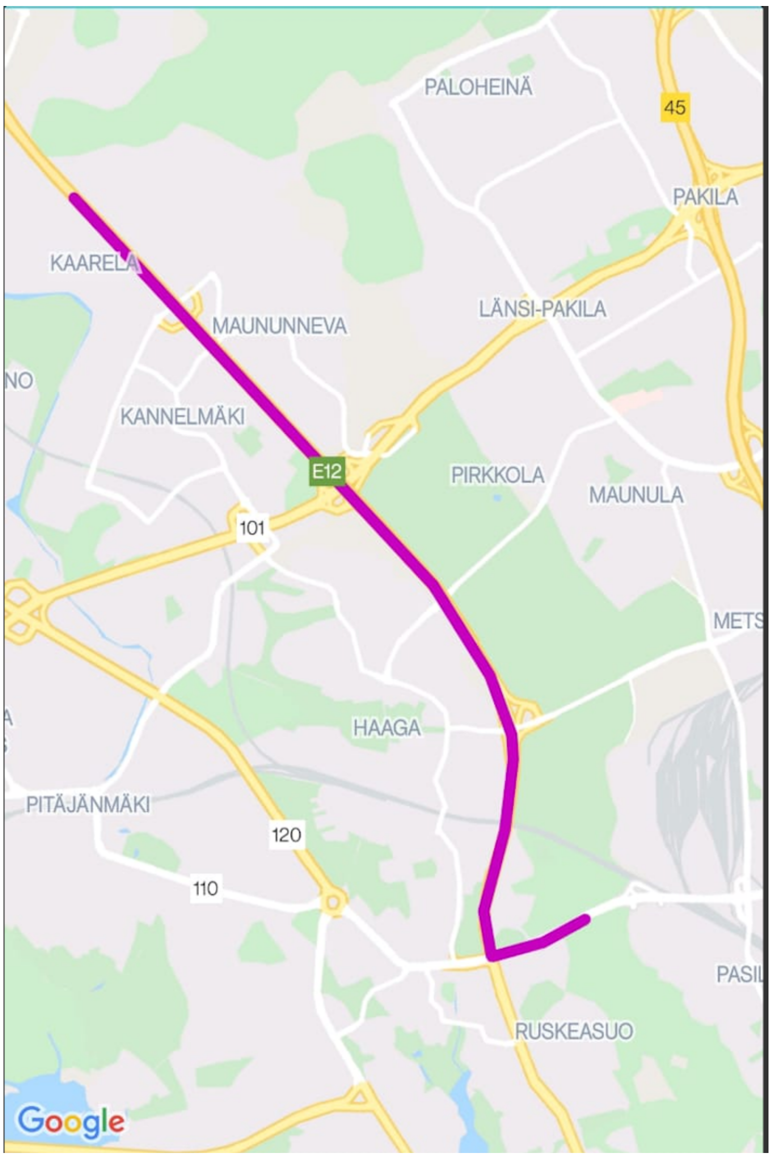


Figure 16. Trip drawing on the map

4.4.6. Tracking user activity

This chapter covers the implementation of enriching the location recording completed in the above section with the activity type. For the purpose of this thesis, it was decided that initially it would be sufficient if the application got the activity data from an external service such as Google Fit. For this purpose, Google Fit is installed using the code in Listing 36:

```
yarn install react-native-google-fit
```

Listing 36. Google fit installation

When the activity is running, the start Google Fit permission is requested from the user. It is necessary to require Google Fit permission before the start of the trip because otherwise Google Fit will not be able to identify the user activity. Listing 37 presents the code which was added to startActivity method to request the permission:

```
GoogleFit
.authorize(options)
.then(({success}: any) => resolve(success))
.catch(() => {
  console.log('Unknown error fetching Google fit authorization', 'ERROR');
  resolve(false);
});
```

Listing 37. Google Fit authorization

Google Fit provides the list of activities between the given start and end time. It also requires at least 60 seconds to determine the current activity: until then, it is shown as "unknown activity type". Instead of fetching the activity on each location change, it was agreed with the client that once the trip is ended the application can fetch all the activities during that time and update the trip by finding out the start and end point of each and every activity.

The following code presented in Listing 38 was added to fetch the activities, filter the unknown activities and merge them with the current trip.

```
const allacts = await getActivitySamples(startDate, endDate);
const activities = allacts.filter(a => a.activityName !== UNKNOWN);
```

Listing 38. Fetching activating from Google Fit

4.4.7. Calculating Incentive Model

This section of the thesis implements the incentive model which was presented in the previous sections. As per formula, the application needs to calculate the distance covered in each activity. It was decided by discussion with the client to use the haversine formula.

The haversine formula determines the great-circle distance between two points on a sphere, given their longitudes and latitudes (23) which is shown in Listing 39 below (24):

```
const distance = (p1: GeoPosition, p2: GeoPosition): number => {
       const lat1 = p1.coords.latitude;
       const lon1 = p1.coords.longitude;

       const lat2 = p2.coords.latitude;
       const lon2 = p2.coords.longitude;

       const R = 6371e3; // earth radius in meters
       const φ1 = lat1 * (Math.PI / 180);
       const φ2 = lat2 * (Math.PI / 180);
       const Δφ = (lat2 - lat1) * (Math.PI / 180);
       const Δλ = (lon2 - lon1) * (Math.PI / 180);

       const a = (Math.sin(Δφ / 2) * Math.sin(Δφ / 2)) +
           ((Math.cos(φ1) * Math.cos(φ2)) * (Math.sin(Δλ / 2) * Math.sin(Δλ /
2)));

       const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

       return R * c;
   };
```

Listing 39. Haversine formula

In Cleancentive, there are many coordinates recorded in each activity, so to calculate the total distance covered in each activity the application needs to factor in all the coordinates, calculate the distance between them and sum everything up. Listing 40 shows the code which calculates the distance of the activity by receiving the coordinates:

```
const calculateDistance = (points: GeoPosition[] = []): number => {
       if (!points || points.length === 0) {
           return 0;
       }

       const distObj = points.reduce((dista: any, point) => {
           return {
               point,
                distance: Object.keys(dista).length === 0 ? 0 : (dista.distance
+ distance(dista.point, point)),
           };
       }, {});

       return distObj.distance;
   };
```

Listing 40. Calculating distance of the trip

Now that we have the method to calculate the distance covered in the activity, it just requires the application to implement the formula for calculating the benefit loss or gain in each trip. In the benefit calculation, car journeys have been carefully noted, and ignored when mobility and health benefits for the activities are calculated. The total "own effort" covered distance is calculated, and multiplied with the unit price of each activity to find out the health benefit earned by the user as shown in Figure 17. Lists of all mobilities records, possible benefits earned or lost and the trip details are also calculated, shown below in Figure 18.

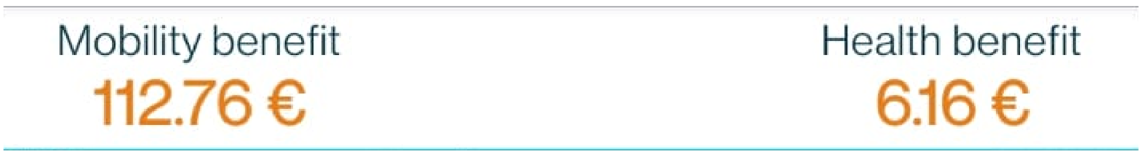
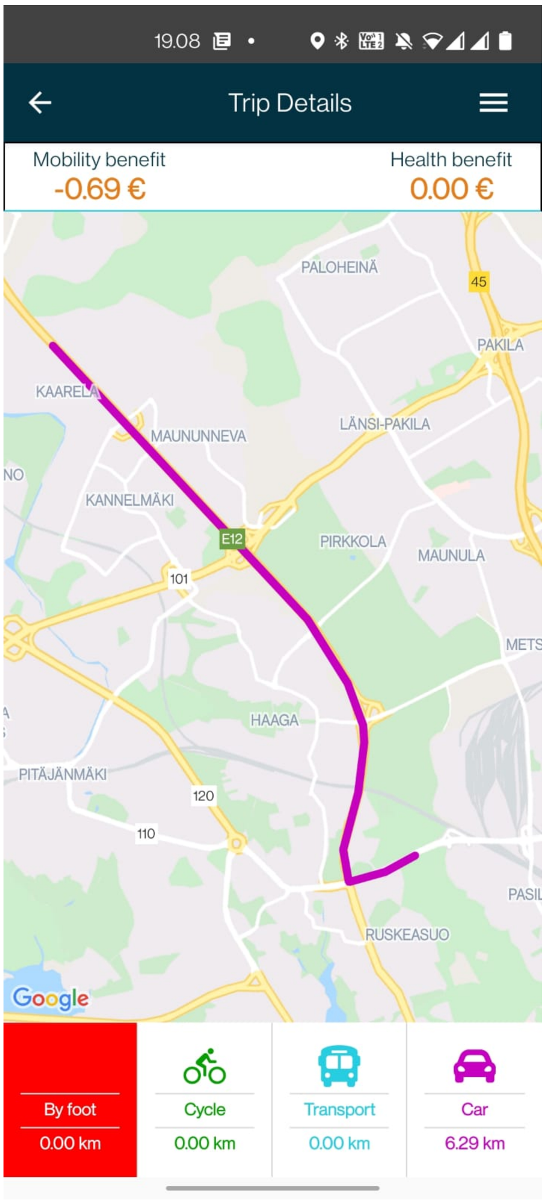
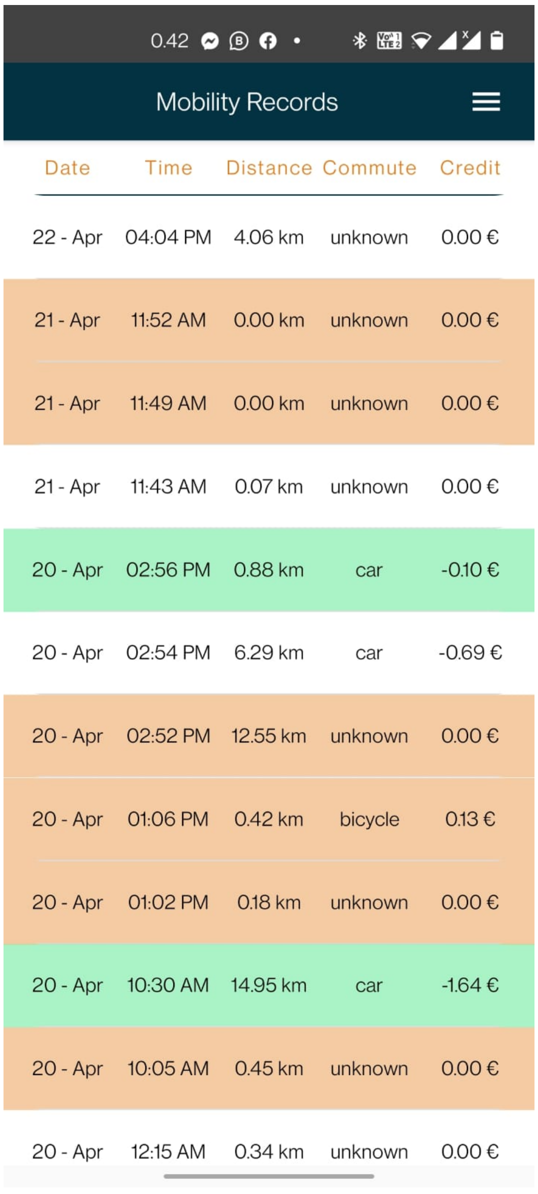


Figure 17. Health and Environment benefits





Date	Time	Distance	Commute	Credit
22 - Apr	04:04 PM	4.06 km	unknown	0.00 €
21 - Apr	11:52 AM	0.00 km	unknown	0.00 €
21 - Apr	11:49 AM	0.00 km	unknown	0.00 €
21 - Apr	11:43 AM	0.07 km	unknown	0.00 €
20 - Apr	02:56 PM	0.88 km	car	-0.10 €
20 - Apr	02:54 PM	6.29 km	car	-0.69 €
20 - Apr	02:52 PM	12.55 km	unknown	0.00 €
20 - Apr	01:06 PM	0.42 km	bicycle	0.13 €
20 - Apr	01:02 PM	0.18 km	unknown	0.00 €
20 - Apr	10:30 AM	14.95 km	car	-1.64 €
20 - Apr	10:05 AM	0.45 km	unknown	0.00 €
20 - Apr	12:15 AM	0.34 km	unknown	0.00 €

Figure 18. Mobility record and Trip Details

5. Discussions and Conclusions

This thesis was about developing and designing an MVP of the mobile application for Codistan employees to encourage healthy lifestyles. The main question were:

- What technologies could be used to develop such a tracking application?
- What benefit would the application bring to the employees and employer?
- What kind of incentive model works best?

Many companies are trying out different methods to keep their employees fit and healthy, especially in the IT industry where most work requires sitting in one place in front of a computer. This kind of application can bring a possibility for the employees to earn some incentive by keeping themselves fit and active.

The requirements for this solution were that Cleancentive could bring a healthy lifestyle to its employees and clients who are IT professionals. One of the requirements set by the client was that Cleancentive should be used as part of Codistan’s normal everyday operation, where employees come to the office by using public transport or by walking/bicycling and avoid using cars to reduce their carbon footprints. There are environmental factors currently in development as well: these would offer the option to avoid car usage, reduce traffic congestion and encourage users to use public transportation.

The thesis presents a comparison between different development options for tracking applications. Since the application was actually intended to be put into service, this thesis contains recommendations of what to consider in developing web applications, along with technology choices and methods.

The MVP created in this thesis includes designing a Cleancentive mobile application as a whole: the UI and UX of the application, technical method and programming language recommendations, recording of the user location with activity and calculating the incentives. The main finding about technical solutions was that, when developing tracking applications involving a great deal of hardware and mobile sensors, it is better to develop native applications instead of hybrid applications. This brings the possibility of developing the application in a better way, and avoiding unnecessary complications. React Native might not be the correct language choice for developing such a tracking application, because it might need to interact with phone sensors and background activities. React Native also requires a lot of custom module writing and that work has to be done separately for iOS and Android platforms. If 50% of the work requires custom module writing, then it is better to write native code. React Native works just fine now for the purpose of this thesis, but it would be wise in the future to expand things and create a native application which brings more expansion options for this product.

It was also identified while testing the application that it brought a somewhat healthy lifestyle to the author's daily routine, and if that also comes with a fiscal compensation then this would potentially increase the usage of the application.

It was noticed during testing that the author has started to feel more energetic during everyday routines, and during a period when most of the time was spent on the computer without any physical activity, the author felt low energy. The proposed incentive model works for the client, but it could vary from company to company, department to department and country to country. The proposed model is just set as an example, and it could be modified as needed.

This experiment was small, but it brought out the possibilities that a mobile tracking application offers, and such an application could encourage healthy lifestyles for users, if given a proper incentive. The kind of simple user activity and location tracking application presented in this thesis could be expanded to multiple fields, departments, companies and countries.

The next steps include implementing the activity recognition API provided by Android and iOS to record the trip automatically. The logic could be implemented in such a way that when the user leaves home, and the activity recognition senses that the user is moving, it could start the activity automatically and when the user returns home and is again stationary, the application will stop the trip. When the user is still outside and the activity sensor picks up that the user is stationary it will stop the activity and wait for the user to start moving again to begin a new activity. This would also bring real time drawing of different activities on the active trip.

Implementing the same activity recognition on the iOS would also be a positive development, so that iPhone users could also get to use the application. Other development items in the pipeline include user authentication, recording and storing user addresses, onboarding process, wallet, safety, statistics and push notification in the mobile application, web applications where companies can be added easily by the client, a company panel where they can manage their employees, set their benefits, approve the activities and check the statistics related to individuals and expenditure.

References:

1. What's the difference between single-page application and multi-page application?: | ADCI Solutions [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://www.adcisolutions.com/knowledge/whats-difference-between-single-page-application-and-multi-page-application

2. (PDF) THE DIFFERENCE BETWEEN DEVELOPING SINGLE PAGE APPLICATION AND TRADITIONAL WEB APPLICATION BASED ON MECHATRONICS ROBOT LABORATORY ONAFT APPLICATION [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://www.researchgate.net/publication/324380010_THE_DIFFERENCE_BETWEEN_DEVELOPING_SINGLE_PAGE_APPLICATION_AND_TRADITIONAL_WEB_APPLICATION_BASED_ON_MECHATRONICS_ROBOT_LABORATORY_ONAFT_APPLICATION

3. Programming: Single Page Applications Fundamentals [Internet]. 2021 [cited 2021 Apr 18]. Available from: http://thephantomprogrammer.blogspot.com/2016/08/web-fundamentals.html

4. SPA vs MPA: Which One is Better For You? (2021) | Yojji [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://yojji.io/blog/spa-vs-mpa

5. AngularJS - Wikipedia [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://en.wikipedia.org/wiki/AngularJS

6. React (JavaScript library) - Wikipedia [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://en.wikipedia.org/wiki/React_(JavaScript_library)

7. Vue.js - Wikipedia [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://en.wikipedia.org/wiki/Vue.js

8. Comparative study of Svelte vs React vs Angular vs Vue - GeeksforGeeks [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://www.geeksforgeeks.org/comparative-study-of-svelte-vs-react-vs-angular-vs-vue/

9. Typescript vs Javascript: Difference You Should Know [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://hackr.io/blog/typescript-vs-javascript

10. What is an API?: (Application Programming Interface) | MuleSoft [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://www.mulesoft.com/resources/api/what-is-an-api

11. Representational state transfer - Wikipedia [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://en.wikipedia.org/wiki/Representational_state_transfer

12. SOAP - Wikipedia [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://en.wikipedia.org/wiki/SOAP

13. Becker J, Matzner M, Muller O. Comparing Architectural Styles for Service-Oriented Architectures a REST vs. SOAP Case Study. In 2009 [cited 2009 Jan 1]. p. 20715. Available from: http://link.springer.com/content/pdf/10.1007/b137171_22.pdf

14. Node.js vs. Java: An epic battle for developer mindshare | InfoWorld [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://www.infoworld.com/article/2883328/nodejs-vs-java-an-epic-battle-for-developer-mindshare.html

15. Ionic vs React Native: Which Framework you should Choose? | by Team4Solution | Medium [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://team4solution.medium.com/ionic-vs-react-native-which-framework-you-should-choose-4572de4e938a

16. Integrated development environment - Wikipedia [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://en.wikipedia.org/wiki/Integrated_development_environment

17. prettier/prettier: Prettier is an opinionated code formatter. [Internet]. 2021 [cited 2021 Apr 18]. Available from: https://github.com/prettier/prettier

18. createStackNavigator | React Navigation [Internet]. 2021 [cited 2021 Apr 25]. Available from: https://reactnavigation.org/docs/stack-navigator/

19. PermissionsAndroid React Native [Internet]. 2021 [cited 2021 Apr 25]. Available from: https://reactnative.dev/docs/permissionsandroid

20. Headless JS React Native [Internet]. 2021 [cited 2021 Apr 25]. Available from: https://reactnative.dev/docs/headless-js-android

21. react-native-maps/installation.md at master react-native-maps/react-native-maps [Internet]. 2021 [cited 2021 Apr 25]. Available from: https://github.com/react-native-maps/react-native-maps/blob/master/docs/installation.md

22. Haversine formula - Wikipedia [Internet]. 2021 [cited 2021 Apr 25]. Available from: https://en.wikipedia.org/wiki/Haversine_formula

23. Calculate distance and bearing between two Latitude/Longitude points using haversine formula in JavaScript [Internet]. 2021 [cited 2021 Apr 25]. Available from: http://www.movable-type.co.uk/scripts/latlong.html