Road Trip Companion
Interim Report

DT211c

BSc in Computer Science Infrastructure

Erik Grunnér

C17412636

Damian Bourke

School of Computer Science

Technological University, Dublin

16/12/2020

Abstract

This project has that main goal of producing an Road Trip Companion app. The app will help in the creation of Road Trip by use of its own recommender system. The app will guide the user along the road trip and will support some level of sharing between friends.

By using Open-Source map data the app will process and generate unique recommendations tailored to each user. Through use of the app, it will learn to know what the user likes and does not like to create road-trip suggestions for them.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

__erik grunnér________________________

Erik Grunnér

Date

Acknowledgements

I would like to thank Damian Bourke, my project supervisor, for the continued guidance throughout this project.
I would also like to thank my friends and family who agreed to take part in the test group.

Contents

Table of Figures

	September				October				November				December				January				February				March				April				May			
Initial Proposal						x																														
Requirements Gathering							x	x	x	x																										
Initial Design											x	x	x																							
Horizontal Prototype												x	x																							
Interim Report							x	x	x	x	x	x	x	x																						
Interim Demonstration															x																					
Iterative Prototyping									x	x	x	x	x	x	x	x			x	x	x	x	x	x	x											
Final Design																										x	x									
Software Testing															x	x			x	x	x	x	x	x	x	x	x	x								
Project Evaluation																													x	x						
Dissertation Document							x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x					
Final Demonstration																																x				

Introduction

In this section of this report the proposal for the project will be laid out. The project background will be discussed, the reasons for choosing this project. A description of what the project is and what the project aims to achieve with its objectives. Smart objectives will be set as a way of structuring and separating out each major body of work that needs to complete in doing the proposed project. The Scope of the project will lay out what the project is and also what it is not. Finally, a thesis Roadmap will give an overview as to the structure being taken whilst writing out the rest of the project.

1.1. Project Background

While travelling over the years in various places I've found that my lack of knowledge of the area and lack of special awareness to that new area has led to my missing out on various visits and activities. The goal of this app will eliminate all of this while also preserving the sense of adventure that comes along with a holiday.

In this research two similar mobile apps were identified

- Road Trippers has a similar concept, the main difference being that they create curated guides to specific areas, all within the United States, Canada, Australia and New Zealand. This style of curation over automation means that users are being suggested the same trips and have a more limited range of options for their trip. It is also possible to create personal trips, but this is not the focus of the app. In my research this functionality is lacking in many ways.

- Google Trips also has similar app but puts focus on visiting specific areas or cities. This is different in that the journey is not a focus to the trip, but that the journey is an afterthought, i.e. travel is only treated as travel and not a part of the experience.

[1] Fitzgerald C has an abandoned patent application for a recommender system designed for navigation, which would be similar to one part of my proposed project. Although it specifies itself based on a property database, rather than looking for open license data.
Further Reading into a Personalised trip recommender based on points of interest proved to be similar in that time is a focus in their recommender system and this would relate more closely to how I wish to implement the recommender system into the navigational system for on-the-go recommendations.
[2] [3]

1.2. Project Description

The app will be split into two main functions:

- The Planning Phase – This is where to start off the User would enter both starting point and destination, similarly to any other navigation app. Then timing estimations would be presented along with possible on route destinations for each timeframe. Following that will come more in-depth look at the selected route with checkboxes for all desired stops. Finally, a calculated estimation for total trip length.

- The Travel Phase – This would be where the actual navigation would take place. Alongside this would be an (optional) interactive experience guiding you through the areas you visit and suggesting additional smaller pitstops; This could include bathroom breaks, petrol stops or mainly extra destinations such as viewing points or short activities.

1.3. Project Aims and Objectives

The Goal of this project is to develop a mobile app which functions as both a road trip planning application and acts as a local guide while on the trip. Mainly focused on Driving but should work for walking/cycling trips.

1.3.1. Smart Objectives

Figure 1-1-1 Smart Objectives

Smart Objectives are one of the ways available to define separation between a project's tasks. As shown in Figure 1-1 it can be seen that they follow 5 basics rules of definition

1. Specific
2. Measurable
3. Achievable
4. Realistic
5. Timebound

By considering if a task follows all five of these rules one can be confident that the task can be considered a smart objective. Using this structure helps in organisation of development and allows for contemplation on how tasks should be split and how they will be completed.

1.3.2. Project Objectives

- Evaluate Similar applications through the use of Nielsen's Heuristics
- Generate a list of requirements based off the heuristics and the projects goal
- Identify an appropriate design methodology based on the proposed system and other real-world constraints
- Research and identify all the appropriate technologies needed in order to support the development the proposed application
- Construct a design of the proposed application along with Use-Case Diagrams, Data-flow Diagrams and any appropriate diagrams
- Design and create working prototype in order to have users perform UI testing on it in line with the iterative design process
- To research methods in order to develop prototypes for a recommender system.
- To document the progress of the project and report on the findings of the work.
- Continue producing iterative prototypes to progress the project.
- Test and evaluate each prototype based on the type of work completed for said prototype.

1.4. Project Scope

The main parts to the system focus on UX design and are based on developing the recommender system that goes along with the application.

The mapping service and real time navigation services will be taken from API's to facilitate the rest of the projects function.

1.5. Thesis Roadmap

This introductory chapter outlines the goals for the project and how they plan to be subdivided into smart objectives.

Chapter two is a look and evaluation of similar projects with the aim of developing requirements and subsequently researching relevant technologies for the development of the proposed application.

Chapter three identifies in what way the development of the project takes by ways of software methodologies and creating prototype systems.

Chapter four documents the prototyping process and how each one iteratively improves upon its predecessors.

Chapter five provides the methods used in the testing of the proposed system throughout its development in each of the separate components that make it up.

Chapter six is a summary of the entire project and development of the proposed system.

Literature Review

2.1. Introduction

In this chapter multiple projects which relate to this one will be discussed in order to gain insight into the reasons for their User requirements. This is very useful as each of them went through their own design process and have inherent knowledge baked into the applications. Nielsen’s heuristics will be used as a tool of evaluation. Furthermore, possible technologies that could be used in the proposed system will be evaluated and compared in order to decide on appropriate choices for the design phase.

2.2. Alternative Existing Solutions to Your Problem

The Software being investigated and reviewed with the help of these Heuristics included Road Trippers and Google Trips. Grading each application will be from 1 to 5 on each of the 10 Heuristics listed above, with a small explanation for each scoring. In order to do this similar trips on a small scale in the locality will be performed.
The grading system scores:

1. Does not meet heuristic specifications
2. Meets few of the heuristic specifications
3. Meets some of the heuristic specifications
4. Meets most of the heuristic specifications
5. Fully meets the heuristic specifications

2.2.1 Nielsen’s Heuristics

To evaluate these two systems an adjusted set of Nielsen’s Heuristics will be used. Heuristic evaluation is in essence a way to measure a systems usability by going through a series of checks. By using this type of inspection one can identify usability design flaws in each system. Each check should test against specific design principals which have been altered to suit the needs of these types of navigation apps.

1. **Visibility of system status**:
   The system's ability to keep the users informed on the state of system with the right outputs. This becomes especially important with safety concerns in mind due to the nature of the apps. In this case there is both driving and setting up as two modes it should be in.
2. **Match between system and the real world**:
   This refers to the system's ability to convey its information in a reasonable way so that the user can understand. This would include the appropriate language and the use of real-world phrases and terms. This can be seen as how well the app represents the possibilities of a road trip and how well it can guide you when you are on the trip.
3. **User control and freedom**:
   Allows for the users to remain in control of the system without having to go through unnecessary processes, clearly marked options should be available.
4. **Consistency and standards**:
   The structure of the system should follow its own conventions and be clearly structured for the users to understand.
5. **Error prevention**:
   A properly designed system would be better than good error prevention but nonetheless checks with the users before committing to options can help prevent further errors.
6. **Recognition rather than recall**:
   The users should be able to intuitively navigate around the system rather than having to take to the manual. Following industry standards can help in user's recognition of system functions. This point relates to safety during the navigation stage.
7. **Flexibility and efficiency of use**:
   The use of shortcuts or" Accelerators" can be a good way of adding flexibility and efficiency for the advanced users while also not impeding the experience from the new user's point of view.
8. **Aesthetic and minimalist design**:
   The design of a system plays a large role in how the users interacts with it especially in a navigation application. The right information must be shown at appropriate times and should not interfere with driving.
9. **Help users recognize, diagnose, and recover from errors**:
   Error messages should present in plain language as to allow any users to understand the problem and/or allows for some solution to the issue.
10. **Help and documentation**:
    A system which does not need explicit documentation is always preferable but does not rule out the fact that proper and structured documentation can aid in helping users with many specific problems.

2.2.2. Google Trips

Google trip is an online trip creation tool meant to be used in conjunction with Google Maps on the mobile. It provides lots of integration with other google services and generates most of its data procedurally making almost everything available on the web integrated with it.

1. **Visibility of system status**: 3
   The actual navigation on the system is very good but it lacks integration of the whole trip and its status.
2. **Match between system and the real world**: 4
   Google trips does a very good job of organising all the different activities into groups and making searching an easy task.
3. **User control and freedom**: 2
   While the setup process is very extensive and has lots of options, there is no app to accompany the web setup and instead saved points of interest are sent to Google maps where the route still needs to be created.
4. **Consistency and standards**: 3
   The separation of the web app and google maps led to a lot of frustration when the trips changed form on the mobile app.
5. **Error prevention**: 3
   Rerouting when wrong turns are taken.
6. **Recognition rather than recall**: 4
   The extensive use of symbols and concise descriptions made sure that navigation of the system was always known.
7. **Flexibility and efficiency of use**: 2
   While a user can change things on the go, as the trip creation tool is on the website it gets very awkward to edit the trip when it has been converted for use on google maps.
8. **Aesthetic and minimalist design**: 4

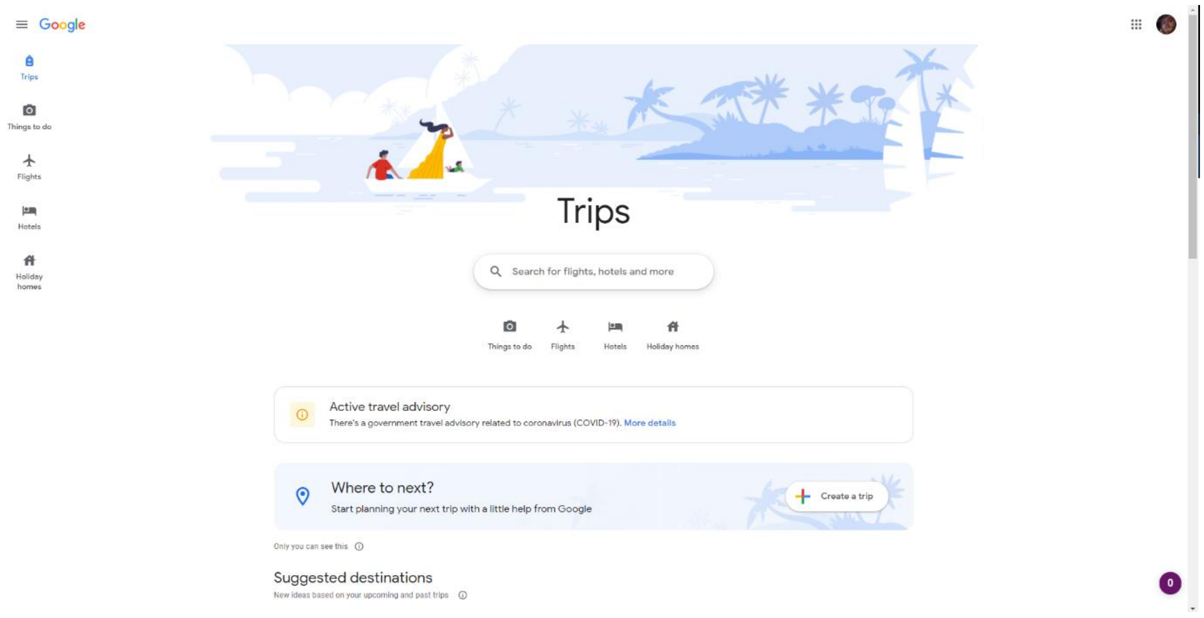
   The design was very clear for the most part and provided a clear view of what functions were available and what information the system wished to show the user.

9. **Help users recognize, diagnose, and recover from errors**: 3

   Overall lack of clarity in regard to the trips integration to the app.
10. **Help and documentation**: 4

    There is very thorough documentation on the site. Everything is tabbed for organisation and FAQ’s are intermingled to make searching for them easier.

Overall, the separation between the web application and the fact that google maps is supposed to be the mobile frontend to use google trips to be very awkward and made it difficult to use while on real-world trips.



Active travel advisory

There's a government travel advisory related to coronavirus (COVID-19). More details

Where to next?

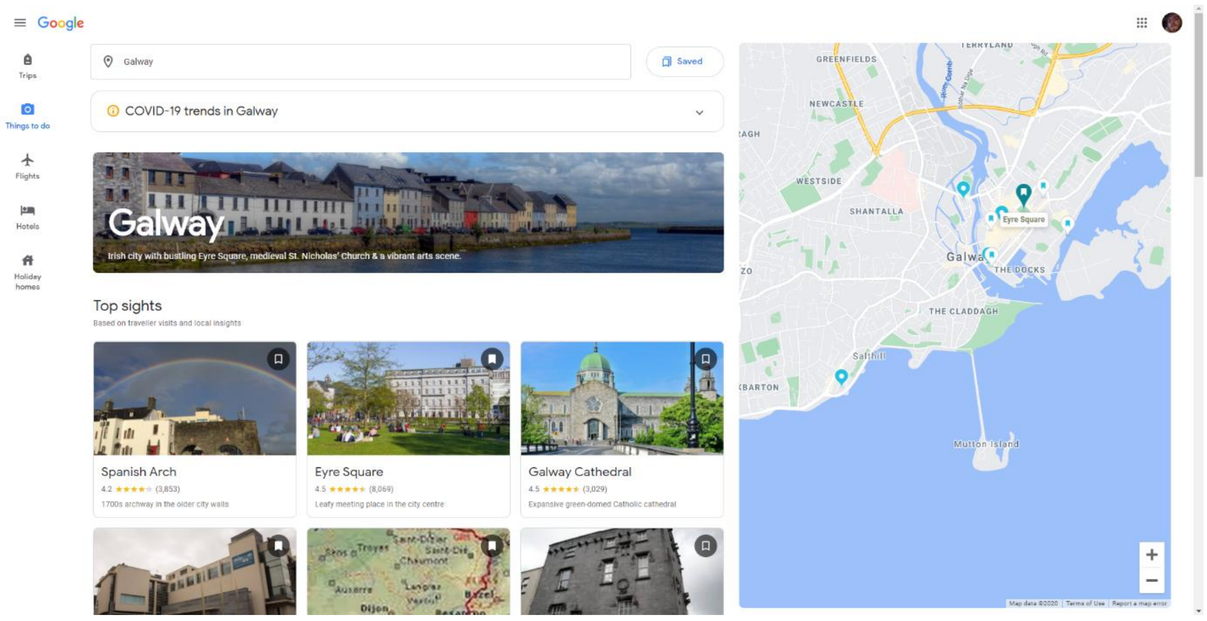
Start planning your next trip with a little help from Google

Only you can see this

Suggested destinations

New ideas based on your upcoming and past trips

Figure 2-1 Google Trips screenshot



Irish city with bustling Eyre Square, medieval St. Nicholas' Church & a vibrant arts scene.

Top sights

Based on traveller visits and local insights

Spanish Arch
4.2 ★★★★ (3,853)
1700s archway in the older city walls

Eyre Square
4.5 ★★★★★ (8,069)
Leafy meeting place in the city centre

Galway Cathedral
4.5 ★★★★★ (3,029)
Expansive green-domed Catholic cathedral

Figure 2-2 Google Trips screenshot

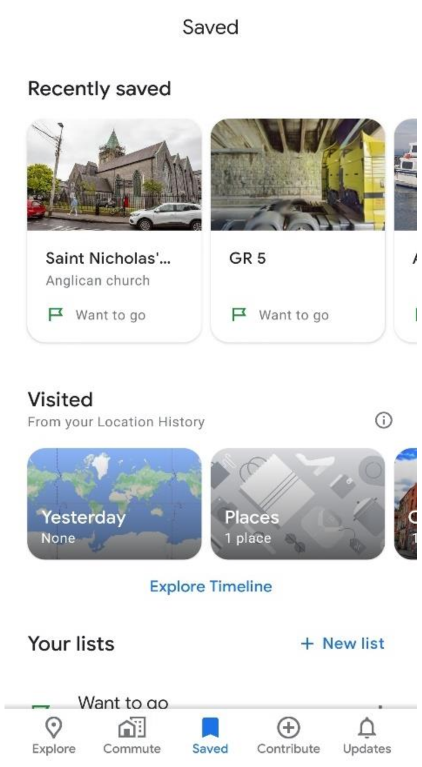


Figure 2-3 Google Maps View

2.2.3. Road Trippers

Road Trippers is both a mobile and web app designed around a road trip concept. THE content on the site is all curated trips that people have made. This makes it a bit limited in that sense. IT also has a self-creation mode for more personal trips.

1. **Visibility of system status**: 3
   The separation of the full road trip and having to route to individual points makes it harder to fully realise your progress of the whole trip. This is most evident when multiple stops are nearby each other.

2. **Match between system and the real world**: 4
   Much in the same way as the previous app, this had excellence relation between the system and the real world.

3. **User control and freedom**: 2
   The creation of trips is where it was very lacking. While it is possible the app seems to aim at delivering curated trips to the users. Although the user must create an account in order to use any of the curated trips.

4. **Consistency and standards**: 5
   Both the mobile and web app have been rigorously designed and match up quite well. There are very few inconsistencies.

5. **Error prevention**: 2
   the navigation is badly implemented into the app's workflow. Each specific stop needs to be selected and routed to, rather than a more integrated feel. This would be difficult for use while driving.

6. **Recognition rather than recall**: 5
   The layout and use of symbols were used in a similar way to google trips but in general the limits of the system to only roads trips made for an more specific but easier experience.
7. **Flexibility and efficiency of use**: 4
   The creation of new route is quite limited on the mobile application.
8. **Aesthetic and minimalist design**: 5
   The Aesthetic of both the web and mobile app had great attention to detail and provided a good experience.
9. **Help users recognize, diagnose, and recover from errors**: 4

   Most of the app is very clearly laid out and has an intuitive nature.
10. **Help and documentation**: 4
    In terms of help, a fleshed out help page on the website which is easily viewable on a mobile device provided great assistance.

To summarise the majority of the app to be well laid out and functional. The only real issue is the navigation being split up into chunks of routes in between places. This becomes a problem when stops are nearby each other.

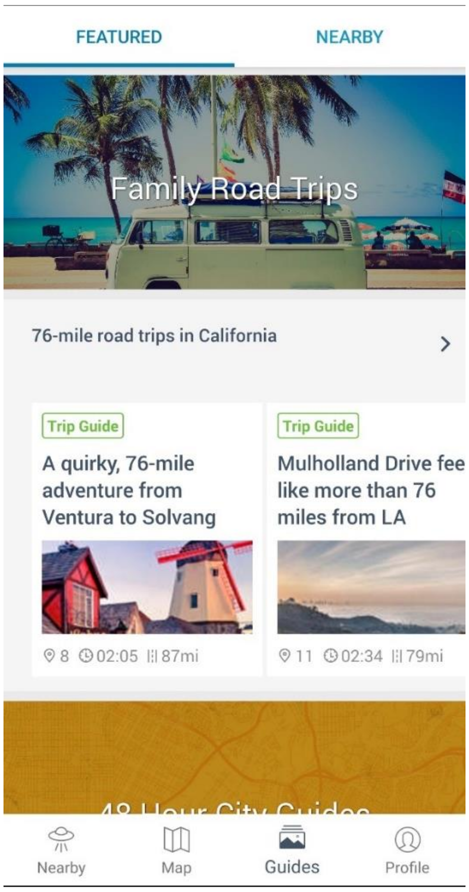
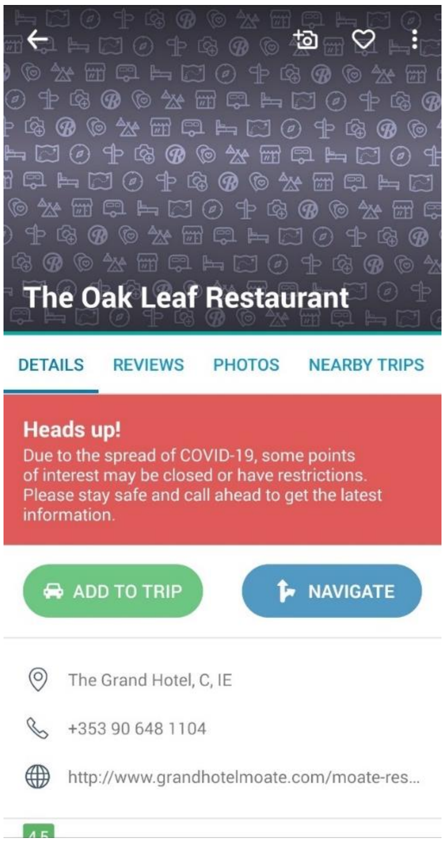

FEATURED
NEARBY
Family Road Trips
76-mile road trips in California
Trip Guide
A quirky, 76-mile adventure from Ventura to Solvang
8 02:05 87mi
Trip Guide
Mulholland Drive fee like more than 76 miles from LA
11 02:34 79mi
48 Hour City Guides
Nearby
Map
Guides
Profile

Figure 2-4 Road Trippers Home page


The Oak Leaf Restaurant

Heads up!
Due to the spread of COVID-19, some points of interest may be closed or have restrictions. Please stay safe and call ahead to get the latest information.

The Grand Hotel, C, IE
+353 90 648 1104
http://www.grandhotelmoate.com/moate-res...

Figure 2-5 Road Trippers Place Example

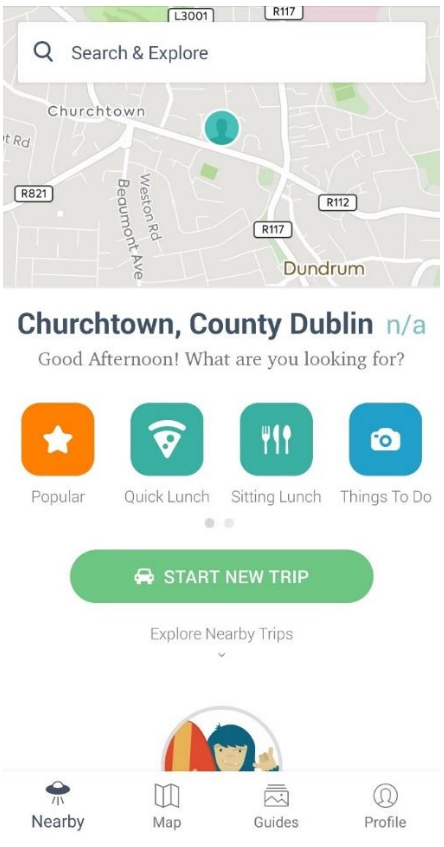
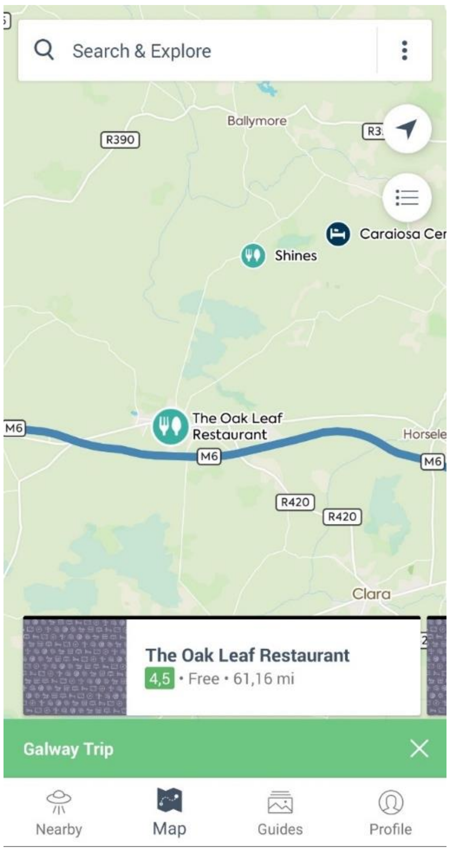

Churchtown, County Dublin n/a
Good Afternoon! What are you looking for?

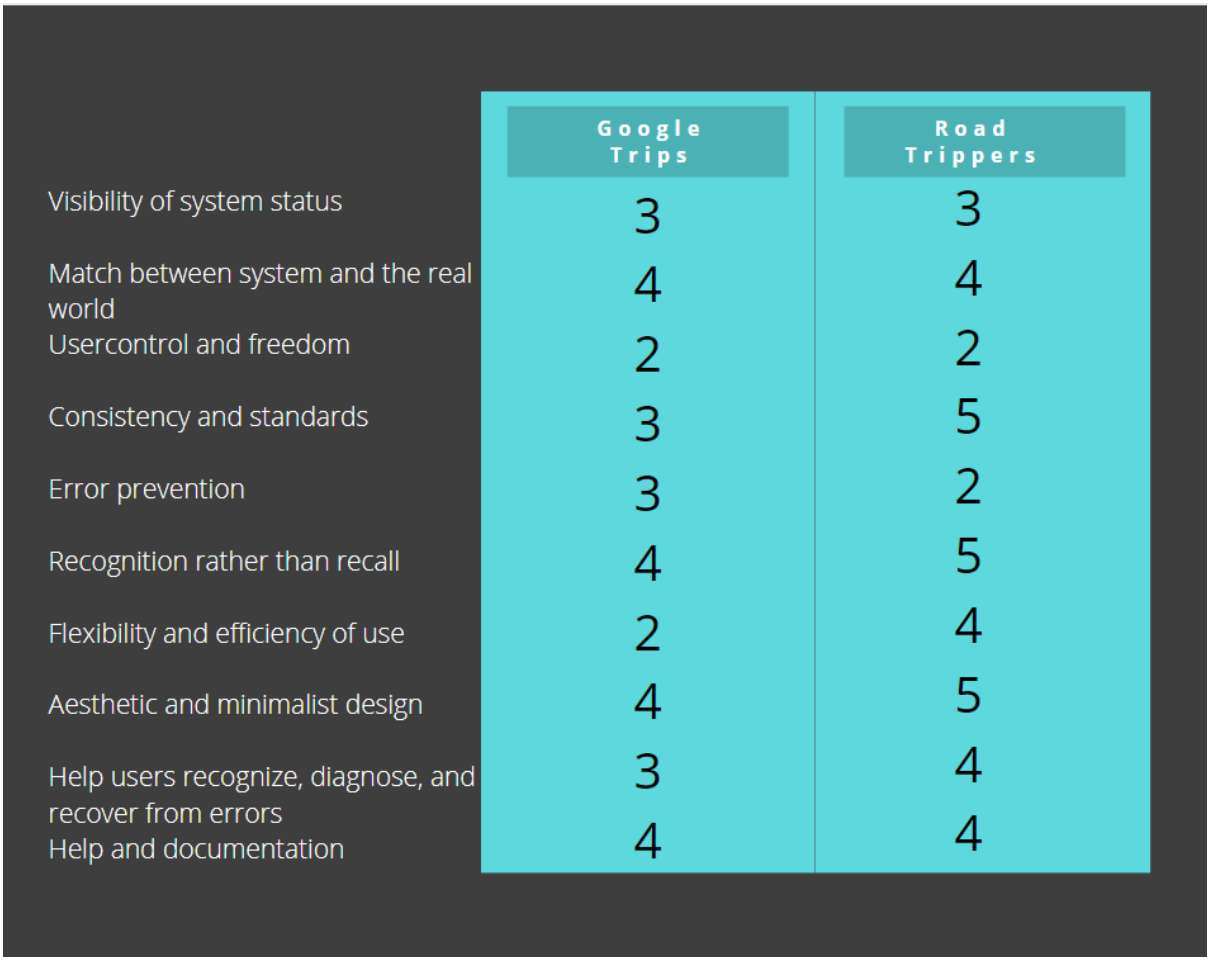
Figure 2-6 Road Trippers Start Trip



The Oak Leaf Restaurant
4,5 • Free • 61,16 mi

Figure 2-7 Road Trippers Map

2.2.4. Heuristic Evaluation



	Google Trips	Road Trippers
Visibility of system status	3	3
Match between system and the real world	4	4
Usercontrol and freedom	2	2
Consistency and standards	3	5
Error prevention	3	2
Recognition rather than recall	4	5
Flexibility and efficiency of use	2	4
Aesthetic and minimalist design	4	5
Help users recognize, diagnose, and recover from errors	3	4
Help and documentation	4	4

Figure 2-8 Results from Heuristic Evaluation

This chart shows what two similar applications do well and what they do badly. Starting from the top it is clear that the system status does have some good qualities that can be drawn from but namely the overall focus on the road trip needs to be considered when making the design for this system. The second metric shows how well both systems take in and show real world data, but this also considered their live navigation. User control and freedom is something that will be handled completely differently and should be well researched in the testing phase of the wireframe mock-ups. The standards will have taken inspiration from the unique but consistent styling Road Trippers managed to implement. The error prevention also needs careful evaluation while designing the proposed application. Aesthetic design combined with minimalism will be very important in this type of app and as has been shown contributes to the usability of the app while on the road. By creating the system in a simple and modular way the users will not be prone to errors which is important in a real-time task like navigation. Documentation is key to resolving any issues without intervention.

From here some conclusions can draw on the requirements the system will need based on these evaluations on similar systems. Stated below in are the requirements from heuristics.

- Simple well-defined UI
- Real time navigation
- Multiple user inputs (planning)
- Users should not be overwhelmed while information while driving
- Continuous recommendations should appear during the journey

- Search by map, voice, and text
- Save, edit and share trips
- In-app documentation (simple things) (help section)
- ^possible first-time tutorial
- Quick access(recents)

2.3. Technologies researched

In this section technologies which will be relevant to the development of the proposed application will be reviewed and compared. Comparisons between competing technologies will help to tease out helpful features that can be more specific to the use case. This research is core to the development of the process and can address possibilities can be achieved through API's or what may need to be developed. These will be talked about in reference to the proposed System Architecture Diagram (figure 2-9).

2.3.1 System Architect Diagram

Below in figure 2-9 is a skeleton architectural Design for the Proposed application. It will be split into two sections: Front-End, Back-End. As the API's form parts of either end they will be in the appropriate section.

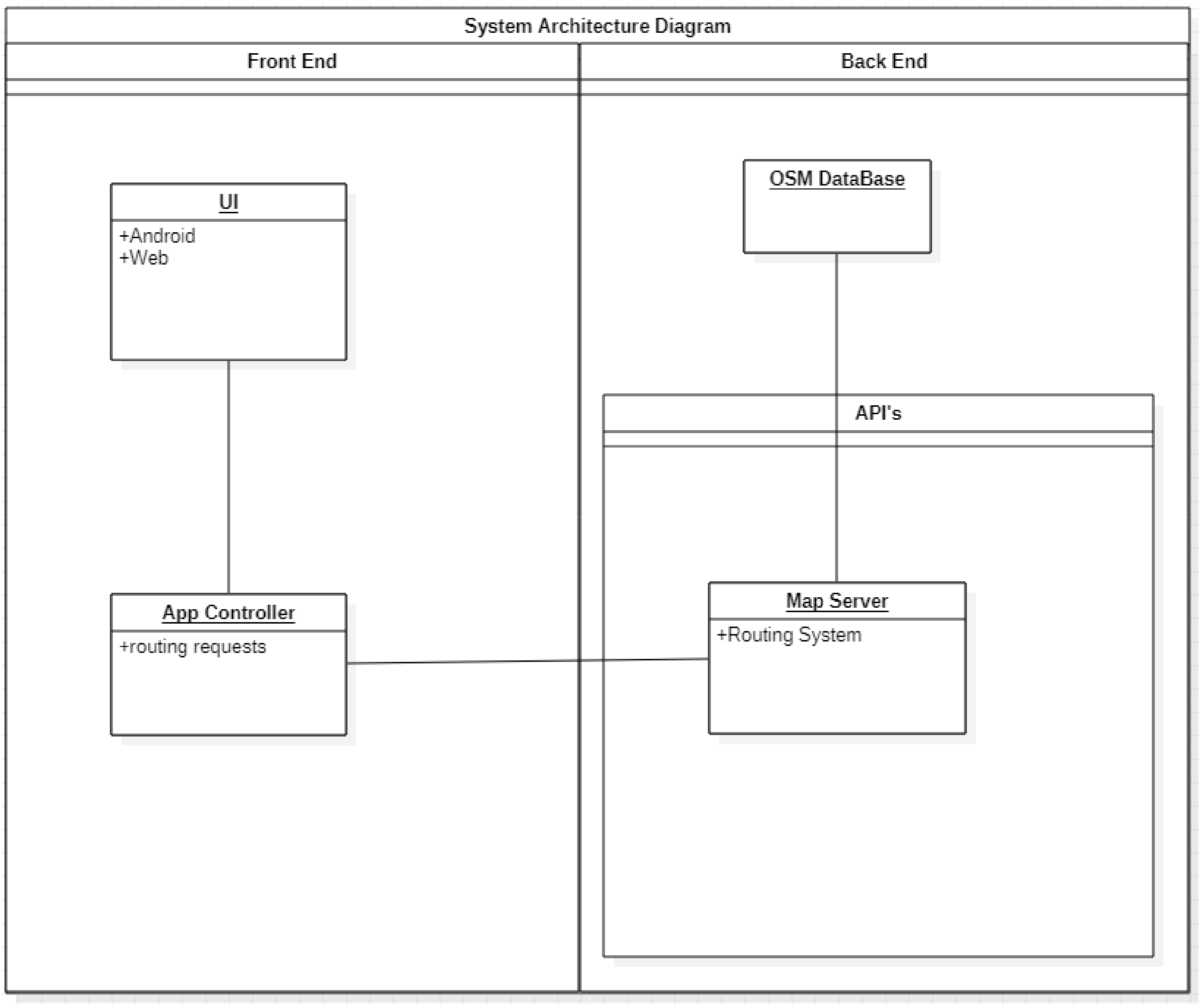


Figure 2-9 System Architech Diagram

2.3.2. Front-End

In this portion of the research section technologies which have relevance to the front-end development of the proposed application will be discussed.

2.3.2.1.Google API & MapBox

In order to perform all of my map related services within the proposed application two technologies have been primarily considered, Google maps API and MapBox. One of the first considerations was what type of map data is available to use with each of the platforms. While Google collects its own data MapBox works with data collected from OpenStreetMap, an open-source map data company. Another advantage MapBox offered over Google was that offline maps were available; while not necessarily a reason to choose one over the other is a welcome feature. Both companies offer free usage of the service for very low volume traffic which is perfect for development. Due to the high personalisation and freedom offered from MapBox it will be used for this project

2.3.2.2. Ionic & React Native

Ionic is a framework based on angular JS. This means that if supports languages such as HTML5, CSS and JavaScript. It offers lots of pros like faster development time thorough dynamically updating a preview web app and good stability with supported from the ionic team.

React is a framework based on JavaScript and is used to build apps natively on both iOS and Android. The developer then uses the same JS codebase to compile each app natively. Due to the personal familiarity of ionic I chose to use it over React as there was no reason to pick one over the other in the case of this project.

2.3.2.3. BootStrap

This technology can be used in combination with either of the two above frameworks discussed. This should be able to deliver the consistency needed to fulfil the requirements of the design. BootStrap is a CSS Framework to standardise a lot of the design choices made throughout the proposed application.

2.3.2.4. FireBase

Due to the current national situation, the testing of the proposed application has been limited by lockdown restrictions. To overcome this firebase can be used in conjunction with a calling service to perform usability testing with test users.

Firebase is a webhosting platform. It is specially designed to host and serve reactive applications via the web. It also provides database functionality which will be needed for backend functionality.

2.3.3. Back-End

This section will discuss all technologies which will be relevant to the development to any of the back-end systems.

2.3.3.1. Oracle

Oracle is a major relational database system and is generally used in enterprise production. It also charges licencing fees to use the technology. Oracle offers very flexible cloud solutions with support for many platforms which may prove to be useful as the proposed system could be based on multiple platforms.

One of Oracles products however, Apex, provides free unlimited use. The Apex platform provides web support also. And has capabilities to be built into customer facing Apps.

2.3.3.2. SQL lite

SQL Lite is another option available when considering data storage for the proposed application. It can provide a lightweight database solution on the device in question. It supports the storage of json files which will be used when receiving the routing information from the server.

2.3.3.3. Sequelize

Sequelize is an ORM (Object-relational mapping tool), this type of technology would be useful in integrating a SQL DB with one of previously proposed technologies ionic.

2.3.3.4. Python

When making the choice about what language to build the proposed recommendation system in there are several choices available: Python being one of the more popular ones. Pythons ability to test code rapidly will allow it to integrate into agile software mythologies. It is an open-source language and has a large set of community supported libraries to help development in many ways. Python is very well suited to handling many different types of data.

2.3.3.5. Django

Django is a Python Web framework. If python is chosen as the language to implement the recommender system Django will be key to integrating it with web applications.

2.4. Other Research

This section consists of any other relevant research that does not fall into the other sections above.

2.4.1. Mobile App Environment

When choosing between the options available for mobile app development there are a number of features that need to be considered. In this section those features along with choices made will be discussed

Native or hybrid

Native apps are apps which are written specifically for one operating system. This could include iOS or Android. Native apps tend to be simpler and more straightforward due to the fact that they need to cater for less variables. A native app has more direct access to a device’s hardware and firmware. This results in more efficient code and faster applications.

Developing in native can also have its drawbacks, namely restricting the userbase to that operating system. It also restricts the developer to staying on that operating system into the future as a switch to another OS will incur lots of work in redevelopment. Certain functionality may not be easily transferred over to another OS.

Developing in hybrid tends to be built upon web technologies. Ranging from JavaScript, typescript, html and CSS. The main advantage to doing this is the ability to port the application to almost any device or OS. Development in these languages tend to be faster that in a native app.

One of the main drawbacks to developing in this way would be the performance of the app

2.4.2. Recommender systems

The recommender systems function is to provide relevant items to a user. This can be in many different forms. There are two major methods in the approach to building recommender systems: Collaborative and Content based systems.

Collaborative Based Recommender Systems

This type of system has a range of methods that are reliant on the users past interaction with the system to produce new recommendations. This can be further subdivided this section of collaborative approaches into two main classes, memory and model approaches.
The memory approach can be boiled down to a nearest neighbour approximation. The system tries to find the most similar things based on what the user has chosen in the past. This can be split into two more approaches within the memory approach, user to user and item to item. The user-to-user method tries to find similar users are recommend those users items which are popular among the group. The item-to-item approach tries to find items which have interacted with in similar ways to other items and when it has determined that they are similar it will then recommend that item to users who like the first one.
The model approach as it suggests creates a model off which to generate prediction of what the user might like based on their past selections.
Collaborative systems have the advantage of not needing any information on the user or data that they are working with. But they do benefit from the continued use of a system in ways of becoming more accurate to the tastes of the user with use of the system. By their design these systems can start out being ineffective. While they are able to spit out recommendations without the training of the systems, they can be locked into bad recommendations by having never received a good reaction and hence not being able to generate one for the user. This is often overcome by simply adding a random element to the recommendations in order to introduce variety to the system.

Content Based Recommender Systems

This type of systematic approach replies on various additional information about both the user and the items in question. The type of information used is the categorisation of the users and the items into groups. For example, it may consider the age, the nationality and gender to determine what sort of movies a user would like. While it might compare these traits to the age rating of the movie, what studio made the film and the genre of the film in order to generate the recommendation.
The general approach to making content-based recommender systems is to build a model using all the information gathered. Grouping users together and recommending them similar items to their peers. In opposition to the collaborative system, content-based systems tend not to have the same problems outlined in their starting phase as the system is not based solely on one users' interaction.
In a similar way to the memory based Collaborative recommender system there are two ways of approaching the content-based system. One being focused on the user and the other being focused on the items.
Item- centred Bayesian classifiers are the first method that will be discussed. The idea being that item features are taken in and the result is whether the user should like or dislike the item. This is them used to create a ratio between the likely hood of the user liking the item and if the user will dislike the item.
User-centred Linear regression item features as their inputs and output a rating for that item. This is done for each user. The ratings are put into a matrix along with the features of those items. And then needs to be solved through the user of an optimisation Equation.

2.5. Existing Final Year Projects

Project 1
Title: Travel Assistant

Student: Cillian McCabe

Description (brief):

This projects aim was to provide improved useful travel information to navigational apps by means of collecting personal data through any sort of mobile or fitness device. The data would then be processed through a machine learning algorithm to provide more accurate time estimations for that specific user when navigating in the future.

Complexities: Extracting enough useful data from the user to provide accurate results from a machine learning algorithm. This also forms a challenge in how to organize the collected data into useful results.

Technical Architectures: Weka (machine Learning), Android, Google Fit Google Maps, Realm Database

Evaluation: This project delivers more accurate data to navigational tools and helps users save time in the real world by providing more accurate estimated times.

As it requires data for the machine learning to takes place, time must be invested in the application before any usable benefits can be seen.

Project 2
Title: Driving Instructor Assistor

Student: Baolach Morrison

Description (brief):

This project aims to provide an android app to assist driving instructors while on a lesson. It keeps track of all records and business info. As well as managing these responsibilities it provides map data for places the instructor can teach certain techniques for driving i.e. 3-point turn.

Complexities: AS this is a real-world based app this means that comprehensive testing must be implemented and would have to checked against real-world tests as opposed to another style of application. This app is to be used by an instructor who is responsible for unexperienced drivers so it must be quick to use and not distracting.

Technical Architectures: Android studio, Java, Django, Python, Django REST framework.

Evaluation: This project gives a useful application to Driving instructors by adding time saving features, namely documenting financial and business documents and by helping plan lessons by saving specific points on maps for testing driving skills.

Having to manually input these places does however mean that the database is user generated. Meaning that an instructor will need to be adding places to the DB while/after using the app.

2.6. Conclusions

From this chapter the evaluation of two similar systems through the use of Nielsen’s heuristics took place. Using those evaluations, a list of requirements for the proposed system were compiled. Knowing the requirements for the proposed system, an investigation into the possible technologies that could be used in the development of the app took place. This research went on to discuss some other fields relevant to the application, mainly recommender systems Finally a look at other similar final year projects in order to give a broader look into the area.

Knowing what technologies that could be used and having defined a list of requirements for the proposed system leads into the design chapter for this project.

Experiment Design

3.1 Introduction

In this section the Design of the system will be broken down into its stages and explained. Each of the following steps have been taken in order to properly implement a working design that will reduce and streamline the development process. From there the new iteration of the System Architectural Design with the proposed technologies from the previous chapter included will be laid out. Following on from this will be a detailed description of the proposed implementation of each of the Sections: Front-End, Back-End and APIs. Through the use of paper wireframes, prototype designs will be laid out for the Front-End GUI. Use case diagrams will be used to visualise the actions which take place/are available to the user. Dataflow Diagrams will inform how the development should pass data to and from the user and around the various system components.

3.2. Software Methodology

The methodology used throughout the project forms the processes and order they are taken in. By selecting the appropriate design methodology, we can structure our project in such a way that makes sense for our project's requirements and the circumstances in which the project is being developed. In this case for a final year project on a tight schedule.

3.2.1. Waterfall

The waterfall design methodology takes its name due to the rigid order its uses when going through steps. There are Seven steps involved in its design. The idea is that each step is done in order and once completed you would move onto the next step. This approach can be useful for fast development cycles; however, it does not allow for any flexibility in the stages. One cannot go back to design when in the testing process.

1. **Conception:** This is where the basic idea for the project is thought up along with basic conclusions around its pros and cons.

2. **Initiation:** Now a team is assembled and the objectives along with deliverables will be defined.

3. **Requirement gathering and analysis**: The feasibility of the proposal should be evaluated, and a set of requirements laid out.

4. **Design:** Now a design spec will be laid and analysed in order to see what the final product should look like and what is needed to be done in order to get to that point.

5. **Implementation/Coding:** Coding will now begin to implement all that set out in the design phase

6. **Testing:** After all coding is completed testing is performed before the delivery of the product.

7. **Maintenance:** This part is up to the relationship the customer wants to maintain after the delivery of the product.

3.2.2. Agile

The Agile programming methodology refers to a range of techniques but defines a general principal as to how they should act and perform. Rather than a long-term ridged plan such as the waterfall model Agile focuses on smaller iterative steps with deliverables being expected much sooner. This structure is repeated in a number of cycles. This type of approach allows for a lot more testing to be incorporated into the process. The customer is much more involved in this type of process as it develops.

Most agile processes follow this structure where you can go back and forth as needed

- Plan
- Design
- Develop
- Test
- Analyse
- Meet

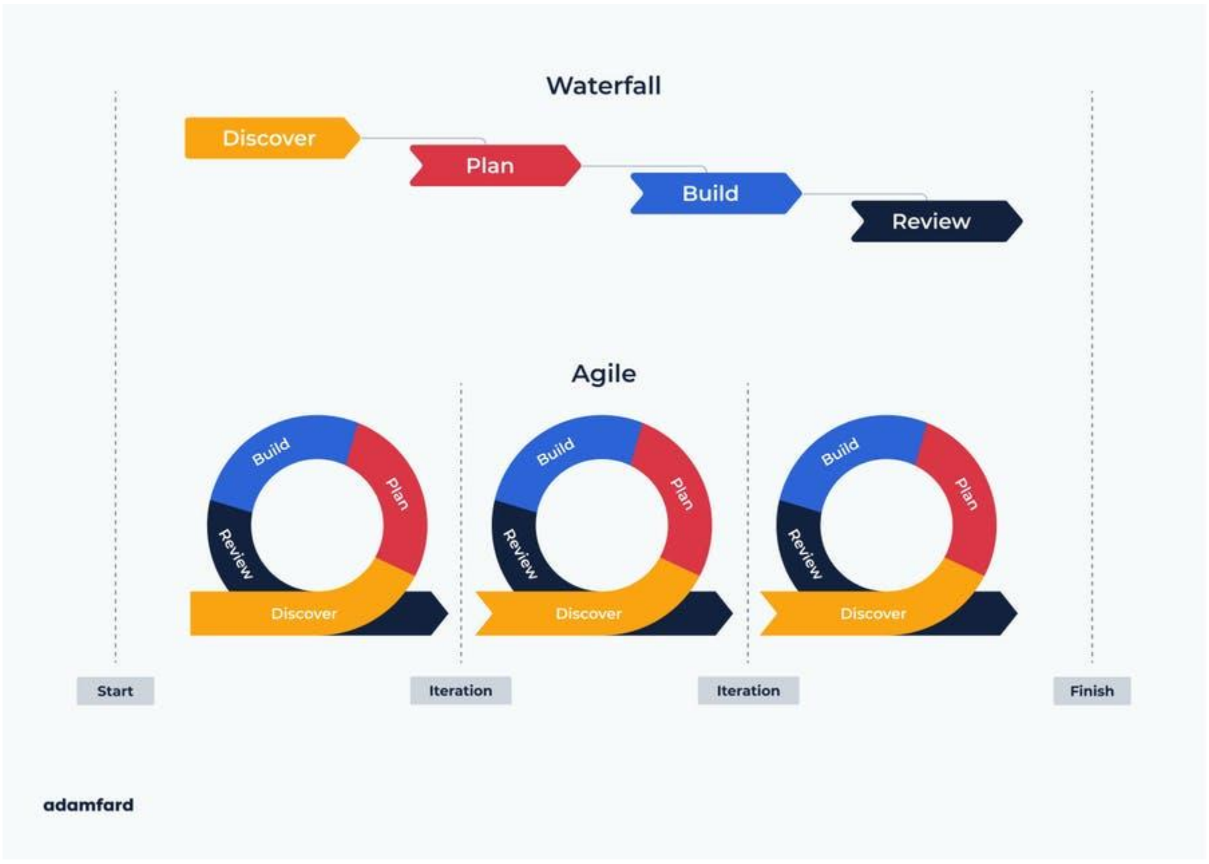


Figure 3-1 Waterfall and Agile Design methodology

3.2.3. Extreme

"XP is a lightweight methodology for small to medium sized teams developing software in the face of vague or rapidly changing requirements" -Kent Beck

Extreme programming puts a lot of focus on customer satisfaction. XP is a time saving methodology so would fit projects with a tight time frame. And relies on Five core values:

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Its cycles consist of the following steps

- Plan
- Design
- Code
- Test
- Deliver/increment

One problem that arises when considering XP is the pair-programming concept as there will not be two people to work on the code, this will not be possible.

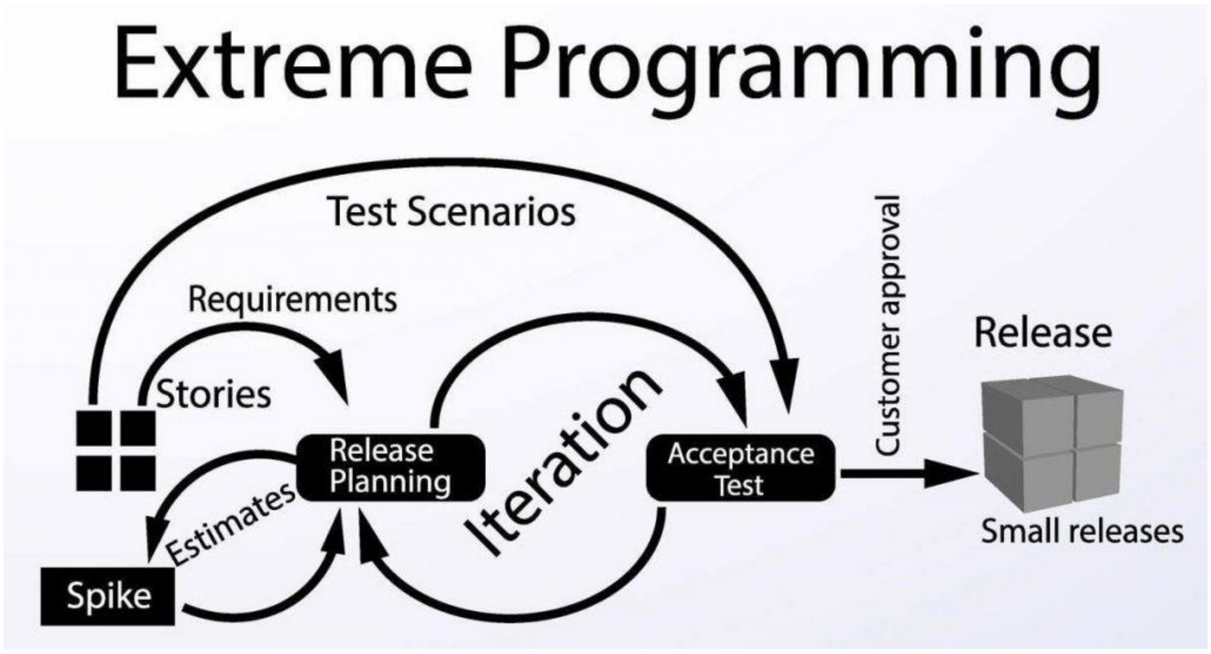


Figure 3-2 Extreme Programming

3.2.4. SCRUM

This methodology puts a lot of its focus on teamwork. Within scrum time is segmented into sprints. This is where a team of people are tasked with completing an allocated workload in a specific timeframe. Scrum is designed to react to unknown challenges during development.

Roles include Product Owner, Scrum Master and the development team. The Owner is there to provide updates requirements. The master manages the team and communicated to the Owner and the Team handles all actual development.

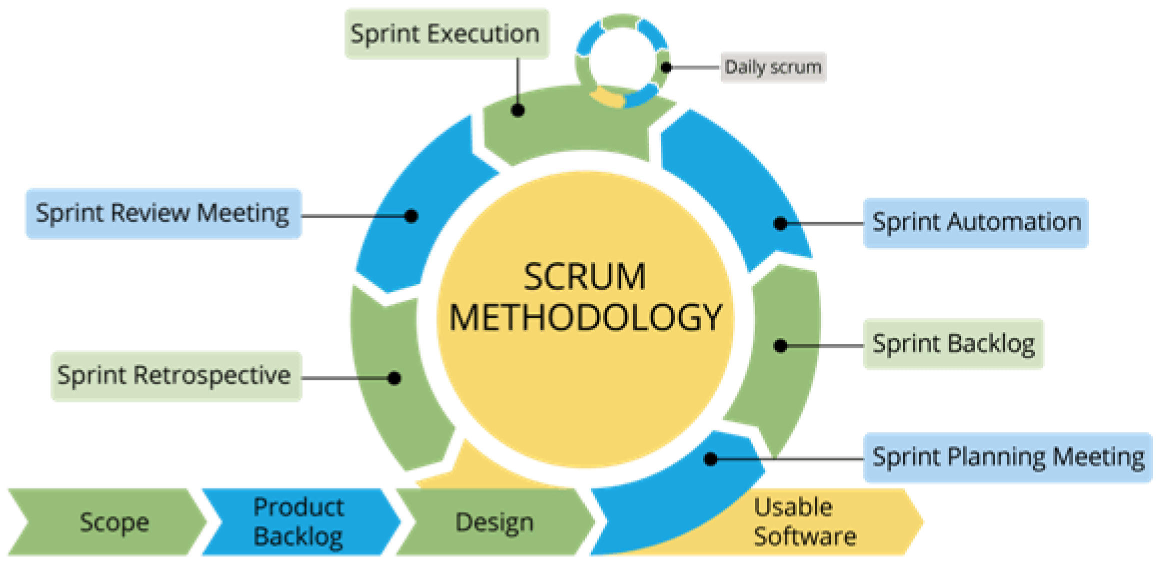


Figure 3-3 Scrum Design Methodology

Due to the limitations on testing and lack of iterative deliverables the Waterfall model was not chosen for this project. When comparing Scrum to XP, XP's shorter iterations with smaller deliverables are be more suitable to the timeframe as it promotes faster and more manageable workloads. Scrum places more focus on a team-based approach while it would be much easier to adapt XP to suit the needs of this project.

3.3. Overview of System

In this section the structure of the proposed application will be discussed. It will be split into three amins parts oi order to make a logical separation between each of the major parts going into the proposed application. Them being Front-end, API's and Back-end.

3.4. Front-End

Including screen prototypes and Use Cases

Below are the Initial paper Wireframes used in the development of the proposed Application. By using paper prototypes in combination with a html prototyping system, the process of iterating designs was made much quicker allowing for more changes to be made quickly after testing stages are completed. As seen in fig. 3-5 a sidebar is used to navigate around the main pages of the proposed application. This consists of five main sections

- Map
- Trip
- Profile

- Help
- Recents

By structuring the proposed application in this way, the user has a simple easy to use way of navigating through the app. From here the Simple Html Prototype can be created in order to facilitate the online testing necessary for the current climate.

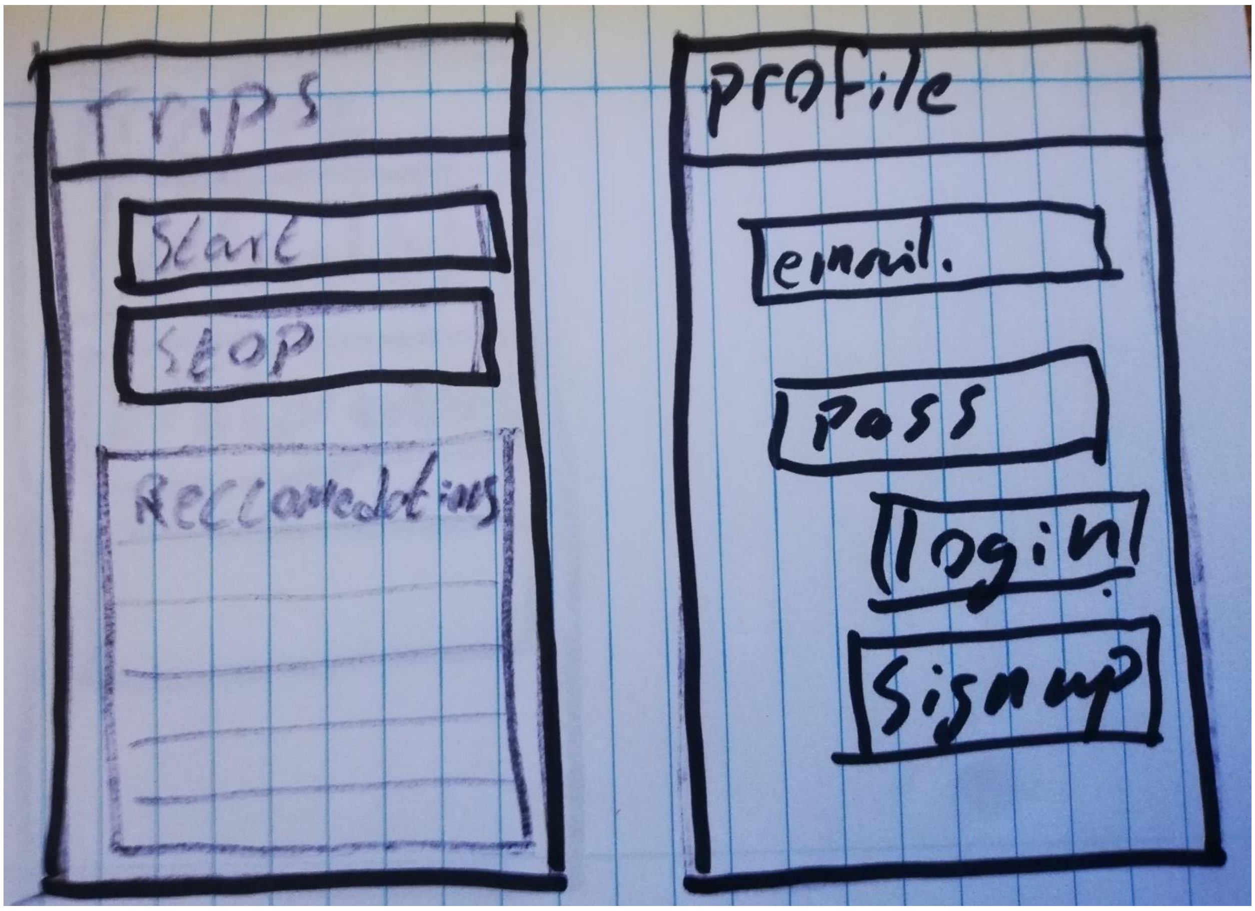


Figure 3-4 Trip and Profile login screens

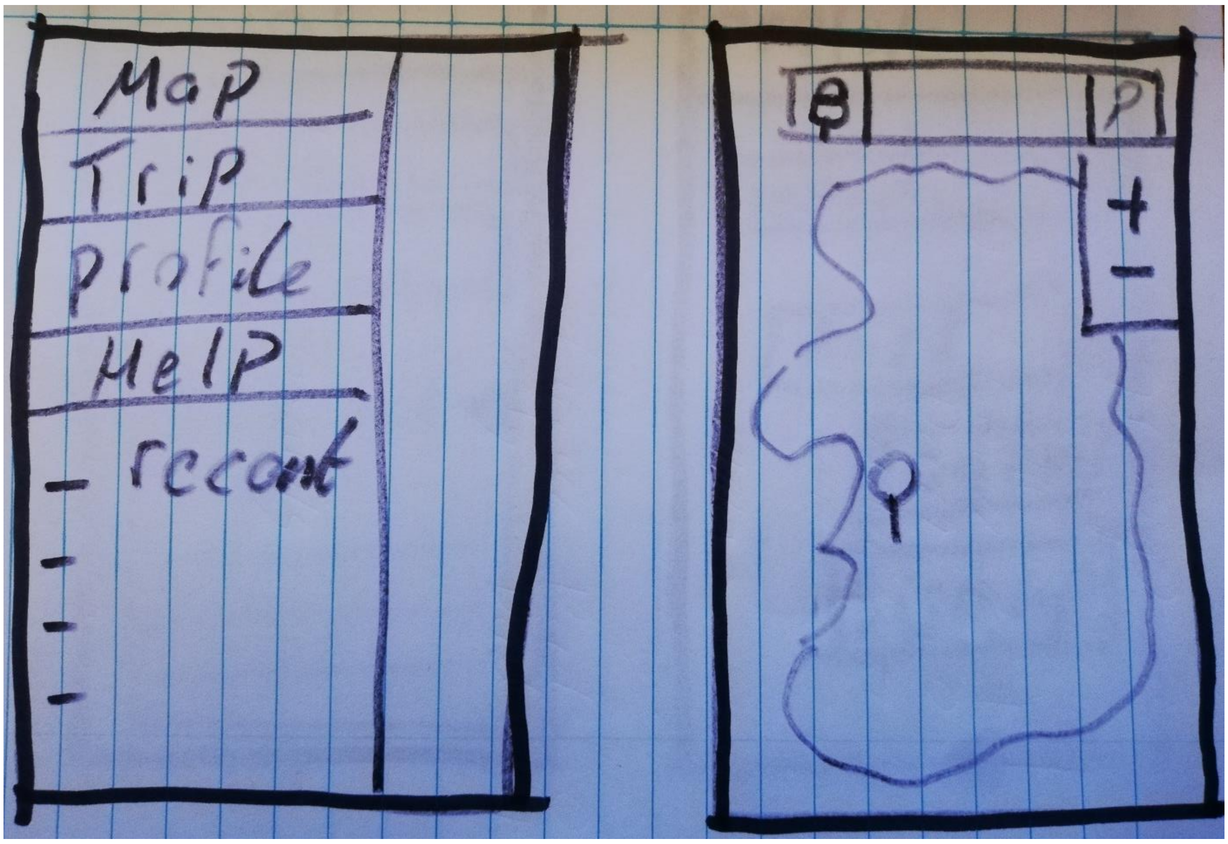


Figure 3-5 Menu and Map screens

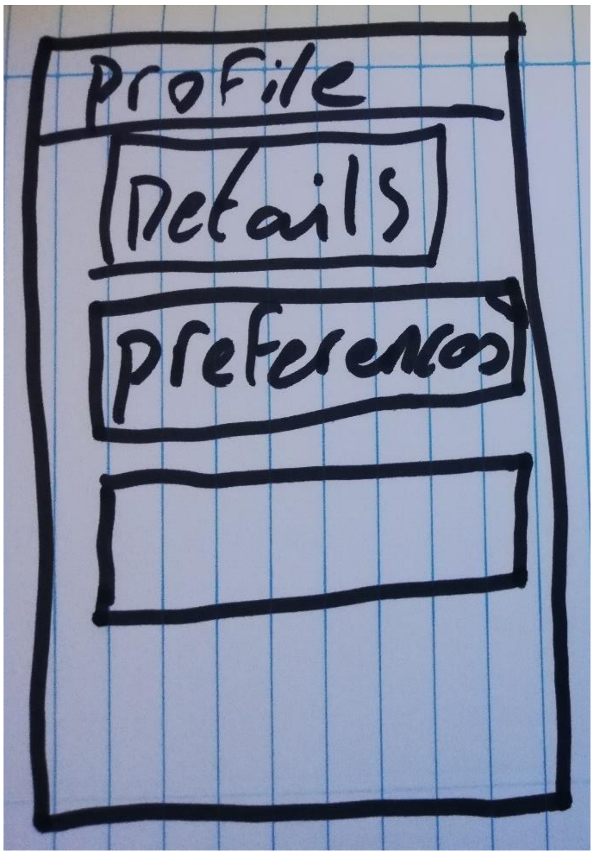


Figure 3-6 Profile Screen

3.4.1. Use-Case Diagrams

The Following Use-Case Diagrams illustrate the options available to the user and how it is intended for them to use the proposed application.

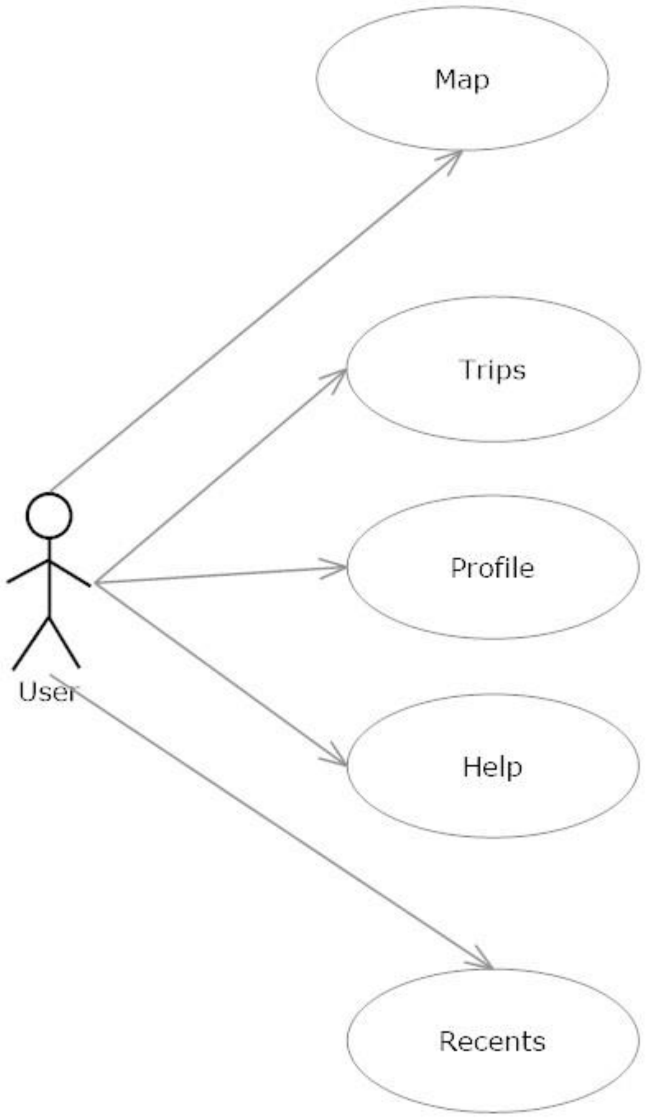


Figure 3-7 Use-case Menu Screen

Figure 3-8 Describes the Menu functions which allows the User to navigate through the proposed application. This menu should be a sidebar that pulls out while the map will shown on the main

page.

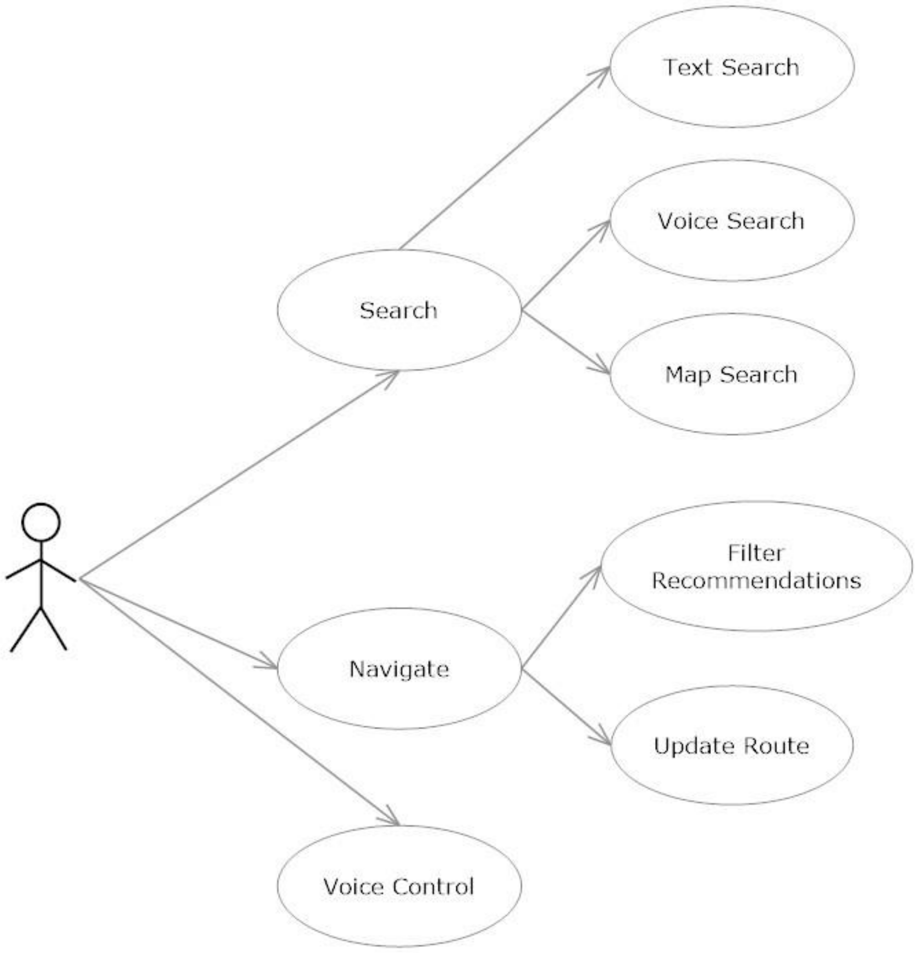


Figure 3-8 Use-case Map screen

Figure 3-9 Illustrates the options available to the user when using the map section of the proposed application. The user will have the ability to search using three different methods: Text, Voice and point and click map search.

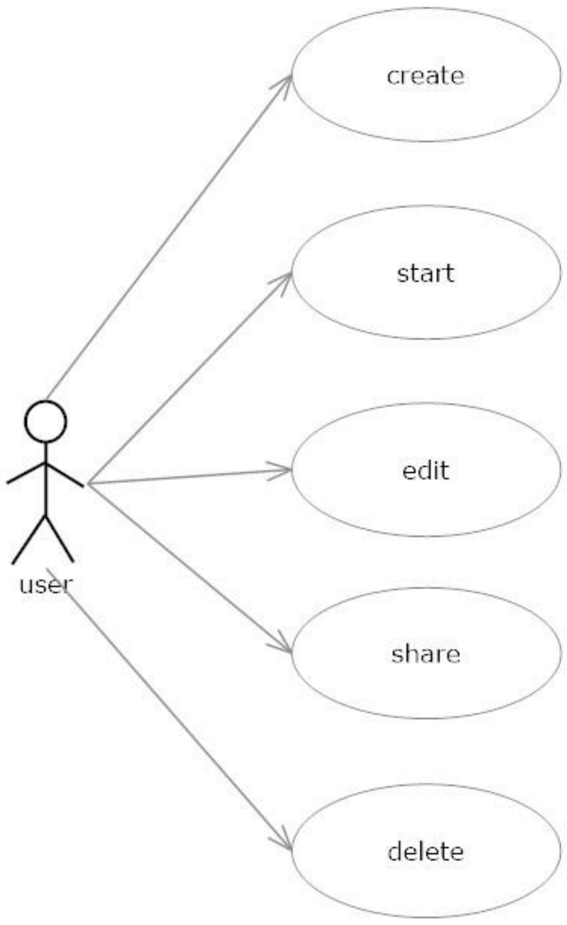


Figure 3-9 Use-case Trips Screen

Figure 3-10 above shows the users options when in the trips section of the proposed application. The user should be presented with the option to create, start, edit, share and delete the trips.

3.4.2. Horizontal Prototype

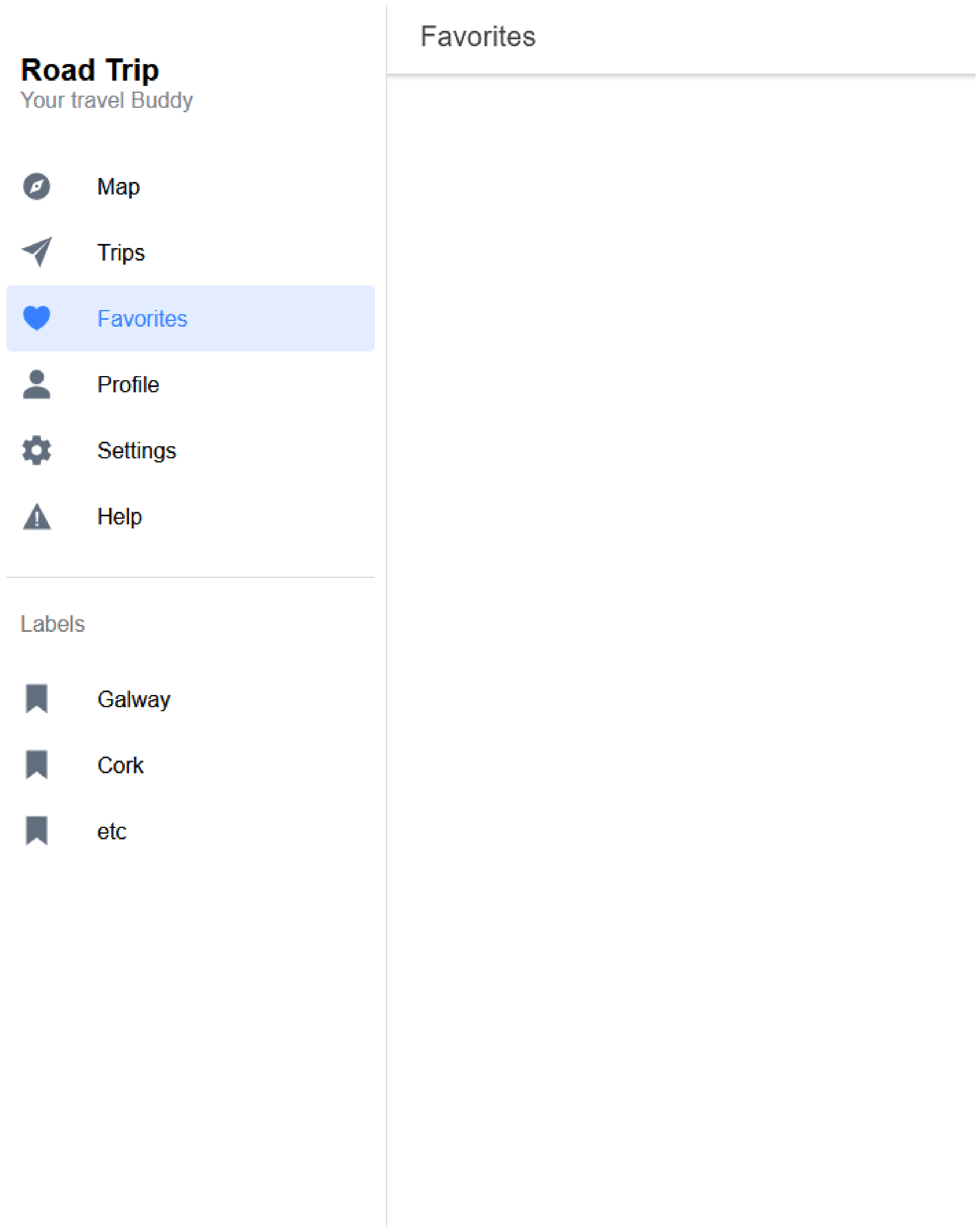


Figure 3-10 Web prototype

Figure 3-11 is the initial html prototype which will be used in user testing. By hosting a web app, the current restrictive climate can be overcome for testing of the proposed application. UI navigation and general layout of the proposed application can be fully tested. Also provides a structure to work off when it comes to the actual development stage of the proposed application.

3.5. API's

This section will outline which API's are being used, the reason for using each API and how they should interact with the rest of the Proposed application.

3.5.1. MapBox

The MapBox API will be used in order to handle all map services. This will include generating a route after the user has selected the appropriate places they wish to visit based on recommendations. Hosting and displaying the map for the user to see and interact with. Loading the feature points onto the map after being loaded from the database.

3.5.2. Django

Django will be used in order to host the python-based recommender system.

3.6. Back-End

In this section the Backend portion of the proposed application will be discussed. The back-end portion has two main parts that make it up. The first being the Map data server which provides Open Street Map (OSM) data to the front-end for display and also to the other portion of the back end the Django-web server. The Second part the Django web server will host a python-based recommender system.

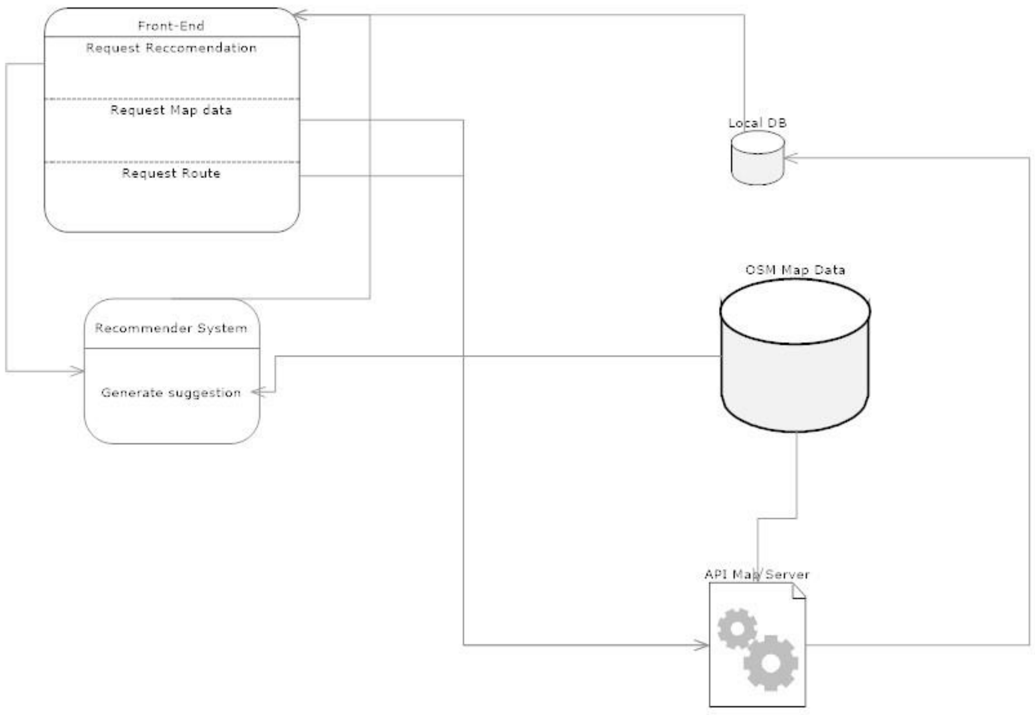


Figure 3-11 Data flow

Above in figure 3-12 we can see the initial structure of the flow of data through the proposed system architecture. It illustrates three examples of requests that would come out of the front-end application. Note the Database may be one database rather than two separate ones.

1. Request recommendation
   The recommender system will receive location data from the frontend system and request

relevant OSM data points for processing. With this data and the users continued interaction recommendations will be served.

2. Request Map data
   Here the Front-end will be served a Map generated by the API in question. The API will request OSM data from the OSM DB in order to populate the map with point of interest.
3. Request Route
   When the User has created the trip, ie aggregated points of interest, the front-end will send these to Map API. A route will then be calculated and served to the front-end through another portion of the Map API for Real-time Navigation.

3.6.1. Recommender system

The recommender system will be written python. This is due to the nature of python and how it can work well with data. It will load its data from the SQL database. Using both a combination of Term Frequency-Inverse Document Frequency and a cosine similarity a list of recommendations will be generated.

Due to the nature of the data being used a content-based recommender system will be used. This is useful in so far that it greatly reduces the cold start problem that a collaborative system would suffer from. This is especially noteworthy as the size of the data in question is very large. It would not be uncommon for items to have few to no interactions. Which would be the main reason for cold start problem in a collaborative system. Instead using a content-based system relies on processing metadata in relation to the item.

For the content-based system a series of specific questions can be asked to the user in order to prevent the cold start issue.

Firstly, the source of data needs to be structured in such a way that it is easily workable. For now, we will assume we have a table with titles and a description of that item. By tackling the problem with the use of a description we are looking at it in terms of natural language processing.

To do the similarity of each item in relation to all other items will need to be generated. As this will lead to exponentially bigger matrices before the matrix is created a simple filtration must be applied to the data in order to only include relevant data. This can be done in the form of filtering based on the geolocation data to only include items within a certain range of the desired route the user wishes to take.

To generate the similarity matrix between all the items TF-IDF (Term Frequency-Inverse Document Frequency) vectors will be used in combination with a similarity metric equation. This method will produce a score for each relationship between all items. The score is calculated based on two factors. Each word is given a frequency of occurrence throughout all item descriptions and a score based on the frequency. Then each relationship adds up the scores of the words in common. Thus, giving the more uniquely identifying words a larger influence on the final score. Figure 3-15 denotes the cosine similarity equation often used in this type of situation.

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}^\mathsf{T}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \cdot \mathbf{y}^\mathsf{T}{}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{y}_i)^2}}$$

Figure 3-12 [4]

3.6.2. Database

The function of the database will be to store all of the places of interest for use by both the MapBox API to load them on the map and for the recommender system to retrieve and be able to filter based on certain criteria.

The database will consist of Four tables to facilitate the running of the proposed system. These being the login table, the user profile, Feature Reference and the map data table.

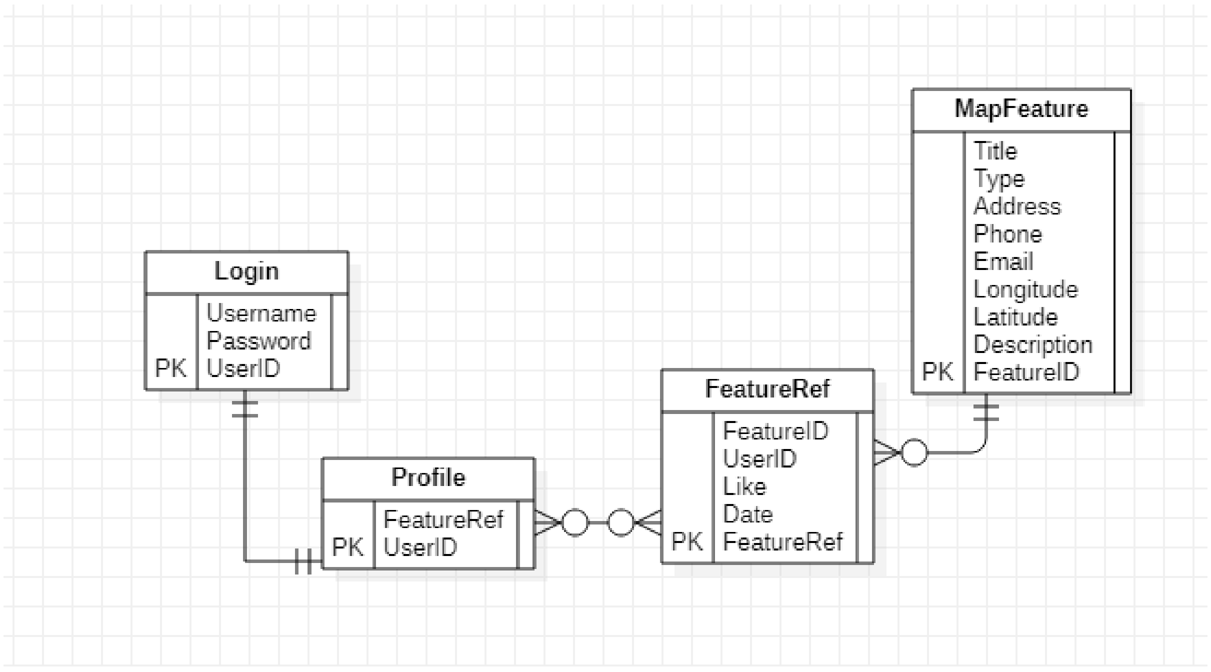


Figure 3-13 ERD

The Login Table Stores the username and password along with a UserID. The Profile Table stores all of the liked and disliked Places that the User has interacted with. This is handled in Feature Reference where the user, specific feature and date are stored. The date is important to ensure the users most recent interactions with the system have more weight when deciding on recommendations. Then we have the map features where all the places on the map are stored. This will be used by both the MapBox API and the recommender system.

3.7. Conclusions

Having chosen the Extreme Programming methodology, the Design itself should remain mostly intact the whole way through But lead onto the next chapter where each Prototype iteration made for this project will be spoken about.

Experiment Development

4.1. Introduction

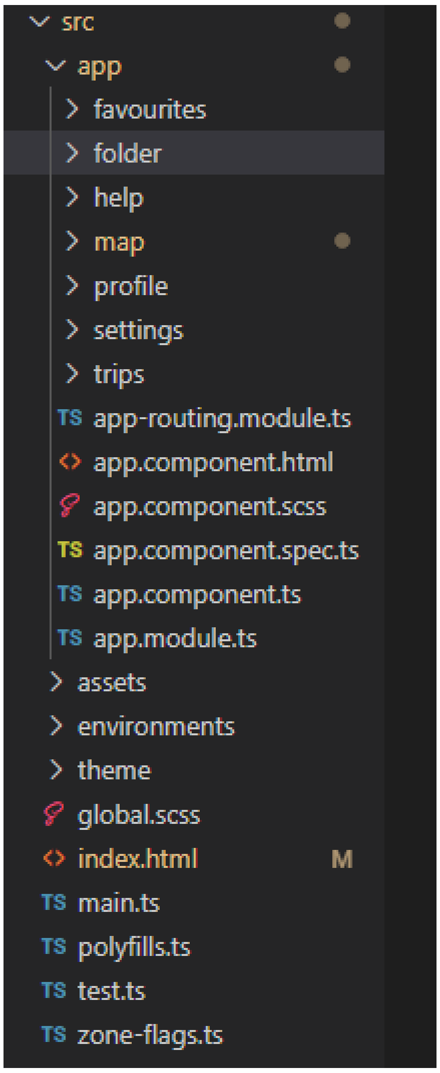
This section will speak about the processes involved in the development of the proposed application. It is split into each prototype phase. All updates to the code have been logged at this git address: https://github.com/ErikGrunner/RoadTripCompanion

4.2. Prototype One

Prototype One consists of three separate sections of the proposed application. The first being the ionic app which represents the front-end UI of the application. The second being the MapBox API implementation within the ionic app. Third, the recommender system written in python.

4.2.1. Ionic app

To build the ionic App the terminal can be used with a range of commands to progenerate components that make up ionic apps. The general structure of the app looks like this.

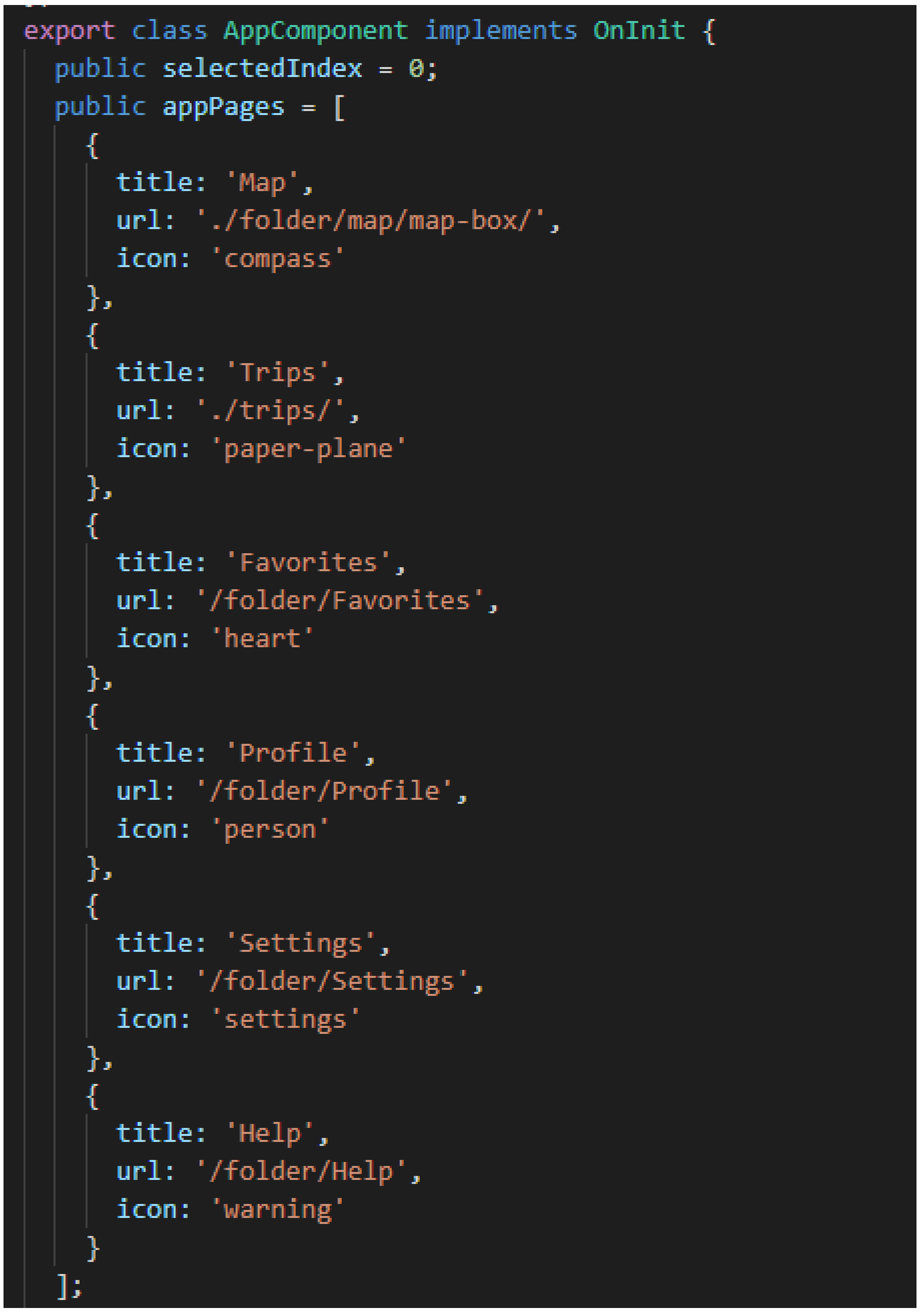


```
src
  app
    favourites
    folder
    help
    map
    profile
    settings
    trips
    app-routing.module.ts
    app.component.html
    app.component.scss
    app.component.spec.ts
    app.component.ts
    app.module.ts
  assets
  environments
  theme
  global.scss
  index.html            M
  main.ts
  polyfills.ts
  test.ts
  zone-flags.ts
```

Figure 4-1 Ionic Folder Structure

Each page folder is created with “ionic generate page -name-“

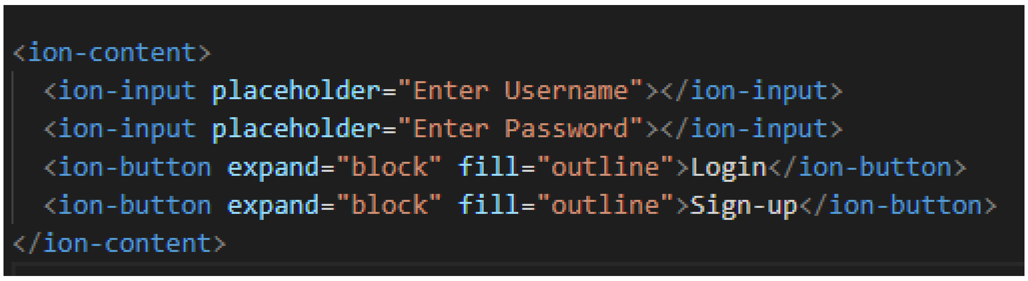
In the app.components.ts file a list of the main pages for the app will be listed with their respective URL.



```
export class AppComponent implements OnInit {
  public selectedIndex = 0;
  public appPages = [
    {
      title: 'Map',
      url: './folder/map/map-box/',
      icon: 'compass'
    },
    {
      title: 'Trips',
      url: './trips/',
      icon: 'paper-plane'
    },
    {
      title: 'Favorites',
      url: '/folder/Favorites',
      icon: 'heart'
    },
    {
      title: 'Profile',
      url: '/folder/Profile',
      icon: 'person'
    },
    {
      title: 'Settings',
      url: '/folder/Settings',
      icon: 'settings'
    },
    {
      title: 'Help',
      url: '/folder/Help',
      icon: 'warning'
    }
  ];
```

Figure 4-2 app.component.ts pages

Within each page folder the HTML file needs to have ionic content added in order to create the design laid out earlier.



```
<ion-content>
  <ion-input placeholder="Enter Username"></ion-input>
  <ion-input placeholder="Enter Password"></ion-input>
  <ion-button expand="block" fill="outline">Login</ion-button>
  <ion-button expand="block" fill="outline">Sign-up</ion-button>
</ion-content>
```

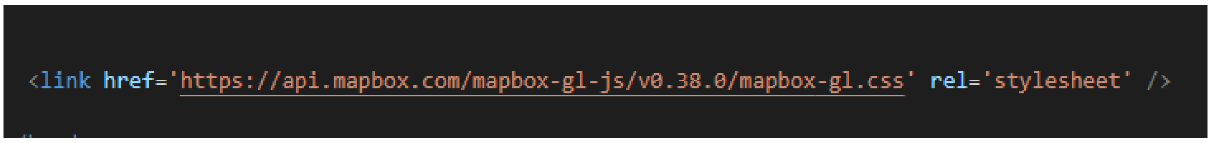
Figure 4-3 Login Screen

Above is an example of how the simple login Screen is constructed.

4.2.2. MapBox API

In order to use MapBox with the ionic app a service needs to be created within ionic. Most of this setup is not used but lays the foundation for more features to be added later on. This version only displays a simple map on screen.

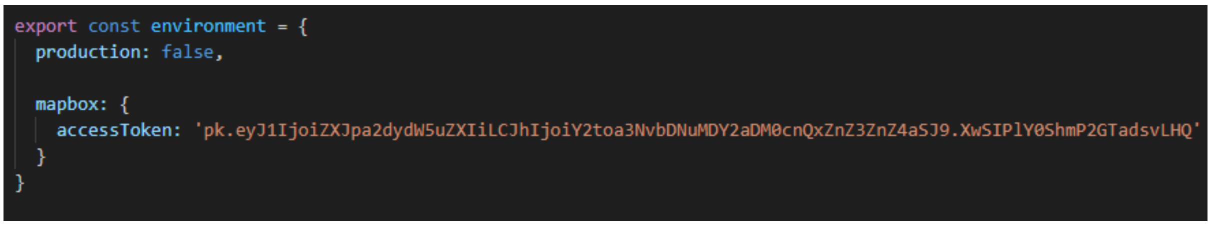
To begin with a reference to any MapBox CSS was added to the index.html file of the app.



```
<link href='https://api.mapbox.com/mapbox-gl-js/v0.38.0/mapbox-gl.css' rel='stylesheet' />
```

Figure 4-4 MapBox Style Sheet

And then in the environments file a token which is attached to a MapBox account is needed.



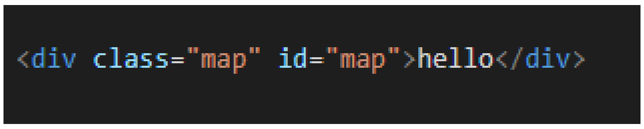
```
export const environment = {
  production: false,

  mapbox: {
    accessToken: 'pk.eyJ1IjoiZXJpa2dydW5uZXIiLCJhIjoiY2toa3NvbDNuMDY2aDM0cnQxZnZ3ZnZ4aSJ9.XwSIPlY0ShmP2GTadsvLHQ'
  }
}
```

Figure 4-5 MapBox environment

After this we can use the terminal to create a map class and a map service within that class and also a map component. The map class and the service will be used in later implementation of other MapBox features needed in the project. For now, the component is only needed for loading in a map.

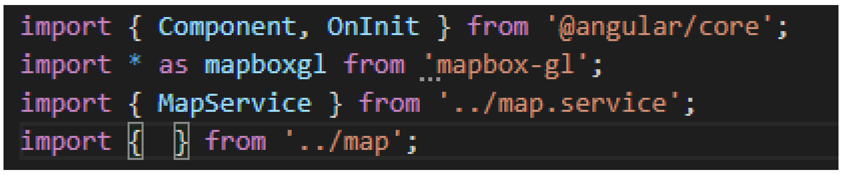
In map-box.component.html we simply need to add one line to load the map. More will be added later for more features.



```
<div class="map" id="map">hello</div>
```

Figure 4-6 Map Class Div

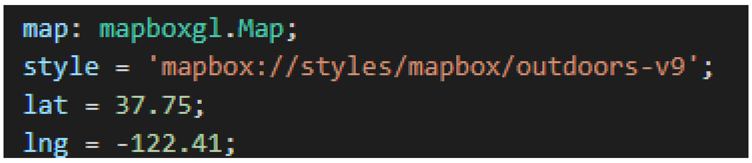
Now the map-box.component file is where the generation of the map takes place. If we add the following lines we will load the data from the MapBox server and be able to respond to simple movement inputs from the user. Starting with the required imports for the map and future layouts.



```
import { Component, OnInit } from '@angular/core';
import * as mapboxgl from 'mapbox-gl';
import { MapService } from '../map.service';
import {  } from '../map';
```

Figure 4-7 MapBox Component imports

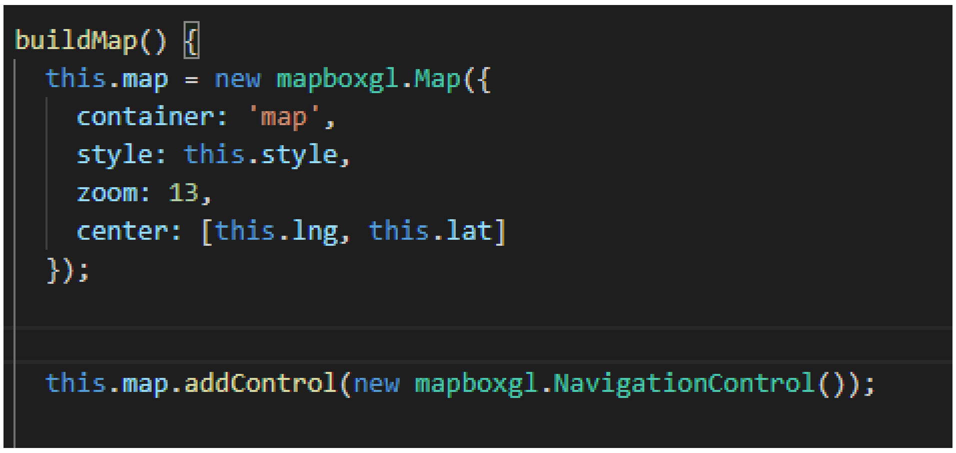
Then we can setup some defaults for the map to use.



```
map: mapboxgl.Map;
style = 'mapbox://styles/mapbox/outdoors-v9';
lat = 37.75;
lng = -122.41;
```

Figure 4-8 Default settings for MAp

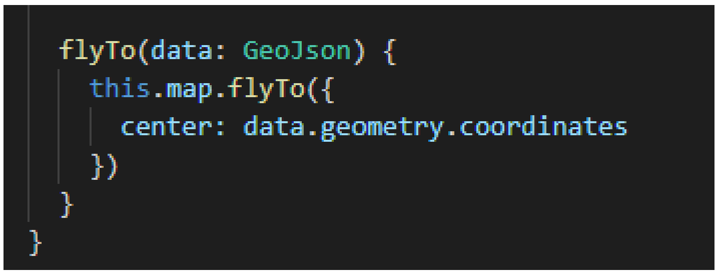
Then we need to create the buildMap function. For the creation of the map



```
buildMap() {
  this.map = new mapboxgl.Map({
    container: 'map',
    style: this.style,
    zoom: 13,
    center: [this.lng, this.lat]
  });


  this.map.addControl(new mapboxgl.NavigationControl());
```

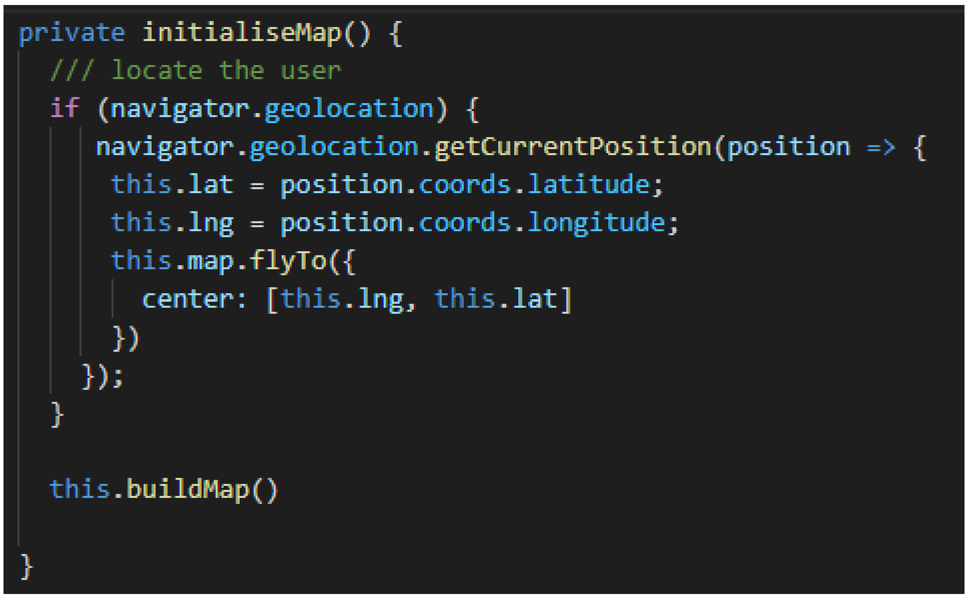
Figure 4-9 BuildMap function



```
   flyTo(data: GeoJson) {
     this.map.flyTo({
       center: data.geometry.coordinates
     })
   }
 }
```

Figure 4-10 flyTo function

This function is in order to load the map where the user is the initialise map function changes the latitude and longitude variables if they can be detected.



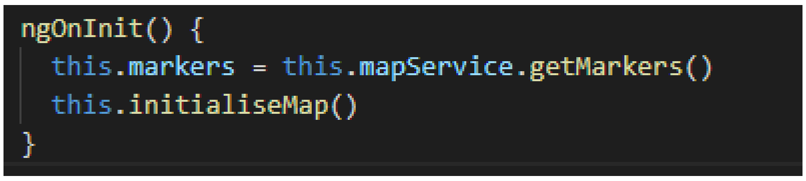
```
private initialiseMap() {
  /// locate the user
  if (navigator.geolocation) {
     navigator.geolocation.getCurrentPosition(position => {
      this.lat = position.coords.latitude;
      this.lng = position.coords.longitude;
      this.map.flyTo({
        center: [this.lng, this.lat]
      })
    });
  }

  this.buildMap()

}
```

Figure 4-11 initilaliseMap function

Finally, in order to get ionic to start this process we need to add a reference into the init function



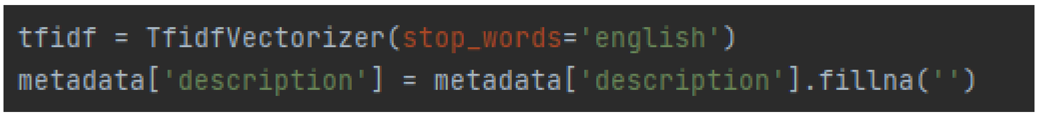
```
ngOnInit() {
  this.markers = this.mapService.getMarkers()
  this.initialiseMap()
}
```

Figure 4-12 Innit function

4.2.3. Recommender system

In the following section the design of the recommender system will be outlined

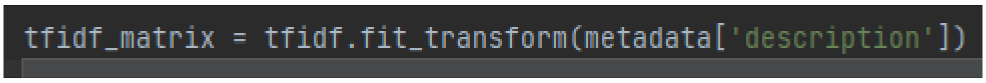
To generate the suggestions a similarity score will be generated between Items the user has liked and the other items available. This will be done based off the description of the item. Before we do this first the description of each item must be cleaned.



```
tfidf = TfidfVectorizer(stop_words='english')
metadata['description'] = metadata['description'].fillna('')
```

Figure 4-13 tfidf

Figure 3-13 shows the removal of subordinating conjugations such as: if, as, and, before. Secondly any empty descriptors are replaced with an empty string in order to avoid faults later on in the process.

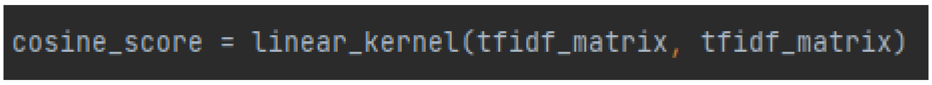


```
tfidf_matrix = tfidf.fit_transform(metadata['description'])
```

Figure 4-14 tfidf description transform

Then we will be creating a TF-IDF (Term Frequency-Inverse Document Frequency) Vector for each description. This will produce a matrix with the columns being each unique word that showed in throughout all the descriptions. Then from this we can generate TF-IDF scores. These are constructed so that overly common words are weighted less than more unique words, as they would be less useful to establishing similarity by being so common.

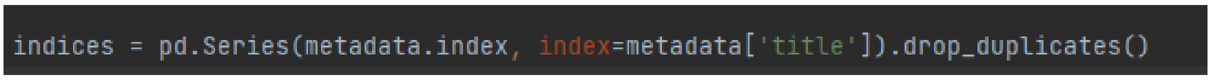
Next using the previously generated matrix we can move on to create similarity scores between items. To do this the above equation is used in figure 3-15. But can easily be achieved with the following code in figure 3-16.



```
cosine_score = linear_kernel(tfidf_matrix, tfidf_matrix)
```

Figure 4-15 cosine implementation

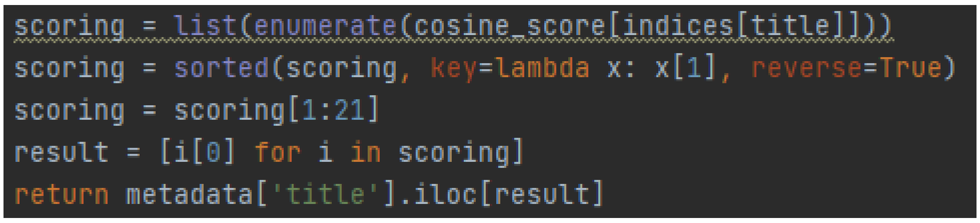
Finally, in order to make all of this usable a reverse map of the item titles with indices will be created while also dealing with any duplicated that may occur in the data. In figure 3-17.



```
indices = pd.Series(metadata.index, index=metadata['title']).drop_duplicates()
```

Figure 4-16 createinh indices

Now that all the similarity scores have been computed between each item we can take any item that the user has expressed interest in previously and generate a list of other items to show them. This is done by extracting the relevant column from the matrix and sorting the list, then the top results are returned to use as the most similar items. AS shown in figure 3-18



```
scoring = list(enumerate(cosine_score[indices[title]]))
scoring = sorted(scoring, key=lambda x: x[1], reverse=True)
scoring = scoring[1:21]
result = [i[0] for i in scoring]
return metadata['title'].iloc[result]
```

Figure 4-17 sorting results for recommendations

4.3. Conclusions

The prototyping approach to the development of the proposed system has allowed for a simple prototype as a proof of concept to be created and for the application to become iteratively more feature rich. This approach ensures that core components are developed properly and only after testing allows for the addition of other features.

Testing and Evaluation

5.1. Introduction

In this Section the various methods of testing that relate to different parts of the Proposed application will be discussed. This will vary from general overviews of how Whitebox or Blackbox testing will be utilised to more specific software tools that would be more appropriate for very specific test cases.

5.2. System Testing Methodology

5.2.1. Black Box Testing

This type of testing like the name infers refers to a method of testing where the underlying code or design is not looked at but only what would be presented to a potential user is used in the evaluation of the system. By doing testing in this way a designer can gain insight in the behaviour of a user when faced with the system.

Scenarios would be created and then the user would be asked to perform a task on the system. Through this type of testing the intuitiveness and clarity of the system is tested. While the user is doing this an observer may be sitting near them trying to understand what parts of the system does and does not understand. Certain variants of this technique include eye tracking software to determine where the user is looking at all times during a given test or scenario.

There are two subsets to this type of testing. whether they are testing a functional aspect of the system or not. A functional test is if the system can perform a task. A non-functional test is looking at how well the system performs overall.

Several techniques exist in order to achieve a systematic way of choosing test cases, saving time in testing and ensuring good test coverage.

- Equivalence partitioning
  Divide test conditions into groups and from each group test only one condition. It assumes that all the conditions work in the same way. If that condition from a group works, it is assuming all conditions from that group work. Good test coverage.
- Boundary value analysis
  Using this technique test conditions are used to create partitions. Test cases are designed by getting the boundary values of the partition. Test conditions either side of the boundary are boundary values. If both values are valid, we have valid partitions if not the partitions are invalid. This reduces testing.
- Decision table
  Using this technique we take all conditions as inputs and actions as outputs. This is used when analysing logical relationships between inputs and outputs. This is used when dealing with multiple inputs. Listing all possible inputs in combination ensures that every combination has been tested.
- State transition
  This is used when testing the transitions of a system. !!!write more
- Exploratory testing
  This is a method of testing which is just the exploration of a system without an y prior

knowledge of the system with the intension of finding functionality and possible errors. This can be similar to a trial-and-error approach

- Error guessing
  This technique does not have any specific design approach but uses testers with prior experience and may know common pitfall of applications designs in that area.

5.3.2. White Box Testing

This type of testing is a way of testing that allows for full internal inspection of a given system. It puts focus on the flow inf information through the given system, identifies sways to improve design and usability of the system. In opposition to its counter-part black box testing does its testing inside the given system. White Box testing tests the following areas in regard to the system:

- Internal security holes
- Broken or poorly structed paths within the code
- The flow specific information streams through the code
- The expected results or outputs from the system
- The functionality of conditional loop statements
- Testing each statement object and function individually

The actual testing process for white box testing can be broken into a simple two step model. The first being the understanding of the code, technologies, and intentions of the given unit to be tested, as the intention is to study the internal structure. The second step would be the creation of the test cases and performing the appropriate tests. Like Blackbox testing there are server techniques usually used in order to streamline the process.

- Code Coverage analysis
  This is the identification of untested part of code in the system. Then test cases are made specifically for that piece of code.
- Statement coverage
  This requires every statement within the code to be tested at least once during testing of the system.
- Branch coverage
  This requires that every possible path through the system including all conditional cases are tested.

Using the above coverage methods, it is expected that 80%+ of the code test cases should be covered, which should be enough in most cases. Although for more granular coverage the following techniques could be used

- Conditional Coverage
- Multiple condition coverage
- Path Coverage
- Function Coverage

By Using both White box testing and black box testing on the proposed application a wide area of test cases will be covered and with the iterative design choices that have been made will contribute greatly to the progression of the system development. In this next section specific tools for testing will discussed.

5.3. System Testing application

5.3.1. Test Groups

For the testing of the Front-End GUI of the proposed application black box testing will be used with a test group in order to test the system. The test group will be made up of various people based on two key attributes, age and technical proficiency. The idea being to get as even a spread as possible and taking into account if certain attributes outweigh the others. For example, if the group ends up being mostly younger participants, the fewer older participants should not be overshadowed when considering the test results.

AS the current national outlook is uncertain in regard to future lockdown measures, it will be assumed that this testing will take place in a virtual setting. This will be achieved by hosting a web application in conjunction with a screenshare for observation and call for directions.

The test itself is looking for GUI problems and or optimisations that can be made. The process will consist of asking the tester to perform certain tasks that would be expected from a normal user of the proposed application. Then the observer will take notes as to what the user seems to be doing while trying to perform the tasks and will be able to help also noting what questions were asked. After the tasks are completed the observer will also ask the tester a series of questions in order to gain more insight. As this is an iterative process questions may be updated from user to user when the observers deem helpful to the evaluation.

5.4. Conclusions

A combination of white box and black box testing is necessary to the proper evaluation of the proposed system as it has elements that will benefit from both. Due to the chosen XP methodology testing is being done intermittently as iterative prototypes are being produced. Thus making testing a core pillar to the development cycle rather than having a distinct testing period at the end of the development process.

Conclusions and Future Work

6.1. Introduction

In this chapter the progress made towards the proposed application will be discussed. How things were achieved, whether the right decisions were made and how the chosen methods affected the outcome of the project as a whole. Then the goals and the work involved in completing them will be discussed.

6.2. Progress

So far in the literature review chapter two similar systems have been evaluated through the use of Nielsen's Heuristics. From doing this evaluation a list of requirements was made for the proposed application. With the requirements list compiled an investigation into relevant technologies was done in order to identify appropriate candidates. This was split into front-end and back-end research. Along with this investigation other approaches to the area including a comparison between hybrid and native app development and recommender systems took place. This was important to identify what style of dev3elopment would be most appropriate. A hybrid model was chosen for the fast development time due to its web framework. For the recommender system a content-based system was chosen due to the characteristics of the data being used.

In the design chapter the system was broken up into front-end and back end sections. In the front-end section paper prototypes were used for the general outline of the system and then a HTML prototype was proposed to be used specifically for testing purposes in case of any further lockdowns. This prototype could be served via the internet and has the benefit of being used in later development. Back-end data-flow diagrams outlined how the system would pass data. An ERD outlined the structure that would be taken for the database. The methods used for the recommender system were outlined and explained. The systems integration with the rest of the system was also laid out.

In the design chapter the work done towards prototype one was discussed. This includes the ionic app front-end, the MapBox integration into the ionic app and the simple recommender system.

In the testing and evaluation chapter a general outline of what white box and black box testing has been established, along with several abstract methods that fall under each category. The user testing group method has been outlined. This is mainly needed for the UI testing of the front-end ionic system.

6.3. Future Work

As it stands the project has a very simple recommender system. This will be the main focus for further research and development over the Christmas period. While there is a basic prototype that returns results, part of more investigation will be weather a hybrid model can be created in order to take advantage of other users and possibly reduce or remove any cold start problems that may occur when beginning to use the proposed application.

The ionic app will be developed alongside other features as they are needed first in order to do any integration work. As the app needs a user group to test the UI functionality a lot of that can be don

Further work:

- Setting up the Django Back-end in order to host the recommender system needs to be done.
- Setup of the database needs to take place.
- After the database has been setup it can be integrated into both the recommender system and the ionic app.
- The testing methods need to be developed in order to test any black box testing required.
- Continuing the integration of MapBox into the ionic application
- Continued testing of the Front-end UI development. Using the test group.

Challenges

- Very Little prior knowledge to app development.
  This will be tackled by online learning course during Christmas break.
- Currently experiencing some difficulty with the MapBox integration
  This will benefit from the mentioned online ionic course.
- The use of test groups in the face of possible lockdowns
  This has been overcome by setting up a firebase to host the web application developed.
- Time Constraints
  Due to the nature of this year it will be necessary to do a substantial amount of development over the Christmas period before exams begin in the new year.

Bibliography

[1] V. D. i. Fitzgerald C, “Advice engine delivering personalized search results and customized roadtrip plans”. United States Patent 12/113,911, 9 December 2010.

[2] Google, “Google Trips,” Google, [Online]. Available: https://www.google.com/travel/. [Accessed 12 October 2020].

[3] “Roadtrippers,” Roadtrippers, [Online]. Available: https://maps.roadtrippers.co,. [Accessed 15 October 2020].

[4] wikipedia, “Cosine similarity,” [Online]. Available: https://en.wikipedia.org/wiki/Cosine_similarity.

[5] B. Rocca, “Introduction to recommender systems,” 3 June 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

[6] scikit-learn, [Online]. Available: https://scikit-learn.org/stable/.

[7] K. H. &. C. J. &. L. C. &. K. S. Lim, “Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. Knowledge and Information Systems,” 2018.