

Road Trip Companion

Final Project 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

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

[1. Introduction 11](#_Toc67159059)

[1.1. Project Background 11](#_Toc67159060)

[1.2. Project Description 11](#_Toc67159061)

[1.3. Project Aims and Objectives 12](#_Toc67159062)

[1.3.1. Smart Objectives 12](#_Toc67159063)

[1.3.2. Project Objectives 13](#_Toc67159064)

[1.4. Project Scope 13](#_Toc67159065)

[1.5. Thesis Roadmap 13](#_Toc67159066)

[2. Literature Review 14](#_Toc67159067)

[2.1. Introduction 14](#_Toc67159068)

[2.2. Alternative Existing Solutions to Your Problem 14](#_Toc67159069)

[2.2.1 Nielsen’s Heuristics 14](#_Toc67159070)

[2.2.2. Google Trips 16](#_Toc67159071)

[2.2.3. Road Trippers 18](#_Toc67159072)

[2.2.4. Heuristic Evaluation 24](#_Toc67159073)

[2.3. Technologies researched 25](#_Toc67159074)

[2.3.1 System Architect Diagram 25](#_Toc67159075)

[Front-End 26](#_Toc67159076)

[Google API & MapBox 26](#_Toc67159077)

[Ionic & React Native 26](#_Toc67159078)

[BootStrap 26](#_Toc67159079)

[FireBase 26](#_Toc67159080)

[Back-End 27](#_Toc67159081)

[Oracle 27](#_Toc67159082)

[SQL lite 27](#_Toc67159083)

[FireStore 27](#_Toc67159084)

[Sequelize 27](#_Toc67159085)

[Python 27](#_Toc67159086)

[Django 27](#_Toc67159087)

[2.4. Other Research 28](#_Toc67159088)

[2.4.1. Mobile App Environment 28](#_Toc67159089)

[2.4.2. Recommender systems 28](#_Toc67159090)

[2.4.3. Fitzgerald patent 29](#_Toc67159091)

[2.5. Existing Final Year Projects 29](#_Toc67159092)

[2.6. Conclusions 30](#_Toc67159093)

[3. Experiment Design 32](#_Toc67159094)

[3.1 Introduction 32](#_Toc67159095)

[3.2. Software Methodology 32](#_Toc67159096)

[3.2.1. Waterfall 32](#_Toc67159097)

[3.2.2. Agile 33](#_Toc67159098)

[3.2.3. Extreme 33](#_Toc67159099)

[3.2.4. SCRUM 35](#_Toc67159100)

[3.3. Overview of System 36](#_Toc67159101)

[3.4. Front-End 36](#_Toc67159102)

[3.4.1. Use-Case Diagrams 39](#_Toc67159103)

[3.4.2. Horizontal Prototype 41](#_Toc67159104)

[3.5. API’s 43](#_Toc67159105)

[3.5.1. MapBox 43](#_Toc67159106)

[3.5.2. Django 43](#_Toc67159107)

[3.6. Back-End 43](#_Toc67159108)

[3.6.1. Recommender system 44](#_Toc67159109)

[3.6.2. Database 45](#_Toc67159110)

[3.7. Conclusions 45](#_Toc67159111)

[4. Prototyping Development 47](#_Toc67159112)

[4.1. Introduction 47](#_Toc67159113)

[4.2. Prototype One 47](#_Toc67159114)

[4.2.1. Ionic app 47](#_Toc67159115)

[4.2.2. MapBox API 50](#_Toc67159116)

[4.2.3. Recommender system 52](#_Toc67159117)

[4.3 Prototype Two 53](#_Toc67159118)

[Ionic App 53](#_Toc67159119)

[MapBox API 54](#_Toc67159120)

[Firebase DB 55](#_Toc67159121)

[Recommender system 57](#_Toc67159122)

[Issues 58](#_Toc67159123)

[Summary 58](#_Toc67159124)

[4.4 Prototype Three 58](#_Toc67159125)

[Map Interface 58](#_Toc67159126)

[Recommender system 61](#_Toc67159127)

[Database 62](#_Toc67159128)

[Issues 63](#_Toc67159129)

[4.5 Prototype Four 63](#_Toc67159130)

[GUI interface 63](#_Toc67159131)

[Recommender system 64](#_Toc67159132)

[Database 65](#_Toc67159133)

[Issues 65](#_Toc67159134)

[4.6. Conclusions 66](#_Toc67159135)

[5. Testing and Evaluation 68](#_Toc67159136)

[5.1. Introduction 68](#_Toc67159137)

[5.2. System Testing Methodology 68](#_Toc67159138)

[5.2.1. Black Box Testing 68](#_Toc67159139)

[5.3.2. White Box Testing 69](#_Toc67159140)

[5.2.3. Nielsen’s Heuristics 70](#_Toc67159141)

[5.3. System Testing application 71](#_Toc67159142)

[5.3.1. Test Groups 71](#_Toc67159143)

[5.3.2. Metric testing 71](#_Toc67159144)

[5.4. Conclusions 71](#_Toc67159145)

[6. Conclusions and Future Work 73](#_Toc67159146)

[6.1. Introduction 73](#_Toc67159147)

[Project planning 73](#_Toc67159148)

[Application results 73](#_Toc67159149)

[Nielsen’s Heuristics 73](#_Toc67159150)

[6.2. Conclusions 74](#_Toc67159151)

[6.3. Future Work 74](#_Toc67159152)

[7. Bibliography 76](#_Toc67159153)

**Table of Figures**

[Figure 1-1‑1 Smart Objectives 11](#_Toc66782017)

[Figure 2‑1 Google Trips screenshot 16](#_Toc66782018)

[Figure 2‑2 Google Trips screenshot 16](#_Toc66782019)

[Figure 2‑3 Google Maps View 17](#_Toc66782020)

[Figure 2‑4 Road Trippers Home page 19](#_Toc66782021)

[Figure 2‑5 Road Trippers Place Example 20](#_Toc66782022)

[Figure 2‑6 Road Trippers Start Trip 20](#_Toc66782023)

[Figure 2‑7 Road Trippers Map 21](#_Toc66782024)

[Figure 2‑8 Results from Heuristic Evaluation 22](#_Toc66782025)

[Figure 2‑9 System Architech Diagram 23](#_Toc66782026)

[Figure 3‑1 Waterfall and Agile Design methodology 30](#_Toc66782027)

[Figure 3‑2 Extreme Programming 31](#_Toc66782028)

[Figure 3‑3 Scrum Design Methodology 32](#_Toc66782029)

[Figure 3‑4 Trip and Profile login screens 33](#_Toc66782030)

[Figure 3‑5 Menu and Map screens 34](#_Toc66782031)

[Figure 3‑6 Profile Screen 34](#_Toc66782032)

[Figure 3‑7 Use-case Menu Screen 35](#_Toc66782033)

[Figure 3‑8 Use-case Map screen 36](#_Toc66782034)

[Figure 3‑9 Use-case Trips Screen 37](#_Toc66782035)

[Figure 3‑10 Web prototype 38](#_Toc66782036)

[Figure 3‑11 Data flow 39](#_Toc66782037)

[Figure 3‑12 [13] 40](#_Toc66782038)

[Figure 3‑13 ERD 41](#_Toc66782039)

[Figure 4‑1 Ionic Folder Structure 42](#_Toc66782040)

[Figure 4‑2 app.component.ts pages 43](#_Toc66782041)

[Figure 4‑3 Login Screen 44](#_Toc66782042)

[Figure 4‑4 MapBox Style Sheet 44](#_Toc66782043)

[Figure 4‑5 MapBox environment 44](#_Toc66782044)

[Figure 4‑6 Map Class Div 44](#_Toc66782045)

[Figure 4‑7 MapBox Component imports 45](#_Toc66782046)

[Figure 4‑8 Default settings for MAp 45](#_Toc66782047)

[Figure 4‑9 BuildMap function 45](#_Toc66782048)

[Figure 4‑10 flyTo function 45](#_Toc66782049)

[Figure 4‑11 initilaliseMap function 46](#_Toc66782050)

[Figure 4‑12 Innit function 46](#_Toc66782051)

[Figure 4‑13 tfidf 46](#_Toc66782052)

[Figure 4‑14 tfidf description transform 46](#_Toc66782053)

[Figure 4‑15 cosine implementation 47](#_Toc66782054)

[Figure 4‑16 createinh indices 47](#_Toc66782055)

[Figure 4‑17 sorting results for recommendations 47](#_Toc66782056)

[Figure 4‑18 Prototype two routing example 48](#_Toc66782057)

[Figure 4‑19 Map points 49](#_Toc66782058)

[Figure 4‑20 FireStore first implementation 50](#_Toc66782059)

[Figure 4‑21 Login Method prototype two 51](#_Toc66782060)

[Figure 4‑22 Register Method prototype two 51](#_Toc66782061)

[Figure 4‑23Django requests 52](#_Toc66782062)

[Figure 4‑24 Prototype Three Navigation page 53](#_Toc66782063)

[Figure 4‑25 map view with recommendations 54](#_Toc66782064)

[Figure 4‑26 55](#_Toc66782065)

[Figure 4‑27 populate\_db function 56](#_Toc66782066)

[Figure 4‑28 Recommender training page 58](#_Toc66782067)

[Figure 4‑29 type selection on recommender system 58](#_Toc66782068)

[Figure 4‑30 Firebase Usage report 59](#_Toc66782069)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 | x | x |  |  |  |  |  |  |  |  |  |  |  |
| **Final**  **Design** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x |  |  |  |  |  |  |  |  |  |
| **Software**  **Testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | 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 | x | x |  |  |  |  |  |
| **Final**  **Demonstration** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |

# Introduction

In this section of this report the proposal for the project will be presented. 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 present 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.

## Project Background

When travelling over several years in various places one can see a lack of knowledge of the area and lack of special awareness to that new area can lead to 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. It is impossible to discover everything while traveling within time restraints which posed another challenge to the average traveler. The author has confidence that through the development of a smart companion application these challenges can be overcome and a system for both guiding the user and helping the user can be created to overcome these challenges.

In this research two similar mobile apps were identified with detailed heuristics evaluation taking place. Both were found to be lacking in certain areas, which will be discussed further. This initial research gave more reason for the need for an application of this sort.

## 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.

## 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-‑ 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   
   Meaning that the task is laid out in somewhat technical terms, not being vague enough for alternate interpretation.
2. Measurable  
   This means that the task has some form of metric that allows for the progress to be calculated.
3. Achievable  
   Based on current understanding of the skills involved is it possible for the task to be completed. Links in with point five.
4. Realistic  
   Is it reasonable to believe that the task can be completed.
5. Timebound  
   Finally to set a timescale on the task/tasks that are being set out.

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. This type of task setting is very applicable to this project as it timebound and requires a defined structure in order to meet targets.

### 1.3.2. Project Objectives

Based off the principles of the smart objectives the following list of objectives have been developed in order to divide the proposed project into smart objectives.

* Evaluate Similar applications through the use of Nielsen’s Heuristics and 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.

## Project Scope

This project aims to develop a system with the ability to create road trips based on recommendations using a collaborative approach with multiple users. It will provide a real time navigation services together with personalised recommendations based on the user and current planned destination. This will be possible by using a webapp serving as the front-end portion of the

proposed application, where the user will be able to create share and customise personal trips. The backend of the app supported by a database and the recommendation system.

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.

## 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 similar systems 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

### 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]

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.

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 trip 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.

### 2.2.2. Google Trips

Google trip is an online trip creation tool meant to be used in conjunction with Google Maps on the mobile. IT provide lots of integration with other google services and generates most of its data procedurally making almost everything available on the web integrated with it. 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. [2]

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.

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

Overall lack of clarity in regard to the trips integration to the app.

1. **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.

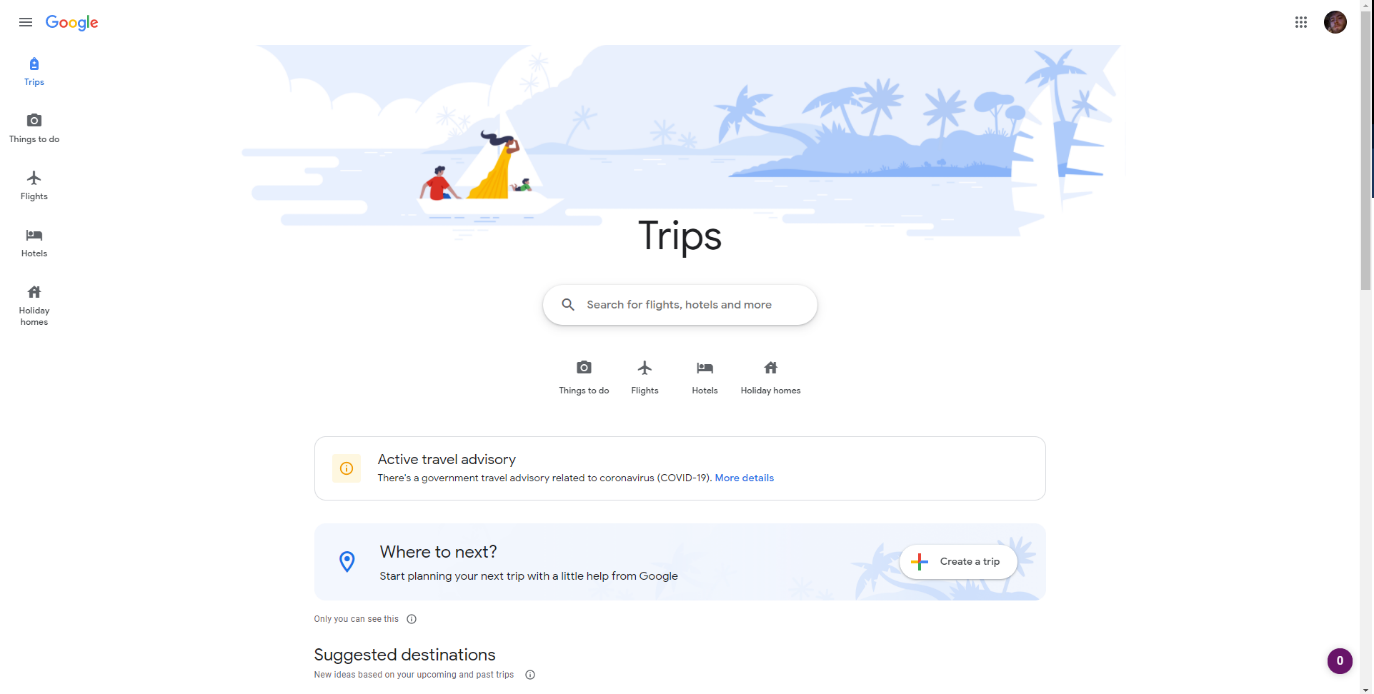


Figure ‑ Google Trips screenshot

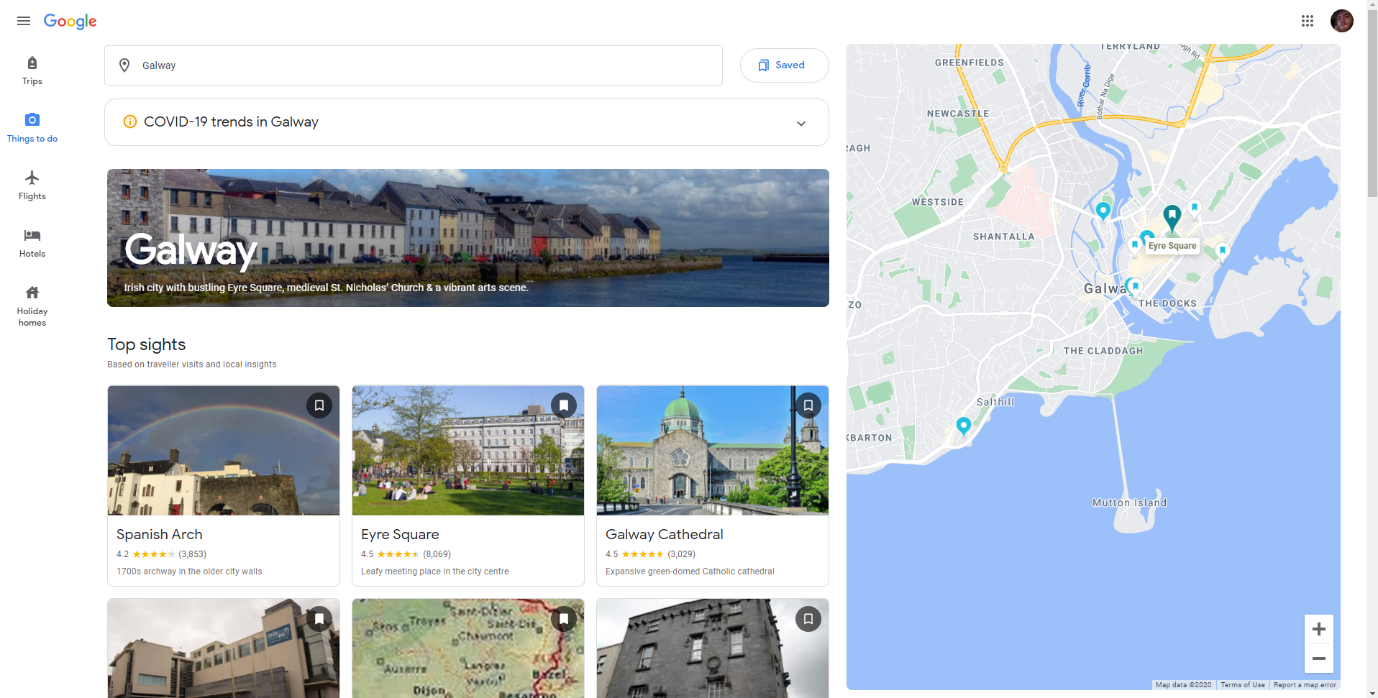


Figure ‑ Google Trips screenshot

Above in figure 2.1 and 2.2 are two of the starting pages a user would encounter when using Google Trips.

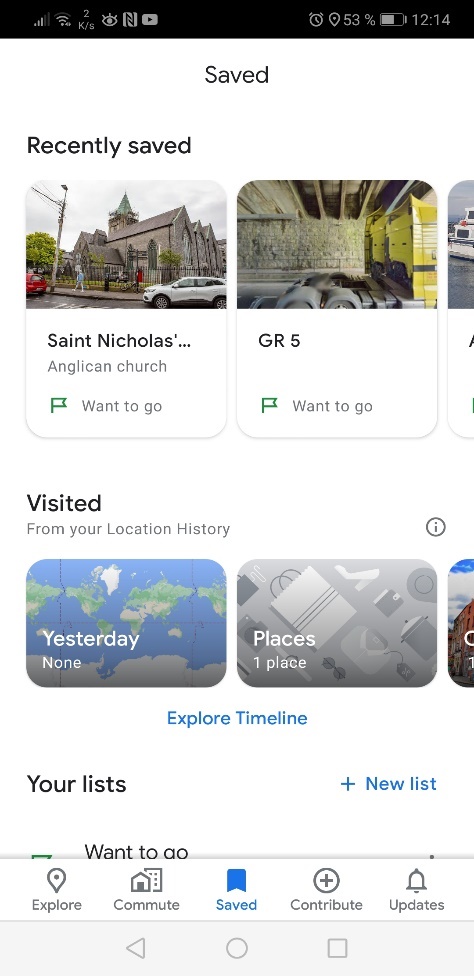


Figure ‑ Google Maps View

This third screen grab figure 2.3 is showing off the mobile view of what the user would see within the google maps app, showing what was saved as a trip on the web application.

### 2.2.3. Road Trippers

Road Trippers is both a mobile and web app designed around a road trip concept. The Company behind the app 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. [3]

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. This included detailed maps and easy controls to manipulate the map.
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. This is due to the method of creation being more suited to a web based platform.
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.

1. **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.

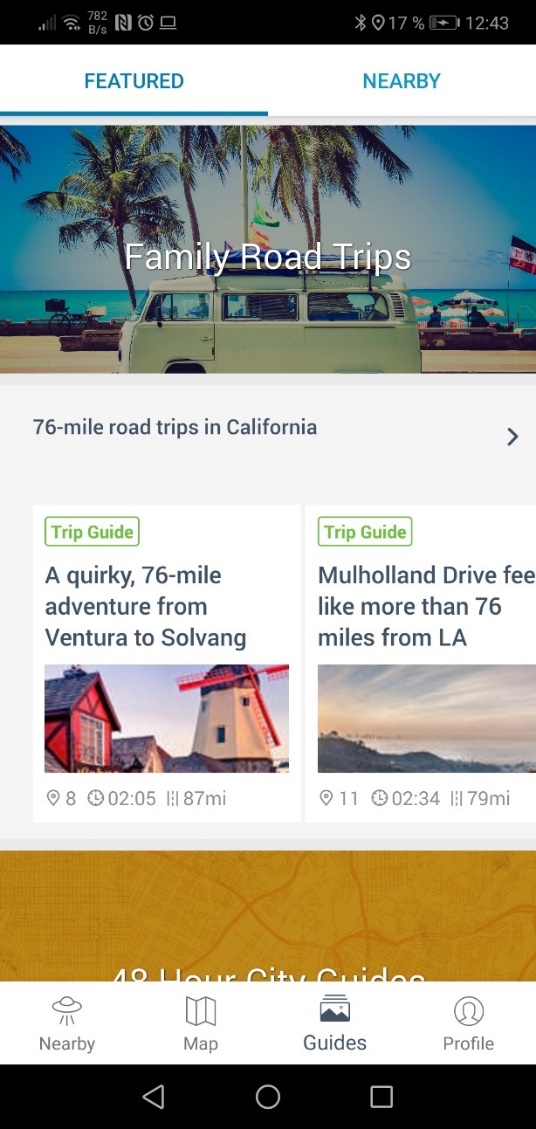


Figure ‑ Road Trippers Home page

Figure 2.4 above shows the home page of the road trippers mobile application. It can be seen that the featured curated trips are presented to the user first.

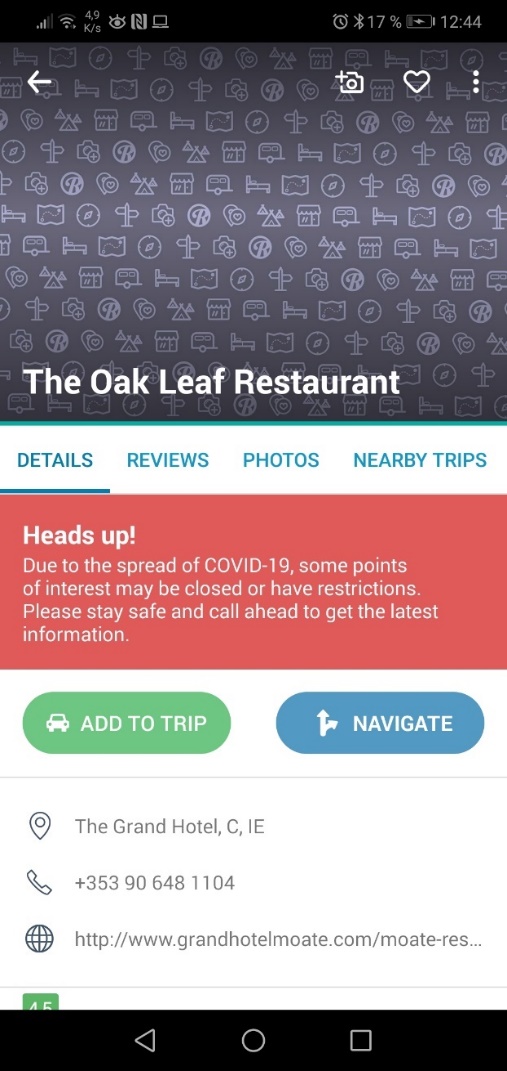


Figure ‑ Road Trippers Place Example

Figure 2.5 Shows how the app presents each map feature with relevant information to the feature and options to either navigate or include it into a trip.

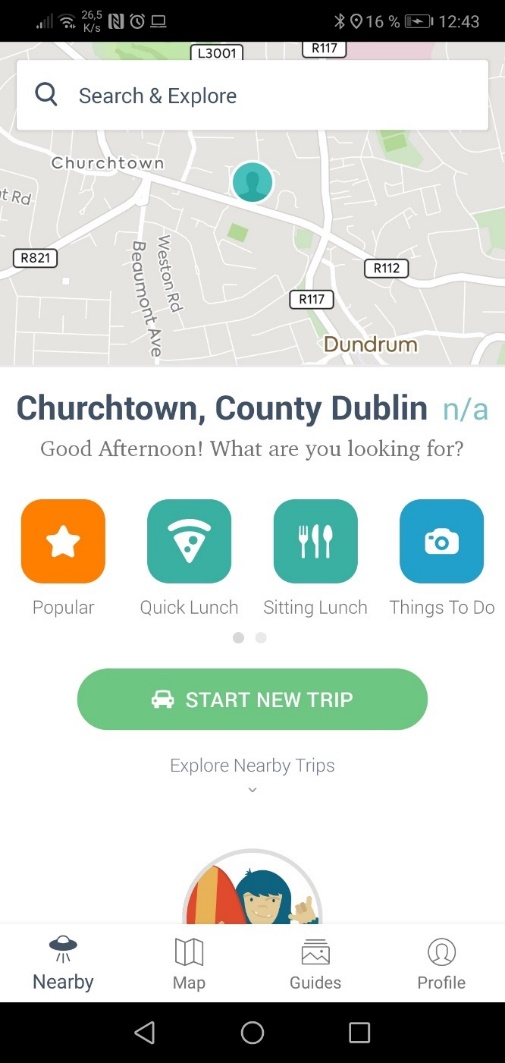


Figure ‑ Road Trippers Start Trip

Figure 2.6 shows a view when browsing the map. A popup tab prompts the user to start a trip or have a look at various subsections of features. The overall section is about discovery of the area.

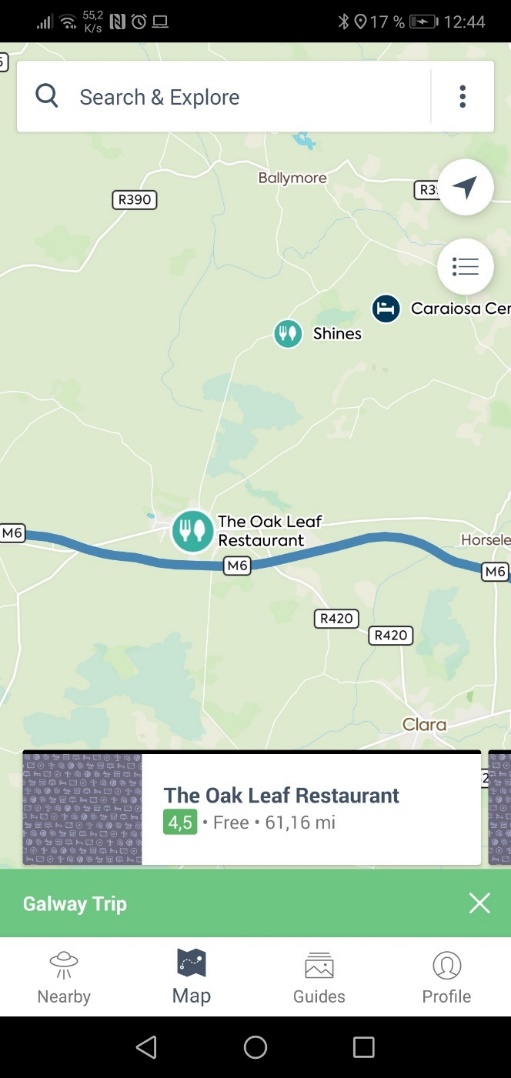


Figure ‑ Road Trippers Map

Figure 2.7 is another page of the live navigation. This view is for looking ahead in the trip and being able to select more points or to check what has already been selected. This type of page would be very useful for a passenger on a trip being able to manage the trip in more detail.

### 2.2.4. Heuristic Evaluation

Figure ‑ 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 each 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. This diagram has been designed with some idea of what types of technologies may be used in the proposed application.

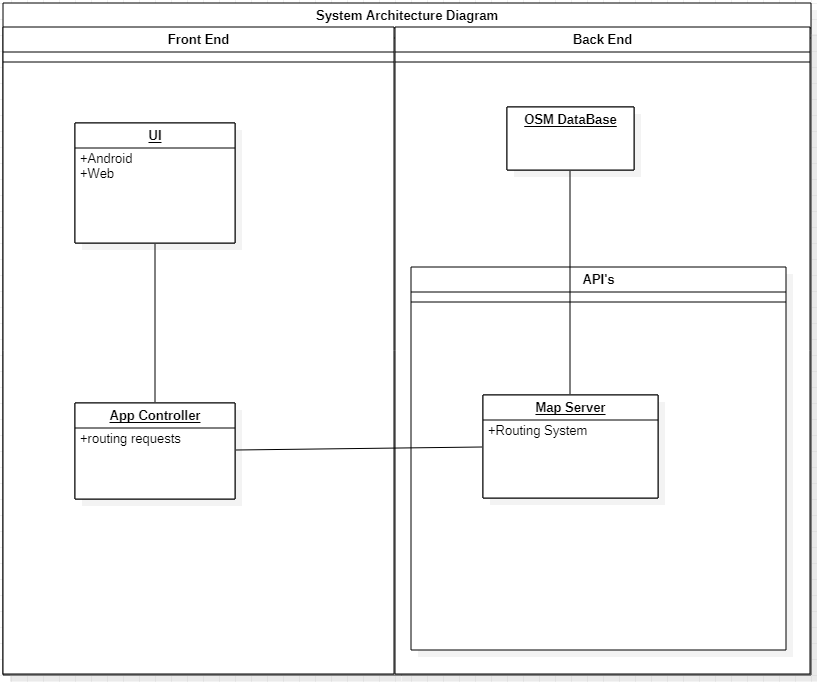


Figure ‑ System Architech Diagram

## 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. A front end section is required for the proposed application as the design specifies online recommender system and database which the front end application would be in communication with.

### 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 extensive customisation features offered from MapBox it will be used for this project. [4] [5]

### 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. Rapid expansion of the project though the generate command allows for large repetitive tasks to be taken care of. This being the generation of components pages or services. All the necessary links to the rest of the project are taken care of. Ionic also provides a modified suit of html components designed for operative design. This means the html for a webapp can dynamically reshape itself for an iPhone app or Android app.

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 with ionic it was chosen over React as there was no reason to pick one over the other in the case of this project. [6] [7]

### 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.

### 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. As well as Database functionality FireBase includes an authentication service. This means when developing no time needs to be spent on developing security as it is not within the project scope but cannot be ignored.

## Back-End

This section will discuss all technologies which will be relevant to the development to any of the back-end systems. There is need for a backend to the system in order to host the recommender system along with the database that will support that recommender system. These will then be in communication with the front-end web or mobile application.

### 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. [8]

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.

### SQL lite

SQL Lite is another option available when considering data storage for the proposed application. It can provide a lightweight SQL 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. SQLite is available as a plugin for many web frameworks including ionic. [9]

### FireStore

FireStore is the database software that comes as a part of the firebase package. It requires no use of SQL commands as it handles its input through the use of its API. It a large degree of scalability which would be useful when it comes to launching an application. As it is designed to be used with firebase, an option being considered for hosting the app, it would make integration seamless between the main app and its database. It is easily integrated into both web apps and python code, which would make it an appropriate database system. [10]

### 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.

### 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. [11]

### 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. Like ionic it allows for a live development process. Making changes easy to implement and test. Django allows for easy communication between the front and back-end systems via HTTP requests and in response Json files.

## 2.4. Other Research

This section consists of any other relevant research that does not fall into the other sections above. While previous research put focus on choosing particular technologies to suit tasks, this section will examine different approaches that may be taken to tackle a given task.

### 2.4.1. Mobile App Environment

When choosing between the options available for mobile app development there are several features that can be considered. In this section Native and Hybrid 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, but due to the fast-paced development that comes with the hybrid app development it has been chosen for the proposed application.

### 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. [12]

### 2.4.3. Fitzgerald patent

Fitzgerald Chas an abandoned patent application for a recommender system designed for navigation, which would be similar to one part of the 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. [13]

## 2.5. Existing Final Year Projects

In this section two projects from previous final years will be discussed. Both of the projects have been picked due to similar aspects in the projects that relate to the proposed system in this project. Through the examination of other previous final year projects, an idea of the scale of the project and the styles of work required can be ascertained.

**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. The examination of two previous final year projects also helped in how the planning should go regarding the volume and style of work required.

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 stages and each of them will be explained. 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, project can be structured in such a way that makes sense for the 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. [14]

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

* Plan
* Design
* Develop
* Test
* Analyse
* Meet

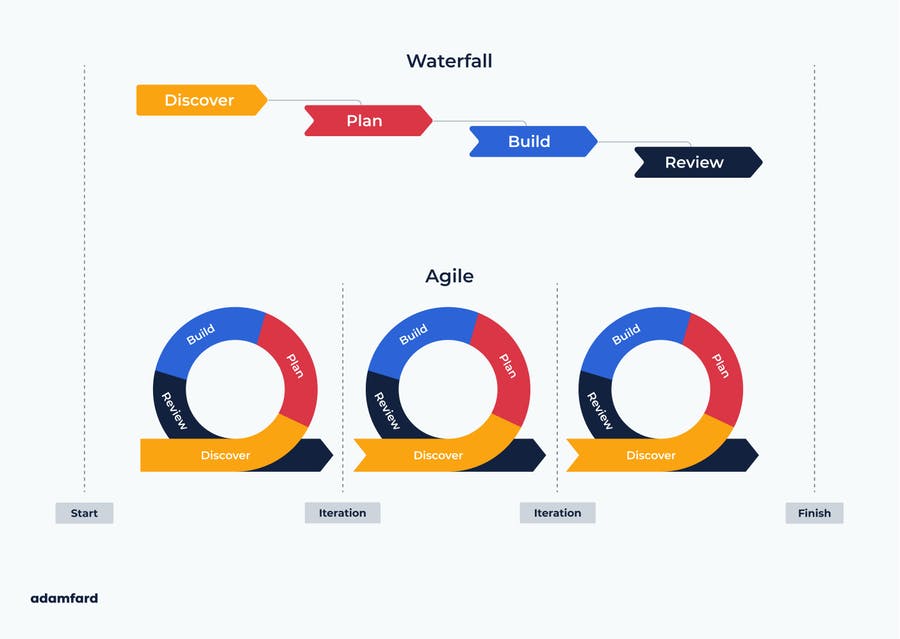


Figure ‑ Waterfall and Agile Design methodology

### 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/Stories  
  This is where the changes or new features are laid a out and considered as to the requirements they may have. Together with this testing scenarios are developed. This is a more high-level way of looking at what will be produced. Tis section will also put consideration into the timeframe and what team members will be involved in what.
* Design  
  In this section the plan taken from the previous step is looked at with a technical mindset and a set of technical requirements are produced and tackled to make a detailed design of the product.
* Code/Spike  
  This is where the team goes off and implements all the features laid out in the design plan. Pair programming would be apart of this step. Teaming up two developers to code together on one machine.
* Test  
  Here all of the features that have been newly developed are tested. This goes along with automated testing of all previous features.
* Deliver/increment  
  This final step of the cycle involved giving the product to the customer or showing the customer what has been changed, looking for feedback and considering what a future intention might be.

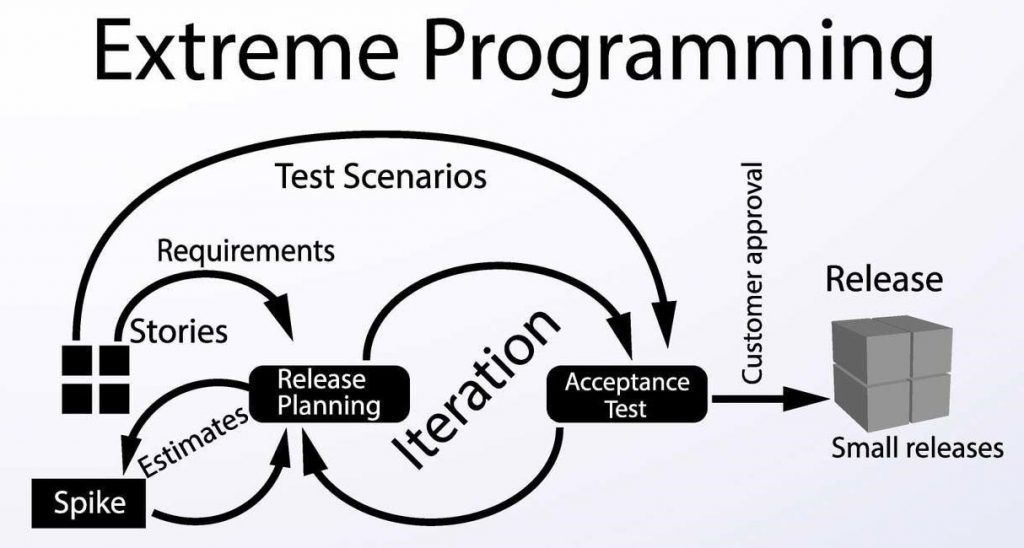
One problem that arises when considering XP is the pair-programming concept as there will not be two people to work on the code, this will not be possible. XP delivers smaller releases in shorted timeframes. This rapid cycle allows the customer to be more involved in the development of their product. As well as this it makes finding bugs easier as newly added features are tested straight away while also being fresh in the mind of the team when/if any bugs are found. [15]

Figure ‑ 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 communicates to the Owner and the Team handles all actual development.

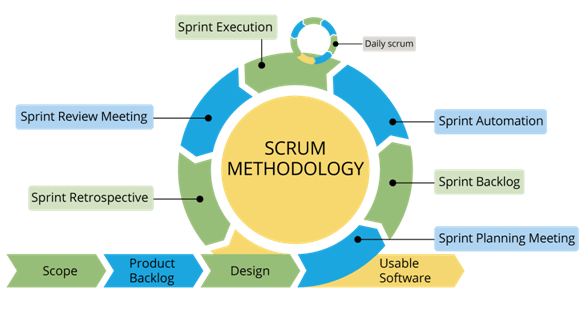


Figure ‑ Scrum Design Methodology

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. [16]

## 3.3. Overview of System

In this section the structure of the proposed application will be discussed. It will be split into three main parts in order to make a logical separation between each of the major parts going into the proposed application. There will be a Front-end which will integrate several API’s and a Back-end which will host the Database and Recommender system.

## 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.

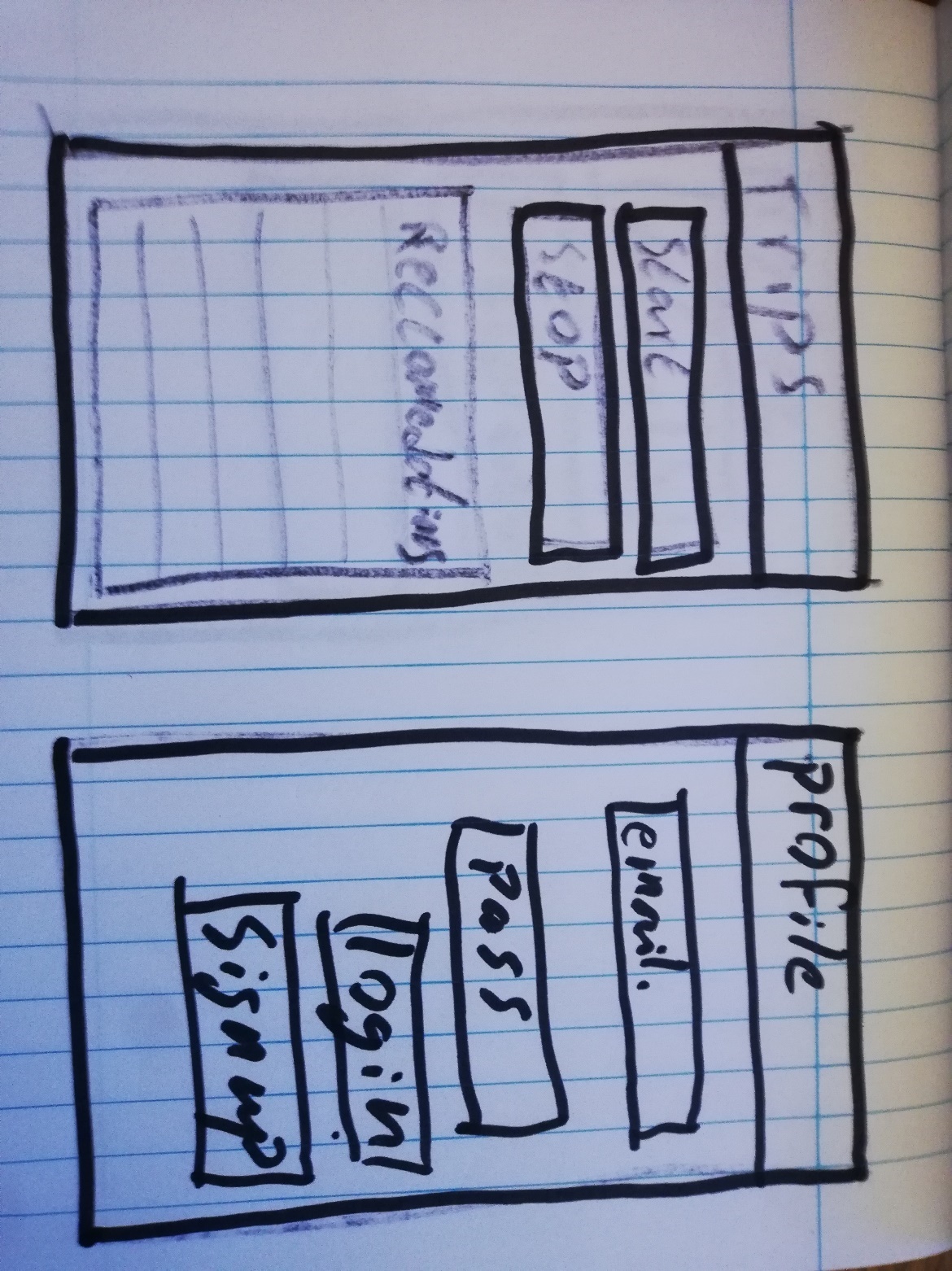


Figure ‑ Trip and Profile login screens

The above screens show the initial proposals for trips page, where users will input their start and destination to then be presented with a range of recommendations based off that info. The second screen outlines a simple login screen for the profile section.

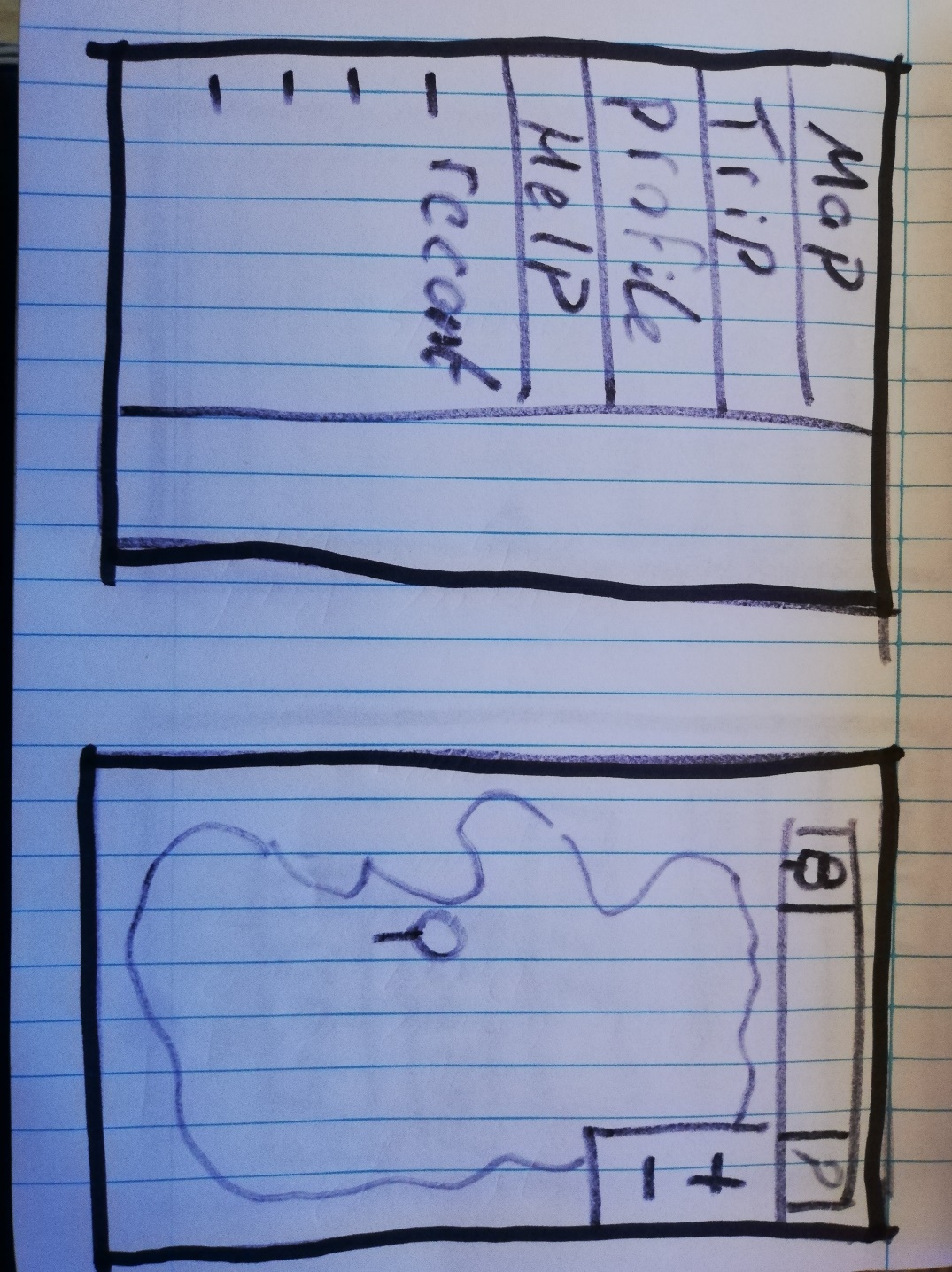


Figure ‑ Menu and Map screens

The first screen shows the slide in menu that should always be available to the user in order to make navigation though the app easy and always available. The second screen outlines the map page where the user should be able to search for locations and manipulate the map using touch and side buttons.

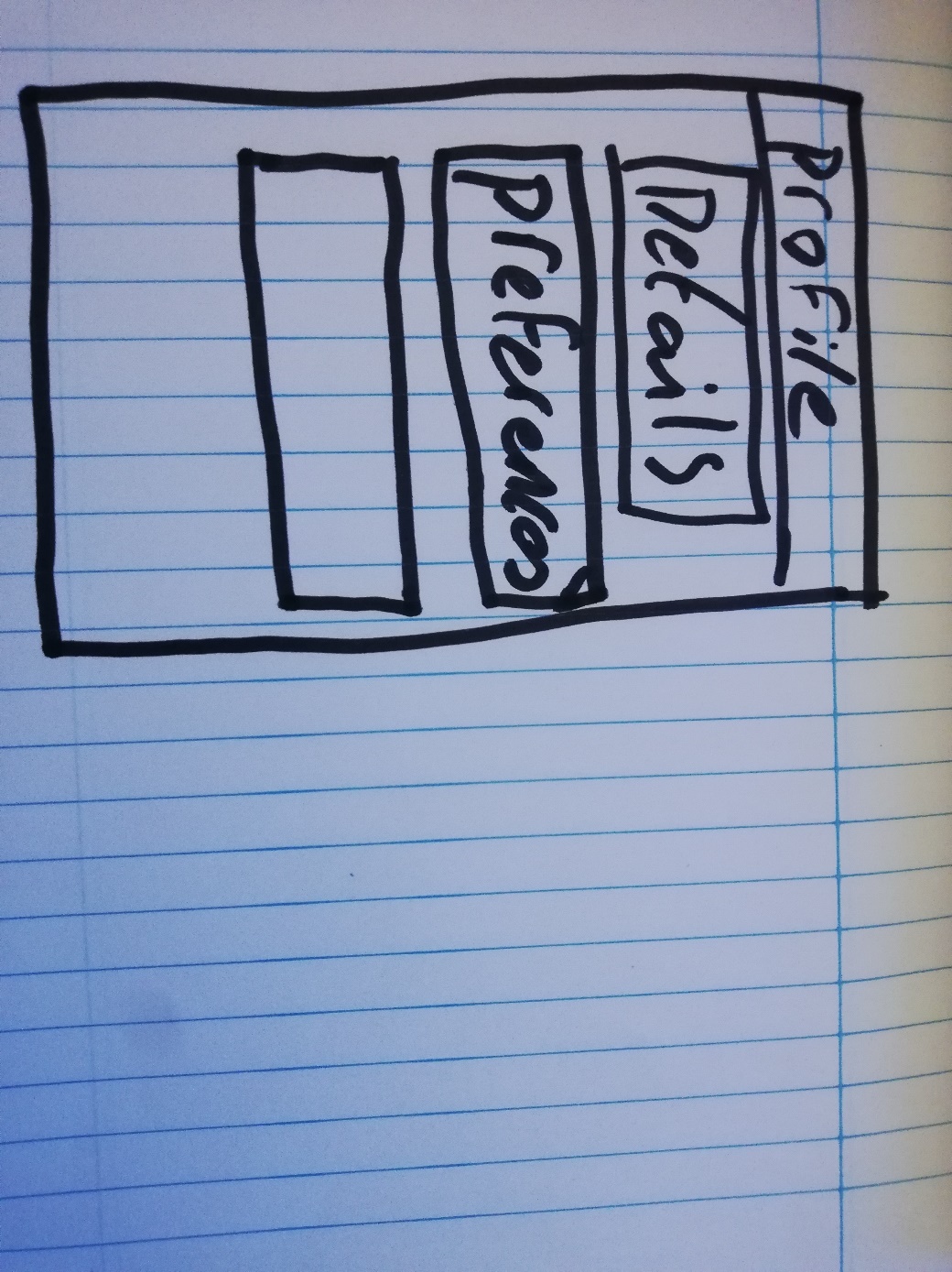


Figure ‑ Profile Screen

This profile screen is a gateway to deeper options within the app in order to personalise it to the users own liking. Details would refer only to the user account details, whereas the preferences would relate to the style of trip creation and alter how the recommender interacts with the user.

### 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.

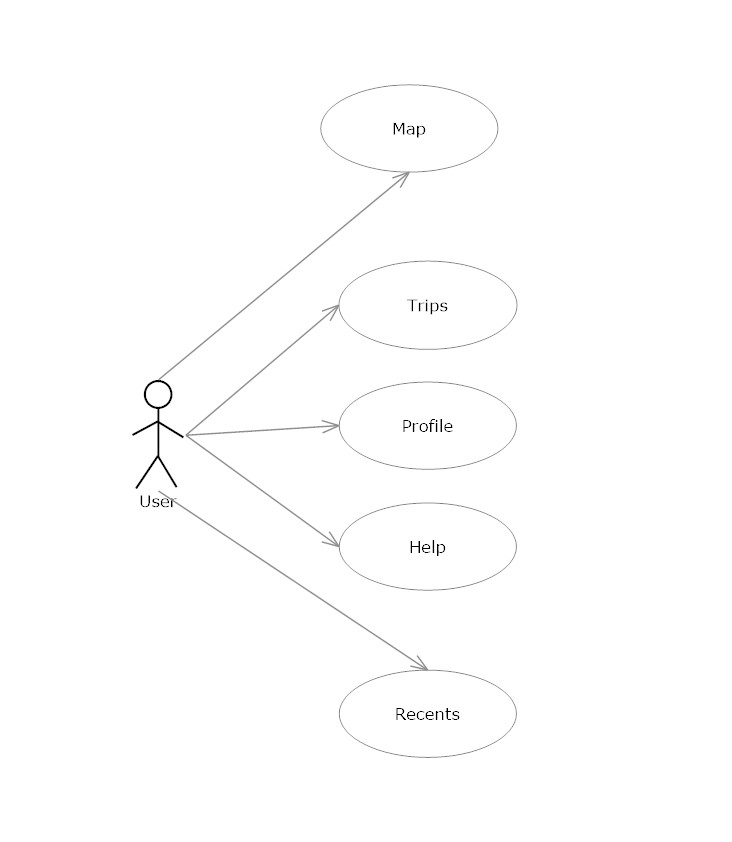


Figure ‑ Use-case Menu Screen

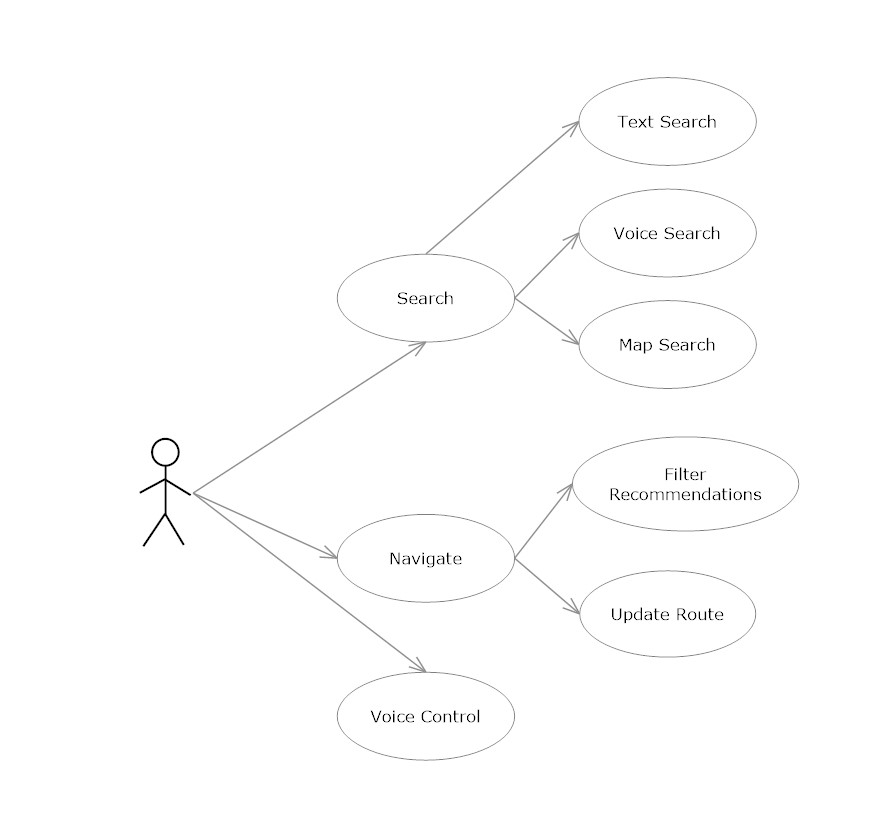
Figure 3-8 Describes the Menu functions which allows the User to navigate through the proposed application. This menu should be a sidebar that pulls out while the map will shown on the main page. This sidebar allows the user to oriantate themselves in the app at any time. It also provides easy switching between tabs which saves time and decreases the time a user needs to learn the layout. 

Figure ‑ 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. By providing multiple types of user input it increases the usability of the app and allows some disabilities users may have to be overcome.

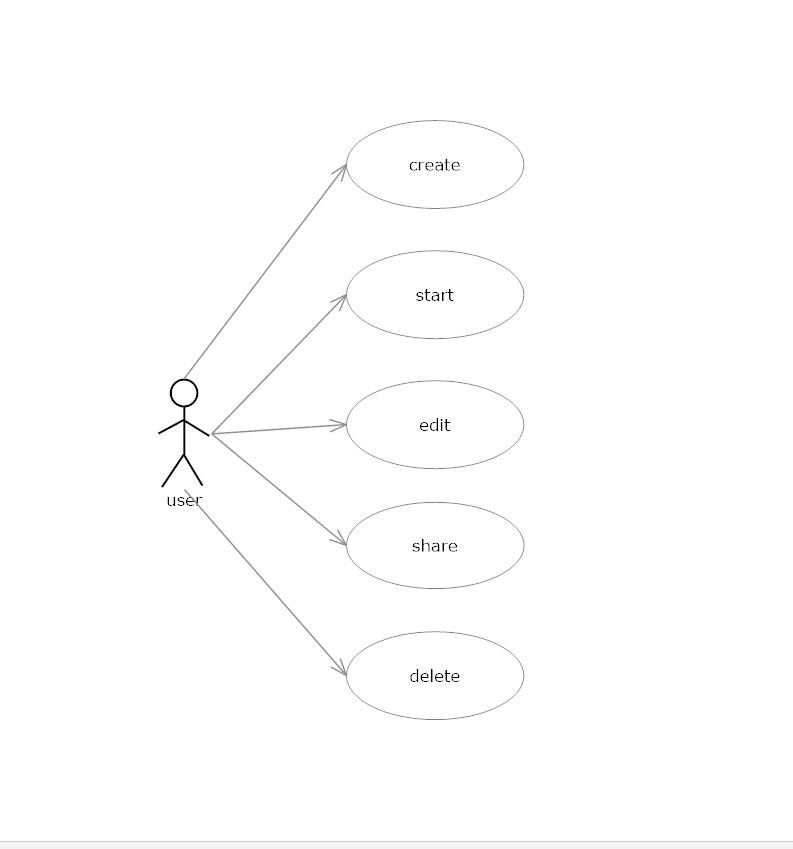


Figure ‑ Use-case Trips Screen

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. By organising the trips section in this way users can prepare for trips ahead of time and edit them as they see fit. By allowing the ahead of time creation for multiple trips the user is more likely to plan to use the app in the future.

### 3.4.2. Horizontal Prototype

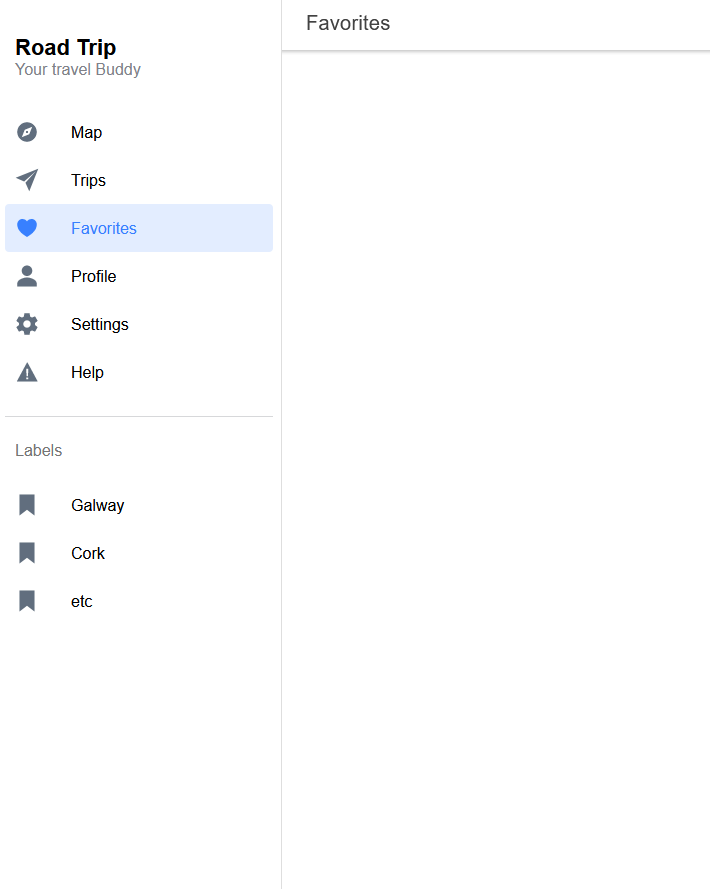


Figure ‑ Web prototype

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. [5]

### 3.5.2. Django

Django will be used in order to host the python-based recommender system. It will facilitate the receival of http requests and extract any data passed through with them. The use of the URL will specify what function in the recommender system is being called and what arguments are being sent to said function. Then will be able to serve the application a json response that the recommender system has produced.

## 3.6. Back-End

In this section the Backend portion of the proposed application will be discussed. The back-end portion consists of two main parts. 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.

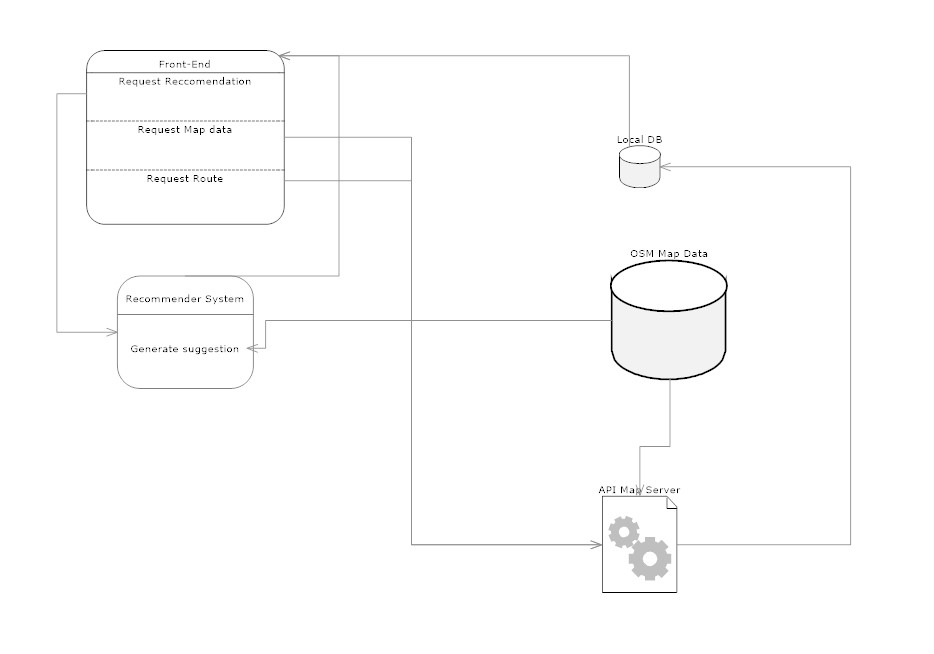


Figure ‑ Data flow

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 in python code. 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.



Figure ‑ [17]

### 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.

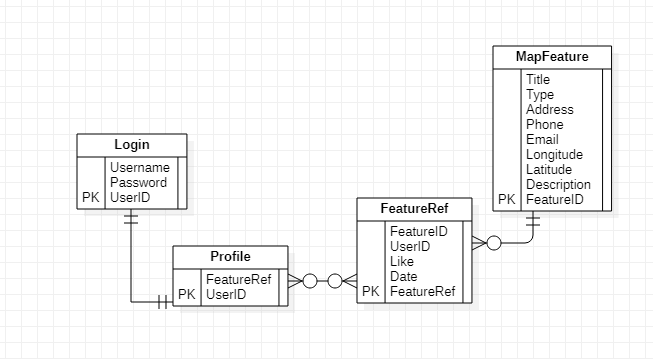


Figure ‑ ERD

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 with specific changes being made as testing a requirements analysis take place. This leads onto the next chapter where each Prototype iteration made for this project will be spoken about.

# Prototyping Development

## 4.1. Introduction

This section will speak about the processes involved in the development of the proposed application. Each of the protypes below are an iterative improvement over their predecessor and aim improve the application so that each is closer to the final product.

## 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. These three parts have been identified as core parts of the proposed project so they have been chosen for prototype one in order to ensure that they can work and integrate with each other moving onto later prototyping phases. The second prototype will be where all required technologies are being implemented as this puts more focus onto the front-end technologies; With a small amount going into starting work on how the recommender system will work so that its further development can be planned for the next prototype.

### 4.2.1. Ionic app

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

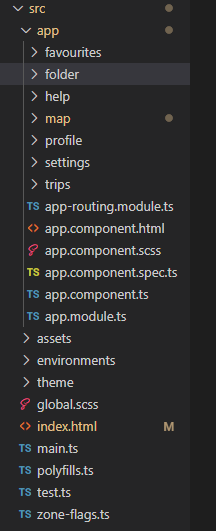


Figure ‑ Ionic Folder Structure

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.



Figure ‑ 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.

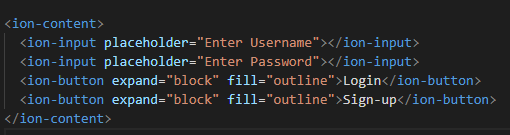


Figure ‑ Login Screen

Above is an example of how the simple login Screen is constructed. This will be altered in a later prototype to connect with an authentication system.

### 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.

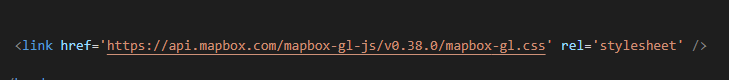


Figure ‑ MapBox Style Sheet

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

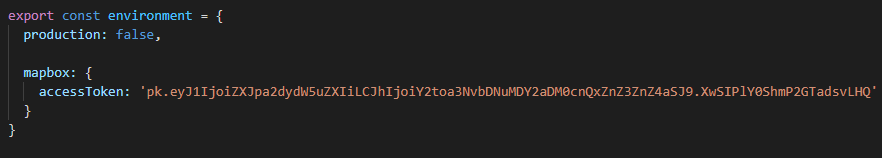


Figure ‑ MapBox environment

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.



Figure ‑ 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.

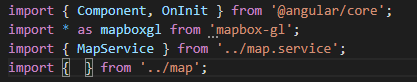


Figure ‑ MapBox Component imports

Then we can setup some defaults for the map to use. This is initially for testing as browsers don’t provide any sort of accurate geolocation data.

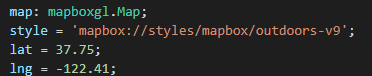


Figure ‑ Default settings for MAp

The buildMap function is used for the creation of the map when the Map tab on the app is initialised.

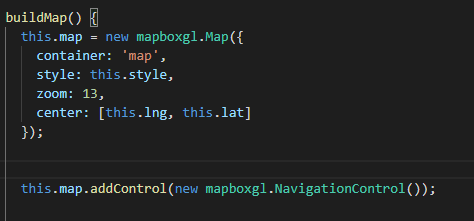


Figure ‑ BuildMap function

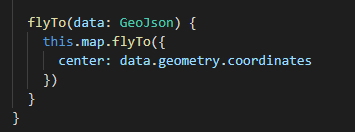


Figure ‑ flyTo function

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.

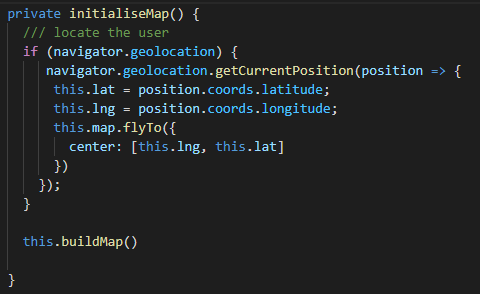


Figure ‑ initilaliseMap function

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

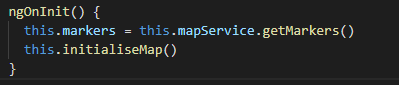


Figure ‑ Innit function

### 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.



Figure ‑ 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.



Figure ‑ 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.



Figure ‑ 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.



Figure ‑ 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

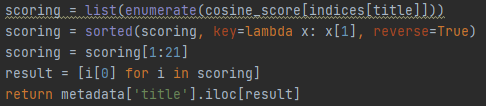


Figure ‑ sorting results for recommendations

## 4.3 Prototype Two

Prototype two consists of the ionic app using the MapBox API, a Django server and the implementation of the firebase database. This prototype represents all the core technologies required for the final design. While the previous prototype focused more on the front-end aspects of the proposed application. AS we can see that these technologies can and have been integrated together it give a platform for future prototypes to be developed.

### Ionic App

As part of prototype two the ionic app has taken the majority of the integration work as it has done most of the facilitating for this to take place. The first part of this integration was setting up a http service on the app so that the Django server could be reached, as this would be the method for communicating to it.   
More pages were added to the app as functionality expanded for example A registration page along with a login page was added to facilitate future profile settings for the app.   
Form functionality has been added to the proposed application as this will be a core part of the proposed application and it also served as providing testing functionality to the integration of the other core functionality.

### MapBox API

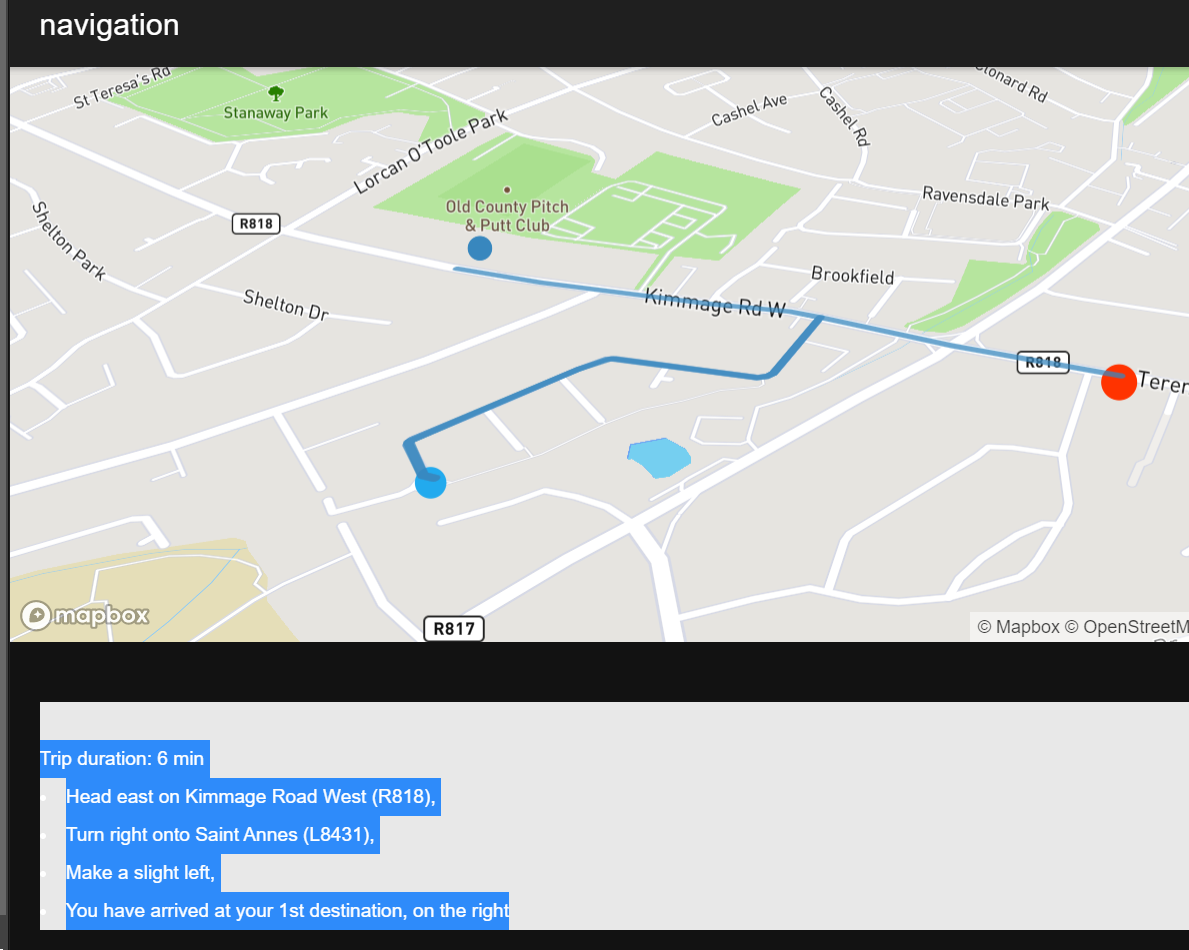
Since the last prototype, routing and directions frameworks have been constructed in order to facilitate future functionality. A basic proof of concept shows that all the necessary parts work. 

Figure ‑ Prototype two routing example

In the above figure it can been seen that a starting and end point with a stop along the way has been routed and draw onto the map. Then the directions for this route have been displayed below it.

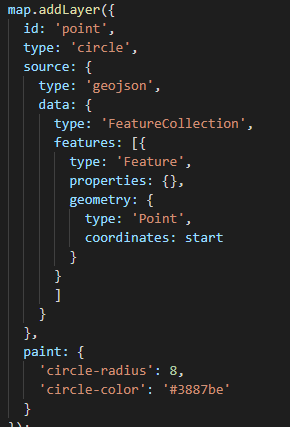


Figure ‑ Map points

This code shows points are added to the map via adding layers on top and specifying Geo Features. The data being used is formatted to how it is stored and retrieved from the database. This being important for ease of compatibility by using the same standard across the whole application.

To use the API a http request is performed where the specified route is defined and in return a json file is given. The json file contains the route information. In a similar fashion to adding the points on the map the route is drawn on the map. In order to display the direction instructions a simple function has been built to extract them from the json route file that was returned.

### Firebase DB

The firebase database utilises the Cloud firestore feature. This in conjunction with the firebase authentication service both allows for secure logins and allowing for the profile features to be implemented.

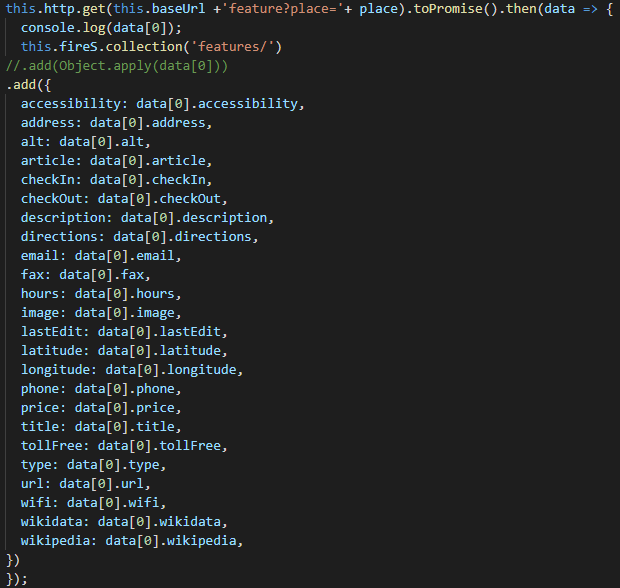


Figure ‑ FireStore first implementation

Figure 4.20 above shows in a very simple fashion how the future protypes will e able to store data taken from the recommender system in the FireStore DB. In order to organise the data into a form the FireStore API could manage the data is first stored in a promise but this is likely to change in future iterations. As the Recommender service returns a json format it can be seen how easily the transfer is made after that by simply calling “data[0].title” to retrieve the title and place it in a title attribute on the DB.

The second use of the FireStore API was the authentication method that FireBase provides with the service. By using this authentication method, it means that unnecessary development time is not spent on security, but is provided by professional services.

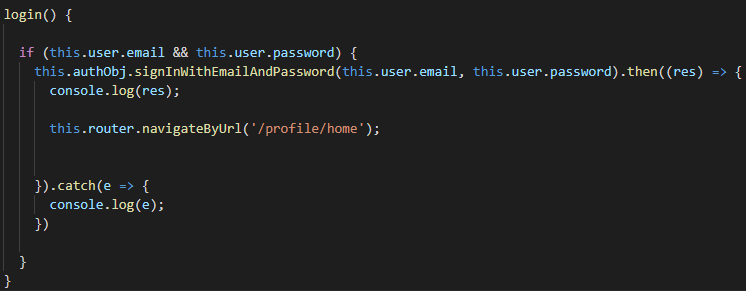


Figure ‑ Login Method prototype two

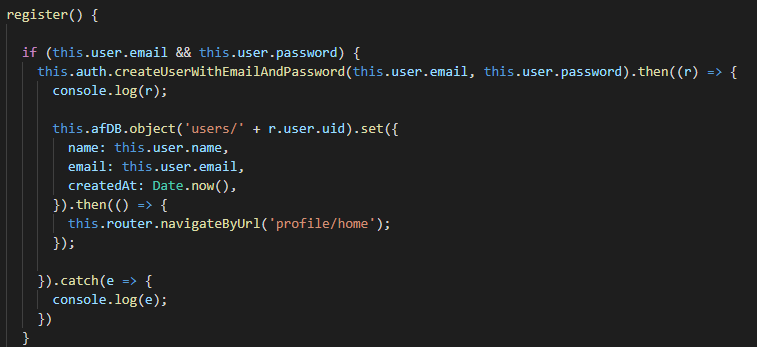


Figure ‑ Register Method prototype two

The registration and login process have been handled above in these two very simple functions. Where the information was taken from forms on the html side of the app. Each function then routes the user to the appropriate page after the process comes back as successful.

### Recommender system

The recommender system has seen the integration into a Django server as part of the second prototype. This is then accessed via HTTP requests through the ionic app. As an example, the system may ask for a set of recommendations for the specific place of ‘Kruithuis’ and would send the following request <http://localhost:8000/recommend?place=Kruithuis>. This would return a json file with all the names and indexes of the recommended places the system generated back to the app.

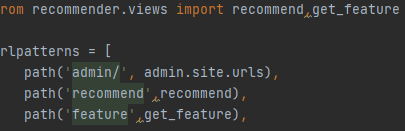


Figure ‑Django requests

Here it can be seen all available requests that can be made from the Django server. Currently there are two on as mentioned for getting the recommendations and the other is to get more info on a specific feature. This would be used in displaying details about a place or getting the co-ordinates for routing to the feature.

### Issues

For the most part the issues encountered during the development for protype two all had to do with version compatibility.   
MapBox presented the most issues. The main one being an error on my own part. This was due to confusion with the applicable technologies when trying to display the map initially. As the API is called through typescript code no consideration was given to the overarching ionic platform which needed to initialise the API itself before allowing the underlying angular code to initialise and display the map.

### Summary

Prototype two has achieved all its goals which were to provide all the basic functionality required for the future development of the proposed application. Now that these technologies have been integrated it has been shown that they can work well together, and focus can be put onto expanding the features of the application as well as enriching the User experience.

## 4.4 Prototype Three

In this the third prototype the aim puts focus in three areas which relate closely to each other. The first being the expansion of the options given by the recommender system, specifically so that recommendations can be specified by location. This is necessary to both improve importance but more so to remove destinations that are not within reach for the given trip. Secondly the Map options should be updated to support the choosing of recommendations for the trip. Third is to expand the database to store the preferences of each user, taking the recommendation system to a place where is it useable rather than the proof on concept in previous iterations.

### Map Interface

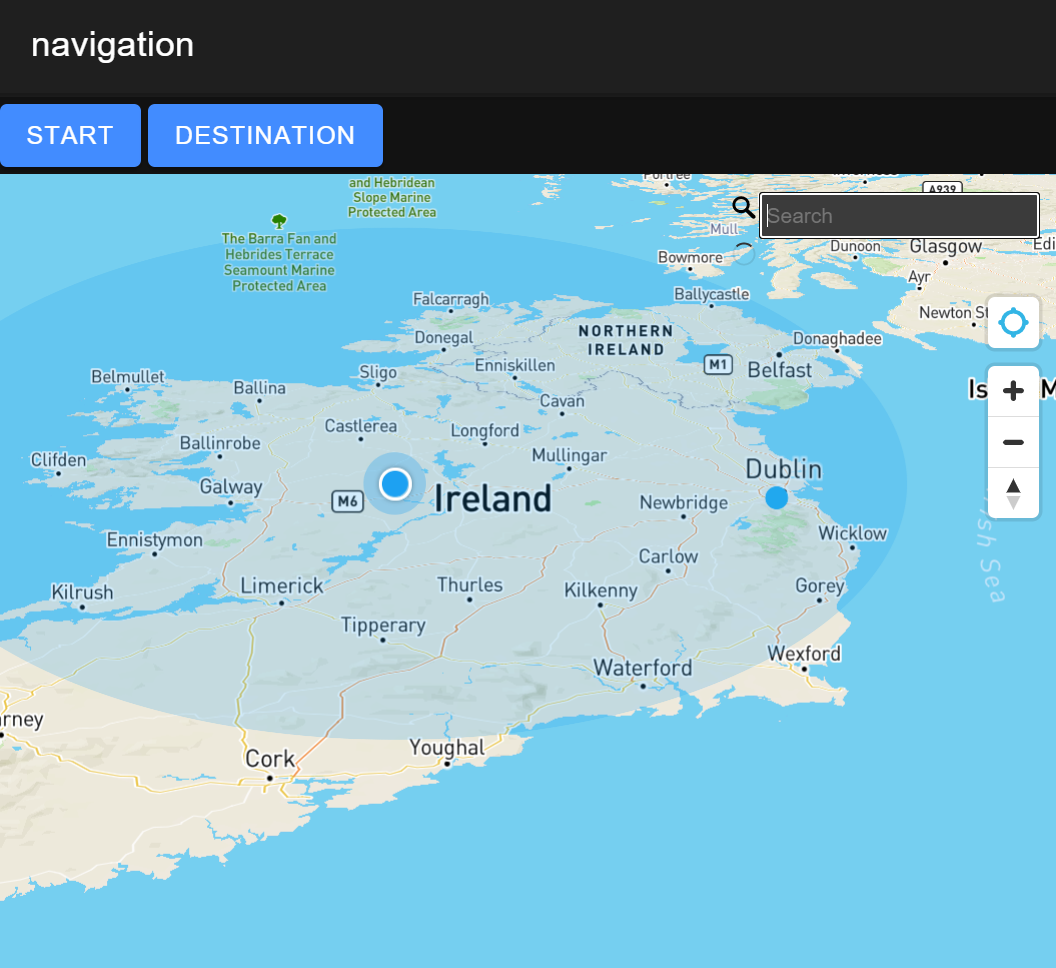
This portion of work is where the users will be interacting with the various background systems. As the system has aspects of safety involved with its use this will have to be considered when making any design decisions. One of the first updates was to add a here location button so that users can easily zoom into where they are, this not being so useful now but when being used on a mobile device with an accurate GPS location. After that, a geocoder was added so that users could now search for locations on the map rather than previously only having panning and zooming available. AQs well as that a start and Destination buttons were added to replace the testing code before it which was in place for the proof of concept. Below in figure 4.24 the updated map can be seen without the recommendations tab.

Figure ‑ Prototype Three Navigation page

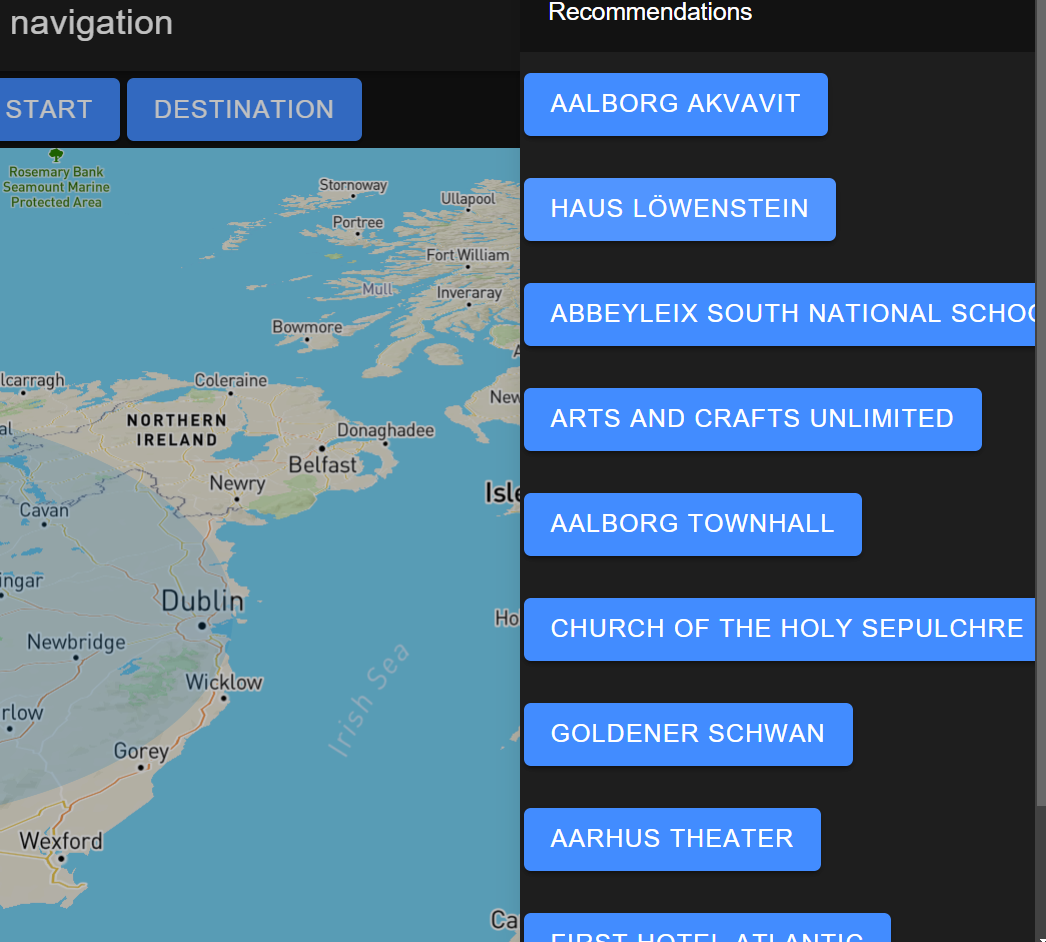


Figure ‑ map view with recommendations

Figure 4.25 shows the recommendations tab that can be slid in from the right-hand side as no to obscure view of the map when in use. On the Typescript side of the app this is a generated list taken from the recommender system. To make use of the generated list, in a similar fashion a group of listeners is also generated at the same time. This can be seem in figure 4. 26 below. Where ‘places’ represents the object holding all of the recomendations and ‘recArr’ simply hold the names of those places as a running list of the selected recommendations in use at the time.

Figure ‑

### Recommender system

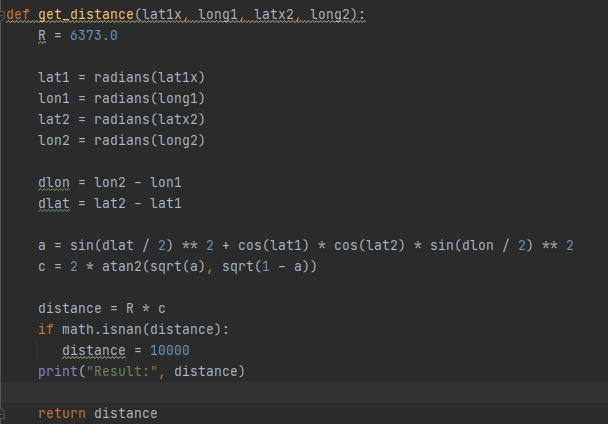
As mentioned above this prototype saw the expansion of the recommender system so that recommendations could be generated based coordinate locations and a set radius along with the normal parameters.  
For this feature to work a new ‘get\_distance’ function had to be added to the recommender system. This was based on longitude and latitude as this is a publicly available equation it was simple to find and implement.   
Aswell as this another function was added in order to load all the CSV map data into the firebase database. This took advantage of the distance function in order to create a smaller world for testing. 300km from Dublin was chosen. This populate function would only be intended for use by an administrator to the system to upload data into the system. But could be used a as a way to procedurally update the database as the souce data became more up to date. As seen in figure 4.27. 

Figure ‑ populate\_db function

### Database

This part of the prototype was very simple and consisted of adding some tables and rewriting how the database is accessed in order for it to continue to work with the system after some structural changes were made in the prototype. This was mainly switching the recommender systems source of data from a local csv file over to the FireBase database. Adding a way for each user to store their own likes and dislikes was done to facilitate what will be the fourth prototype.

### Issues

During the production of this prototype some structural reorganisation had to be made in order facilitate the flow of data from supporting systems into the MapBox API.

Now that the project has become more integrated it is not possible to test a lot of the app’s features in isolation. This is due to the reliance on the supporting systems being accessed as things are loaded up. This has highlighted potential issue in the running of the app. In the next prototype I hope to make the reliance on supporting systems more dynamic so that if any one system fail it should not affect any unrelated systems.

## 4.5 Prototype Four

This prototype aims to deliver a more personalised experience to each user. This will focus on each user getting personalised recommendations. A Profile section will be expanded to allow users to train the system to their likes and dislikes.

### GUI interface

The GUI section of the prototype has a main training page under each user profile. As can be seen in figure 4.28. The user has the option to train a number of sections that the data has been broken up into appropriate subsections. This allows users to train individual aspects of their profile, allowing for greater personalisation to be made. Extra relevance infor is displayed with the recommendation when it is avaliable in the metadata.

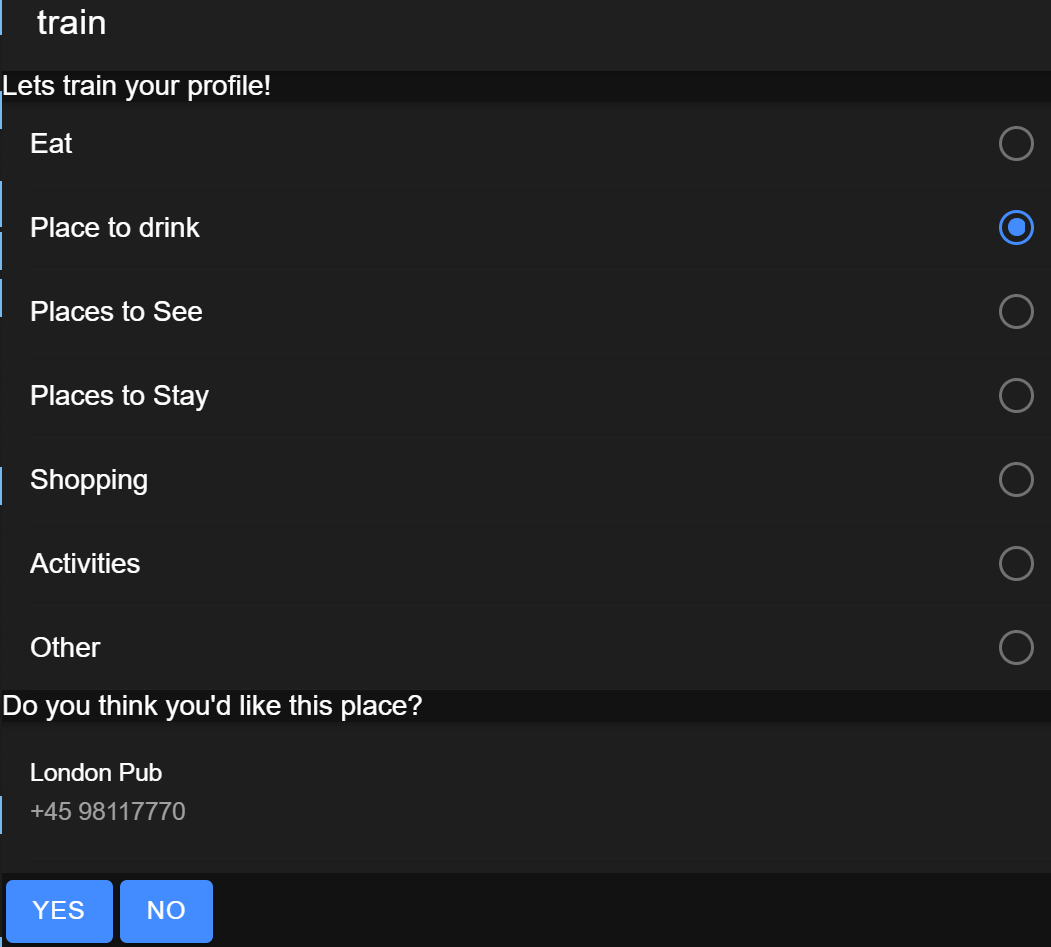


Figure ‑ Recommender training page

### Recommender system

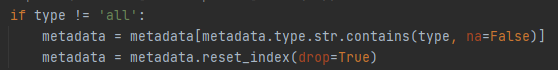
In this prototype the recommender system saw a large amount of rewriting code needed to provide the services for the personalisation features. For the training portion of the recommender, a new function was created to take the input of the user ID and the type of recommendation they would like to get back. You can see below in figure 4.29 how the system takes the type passed to it and selects the relevant data for the program to run.

Figure ‑ type selection on recommender system

This function will work both as a further part of the main navigation system when the user is logged in. And allows for use in the training system. When generating what results are returned to the user it uses the User ID to check what likes the user has made in the past for generating relevant results. The system will access the dislikes that the user has made in the same way and remove them as options when generating new recommendations.

### Database

For this prototype, the way data is stored had to be changed slightly. The main change was that when users created an account and used the app logged in their personalisation data would be stored under their own user ID. As opposed to the earlier prototypes where multiple users were not separated out logically on the database. The User ID being created through the Firebase authentication service makes it easier to associate the database information with a particular user. This setup will be key to any future sharing capabilities that will be implemented into a later prototype. This will likely be seem in the sharing of either active or save trips.

### Issues

This prototype contains the largest number of problems out of all the previous ones. This was due to a lack of consideration when planning out the prototype. Certain systems in particular the recommender system had to change to allow for the input and processing of user ID’s when returning personalised recommendations to the users.

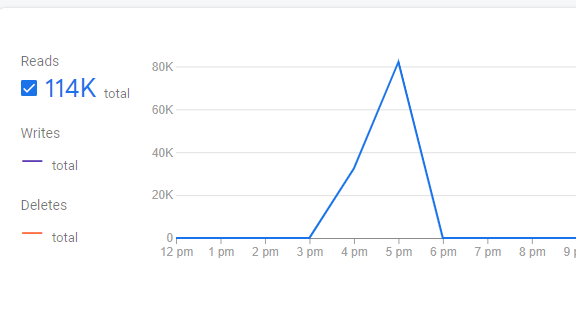
During the development of this prototype the free account provided by firebase intended for use in development posed a problem. During the testing of the Profile Training page, a function had been written incorrectly and sent a request for the entire contents of the features collection in the database. This resulted in one hundred and fourteen thousand requests being sent to the database in the space of few short hours. As shown in figure 4.30 As this exceeded the daily allowance given by the free account the account was temporarily suspended.

Figure ‑ Firebase Usage report

## 4.6. 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.  
As a lot of the technologies were unfamiliar it allowed time to put a lot of fucus onto one particular part of the application thusly allowing focus to be put on learning one particular technology associated with the part in question. While being able to leave integration to later stages.  
One of the main issues facing this style of development was that some features found themselves in-between prototypes and they had not been clearly defined so had not been considered when drawing up the plan for each one. This led to some prototypes being delayed as some features needed to be in place before the new planned features were added. This was evident when implementing the personalisation through profiles. The recommender system had not been updated to take in and look for individual profiles when in the previous prototype and was not initially considered when planning the next prototype which led to delays in the development of said prototype. These types of issue were also partly caused by a lack of knowledge in the area. In the future more detailed analysis and planning can avert these types of issues.

Through the use of the prototyping development cycle based in the style of extreme programming as noted in the previous chapter a working application has been produced. Due to both time constraints and the learning of new technologies, the entire system as laid out in the opening proposal has not been met. Many aspects of that original goal have been satisfied. As the application currently sits there is mainly aesthetic work and general fleshing out and utilisation of backend function required to complete the application. An example of this would be how the trips tab had not been utilised but rather the same functions required were only implemented in the navigation tab.

Each prototype can be summarised to what its general goal was and how it turned out.

* Prototype One put focus on mainly front-end technology utilisation and included a small part of the testing recommendation styles.
* Prototype two finishes the core technologies testing by adding in the Django server and database.
* Prototype Three saw more focus put on the development of the recommender system and the supporting database features with some improvements to the map interactivity.
* Prototype Four aimed at making the system more personalised to each user by adding the training system to the app and separating out the users on the database based on generated login ID.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Protype One** |  |  |  |  |  |  |  |  | x | x | x | x | x | x | x | x | x | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Protype Two** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x | x | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Protype Three** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x | x |  |  |  |  |  |  |  |  |  |  |  |
| **Protype Four** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x |  |  |  |  |  |  |  |  |  |
| **Protype Five** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Software**  **Testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | 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 | x | x |  |  |  |  |  |
| **Final**  **Demonstration** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x |  |  |  |  |

# 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. !!!write5 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 of information through the given system, identifies ways to improve design and usability of the system. Unlike its counter-part black box testing performs testing inside the given system. White Box testing tests the following areas in 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. [18]

### 5.2.3. Nielsen’s Heuristics

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 a system. Each check should test against specific design principals which have been altered to suit the needs of these types of navigation apps. [1]

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.

Grading 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 trip 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.

## 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.

### 5.3.2. Metric testing

In order to use metric testing specific metrics for the proposed application must be developed. In the case of this project the main focus for metric testing will be on the output from the recommender system. That being the product metrics, but the process metric i.e., the efficiency will also need to be tested. This can more easily tested simply by using the time import ion python and adding it into the appropriate sections and then performing test scenarios. For example, inputting a small medium and large trip and a set number of recommendations being chosen.

## 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.

While a lot of very specific testing methods have been looked at in the above chapter only those which are deemed appropriate will be used. In a lot of cases this can only be known after the code has been written.

As part of the final evaluation, the Nielsen’s heuristics will be used again, as discussed earlier in the research section. By doing this the proposed application will be compared to the applications that had been evaluated at the beginning of the project.

# 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.

## Project planning

Overall, the method utilised, Extreme Programming, for the running of this project was very appropriate and worked well. It allowed for an iterative cycle of prototypes that each put focus on various aspects of the proposed application. Starting from a base of the core technologies and finishing with specific features sets. This method also allowed for some degree of flexibility while not resulting in no deliverable allocation. This being that separate parts were made to work completely before moving onto other features, rather than developing the system as a whole in one development phase. This flexibility was needed towards the end of the development cycle when the decision not to implement a prototype five was made. This was a time constraint caused by bugs during the testing of the fourth protype.

Although the project was a lone effort the concept of pair programming from extreme programming would have helped a great deal as a large portion of time was spent troubleshooting.

## Application results

In this section the application outcome of this project will be discussed. This includes the heuristic evaluation as alluded to earlier in the early research phase and the testing chapter. As well as this what deliverable features within the application and what was not delivered will be spoken about. This will make reference to the proposal of the project and include any reasons for change.

### Nielsen’s Heuristics

Below is the heuristic report performed on the application produced. This being a key part of the results as the heuristic method had been part of the requirements gather process performed in the research stage of this project. It also provides a comparison to existing applications of a similar nature.

**Visibility of system status**:

**Match between system and the real world**:

**User control and freedom**:

**Consistency and standards**:

**Error prevention**:

**Recognition rather than recall**:

**Flexibility and efficiency of use**:

**Aesthetic and minimalist design**:

**Help users recognize, diagnose, and recover from errors**:

**Help and documentation**:

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.

Below to reiterate are the requirements generated from the heuristic reports done on two similar systems, Roadtrippers and Google Trips.

* Simple well-defined UI  
  Through the use of the ionic framework and its adaptive design a simple and aesthetically pleasing application was successfully created.
* Real time navigation   
  This portion was partially achieved with directions being generated and tracking of the user on the map but active navigation was implemented.
* 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  
  This was not implemented as the project progressed the decision to favour an intuitive design made less relevant.
* Quick access(recents)

## 6.2. Conclusions

While all the backend parts to make up the system were implemented by the end of the project, not all of the features planned for the front-end GUI were implemented. As the application was aimed mainly for use in cars, the fact that the usability aspect of the application wasn’t fully tackled affected the outcome of this project. This does not change the order that things should have been implemented though, reason being that the features that would provide usability to the user needed to rely on many behind the scenes features both for testing and for proper system feedback in real-world settings.

The way in which this project was planned and executed through the extreme programming methodology proved to be very effective given constraints of time, allowing for a deliverable application before all features had been fully implemented. This of course refers the prototyping involved the development cycle. Ideally another two prototypes would have been added to the project timeline in order to implement the remaining planned features.

From a personal standpoint undertaking this project has allowed me investigate and experience what is involved in developing a project of this scale. While the beginning may have been a bit slow and less interesting that the later development stages it proved to be very useful and informative. Without the initial research stage later development would have taken longer due to a lack of knowledge in choosing the appropriate technologies to suit the needs of the project. I p[personally really enjoyed breaking the project in various prototyping phases as it gave me a sense of progress and achievement along the way when I was able to get various parts working. This continued as the project grew in complexity knowing that I had spent the time to both develop these pieces but having them work together as well to serve a tangible purpose. Finally the writing up portion would never have been a favourite of mine but having completed a document of this scale on my own work helped developed my documentation skills which will be invaluable in the future.

## 6.3. Future Work

First in mind when it comes to future work is the planned prototype five which was cancelled due to time constraints. This prototype was to focus on the multiuser aspects of the application. This would include a way to share live trip with friends so that for example multiple cars could be used on a road trip without becoming unsynchronised when any live changed were to be made to the trip. The other main aspect apart from sharing trips was to focus on the way in which the recommender system generated its results. As mentioned previously this type of system can suffer from cold start problems but in this multi users prototype the system would be modified to be a hybrid system of recommendation that would make associations based on other previous user input. This would work in a similar way to Amazon’s recommender of products, which has the section on “Customers also bought these items”. Meaning that other users who liked a particular place would then have their other similar likes show up suggestions to new users who also liked that particular place.

Furthermore, as envisaged in the project proposal, making the application into a more intelligent companion or recommender is another thing to do for the project. This would mean that things like suggested coffee breaks or petrol stops would be suggested slightly ahead of time but would be at an appropriate part of the trip. So, a popup would appear during navigation and the user could either ignore or easily add it to the trip.

The way in which the recommender system calculates distance could be made more intelligent. While checking straight line distance between places makes sense when planning out a long road trip, it would make more sense to calculate the distance by road for a place is on a shorter trip or when actively adding stops during navigation on a motorway for example.

Lastly, time would be spent on reasserting and implementing the scalability of the proposed application. While the technologies and methods used in the application had scalability in mind, no real-world testing of this took place as it was not in the project scope. If this were to be a commercial product it would need to a core aspect of the product. Together with this would be optimisation of the recommendation system. As it was being run on a standard version of python and nothing had been specifically changed, the program only ran on one thread which in testing on longer distance road trips had an effect of some latency when processing the larger number of possible location features.

# Bibliography

|  |  |
| --- | --- |
| [1] | J. Nielsen, “How to conduct a heuristic evaluation.,” *Nielsen Norman Group,* pp. 1-8, 1995. |
| [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] | Google, “Google Maps Platform APIs,” Google, [Online]. Available: https://developers.google.com/maps/apis-by-platform. [Accessed 05 02 2021]. |
| [5] | MapBox, “Web Services APIs,” MapBox, [Online]. Available: https://docs.mapbox.com/api/overview/. [Accessed 02 02 2021]. |
| [6] | Facebook Inc., “React Homepage,” React, [Online]. Available: https://reactjs.org/. [Accessed 02 02 2021]. |
| [7] | Ionic, “Ionic Home,” [Online]. Available: https://ionic.io/. [Accessed 05 02 2021]. |
| [8] | Oracle, “Oracle,” Oracle, [Online]. Available: https://www.oracle.com/index.html. [Accessed 11 10 2020]. |
| [9] | SQLite Consortium , “SQLite,” SQLite Consortium , [Online]. Available: https://sqlite.org/index.html. [Accessed 10 10 2020]. |
| [10] | Firestore, “Firestore,” Google, [Online]. Available: https://cloud.google.com/firestore/. [Accessed 02 11 2020]. |
| [11] | Python, “python,” Python Software Foundation, [Online]. Available: https://www.python.org/about/. [Accessed 10 10 2020]. |
| [12] | B. Rocca, “Introduction to recommender systems,” 3 June 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada. |
| [13] | V. D. i. Fitzgerald C, “Advice engine delivering personalized search results and customized roadtrip plans”. United States Patent 12/113,911, 9 December 2010. |
| [14] | A. Fard, “What is Agile Design and how to apply it?,” AdamFard, [Online]. Available: https://adamfard.com/blog/agile-design. [Accessed 01 10 2020]. |
| [15] | Agile Alliance, “Extreme Programming (XP),” Agile Alliance, [Online]. Available: https://www.agilealliance.org/glossary/xp/#q=~(infinite~false~filters~(postType~(~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'xp))~searchTerm~'~sort~false~sortDirection~'asc~page~1). [Accessed 08 10 2020]. |
| [16] | A. K. Verma, “DBAs: Why Use Scrum?,” Vertabelo Academy, 15 03 2018. [Online]. Available: https://academy.vertabelo.com/blog/dbas-use-scrum/. [Accessed 05 10 2020]. |
| [17] | wikipedia, “Cosine similarity,” [Online]. Available: https://en.wikipedia.org/wiki/Cosine\_similarity. |
| [18] | U. Eriksson, “White Box Testing – Step by Step Guide on Everything you Need to Know,” ReQuest, 28 11 2016. [Online]. Available: https://reqtest.com/testing-blog/white-box-testing-example/. [Accessed 05 10 2020]. |
| [19] | scikit-learn, [Online]. Available: https://scikit-learn.org/stable/. |
| [20] | 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. |