

Road Trip Companion

Final Project Report

DT211c

BSc in Computer Science Infrastructure

**Erik Grunnér**

**C17412636**

**Damian Bourke**

School of Computer Science

Technological University, Dublin

**Date**

Abstract

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

Contents

[1. 1. Introduction 9](#_Toc58414416)

[1.1. Project Background 9](#_Toc58414417)

[1.2. Project Description 10](#_Toc58414418)

[1.3. Project Aims and Objectives 10](#_Toc58414419)

[Smart Objectives 11](#_Toc58414420)

[1.4. Project Scope 12](#_Toc58414421)

[1.5. Thesis Roadmap 12](#_Toc58414422)

[2. 2. Literature Review 13](#_Toc58414423)

[2.1. Introduction 13](#_Toc58414424)

[2.2.1 Nielsen 13](#_Toc58414425)

[2.2. Alternative Existing Solutions to Your Problem 14](#_Toc58414426)

[2.3. Technologies researched 20](#_Toc58414427)

[2.3.1 System Architect Diagram 21](#_Toc58414428)

[Front-End 21](#_Toc58414429)

[Google API & Mapbox 21](#_Toc58414430)

[Ionic & React Native 22](#_Toc58414431)

[BootStrap 22](#_Toc58414432)

[FireBase 22](#_Toc58414433)

[Back-End 22](#_Toc58414434)

[Oracle 22](#_Toc58414435)

[SQL lite 22](#_Toc58414436)

[Sequelize 22](#_Toc58414437)

[Python 22](#_Toc58414438)

[Django 23](#_Toc58414439)

[2.4. Other Research 23](#_Toc58414440)

[Mobile App Environment 23](#_Toc58414441)

[Recommender systems 23](#_Toc58414442)

[2.5. Existing Final Year Projects 24](#_Toc58414443)

[2.6. Conclusions 25](#_Toc58414444)

[3. 3. Experiment Design 26](#_Toc58414445)

[3.1 Introduction 26](#_Toc58414446)

[3.2. Software Methodology 26](#_Toc58414447)

[Waterfall 26](#_Toc58414448)

[Agile 27](#_Toc58414449)

[Extreme 27](#_Toc58414450)

[SCRUM 28](#_Toc58414451)

[Smart Objectives 30](#_Toc58414452)

[3.3. Overview of System 31](#_Toc58414453)

[3.4. Front-End 31](#_Toc58414454)

[Use-Case Diagrams 34](#_Toc58414455)

[Horizontal Prototype 36](#_Toc58414456)

[3.5. API’s 38](#_Toc58414457)

[Django 38](#_Toc58414458)

[3.6. Back-End 38](#_Toc58414459)

[3.7. Conclusions 39](#_Toc58414460)

[4. 4. Experiment Development 40](#_Toc58414461)

[4.1. Introduction 40](#_Toc58414462)

[4.2. Software Development 40](#_Toc58414463)

[4.3. Front-End 40](#_Toc58414464)

[4.4. API’s 40](#_Toc58414465)

[4.5. Back-End 40](#_Toc58414466)

[4.6. Conclusions 40](#_Toc58414467)

[5. 5. Testing and Evaluation 41](#_Toc58414468)

[5.1. Introduction 41](#_Toc58414469)

[5.2. System Testing Methodology 41](#_Toc58414470)

[Black Box Testing 41](#_Toc58414471)

[White Box Testing 42](#_Toc58414472)

[5.3. System Testing application 43](#_Toc58414473)

[Test Groups 43](#_Toc58414474)

[5.4. Conclusions 43](#_Toc58414475)

[6. 6. Conclusions and Future Work 44](#_Toc58414476)

[6.1. Introduction 44](#_Toc58414477)

[6.2. Conclusions 44](#_Toc58414478)

[6.3. Future Work 44](#_Toc58414479)

[7. Bibliography 45](#_Toc58414480)

[Figure 1-1 11](#_Toc58231757)

[Figure 3‑1 15](#_Toc58231758)

[Figure 3‑2 16](#_Toc58231759)

[Figure 3‑3 16](#_Toc58231760)

[Figure 3‑4 18](#_Toc58231761)

[Figure 3‑5 18](#_Toc58231762)

[Figure 3‑6 19](#_Toc58231763)

[Figure 3‑7 19](#_Toc58231764)

[Figure 3‑8 20](#_Toc58231765)

[Figure 3‑9 21](#_Toc58231766)

[Figure 4‑1 Waterfall and Agile Design methodology 27](#_Toc58231767)

[Figure 4‑2 Extreme Programming 27](#_Toc58231768)

[Figure 4‑3 Scrum Design Methodology 27](#_Toc58231769)

[Figure 4‑4 Smart Objectives 27](#_Toc58231770)

[Figure 4‑5 Trip and Profile login screens 27](#_Toc58231771)

[Figure 4‑6 Menu and Map screens 27](#_Toc58231772)

[Figure 4‑7 Profile Screen 27](#_Toc58231773)

[Figure 4‑8 Use-case Menu Screen 27](#_Toc58231774)

[Figure 4‑9 Use-case Map screen 27](#_Toc58231775)

[Figure 4‑10 Use-case Trips Screen 27](#_Toc58231776)

[Figure 4‑11 Web prototype 27](#_Toc58231777)

[Figure 4‑12 Data flow 27](#_Toc58231778)

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

# 1. Introduction

In this section of this report I will be laying out the proposal for the project. 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 completed in doing the proposed project. The Scope of the project will lay out what the project is and also what it is not. Finally, a thesis Roadmap will give an overview as to the structure being taken whilst writing out the rest of the project.

## Project Background

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

In my research I identified two similar mobile apps

* Road Trippers has a similar concept, the main difference being that they create curated guides to specific areas, all within the United States, Canada, Australia and New Zealand. This style of curation over automation means that users are being suggested the same trips and have a more limited range of options for their trip. It is also possible to create personal trips but this is not the focus of the app. In my research this functionality is lacking in many ways.
* Google Trips also has similar app but puts focus on visiting specific areas or cities. This is different in that the journey is not a focus to the trip, but that the journey is an afterthought, i.e. travel is only treated as travel and not a part of the experience.

(United States Patent No. 12/113,911, 2010) Fitzgerald Chas an abandoned patent application for a recommender system designed for navigation, which would be similar to one part of my proposed project. Although it specifies itself based on a property database, rather than looking for open license data.

Further Reading into a Personalised trip recommender based on points of interest proved to be similar in that time is a focus in their recommender system and this would relate more closely to how I wish to implement the recommender system into the navigational system for on the go recommendations.

Roadtrippers. 2020. *Discover The Best Diners, Scenic Spots, Attractions, Hotels, And Much More With Over A Million Amazing Points Of Interest. Roadtrippers Has Places You Just Won’T Find Anywhere Else!*. [online] Available at: <https://maps.roadtrippers.com/> [Accessed 12 October 2020].

Google Trips. 2020. Looking for your trips and travel research? Sign in to the Google Account where you receive confirmation emails to see your booked reservations and travel [ online] Available at: < https://www.google.com/travel/[/> [Accessed 12 October 2020].

Lim, Kwan Hui & Chan, Jeffrey & Leckie, Christopher & Karunasekera, Shanika. (2018). Personalized trip recommendation for tourists based on user interests, points of interest visit durations and visit recency. Knowledge and Information Systems. 54. 375-406. 10.1007/s10115-017-1056-y.

## Project Description

An overview of the project

Include a diagram

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.

### Smart Objectives



Figure 1-

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

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

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

Smart Objectives

* Evaluate Similar applications through the use of Nielsens Heuristics
* Generate a list of requirements based off the heuristics and the projects goal
* Review possible technologies that could be used to created the proposed system based
* 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
* Develop the Collaborative Recommendation System
* Android application

## Project Scope

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

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

## Thesis Roadmap

One sentence summary of the following chapters

# 2. Literature Review

## 2.1. Introduction

In this chapter multiple projects which relate to this one will be discussed in order to gain insight into the reasons for their User requirements. This is very useful as we know that 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.

### 2.2.1 Nielsen

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 I have altered to suit the needs of these types of navigation apps.

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

## 2.2. Alternative Existing Solutions to Your Problem

The Software being investigated and reviewed with the help of these Heuristics included Road Trippers and Google Trips. Grading each application will be from 1 to 5 on each of the 10 Heuristics listed above, with a small explanation for each scoring. In order to do this, I will try and perform the same of similar trips on a small scale in my locality. I hope this will give me enough to work with regarding the evaluation. The grading system scores:

1. Unusable
2. Bad design implementation
3. Usable but nothing more or less
4. Good implementation of design
5. Excellent implementation of design

Google Trips

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 regards 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.

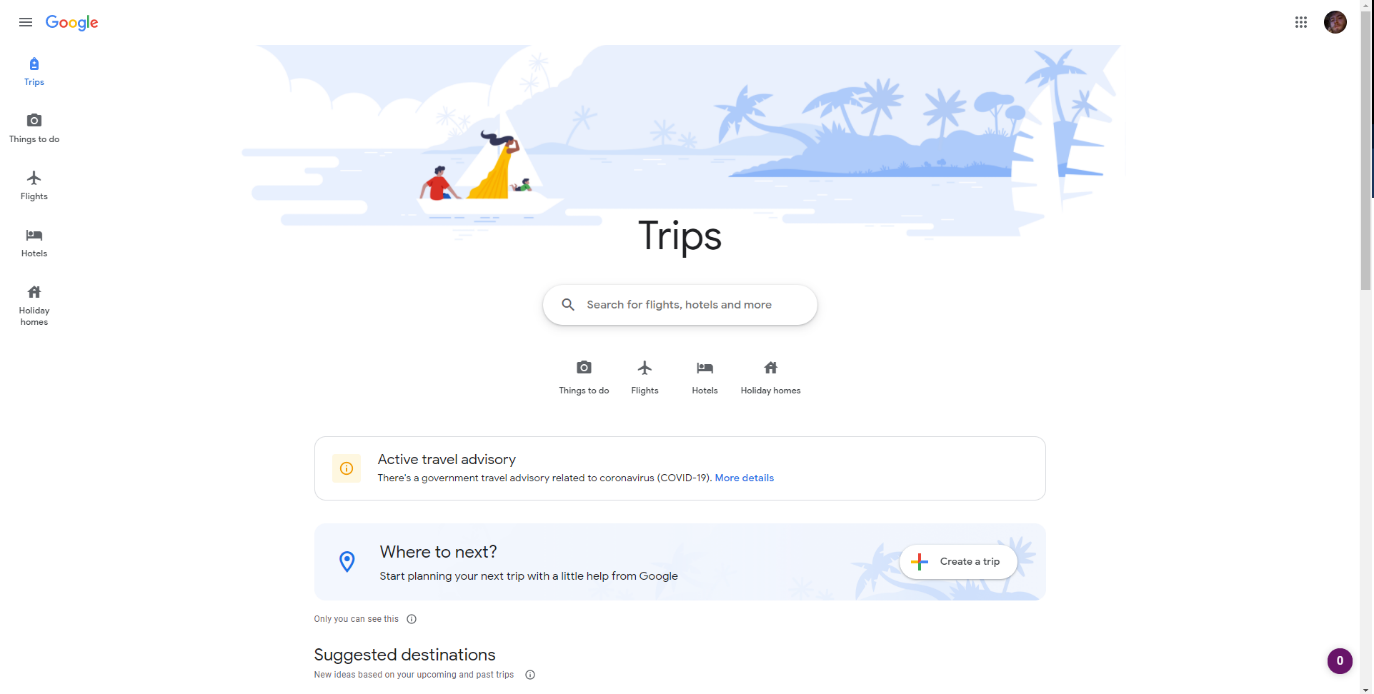
Overall the separation between the web application and the fact that google maps is supposed to be the mobile frontend to use google trips to be very awkward and made it difficult to use while on real-world trips.  


Figure ‑

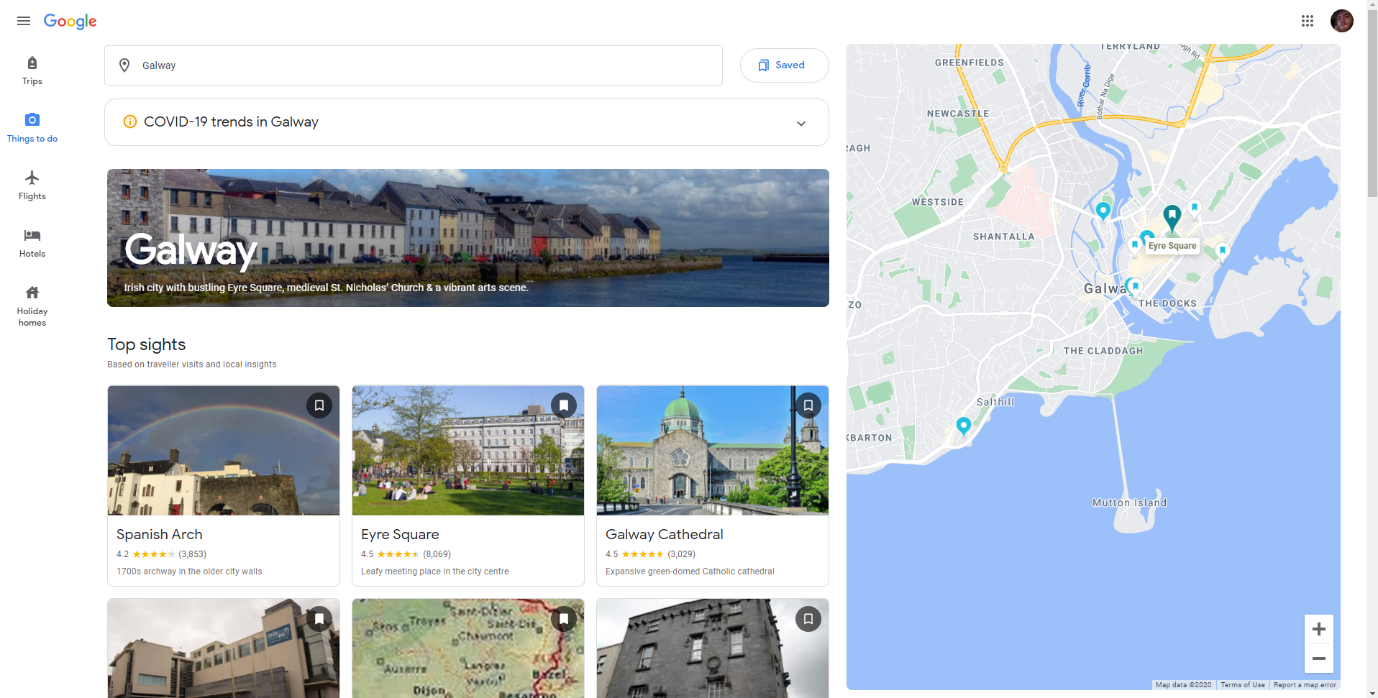


Figure ‑

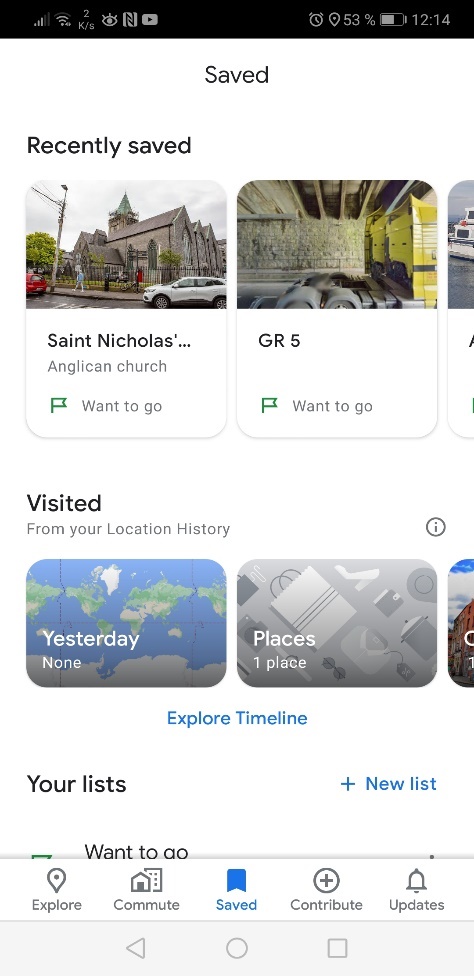


Figure ‑

Road Trippers

1. **Visibility of system status**: 3  
   The separation of the full road trip and having to route to individual points makes it harder to fully realise your progress of the whole trip. This is most evident when multiple stops are nearby each other.
2. **Match between system and the real world**: 4  
   Much in the same way as the previous app, this had excellence relation between the system and the real world.
3. **User control and freedom**: 2  
   The creation of trips is where it was very lacking. While it is possible the app seems to aim at delivering curated trips to the users. Although the user must create an account in order to use any of the curated trips.
4. **Consistency and standards**: 5  
   Both the mobile and web app have been rigorously designed and match up quite well. There are very few inconsistencies.
5. **Error prevention**: 2  
   the navigation is badly implemented into the apps workflow. Each specific stop needs to be selected and routed to, rather than a more integrated feel. This would be difficult for use while driving.
6. **Recognition rather than recall**: 5  
   The layout and use of symbols were used in a similar way to google trips but in general the limits of the system to only roads trips made for an more specific but easier experience.
7. **Flexibility and efficiency of use**: 4  
   The creation of new route is quite limited on the mobile application.
8. **Aesthetic and minimalist design**: 5  
   The Aesthetic of both the web and mobile app had great attention to detail and provided a good experience.
9. **Help users recognize, diagnose, and recover from errors**: 4

Most of the app is very clearly laid out and has an intuitive nature.

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.

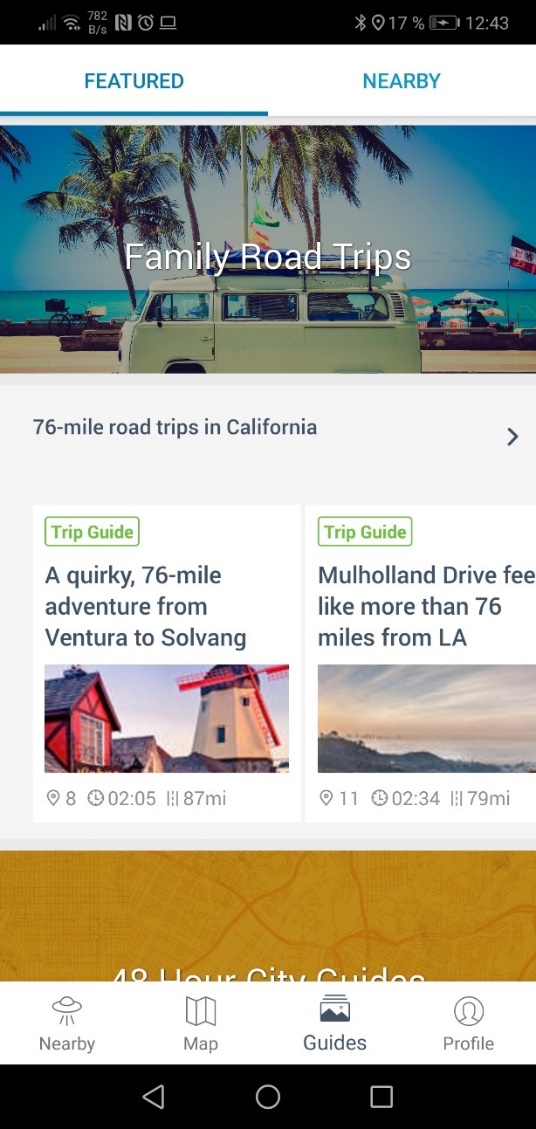


Figure ‑

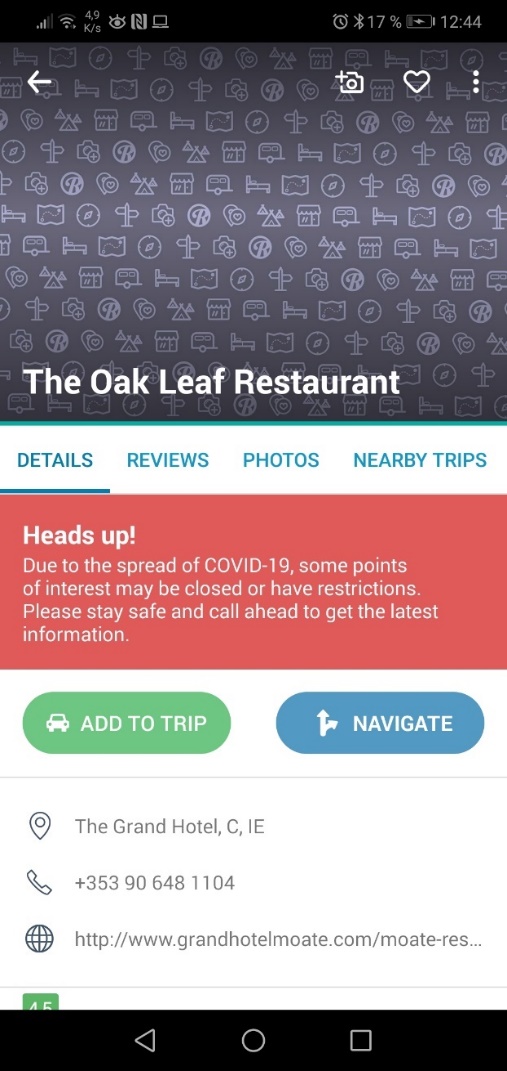


Figure ‑

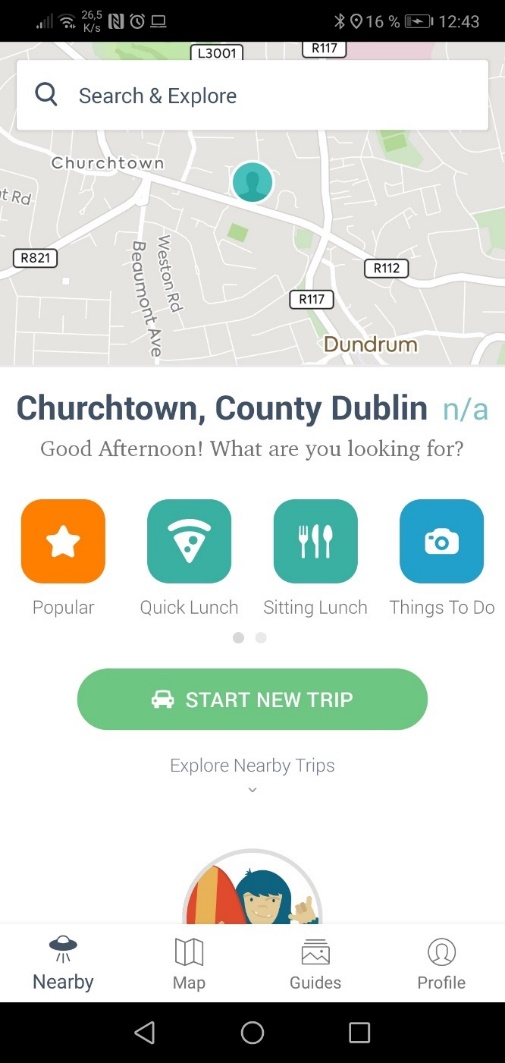


Figure ‑

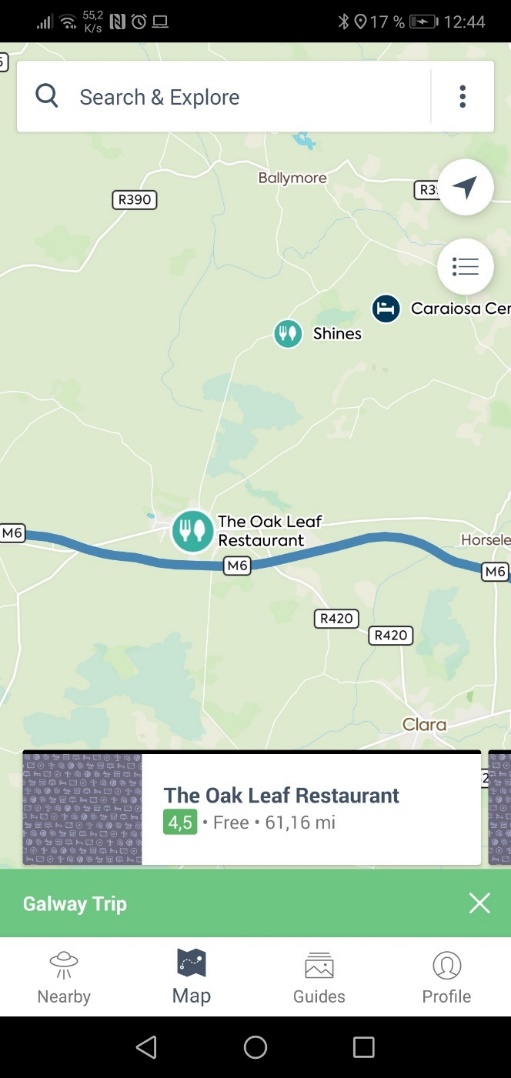


Figure ‑

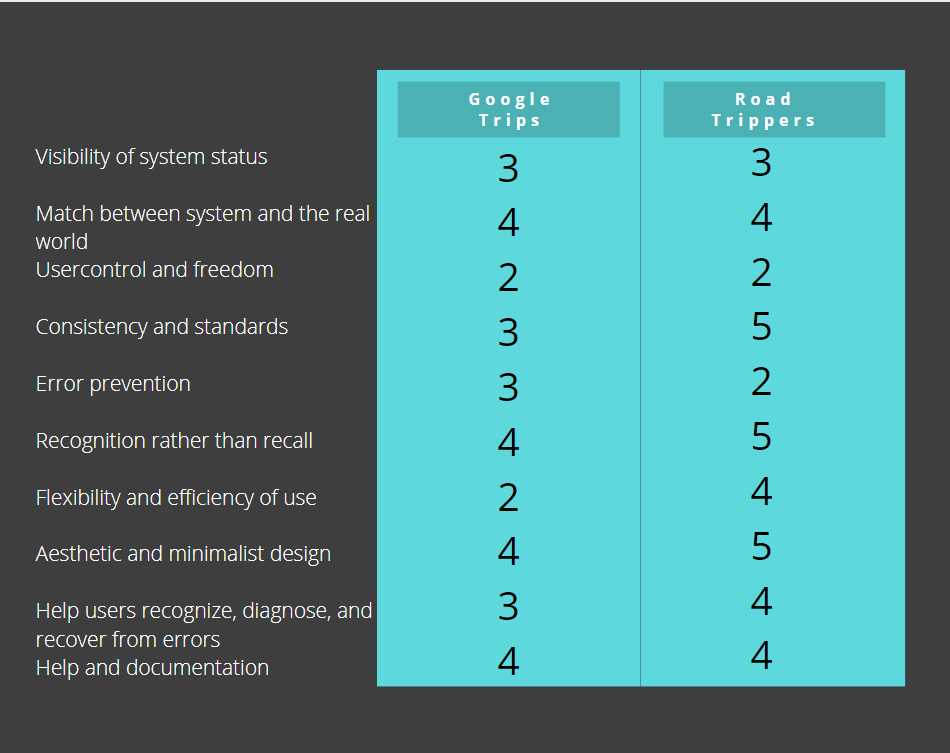


Figure ‑

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

From here we can draw some conclusions on the requirements the system will need based on these evaluations on similar systems. This will be discussed in the conclusion of the chapter

## 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 three sections: Front-End, Back-End and the API sections, respectively.

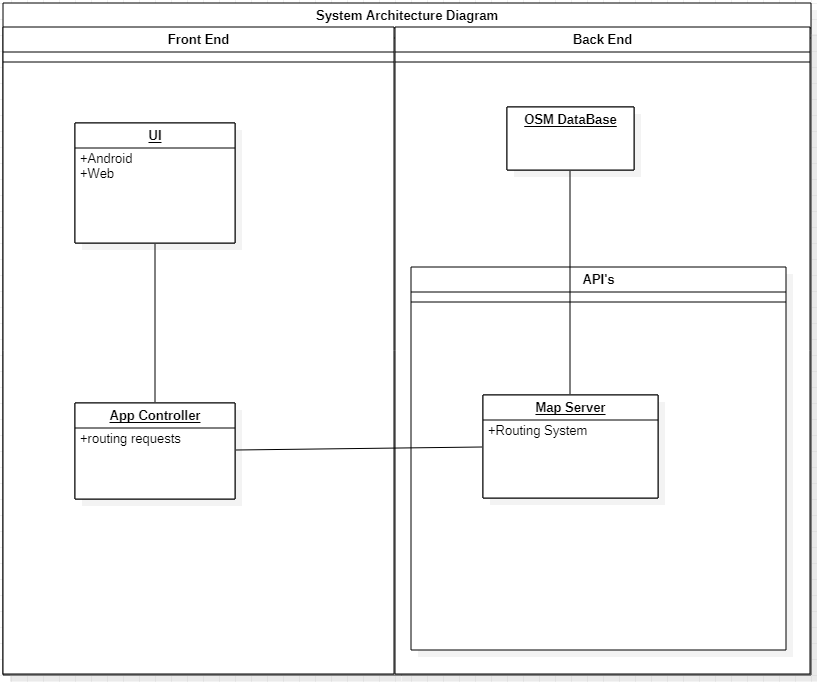


Figure ‑

## Front-End

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

### 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.

### Ionic & React Native

Ionic is a framework based on angular JS. This means languages like HTML5, CSS and JavaScript are used. It offers lots of pros like Faster development time, good reliability and great design.

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.

### 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.

## Back-End

### Oracle

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

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

### SQL lite

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

### 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.

### Django

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

## 2.4. Other Research

### Mobile App Environment

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

Native or hybrid

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

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

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

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

### 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. WE can further subdivide 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 we have 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.

## 2.5. Existing Final Year Projects

**Project 1**

**Title:** Travel Assistant

**Student:** Cillian McCabe

**Description (brief):**

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

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

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

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

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

**Project 2**

**Title:** Driving Instructor Assistor

**Student:** Baolach Morrison

**Description (brief):**

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

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

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

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

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

## 2.6. Conclusions

Requirements

* 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)

UI

Map Service geolocation

Real time navigation

Voice commands

Multiuser inputs/ user profiles

Hybrid app

Database/cloud service

Programming language

# 3. Experiment Design

## 3.1 Introduction

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

## 3.2. Software Methodology

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

### 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.

### Agile

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

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

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

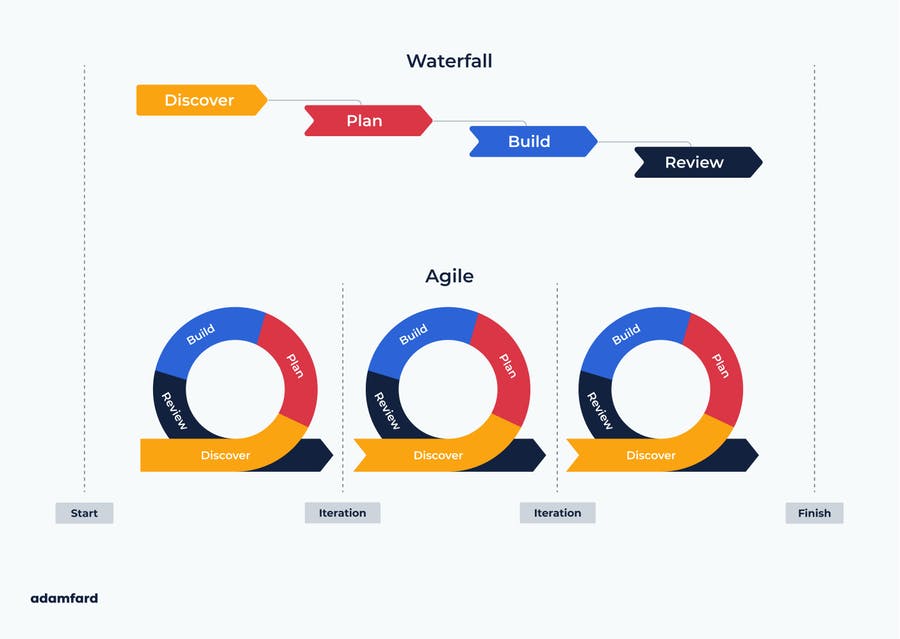


Figure ‑ Waterfall and Agile Design methodology

### Extreme

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

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

* Communication
* Simplicity
* Feedback
* Courage
* Respect

Its cycles consist of the following steps

* Plan
* Design
* Code
* Test
* Deliver/increment

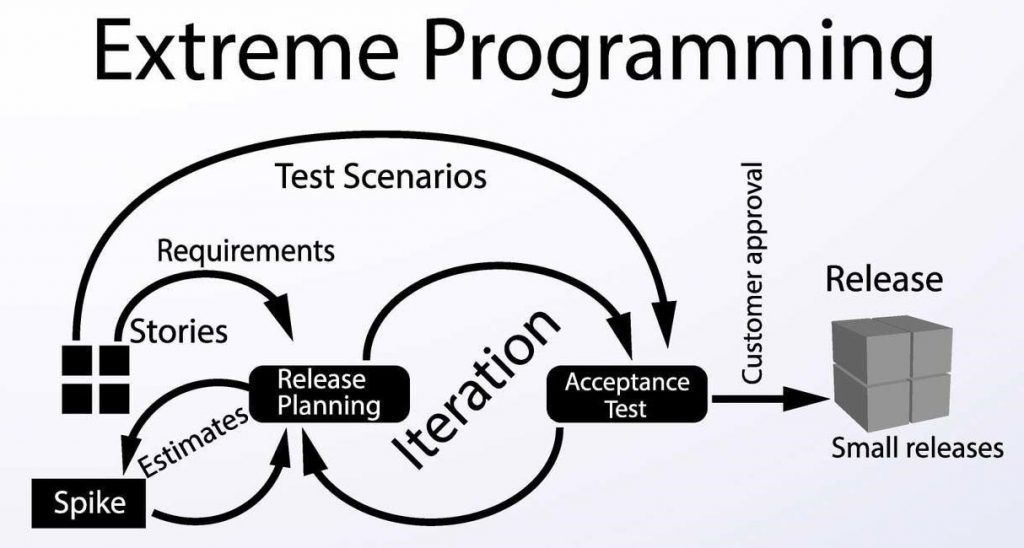
One problem that arises when considering XP is the pair-programming concept as I will be working the coding myself this will not be possible. 

Figure ‑ Extreme Programming

### SCRUM

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

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

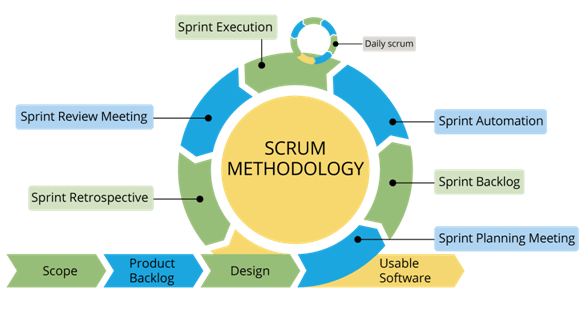


Figure ‑ Scrum Design Methodology

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 I see XP’s shorter iterations with smaller deliverables to be more suitable to my 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.

### Smart Objectives



Figure ‑ Smart Objectives

Smart Objectives are one of the ways available to define separation between a project’s tasks. As shown in Figure 3-4 we can see that they follow 5 basics rules of definition

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

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

## 3.3. Overview of System

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

## 3.4. Front-End

Including screen prototypes and Use Cases

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

* Map
* Trip
* Profile
* Help
* Recents

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

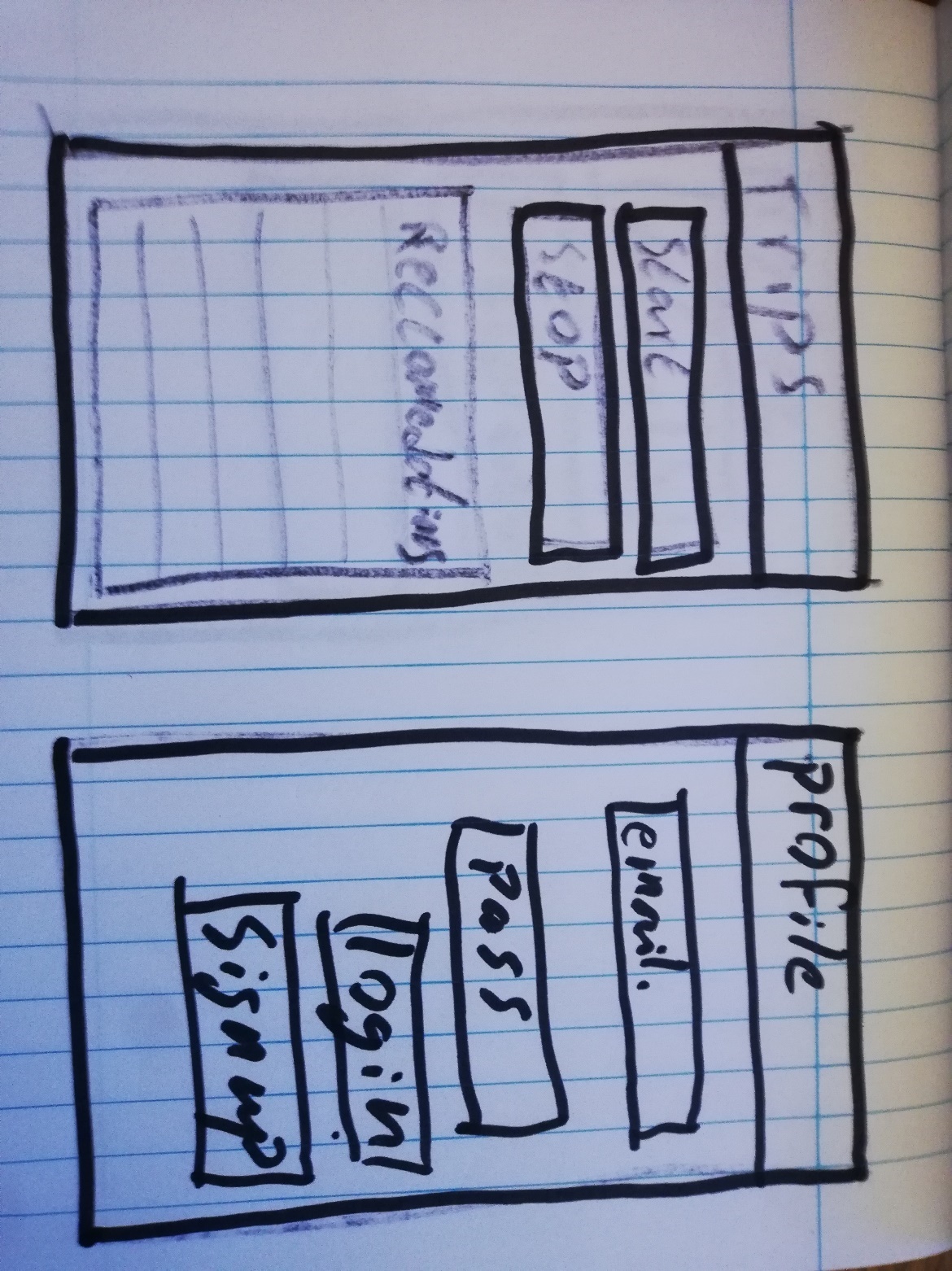


Figure ‑ Trip and Profile login screens

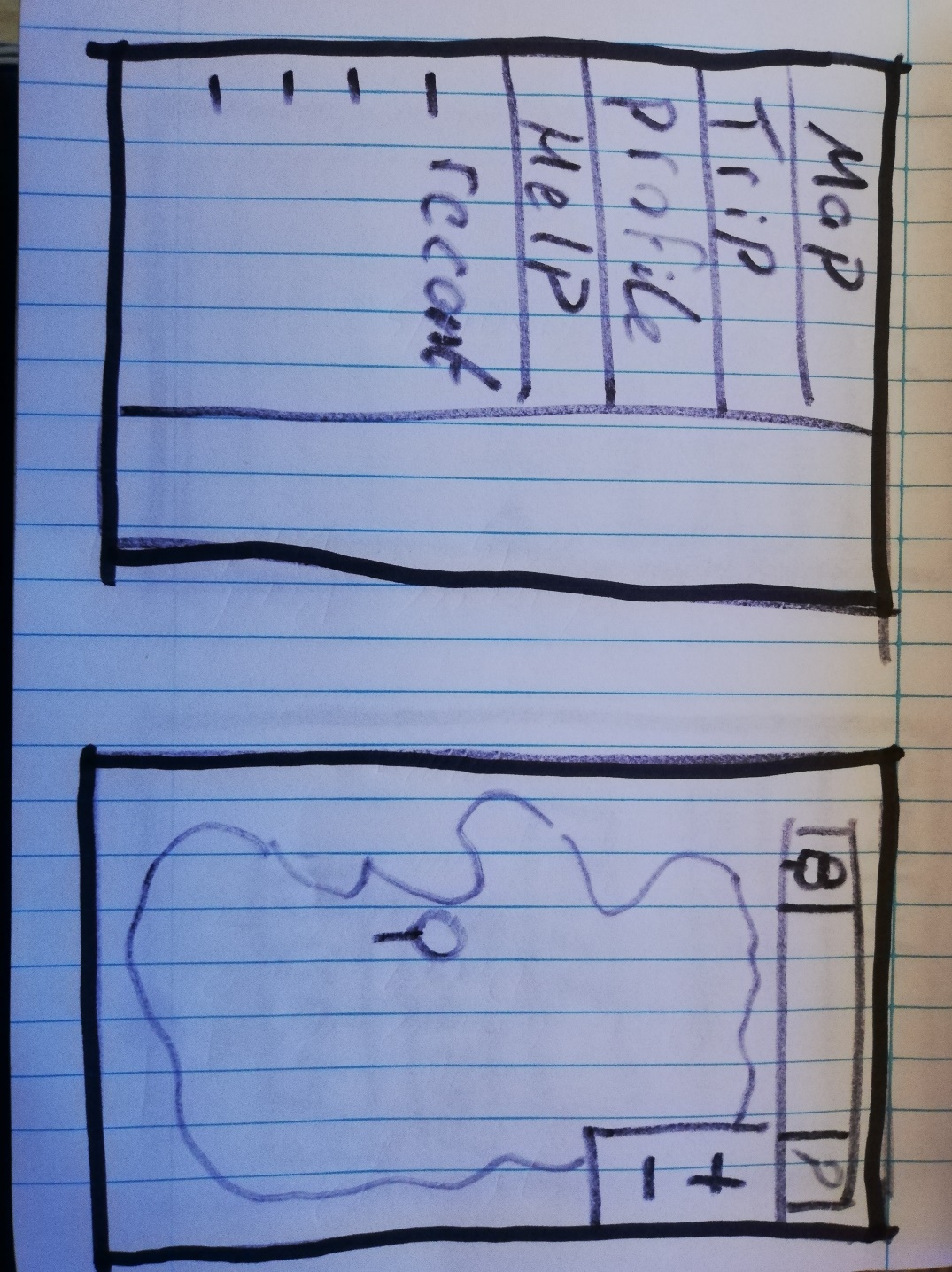


Figure ‑ Menu and Map screens

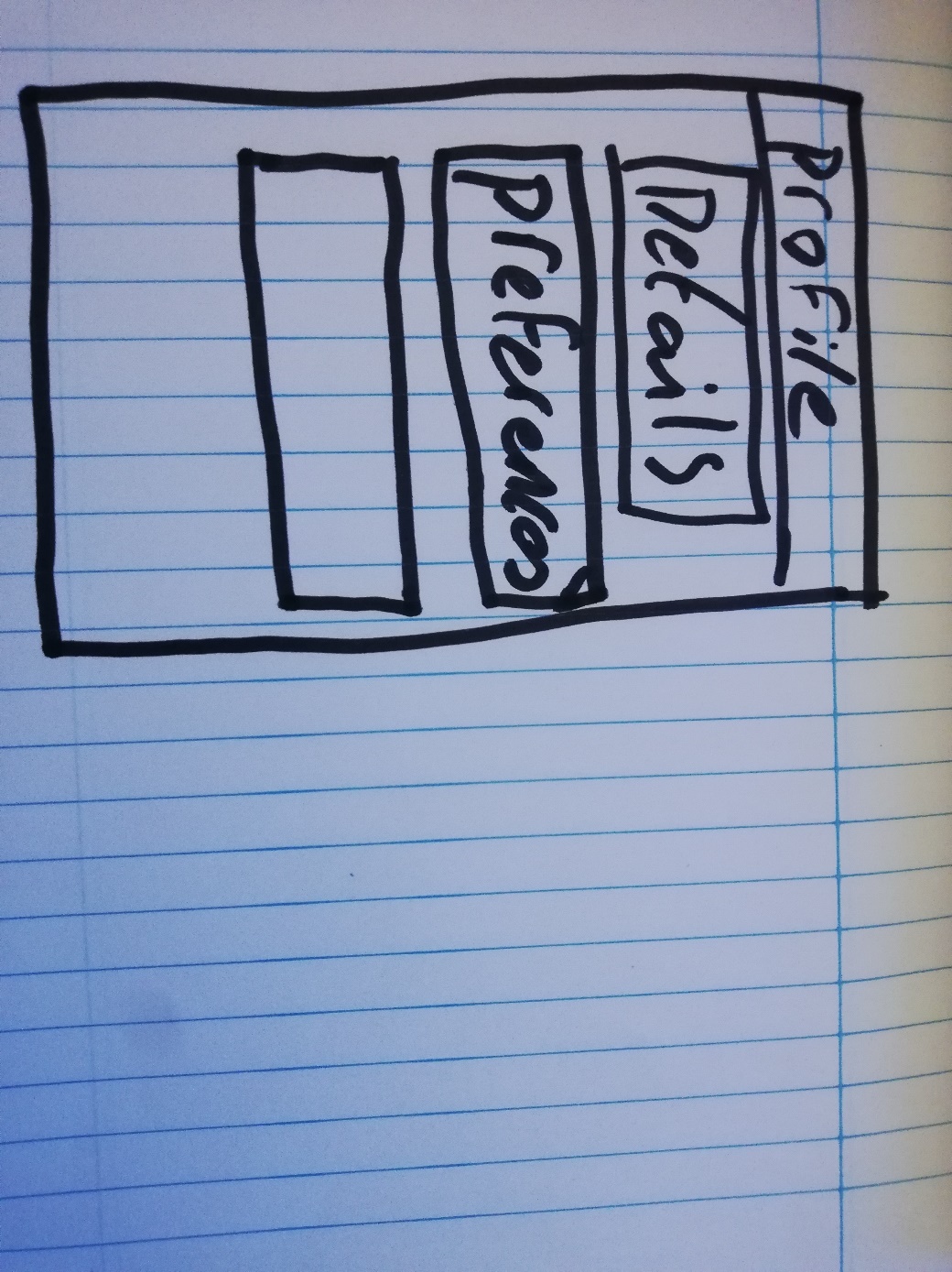


Figure ‑ Profile Screen

### 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.

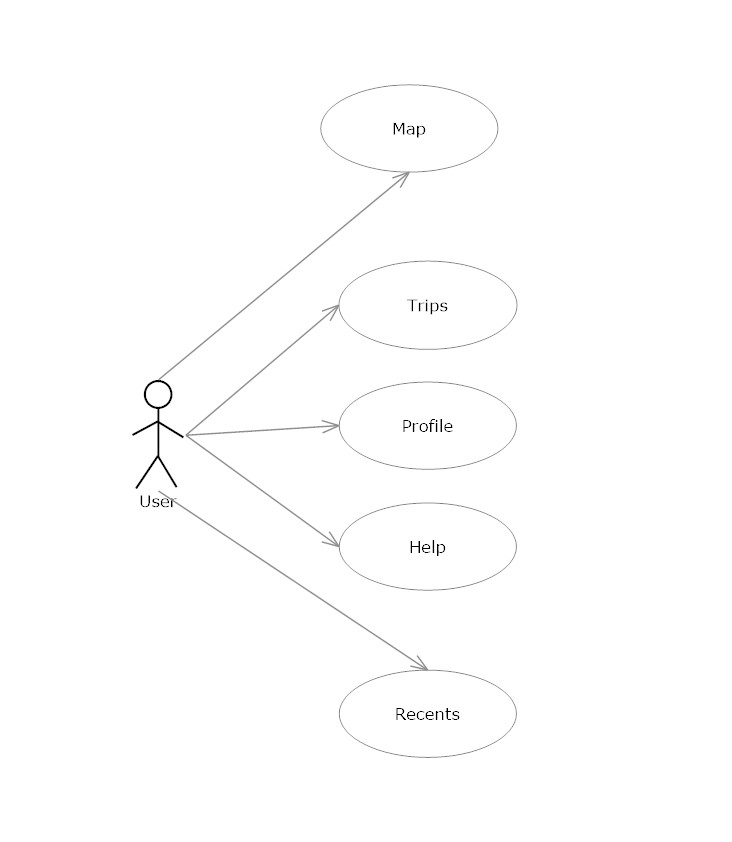


Figure ‑ Use-case Menu Screen

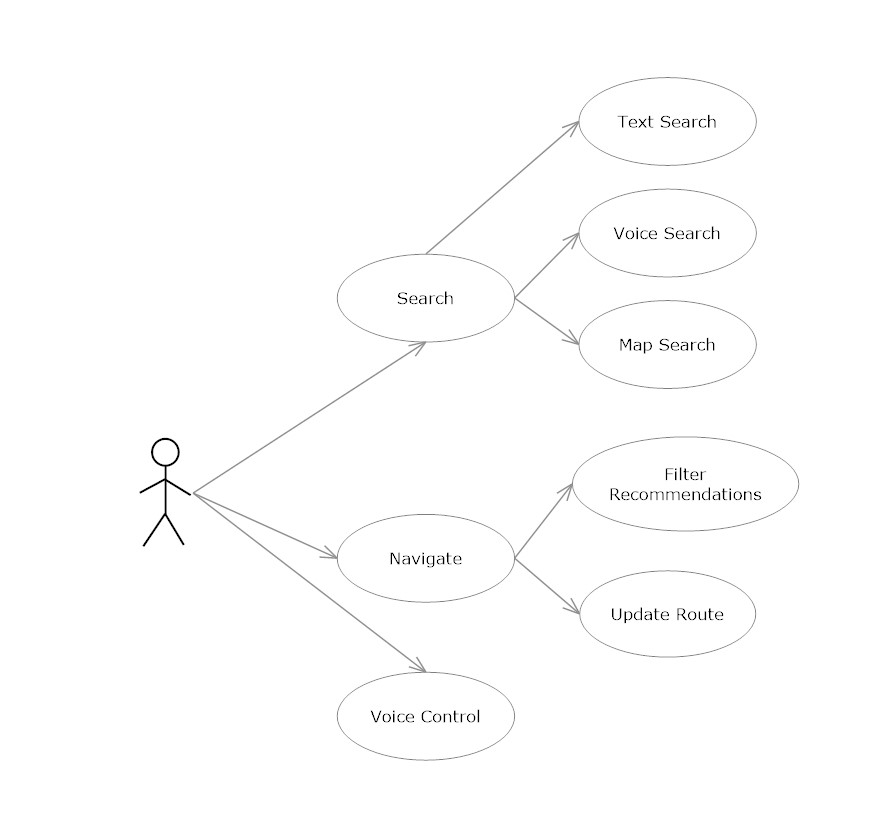
Figure 3-8 Describes the Menu functions which allows the User to navigate through the proposed application. 

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.

### Horizontal Prototype

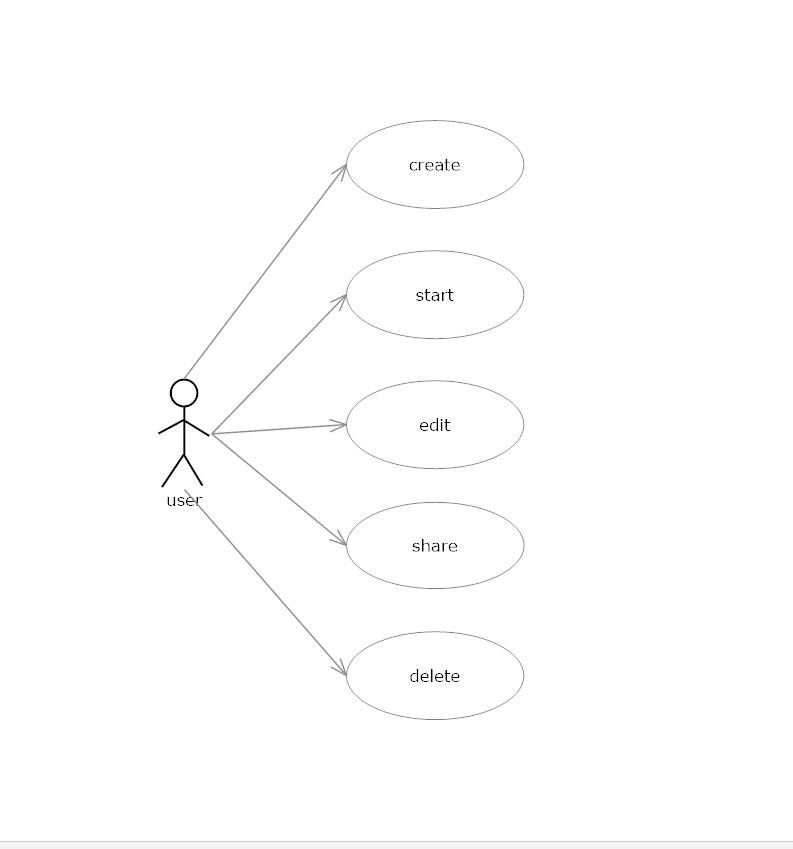


Figure ‑ Use-case Trips Screen

Figure 3-10 above shows the users options when in the trips section of the proposed application.

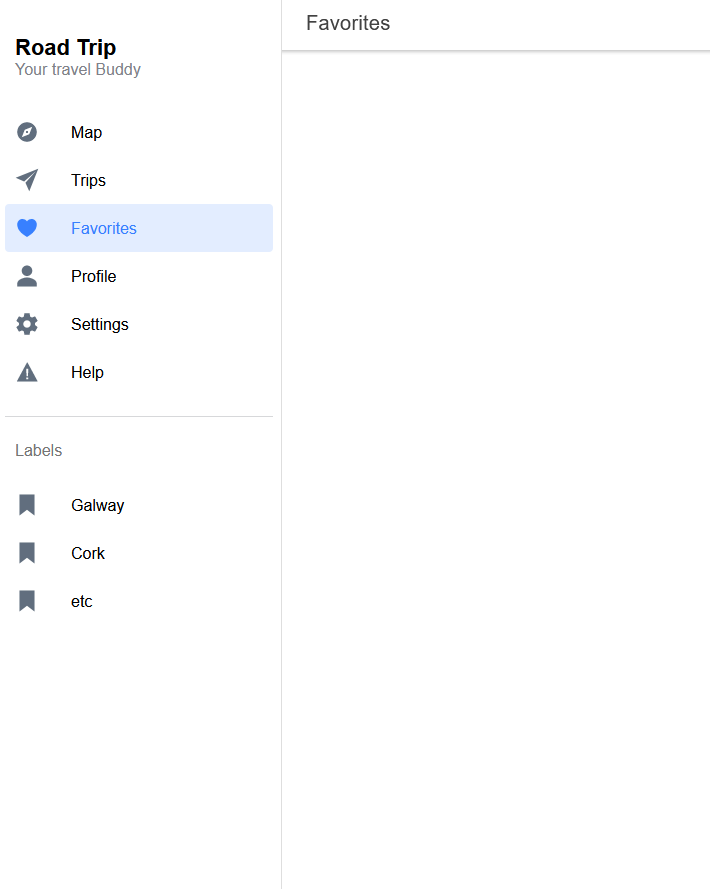


Figure ‑ Web prototype

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.

### Django

## 3.6. Back-End

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

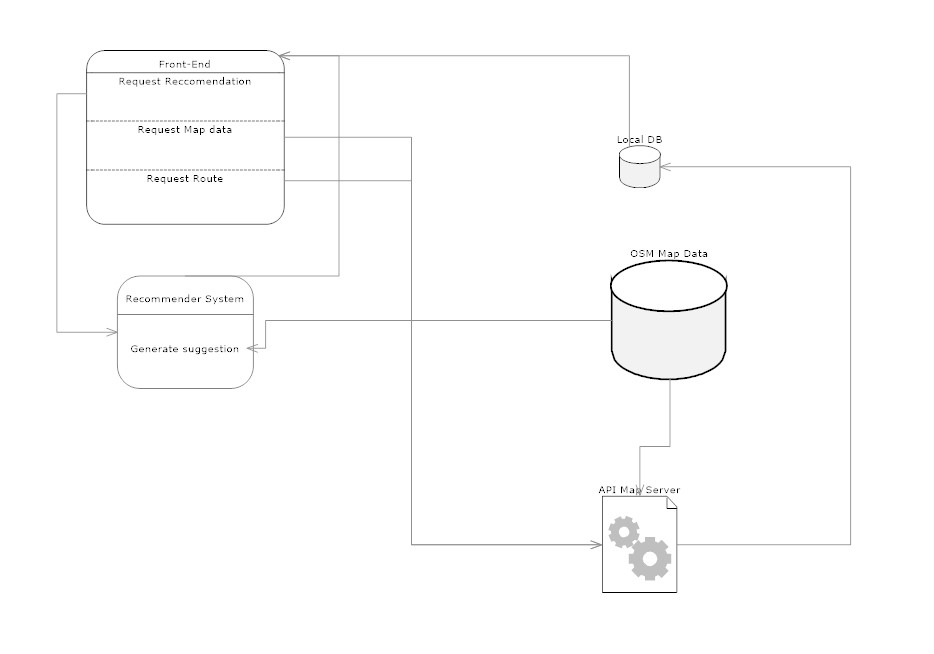


Figure ‑ Data flow

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.

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.

Including ERDs, and maybe ISDs

## 3.7. Conclusions

# 4. Experiment Development

## 4.1. Introduction

This section will speak about the processes involved in the development of the proposed application.

## 4.2. Software Development

## 4.3. Front-End

## 4.4. API’s

## 4.5. Back-End

## 4.6. Conclusions

# 5. 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

### Black Box Testing

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

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

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

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

* Equivalence partitioning  
  Divide test conditions into groups and from each group test only one condition. It assumes that all the conditions work in the same way. If that condition from a group works, it is assuming all conditions from that group work. Good test coverage.
* Boundary value analysis  
  Using this technique test conditions are used to create partitions. Test cases are designed by getting the boundary values of the partition. Test conditions either side of the boundary are boundary values. If both values are valid, we have valid partitions if not the partitions are invalid. This reduces testing.
* Decision table  
  Using this technique we take all conditions as inputs and actions as outputs. This is used when analysing logical relationships between inputs and outputs. This is used when dealing with multiple inputs. Listing all possible inputs in combination ensures that every combination has been tested.
* State transition  
  This is used when testing the transitions of a system. !!!write more
* Exploratory testing  
  This is a method of testing which is just the exploration of a system without an y prior knowledge of the system with the intension of finding functionality and possible errors. This can be similar to a trial-and-error approach
* Error guessing  
  This technique does not have any specific design approach but uses testers with prior experience and may know common pitfall of applications designs in that area.

### White Box Testing

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

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

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

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

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

* Conditional Coverage
* Multiple condition coverage
* Path Coverage
* Function Coverage

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

## 5.3. System Testing application

### 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.

### Metric testing

## 5.4. Conclusions

# 6. Conclusions and Future Work

## 6.1. Introduction

## 6.2. Conclusions

It was good

## 6.3. Future Work

# Bibliography

Fitzgerald C, V. D. (2010, December 9). *United States Patent No. 12/113,911.*

Rocca, B. (2019, June 3). *Introduction to recommender systems*. Retrieved from Towards Data Science: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada