WhereNext - An Autonomous Interactive Tourist Attraction Recommender Platform

Charlie Maguire, BAI

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master in Engineering (Computer and Electronic Engineering)

Supervisor: Ciaran Mc Goldrick

April 2022
Declaration

☒ I agree that this thesis was completed in line with the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

☒ I have completed the Online Tutorial on avoiding plagiarism ‘Ready Steady Write’, located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

☒ I agree that this thesis will not be publicly available but will be available to TCD staff and students in the University’s open access institutional repository on the Trinity domain only.

Signed:
Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Charlie Maguire

April 19, 2022
Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Charlie Maguire

April 19, 2022
WhereNext - An Autonomous Interactive Tourist Attraction Recommender Platform

Charlie Maguire, Master in Engineering
University of Dublin, Trinity College, 2022

Supervisor: Ciaran Mc Goldrick

The Irish tourism industry is an important part of Ireland’s economy and identity. Figures within the industry are constantly looking for new technologies and innovative solutions to the industry’s biggest challenges. The goal of this project was to design, develop and test a solution for improving a key facet of the tourist experience in Ireland. This goal was achieved through the development of an attraction display device for collecting information from tourists and providing them with recommendations based on their travel history, and a server for coordinating these devices and providing them with additional resources. The WhereNext platform is capable of generating accurate recommendations for tourists while keeping their identity private, and without the need for tourists to register to the platform via an app or website. The proposed platform will allow its operators to more successfully manage tourist attractions in a region, as well as create more informed plans for future developments in the area. This project features a lot of potential for future development, including application to industries outside of tourism.
Acknowledgments

Thanks to my parents for their constant love, support and encouragement over the past 23 years. I would not have been able to enjoy this journey as much without knowing that they would always be there for me.

Thanks to my brothers and sister for always helping me feel at home whenever I arrived back from weeks away at college.

Thanks to my friends for providing me with much needed downtime and respite throughout the past few months.

Thanks to Ciaran, my supervisor, for all of his guidance and insights. I have been able to achieve something much more ambitious with this project than I originally imagined thanks to his help.

Charlie Maguire

University of Dublin, Trinity College
April 2022
Contents

Abstract iii
Acknowledgments iv

Chapter 1 Introduction 1

Chapter 2 State of the Art 3

2.1 Background .................................................. 3
2.2 Related Works .................................................. 4
   2.2.1 Tourism Recommender Systems ......................... 4
   2.2.2 User Privacy and Identification ....................... 6
   2.2.3 Communications .................................. 7
2.3 Summary ...................................................... 8

Chapter 3 Design 11

3.1 Overview of the Approach .................................. 11
3.2 Attraction Displays ......................................... 12
   3.2.1 User Identification .................................. 13
   3.2.2 Display ............................................. 14
   3.2.3 Communications .................................. 16
   3.2.4 Power ............................................. 17
3.3 Recommendation Server .................................... 18
   3.3.1 Communications .................................. 18
   3.3.2 Recommender System ................................ 19
   3.3.3 Ratings Database .................................. 21

Chapter 4 Implementation 22

4.1 Implementation Overview .................................. 22
4.2 Attraction Displays ......................................... 22
   4.2.1 User Identification .................................. 22
### Table of Contents

#### Chapter 4

4.2.2 Display ................................................. 24  
4.2.3 Communications ....................................... 25  
4.2.4 Power ................................................. 27  
4.3 Recommendation Server ................................. 27  
4.3.1 Communications ....................................... 27  
4.3.2 Recommender System ................................. 29  
4.4 Ratings Database ......................................... 32  

#### Chapter 5 Evaluation  
5.1 Bluetooth .................................................. 34  
5.1.1 Experiment .............................................. 34  
5.1.2 Results .................................................. 34  
5.2 User Identification ........................................ 35  
5.2.1 Experiment .............................................. 35  
5.2.2 Results .................................................. 36  
5.3 Recommender System .................................... 38  
5.3.1 Experiment .............................................. 38  
5.3.2 Results .................................................. 38  
5.4 Small Scale Deployment ................................. 39  
5.4.1 Experiment .............................................. 39  
5.4.2 Results .................................................. 40  

#### Chapter 6 Conclusions & Future Work  
6.1 Conclusions ................................................ 42  
6.2 Future Work ............................................... 44  

#### Bibliography  

vi
List of Tables

2.1 Comparison of POI Recommender Systems .................................. 10
5.1 Bluetooth Distance Test Results ............................................. 35
5.2 User Identification Test Results ............................................. 36
5.3 Matrix Factorization Implementation Comparison ....................... 39
List of Figures

3.1 Full Platform Overview ............................................. 11
3.2 Pycom FiPy and Pysense 2.0 X Boards .......................... 12
3.3 Resistive Touch Display ............................................ 14
3.4 Mock-ups of Display User Interface ............................. 15
3.5 Attraction Display Solar Panel and Battery .................... 17
3.6 Section of Google Sheets Ratings Database .................... 21
4.1 FiPy connected to the Display via a Breakout Board .......... 24
4.2 Received User Recommendations Printed to Console .......... 25
Chapter 1

Introduction

The Irish tourism industry is an important part of the Irish economy. Every year, millions of tourists travel across the island, generating billions of Euro in revenue for the country (Fáilte Ireland 2021). Irish tourism boards and county councils are constantly looking for ways to improve the tourist experience, and regularly look towards the latest developments in technology to do so. In past years, these innovations have manifested an electronic travel ticket system in the form of Leap cards to simplify the usage of public transport, and high-tech attractions and events, such as coordinated night time displays composed of hundreds of synchronised drones. Recently, tourism boards have been expressing interest in a platform for providing tourists with travel recommendations to further enhance the Irish tourism experience.

The primary goal of this project was to design, develop and test a travel recommendations platform to simplify the process of identifying attractions of interest, and making recommendations and offering guidance to tourists that optimises both their experience and that of the platform operators. This goal can be divided into two primary components. The first component is the development of a series of interactive attraction displays that can be deployed to tourist attractions around a region. The second component is a server that coordinates these devices and provides them with additional storage and processing resources. Together, these components compose the WhereNext platform. Both of these primary components can be further divided into their own sub-components.

The attraction displays will present tourists with recommendations based on their travel history and their ratings of attractions they have visited. The displays will be composed of a user identification system so tourists can be identified as they travel between locations, an interactive display for receiving tourist feedback and providing information, and a robust communications system to allow the attraction displays to interact with the server.
The server will contain a database for tracking user ratings and travel history. A recommender system will utilize this database to generate recommendations for users based on their shared preferences. The server will also feature a robust communications system to manage the requests coming from all deployed attraction displays in a region.

Such a platform will benefit tourists over traditional recommender systems hosted on websites and smartphone apps as it removes all points of friction that prevent users from engaging with those systems, including downloading apps, creating accounts or filling out profiles. The only requirement from a tourist to directly benefit from the platform is for them to interact with an attraction display. This platform also provides tourists with improved privacy over traditional recommendation services as it does not require any personal data from users to operate.

The platform will also benefit the tourism boards or councils that deploy it. Recorded tourist travel patterns can be analysed to influence new travel routes between popular locations. User travel histories can also highlight attractions that are regularly overlooked by most tourists. Tourist types could be identified and package deals created based on these types, e.g. a ticket that provides access to all museums in an area. The recommender system can also be adjusted to highlight certain attractions if, for example, there is an event on that a tourism board would like to encourage tourists to visit.

The remainder of this report will detail the design, development and testing of the WhereNext platform. Chapter 2 details the background of the problem area, and discusses the state of the art regarding tourist attraction recommender systems. Chapter 3 breaks down the design of the various components and sub-components of this platform, justifying the decisions made during its development. Chapter 4 highlights how each of the design decisions made during the development process have been implemented into the platform. Chapter 5 covers the testing undertaken to ensure the platform components operate to a high standard. Finally, Chapter 6 concludes the report with a discussion of the conclusions and takeaways from the design and development of this platform, including recommendations for future work in the area.
Chapter 2

State of the Art

To gain a more complete understanding of the project’s problem space, a thorough investigation of the state of the art was conducted. Firstly, the Irish tourism industry was investigated to understand its strengths and weaknesses, and uncover any areas with potential for improvement. Past attempts at tourism recommender systems were explored to assess different approaches available for tackling the problem of generating recommendations. During this research, privacy was identified as a common theme of recent tourism recommender systems, so user privacy and identification systems were also investigated for their potential for inclusion in the project. Finally, multiple communication methods were explored to determine their suitability for this application, as robust communications would be required for the platform to function. The details of the research conducted for this project are presented below.

2.1 Background

The tourism industry is a key component of Ireland’s economy. According to a 2021 report by Fáilte Ireland concerning the 2019 Irish tourism industry, the number of total oversees visitors into the country continues to climb every year. This number dropped in 2020 and 2021 due to the Covid-19 pandemic, but is expected to increase again as the virus comes under control. The majority of visitors are travelling into the country for vacations, rather than work or visiting family and friends. The most popular activities for these visitors include hiking and visiting castles, parks, gardens and heritage centres.

71% of tourists plan their trips using online resources. The most popular destinations see very large numbers of tourists, but the number of visitors begins to drop off sharply as attractions decrease in popularity. For example, the number one fee-charging attraction is the Guinness Storehouse, which had 1.7 million visitors in 2019. Meanwhile, Blarney...
A 2020 report from the Irish Tourism Industry Confederation (ITIC) proposed solutions for helping the Irish tourism industry recover from the impact of the Covid-19 pandemic, and grow beyond where it had been before 2020. A component of the plans involved transforming Ireland into a “world-class destination” for international tourists. This would be achieved through an increase in oversees marketing, as well as an introduction of more accessible tourism facilities across the country.

The report also suggests a focus on Ireland as a “world leader in digital capability” during this revitalisation of the industry. Many of the world’s largest tech companies reside in Ireland. The ITIC believes this should be represented in the technology employed by the Irish tourism industry to reinforce Ireland’s position as a leader in technological advancements (ITIC (2020)).

2.2 Related Works

2.2.1 Tourism Recommender Systems

Due to the ubiquity of data and the ease of its collection in the modern world, recommender systems are utilized by many industries to personalise and enhance user experiences. The most common approaches to recommender system designs are content-based filtering, collaborative filtering and hybrid filtering, as outlined by Aggarwal (Aggarwal (2016)).

Content-based filtering provides new recommendations to a user based on that user’s history and preferences. These systems are effective if little is known of the user, but information is available for the content. This approach fails if there is not enough data available for predictions, or if the available data is not suitable.

Collaborative filtering systems use the actions of similar users to provide recommendations for another. These systems are based around the principal that if users shared opinions in the past, they are likely to share opinions again in the future. The primary issues this approach faces are cold starts, where nothing is known about a user when they first use the system, and the addition of new content.

Hybrid filtering systems use both content-based and collaborative filtering solutions to provide predictions. This can be achieved through combining the predictions of the two systems, or by combining the systems to provide a shared prediction. This approach eliminates many of the issues found in the other two systems, but still requires a large quantity of data to be effective. Most modern recommender systems are based on this
Many successful attempts have been made to utilise recommender systems in a tourism context to generate travel recommendations for users. In a 2015 paper by Lu et al, a survey of recommender system applications was published, including a section discussing applications in the tourism industry. It found that for simple applications, such as restaurant recommendations, content based and collaborative filtering solutions were sufficient. However, anything more complicated would require a hybrid approach (Lu et al. (2015)). Chaudhari et al published a survey in 2020 that focused on tourism recommender systems and made similar findings (Chaudhari and Thakkar (2020)).

Most recommender system solutions rely on users providing feedback and ratings in order to match them with similar users. This feedback is often obtained through a mobile app or website (Kbaier et al. (2017)). Other solutions collect user information in more novel ways, such as by analysing a user’s social media profile (Figueroedo et al. (2018)). These approaches provide good results, but don’t allow for users without smartphones or internet access while travelling.

Kenteris et al proposed setting up wireless sensor networks at tourist locations to allow travellers to provide feedback from their smartphones while at an attraction (Kenteris et al. (2010)). Recommendations in this system are made based on live and historic feedback, with more weight given to recent ratings. LARS, a recommender system developed by Levandoski et al, accounts for users’ current location and travel time when making recommendations, alongside user feedback (Levandoski et al. (2012)). Costa et al took a similar approach to this, but also included user context. This allowed for tourists to describe what type of attraction they were interested in next, for example, a walk, coffee shop or restaurant, to further customise their recommendations (Costa et al. (2013)). Each of these approaches use temporal and spatial information alongside user ratings to improve the accuracy of recommendations.

Another solution for recommendation generation is based around the idea of social commerce (Esmaeili et al. (2020)). These models use the concept of a social network, along with assigning users ‘trust’ and ‘reputation’ scores, to weigh the impact of their feedback on new recommendations. This gives certain users’ feedback more credibility than others’, and can encourage most users to simply follow in the footsteps of ‘trusted’ users if weighting is not calibrated correctly. Location Based Social Networking (LBSN) solutions are another approach to using social networks for recommendations. In these networks, users can ‘check in’ using an app to share their location. MAPS and MARS are two examples of recommender systems based around this concept (Baral and Li (2016)) (Li et al. (2015)). By their nature, these solutions are unable to provide users with much privacy.
In recent years, papers on tourism recommender systems have placed a higher emphasis on user privacy. Wang et al proposed a solution wherein users rate locations on a mobile app. These ratings are anonymously shared with nearby users, and eventually uploaded to a server in anonymous clusters, where global recommendations can then be generated (Wang et al. (2020)). This approach however loses the ability to personalise recommendations due to the anonymity of the ratings.

Federated-learning based recommender systems have grown in popularity for many applications due to the increased privacy they can afford users. In these systems, a base recommender model is sent from a central server to each edge device in the system. These edge devices individually train the model as users interact with the system, before all of the modified models are sent back to the main server and combined to form a stronger base model. All user data is stored locally at edge devices rather than being sent to a main server (Wang et al. (2021)) (Anelli et al. (2021)). Work in this area assumes that recommendations will be provided to a tourist by a personal device they are carrying with them. This allows the system to have access to the user’s history at all times without the need for their history to leave the device. Complications are therefore introduced with this approach if the recommender devices are stationary, and users travel between them.

2.2.2 User Privacy and Identification

The rise in ubiquity of Internet of Things (IoT) devices has been met with a rise in concerns over user privacy and security. A 2018 survey by Seliem et al highlights the issues that come with providing privacy on resource constrained IoT devices (Seliem et al. (2018)). One of the major areas of concern highlighted by this survey is user identification, an area of particular interest for this project.

A common approach to privacy based user identification is through the use of Radio Frequency Identifier (RFID) tags (Finkenzeller (2010)). These systems are composed of tags and readers. Each user in a system carries around a unique personal tag. These tags are identified by readers using radio waves to allow users to engage with the system. RFID tags are particularly vulnerable to spying and spoofing attacks (Pateriya and Sharma (2011)). They also require unique tags to be provided to each user that wishes to engage with the system, so are more suitable for applications where a defined set of users will be using them regularly. A solution to user identification in a tourism context should therefore focus on identification without the need to provide users with additional hardware.

User identification using biometrics has increased in popularity through inclusion in smartphones. Fingerprint sensors are the most common type of biometric scanner found
in smartphones. These systems read a user’s fingerprints through a combination of lights and an optical sensor, or through solid-state sensor pads that can detect the ridges of a fingerprint (Uchida (2005)). Face identification in smartphones uses a front-facing camera to decompose a user’s face into separate elements, alongside machine learning algorithms to match these elements with a learned face model (Guillaumin et al. (2009)). Both of these systems require training for each user to be able to identify them. This requires a large amount of personal data to be collected for the system to function. A camera-based identifier would also be computationally intensive for a low powered device.

Bellrock is a system that uses BLE to track users in a space while providing them with privacy (Zidek et al. (2018)). An app on users’ phones broadcasts an encrypted universally unique identifier (UUID) that can only be decrypted by the server and the user. This does not provide full privacy, but the concept of collecting UUIDs using BLE has potential in this problem space. According to a 2016 survey, the average person is expected to carry 6.58 network connected devices, many of which will have BLE functionality (PwC (2016)). The idea of anonymizing BLE MAC addresses has been explored by Ali and Dyo. They found that hashing can be used to anonymize addresses with less than 1% collision rate, however the required functions are computationally expensive (Ali and Dyo (2020)). Casanova-Marqués et al have also explored this idea in the context of indoor positioning systems (Casanova-Marqués et al. (2021)). Their system employs an overseer device that generates UUIDs for each user based on their BLE advertising data, and the same UUID is generated for each user each time so they can be tracked as they move around a space.

### 2.2.3 Communications

In order to collect and generate UUIDs for tourists as they travel between sites, a series of BLE equipped devices would be required. These devices would also need to be able to wirelessly communicate amongst themselves and/or with a central server to generate recommendations and share user travel history and feedback. Pycom are one company that offer such devices (Pycom (2022)). Their FiPy board includes WiFi, BLE, LoRa, Sigfox and LTE functionality, and can be combined with a number of available shields to provide environmental sensing functionality too. There are also expansion boards for Raspberry Pi and Arduino devices that provide these boards with more communications options, such as the SODAQ SARA (SODAQ (2022)).

IoT devices can have a selection of communication options to choose between depending on the hardware available to them. LoRaWAN is a communications protocol designed for low powered IoT devices (LoRa Alliance (2017)). A network of LoRa gateways is required to transmit data between devices. There are 24 gateways set up in Ireland, but...
these are primarily located in high population towns and cities, so many more would be required to cover most tourist attractions. LoRa claims it can transmit up to 5km in urban areas, and 10 km in rural areas. Pycom have a proprietary mesh network system based on LoRa called Pymesh available for their devices, designed for easy LoRa network creation (Pycom (2022)).

Sigfox is another common communications method for IoT devices (Sigfox (2017)). Unlike LoRa, which may require a network operator to construct their own network of devices depending on the location, the Sigfox network is provided by private companies in different countries, similar to mobile networks. Users must pay a subscription to access their local Sigfox network. The network has a range of 10km in urban areas and 40km in rural land. The majority of Ireland is covered by the Sigfox network, but the regions that are not covered are remote areas that tourist attractions may reside in.

LTE is a communications standard designed for mobile devices (3GPP (2006)). Many IoT devices have the ability to connect to LTE networks. LTE coverage in Ireland is provided by three major telecommunications companies; Vodafone, Three and Eir. Accessing the network requires a SIM card, and users must pay to transmit on the network. The price will depend on the size and type of information being transmitted.

WiFi is a family of protocols designed for internet access and local area networks that is similarly as ubiquitous as LTE. These protocols are based around IEEE 802.11 (IEEE (2021)). There are a number of messaging protocols available for WiFi devices that are suitable for IoT communications, such as MQTT, WebSocket and CoAP (OASIS (2019))(IETF (2011))(IETF (2014)). Each of these communication methods rely on the device having access to a WiFi network to transmit messages.

2.3 Summary

During research of past recommender systems, no existing solutions were identified that provide recommendations to users through the use of a device located on-site at tourist locations. However, assumptions can be made about what such a device would require to generate recommendations based on the systems covered above. Users would need a way to provide feedback and receive recommendations from the device. The feedback collected should at least include user ratings of locations, but additional user context can be employed to improve recommendation accuracy. Ratings must be anonymous to ensure user privacy in these systems, which limits the contextual user information available.

A system for privacy based user identification cannot be computationally intensive if it is to be used on a low powered IoT device. Such a system should also rely on unique identifiers that users will already have on their person, rather than providing all users will
hardware that can be used to identify them.

Based on these takeaways and the research highlighted above, an outline for the WhereNext platform was produced.

Recommendations will be provided to tourists through interactive devices installed at attractions. This will allow all tourists to avail of the service, whether they have a device with internet access or not. The recommender system will be based off of a collaborative filtering approach. This should reduce complexity as this new context of recommendations from on-site devices is explored. Users will provide the system with feedback through a touchscreen display attached to the on-site device. Collected data will include user ratings and their preferences for the type of attraction they are interested in visiting next, with the potential to collect more information in future iterations of the platform. No personal information will be requested from users in order to ensure their privacy is maintained.

To identify users, the device will contain Bluetooth functionality to read BLE MAC addresses. These MAC addresses will be privatized into user IDs so that individuals cannot be identified from their ID. The platform will therefore be tracking user devices, rather than individual users. This approach allows users to be identified using hardware they will already be carrying on their person, and large amounts of personal data are not required to identify them.

The recommender devices will communicate with a main server. The server will contain a database for tracking user ratings, as well as the locations that user IDs have been identified at. The server will also host the aforementioned recommender system to reduce computational load on the attraction displays. Communications will be implemented using MQTT messaging over WiFi initially, as this method is well documented and tested. The platform will include the potential for more communication methods to be added in the future, including LoRa, LTE and Sigfox.
<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Technology</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Personalized Hybrid Tourism Recommender System</td>
<td>2017</td>
<td>Hybrid - kNN and Decision Trees</td>
<td>Website</td>
</tr>
<tr>
<td>From Photos to Travel Itinerary: A Tourism Recommender System for Smart Tourism Destination</td>
<td>2018</td>
<td>Collaborative Filtering - CNN</td>
<td>Website</td>
</tr>
<tr>
<td>A mobile tourism recommender system</td>
<td>2010</td>
<td>Collaborative Filtering - k-Means Clustering</td>
<td>Mobile App and Website</td>
</tr>
<tr>
<td>A novel tourism recommender system in the context of social commerce</td>
<td>2020</td>
<td>Social Commerce</td>
<td>Mobile App and Website</td>
</tr>
<tr>
<td>LARS: A Location-Aware Recommender System</td>
<td>2012</td>
<td>Collaborative Filtering - Spatial Ratings, kNN</td>
<td>Mobile App</td>
</tr>
<tr>
<td>Recommending POIs Based on the User’s Context and Intentions</td>
<td>2013</td>
<td>Personal Agents</td>
<td>Mobile App</td>
</tr>
<tr>
<td>A group preference-based privacy-preserving POI recommender system</td>
<td>2020</td>
<td>Collaborative Filtering - Matrix Factorization, k-Means Clustering</td>
<td>Mobile App</td>
</tr>
<tr>
<td>MARS: A multi-aspect Recommender system for Point-of-Interest</td>
<td>2015</td>
<td>Location Based Social Network</td>
<td>Mobile App</td>
</tr>
<tr>
<td>MAPS: A Multi Aspect Personalized POI Recommender System</td>
<td>2016</td>
<td>Location Based Social Network</td>
<td>Mobile App</td>
</tr>
<tr>
<td>POI Recommendation with Federated Learning and Privacy Preserving in Cross Domain Recommendation</td>
<td>2021</td>
<td>Federated Learning</td>
<td>Mobile App</td>
</tr>
<tr>
<td>How to Put Users in Control of Their Data in Federated Top-N Recommendation with Learning to Rank</td>
<td>2021</td>
<td>Federated Learning</td>
<td>Mobile App</td>
</tr>
</tbody>
</table>

Table 2.1: Studied recommender systems compared in terms of technology used for generating recommendations, and the medium through which recommendations are delivered. All studied systems rely on users having internet access and a smart device to engage with them.
Chapter 3

Design

3.1 Overview of the Approach

The design of the WhereNext platform was completed using a top-down approach. Initially, the major components of the platform were identified, followed by the ways in which each of these components would connect. The full platform overview can be seen in Figure 3.1. This overview saw some changes throughout the course of the project, but largely remains the same as its initial conception.

![Figure 3.1: Full Platform Overview](image)

There are two primary components to the WhereNext design: the attraction displays and the recommendation server. The attraction displays are devices that will be installed at tourist attractions to collect ratings from tourists and present them with recommendations on where to visit next. The recommendation server is a server that communicates with the attraction displays at each location to collect user ratings, store them, and gen-
erate recommendations for displays that request them. A deployment of this platform would involve installing an attraction display at each tourist attraction in the region, and setting up a server with the recommendation server software to coordinate the deployed displays. Users are detected by the platform through the Bluetooth devices they carry with them.

Both of these primary systems can be divided into multiple sub-components. The attraction displays contain a BLE MAC privatization system for identifying users and privatizing their IDs. A touchscreen display is used for presenting recommendations and receiving user feedback. A communications system is required for transmitting messages to and from the recommendations server. A power system is also included to provide the attraction displays with power when deployed.

The recommendation server similarly contains a communications system to interact with the attraction displays. A recommender system is used to generate recommendations for users based on their travel history and current context. The server also utilises a user ratings database to track user ratings and interactions with the system.

Each of these sub-components are discussed in more detail below to highlight their design and purpose within the larger platform.

### 3.2 Attraction Displays

![Pycom FiPy and Pysense 2.0 X Boards](image)

The majority of the functionality of the attraction displays is provided by the Pycom FiPy board ([Pycom](https://www.pycom.io/)). This is a Micropython based board designed for IoT applications. This board is used to drive the display, handle communications and detect
and privatize users. Other hardware options were investigated, including the SODAQ SARA, Raspberry Pis and Arduinos. The Pycom FiPy was chosen over these options as it comes with the built in functionality and libraries for five different communication methods: WiFi, LTE, Sigfox, LoRa and BLE. Other options either featured only some of these communication methods, or required additional hardware to implement wireless communications. The FiPy also has the capability to incorporate and drive the other components required by the attraction displays.

For development purposes, the FiPy is mounted on a Pycom PySense 2.0 X shield (Pycom (2020)). This board includes sensors to take readings from the local environment, including humidity, temperature and pressure. However, none of these features of the board are currently used by this system. Instead, it acts as a means to connect the FiPy to a computer as the FiPy does not include its own micro-USB connector.

The attraction display devices have two main modes of operation; standby mode and active mode. In standby mode, no user is actively interacting with the device’s touchscreen. The attraction display will sit idly waiting for a user to interact with it, while detecting visitors to the attraction in the background. Information about identified users will occasionally be sent to the server. The device enters active mode once any user interacts with the display. When active, the attraction display will be receiving inputs from a user, displaying information to them, and communicating with the server to update ratings and receive recommendations. After the user finishes with the touchscreen, the attraction display will enter standby mode again.

### 3.2.1 User Identification

Tourists are identified at attractions by the FiPy using its BLE functionality. The FiPy searches for nearby BLE signals, and once one has been identified, its corresponding MAC address is read. Each BLE device a tourist carries on them will have its own unique MAC address. This allows for individual tourists to be identified as they travel between attractions.

After a MAC address has been identified, it is privatized to produce a user ID. No raw MAC addresses are stored by the platform, as user privacy is a priority of this design. All MAC addresses are encrypted using SHA-256 encryption. The same encryption is used at each attraction display, so each MAC address will return the same encrypted string at every display a user visits. SHA-256 encryption was chosen over the development of a lighter, custom encryption algorithm as the time to develop and test a custom encryption algorithm was not available as part of this project. SHA-256 encryption is a known, trusted quantity. A custom encryption algorithm would require extensive testing to ensure
it cannot be reverse engineered, and that it produces minimal collisions when generating user IDs.

The FiPy will routinely perform general scans of the area when in standby mode to identify which tourists are nearby so they can be marked as having visited the attraction. When a user interacts with the display to receive recommendations, a more active scanning is performed as the attraction display enters active mode. The FiPy will scan for BLE signals for a brief period to determine the strongest BLE signal nearby. This signal should correspond to the nearest BLE device, and thus belong to the user interacting with the display. Once the strongest signal is identified, the corresponding privatized MAC address is considered as the active user’s ID. Any ratings or recommendations sent or received will be attributed with this ID.

This approach to identifying users has some faults. Scanning for the strongest BLE signal will not necessarily guarantee that the closest BLE device will be identified due to interference or devices having different relative BLE strengths. However, the device it does detect is likely to be one held by the user interacting with the display. This aspect of the design is tested in Section 5.2 of this report.

If a user carries multiple BLE devices on them, a different one may be identified as the active user each time they interact with the platform. This will mean that their ratings are divided between the user IDs associated with each of their devices. However, this should not affect recommendations too negatively, as each of these user IDs will still be marked as having been detected at each of the attractions the user visits. Therefore, the user will not be recommended attractions that they have already visited.

3.2.2 Display

![Resistive Touch Display](image)

Figure 3.3: Resistive Touch Display

All attraction displays will include a touchscreen display for receiving user feedback.
and displaying user recommendations. The display used by this system is a five inch, resistive touch panel produced by 4D Systems (4D Systems (2021)). This display is powered by an FT812 chip to control the graphics, and connects to the FiPy using SPI.

When a user interacts with the display, the device enters its active mode to identify the user’s ID. Following identification, the display will first ask the user for their rating of the current attraction. This option will be presented as a five-star ranking system, and the user can input their rating by tapping one of the five stars. The next screen will ask the user what type of attraction they are interested in visiting next. The available options are ‘walk’, ‘nature’, ‘activity’, ‘museum’, ‘historical’, ‘tour’ and ‘sports.’ Each attraction in a platform deployment will correspond to one or more of these attraction types. The user can again select their preference by tapping the corresponding option on the screen.

After these user inputs have been collected, the FiPy will request recommendations from the recommendation server. Once these recommendations have been received, they are displayed for the user to view. The recommendations will be ranked in order of preference. Additional information can also be provided alongside recommendations, including opening times, distance from current attraction, or level of busyness. When no users are
interacting with the display, the system will be in standby mode. The display can power
down when in this state to reduce power usage.

The inclusion of a touchscreen in this system offers a lot of potential for future de-
velopments of the platform. The FiPy software could be updated to display time and
weather information when the system is in standby mode, rather than a black screen.
Additional inputs could be collected from users alongside their ratings, such as how far
they are willing to travel to the next location. Users could be provided with a selection
of language options to change the display language, making the system accessible to non-
English speakers. The displays could also provide information about the attraction they
are installed at, such as the location’s history, a map of the area, or a list of available
facilities on-site.

3.2.3 Communications

Communications in the attraction display are handled by the FiPy board. The only de-
vice the attraction displays will communicate with is the recommendations server. The
contents of any message sent by the FiPy will either be a list of user IDs that have been
identified by the FiPy in standby mode, or a user ID and accompanying rating and at-
traction preference if a user is interacting with the system to receive recommendations.
The only messages the FiPy should receive are messages containing a ranked list of rec-
ommendations for a user that has requested them.

The current implementation of the attraction display system conducts all messaging
over WiFi using the MQTT messaging protocol. The FiPy connects to an MQTT broker
and subscribes to feeds available from the broker as they are required. The feeds include a
‘recommendations’ feed for transmitting user ratings and receiving user recommendations.
There is also an ‘updates’ feed that is used to report user IDs identified at a location during
standby scans. All attraction displays in a platform deployment will use the same MQTT
broker.

To reduce power consumption, the FiPy only subscribes to these feeds when it needs
to use them. Rather than constantly polling the feeds for incoming messages and filtering
out messages not relevant to it, the FiPy will subscribe only when it has a message to
send. If it expects a message in return, it will stay subscribed until the message is received.
Otherwise it ends the subscription immediately.

MQTT works over WiFi, meaning that a WiFi connection is required at each location
an attraction display is deployed in. This will introduce challenges in certain remote
attraction locations. A solution for this would be to have other communication methods
available for the FiPy to transmit messages over. No other communication methods were
implemented or tested due to time constraints, but the FiPy communications software was built in such a way during this project as to more easily allow other communication methods to be swapped in should they be implemented in future versions of the platform.

The complexity of implementing other communication methods will depend on the method in question. Using LTE messaging would be relatively straightforward, as Ireland’s LTE infrastructure is well developed. Using LoRa, on the other hand, would be a more difficult process. A LoRa network would need to be created to cover the physical area enveloped by the attraction locations. Depending on the distance between attractions, this may include developing message forwarding nodes. However, once a network like this has been deployed in an area, other devices from different systems would be able to utilize it and further bolster its strength and reliability.

3.2.4 Power

![Attraction Display Solar Panel and Battery](image)

Figure 3.5: Attraction Display Solar Panel and Battery

The attraction displays are powered by a solar panel and battery. The selected solar panel features an adjustable mount, making it easy to reorient depending on the time of year. The battery is built in to the leg of the solar panel, and has a capacity of 3350mA. It connects to the FiPy through the micro-USB port on the PySense 2.0 X shield. During the day, the solar panel will power the attraction display and charge the battery. In
the evenings, or when the solar panel cannot produce enough power, the battery will be employed in addition to power from the panel.

The other components of the attraction display have been designed with reducing power usage in mind. The user identification component will only scan for users when the system is active, or will perform background scans for a few seconds every few minutes when the system is in standby mode. The display will either show some basic information or switch off completely when in standby mode. Communications are only active when they are needed, so no power is spent listening for messages when they are not expected.

If the attraction display is in standby mode, the FiPy can enter a sleep mode in between background BLE scans to further reduce power consumption. The display, however, will need to be at least partially active at all times so that it can detect if a user touches the screen to begin interacting with it.

3.3 Recommendation Server

The recommendations server has been designed as a piece of software that runs on a PC. The server communications and recommendation software is written in Python, and the associated database software is written in JavaScript. It has been tested for Windows 10 computers, but would likely also run on Linux or Mac computers with minimal adjustments. The main software handles incoming messages, generates recommendations using the recommender system, and updates and requests data from the database.

3.3.1 Communications

The recommendation server communicates with all attraction displays using MQTT over WiFi. It connects to the same MQTT broker as the attraction displays, and subscribes to the available feeds at all times while running. Depending on the message received, the server will update the database, and potentially pull from the database to generate recommendations too.

If other communication methods were implemented in the platform, such as LTE, LoRa or Sigfox, the server may need additional hardware to connect to these services. In software, however, the medium through which messages are sent should not matter, as long as the message contents are the same. A message handler would need to be set up for each communication method, but the functions that the handlers call to get recommendations or talk to the database would all be shared.

The ratings database used in this project is a Google Sheets spreadsheet. The recommendations server communicates with this database through the Google Apps Script
API. This is a JavaScript based platform that can be programmed to handle incoming HTTP POSTs and GETs. When the server adds to the database, it compiles all relevant information into a HTTP request and POSTs it to the Apps Script API. Similarly, when the server wants to fetch a user’s ratings history, it will compile a HTTP request with the relevant user ID and make a HTTP GET request.

3.3.2 Recommender System

The recommender system used by the recommendation server is based on a collaborative-filtering approach. It uses both matrix factorization and nearest neighbour algorithms to generate recommendations for users based on their previous ratings, travel history and current context.

To create the recommender system, a pool of simulated user data was first generated. Thousands of simulated users and their ratings were created to imitate the expected ratings and usage patterns of a real population. For example, some users would prefer activity based attractions, so would be more likely to visit these types of attractions and rate them highly. Other users may prefer museums and galleries, and would give activity based attractions low ratings if they visited them. Each simulated user had a probability of visiting any given attraction based on their preferences, and would also rate the attraction in line with these preferences. The attractions selected for this simulated data are 20 popular attractions from County Louth covering a range of attraction types.

This generated data provided a sparse matrix of user ratings. Matrix factorisation was employed to predict the ratings the simulated users would give to attractions they had not rated. Standard matrix factorization uses Formula 3.1 to predict user ratings, where \( \hat{R}_{uv} \) is the predicted rating of item \( v \) for user \( u \), \( \theta^{(u)} \) is user \( u \)'s ratings vector, and \( x^{(v)} \) is the vector of locations.

\[
\hat{R}_{uv} = (\theta^{(u)})^T x^{(v)}
\] (3.1)

Multiple variations of matrix factorization were tested, as is detailed in Section 5.3. From these tests, the best performing matrix factorization algorithm used a radial basis function (RBF) kernel alongside the standard matrix factorization function. The RBF kernel calculates the similarity of two samples using the squared Euclidean distance between the two vectors. The RBF equation is shown in Formula 3.2, where \( ||x - x'||^2 \) is the squared Euclidean distance between \( x \) and \( x' \), and \( \sigma \) is the variance.

\[
K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)
\] (3.2)
After matrix factorization was applied to the generated data, a full matrix of predicted user ratings was created. The recommender system uses this matrix to identify recommendations for a real user by finding the simulated user in the table they most closely match in terms of ratings. The recommendations provided for the user are based on the attraction rankings of the simulated user. The first time a user requests a recommendation, they will only have one rating to be compared against the ratings table with. As a user visits more attractions and leaves more ratings, their recommendations will become more accurate as they get matched with simulated users from the ratings table that more closely match their preferences.

To identify what simulated user a real user most closely matches, a k-Nearest Neighbour (kNN) algorithm is used. kNN algorithms work by calculating the distance between data points to find the option with the shortest distance. For this implementation, Euclidean distance between the real user’s ratings and simulated users’ ratings is calculated. A k value of one is used, as only the simulated user that most closely matches the real user is required.

As more users leave recommendations, the database of real user ratings will grow. Once this database is large enough, matrix factorization can be performed on this data to generate a new ratings table for users to be matched against when generating recommendations. This will allow the recommender system to stay up to date with how users are actually interacting with the platform. For example, during the summer, more people are likely to visit outdoor attractions and rate them highly. This will in turn mean that the recommender system should recommend outdoor attractions more often. Then, during winter time, the recommender system should adapt to provide more indoor recommendations. This is not automated in the system currently, so updates would have to be carried out by the platform operators.

If a new location is added into a platform deployment, or if the platform is deployed in a new region, an initial ratings table will need to be generated. Simulated user ratings can be created using the approach described above by assigning each new location one or more attraction types. As more ratings for the new locations are gathered from real users, the ratings table can be regenerated using this collected data to more accurately reflect how tourists feel about the attraction.

During research, more advanced recommender systems for POI recommendations were studied. However, as there were no examples of recommendations provided from a device located on-site at attractions, preference was given to more standard forms of recommender system to ensure that recommendations in this context were feasible. For a more advanced recommender system, additional factors could be included when generating recommendations, such as distance to nearby attractions, attraction occupancy, weather and
time of day. Other types of recommender systems could also be implemented, such as agent-based recommenders or federated learning systems.

### 3.3.3 Ratings Database

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>User ID</td>
<td>Chartered Adventure Centre</td>
<td>Funpark</td>
<td>Monastery of the Monks</td>
<td>Site of the Boney</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>c9e55f6e7b5d55b3ce</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>f08b506a69d75644</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2f82b929f10b44f1c32</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7b7e1b035f5383bc6</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6be55e1f156f8d1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>9ad6b9f9b2f7f114</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5a9e9e44e19f93f38b</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4e772d44eb05d1006</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c9e55f6e7b5d55b3ce</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>e51b85c95b963917</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>290e660800308000</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>36f16649699993500</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Chapter 4

Implementation

4.1 Implementation Overview

As with the breakdown of the design of the project in Chapter 3, details of the software and hardware implementation of the WhereNext platform are divided into sub-component sections. All code for the attraction displays is written in MicroPython, and runs on a Pycom FiPy device. The recommendation server is programmed in Python 3, and was designed to run on a Windows 10 PC. The database code is written in JavaScript, and is hosted on Google’s Apps Script servers.

4.2 Attraction Displays

The attraction display software contains a main file that loops continuously, and a series of library files to provide the necessary functions for the system to operate. On boot, the main file sets up the necessary connections and clients for communications, and enters the standby mode of the device where it routinely scans for visitors. Functions from the library files as called as the system requires them. Detailed descriptions of the implementation of these functions are included below.

4.2.1 User Identification

User identification is implemented using BLE scanning, and can be divided into two types of detection: standby scanning and active scanning. Standby scanning occurs when no user is interacting with the display, and active scanning occurs when a user interacts with the display to leave ratings and receive recommendations.

When the FiPy boots, a Bluetooth object is created and assigned a callback function. This callback triggers every time a BLE signal is detected by the device. When a device
is detected, the callback reads the device’s MAC address and immediately privatizes it using SHA-256 encryption to create a user ID. If the device has seen this ID previously, it ignores it. If it is a new ID, it gets added to a list of new visitors.

When the attraction display is in standby mode, it will perform a background scan for 30 seconds before sleeping for two minutes. These times have been selected for testing purposes, but actual times may differ in a full deployment. While scanning, the BLE functionality of the device is enabled, so the callback function triggers every time a device is detected. After scanning, the FiPy has a list of all new users at the location. It disables BLE functionality and transmits this list to the recommendations server before sleeping.

When a user interacts with the display, the more active form of BLE scanning occurs. The code for implementing this is shown in Listing 4.1. The FiPy will enable BLE functionality and search for all nearby devices to determine which device is closest. To achieve this, the Bluetooth Received Signal Strength Indicator (RSSI) of each nearby device is read. RSSI values become more negative the farther away a device is, so the device with the lowest absolute RSSI value is considered the closest by the device. After the closest device is found BLE functionality is disabled. The identified MAC address is privatized to get a user ID, which is then considered as the current user of the device.

```
max_strength = 100
max_device = '0'
# Attempt 3 times max
for i in range(3):
    bt.start_scan(scan_time)
while bt.isscanning():
    adv = bt.get_adv()
    if adv:
        ble_strength = abs(adv.rssi)
        if ble_strength < max_strength:
            max_strength = ble_strength
            max_device = ubinascii.hexlify\
                          (uhashlib.sha256(adv.mac).digest())
            max_device = max_device.decode("utf-8")
Listing 4.1: Snippet of the active scanning function for detecting the closest user. Scans RSSI of nearby devices to find the lowest, and encrypts it when found. Performs three attempts before displaying an error.

This more active scanning is intended to be triggered by an interrupt from the touch-
screen display. If the interrupt is triggered while the FiPy is sleeping, it immediately enters active mode to detect the closest user. If the FiPy was in standby mode and in the process of performing a background scan when the interrupt occurred, it first disables background scanning before searching for the active user. The new users detected by the interrupted background scanning are sent to the server before the active scanning is conducted.

Due to issues with the display, this interrupt functionality has not been implemented in the current version of the project. However, the software has been written so that the hand-off between standby and active scanning will go smoothly once the interrupt from the display has been implemented.

### 4.2.2 Display

In the current implementation of the attraction displays, all information that would be shown on the display is instead printed to a console. Due to logistical issues there was limited time available for the implementation of the display, so it has been omitted from the current version of the project. The FiPy can successfully read from the display’s registers using SPI, but the ability to present text on the display was not completed in the available time. The written display software is based off of SPI code written by user ‘vitormhenrique’ from the ‘MicroPython’ forums (vitormhenrique (2019)).

![FiPy connected to the Display via a Breakout Board](image)

Figure 4.1: FiPy connected to the Display via a Breakout Board

The included code has been written in such a way as to make implementation of display related functions as seamless as possible in future iterations. Comments have been included throughout the attraction display code to indicate what should be shown on the display and when. Once the functions to display and read from the touchscreen have been developed, they can be placed in these highlighted locations.
4.2.3 Communications

All communications to and from the attraction displays are done using MQTT over WiFi. The MQTT client is implemented using Pycom’s ‘mqtt’ library. The MQTT broker used for this project is Adafruit IO, a free online broker suitable for small scale projects. The broker has four feeds available: test, recommendations, updates and status. The ‘test’ feed was used for debugging purposes, the ‘recommendations’ feed is used for messages related to recommendations, and the ‘updates’ feed is used to update the server about identified visitors. The ‘status’ feed is currently unused, but could be utilized in the future for updates regarding weather and temperature, or any other information that may need to be collected from a location.

All MQTT messages are sent as strings. Each attraction display in a deployment will have its own unique device ID. This ID is used to identify what attraction a message was sent from, and it is also used by the recommender system to differentiate between locations. All MQTT messages will begin with the device ID of the intended recipient, or an ‘S’ if the intended recipient is the server. This allows devices to ignore messages not intended for them. Information within a string is separated using commas, allowing for the information to be easily interpreted with the Python split() command.

MQTT offers three levels of service quality, which each guarantee a certain level of receipt status. Level 0 guarantees a message will be sent, level 1 guarantees a message will be received at least once, and level 2 guarantees a message will be received exactly once. The attraction displays use a quality level of 2. A quality level of 1 could be implemented by placing some additional checks to ensure that repeat messages are ignored. These checks have been included in some sections of the attraction display software, but further development and testing would need to be conducted to ensure the system can use level 1 quality. A switch to level 1 quality of service would reduce messaging overhead, and should therefore improve the FiPy’s power consumption and network coverage requirements.

During boot up, the FiPy will connect to a nearby WiFi connection. The SSID and password for this WiFi spot are saved in a device config file. After WiFi has been enabled, the FiPy creates an MQTT client using the login details required by the MQTT broker.
A callback function that determines how to handle incoming messages is assigned to the client. This callback will trigger different functions depending on the feed any incoming messages come from. The client will only subscribe to a feed when it has a message to send, or if it is waiting to receive a message.

When the device is in standby mode and has finished scanning for new visitors to the location, it will transmit a message containing these new user IDs to the ‘updates’ feed. The code for this is shown in Listing 4.2. The message transmitted begins with an ‘S’ to mark the server as the target, followed by the device ID of the device sending the message. The rest of the message includes the number of new user IDs being transmitted and the user IDs themselves. The FiPy subscribes to the ‘updates’ feed to post the message and then disconnects.

```python
# Send new visitors
def update_visitors(client, new_devices, visitors):
    print("Running update_visitors()")
    msg_visitors = ''
    msg_vis_count = 0
    for user in new_devices:
        if user not in visitors:
            visitors.append(user)
        if msg_visitors == '':
            msg_visitors = str(user)
        else:
            msg_visitors += ', ' + str(user)
        msg_vis_count += 1
    if msg_vis_count > 0:
        msg = "S," + str(config.DEVICE_ID) + " , " + str(msg_vis_count) + ', ' + msg_visitors
    client.connect()
    client.subscribe(topic=UPDATES_FEED, qos=2)
    client.publish(topic=UPDATES_FEED, msg=msg)
    client.disconnect()
    return visitors
```

Listing 4.2: Function for transmitting a list of new users detected at a location to the server using MQTT. The function first compiles the list of new users in a string. Once the string is ready, the function connects to the MQTT broker, publishes the message, and then disconnects.
If the device is transmitting a user’s rating and requesting a list of recommendations, it will use the ‘recommendations’ feed. The message will contain an ‘S’ to mark the server as the recipient and the device ID to denote the sender. The additional information in the message includes the user ID of the user seeking a recommendation, as well as their rating and context. The user context is the type of attraction they are interested in visiting next. Once the message is prepared, the FiPy will subscribe to the ‘recommendations’ feed and send the message. It remains subscribed until it receives a message back from the server containing the user’s recommendations. These recommendations are then sent to be displayed on the touchscreen display.

4.2.4 Power

Similarly to the display, implementation of the solar panel and battery was hindered due to logistical issues. As such, no thorough power management tests and modifications were undertaken during the course of the project. However, where possible, code was written with the intent to reduce power usage.

The MQTT broker only subscribes to feeds as they are needed. The BLE functionality of the FiPy is only activated as it is required, and it is only set to scan for users every few minutes unless a user interacts with the system. The FiPy also enters a sleep mode when it is not scanning for users or waiting for messages. Further power optimisations could be made, such as tuning the sleep and scanning times of the FiPy, utilizing the FiPy’s deep sleep mode, or reducing the MQTT quality to level 1 and including the necessary checks to account for duplicate messages.

4.3 Recommendation Server

The recommendation server software has a similar structure to the attraction display software. There is a primary file that is called to boot the server and connect to the relevant services. After booting, the server enters a continuous loop waiting for messages, and calls functions from a series of library files to handle any messages it receives. All recommender system related files are also available at all times for the server to access when generating recommendations. The details of the various server files are expanded upon below.

4.3.1 Communications

The server uses MQTT over WiFi to communicate with the various attraction displays, and the details of the MQTT implementation are largely the same as with the displays.
Upon booting, the server establishes an internet connection and connects to the MQTT broker using an MQTT client. The ‘paho’ MQTT library is used to implement MQTT on the server [Eclipse (2021)]. The server subscribes to all four available MQTT feeds at all times so it can handle any incoming messages. A callback function triggers every time a message is received and adds the message to a queue. The server constantly monitors this queue for new messages, and will create a new thread to deal with each message as it is received. This allows for multiple messages to be handled by the server at once. The code to implement the server message handling is shown in Listing 4.3.

```python
def on_message(client, userdata, msg):
    msg_in = msg.payload.decode().split(”,”)
    if (msg_in[0] == ”S”):
        message_queue.put(msg)

while True:
    if not message_queue.empty():
        msg = [message_queue.get()]
        threading.Thread(target=message_handler, args=(msg),
                         daemon=True).start()
```

Listing 4.3: Incoming message handling of the server. Each time a message is received it is added to the message queue. For each message in the queue, a new thread is created to deal with it.

When a message is received in the ‘updates’ feed, the server will reformat the attached information into a JSON string so it can be sent to the database. This string will contain the location the message came from, the number of new visitors, the list of new visitors, and a message type indicator. The indicator tells the ratings database how to handle the message. This JSON string is then POSTed to the ratings database using a HTTP request. The thread handling this message ends once a status message has been received from the database.

When an attraction display is sending a user’s ratings and requesting recommendations for that user, the message will be sent to the ‘recommendations’ feed. The server reads in this message and reformats the relevant information into a JSON string. This string includes the location the message came from and the user’s ID, rating and context. The string also contains a message type indicator. The server POSTs this string to the ratings database using a HTTP request and waits for a status message. The server then makes a GET request to the ratings database to retrieve the user’s entire ratings history. By uploading the new user rating first, it guarantees that the user will have at least one
entry in the database. Once these ratings have been received, the server sends them to
the recommender system to generate recommendations. These recommendations are then
sent to the ‘recommendations’ feed with the device ID and user ID of the device and user
that requested them.

### 4.3.2 Recommender System

The recommender system implementation consists of multiple components: the selected
attractions, simulated data, predicted ratings table generated using matrix factorization,
and a kNN model generator for producing user recommendations.

The recommender system contains 20 attractions that users can be recommended.
These attractions were selected from around County Louth, and were chosen to ensure a
range of attraction types. Each attraction is assigned at least one attraction type from

```python
# Activity user
p = [0.9, 0.8, 0.1, 0.2, 0.1, 0.8, 0.3, 0.2, 0.2, 0.1, 0.6, 0.4, 0.4, 0.5, 0.3, 0.1, 0.8, 0.4, 0.1, 0.4]
generated_ratings = generate_visitors(2000, p, 'activity')
```

Listing 4.4: Example call to the user generation function. The call con-
tains the number of users to generate, the probability of a user visiting
each attraction, and the type of user to generate.

The simulated training data for the recommender system was created by generating
users with preferred attraction types. Each user type was assigned a probability array,
which denoted the probability of that user type visiting each of the 20 attractions. For
each user that is generated, a function loops over this array of probabilities, and deter-
mines if the newly generated user has visited an attraction based on the probability value
associated with each attraction. If the user is marked as having visited the attraction,
they are assigned a rating for that attraction. If the attraction type matches the user
type, they leave a high rating. If the attraction type and user type do not match, they
will leave a lower rating.

This function was used to generate 2,000 users of each type, giving 14,000 total sim-
ulated users. An example call to this function is shown in Listing 4.4, and the function
itself is shown in Listing 4.5. All ratings were placed in a pandas dataframe and converted
to a csv file. The format of the generated user ratings is a sparse matrix of 27 columns
and 14,000 rows. The 27 columns includes the 20 locations, as well as 7 identifier columns
for marking the user type.
```python
def generate_visitors(iterations, probability, context):
    ratings = [None] * iterations
    for i in range(iterations):
        user = [None] * len(config.RATINGS_HEADER)
        for j in range(len(config.RATINGS_HEADER) - len(config.FEATURES_HEADER)):
            chance = random.uniform(0, 1)
            if chance <= probability[j]:
                if features[context][j]:
                    user[j] = random.randint(3, 5)
                else:
                    user[j] = random.randint(1, 4)
        for type in range(len(config.FEATURES_HEADER)):
            user[len(config.RATINGS_HEADER) - len(config.FEATURES_HEADER) + type] = int(context == config.FEATURES_HEADER[type])
        ratings[i] = user

    df = pd.DataFrame(ratings, columns=config.RATINGS_HEADER)
    df = df.fillna(value=np.nan)
    return df

Listing 4.5: User generation function. Each user is randomly provided a rating for each location based on the probability of them visiting it. If the user and location type match they give it a high rating, otherwise they provide a lower rating.

This sparse matrix of generated user ratings was then used to generate a full table of predicted user ratings using matrix factorization. The matrix factorization functionality was implemented using the ‘matrix-factorization’ Python library (Do (2020)). To prepare the data for matrix factorization it had to be formatted into three columns: a user column, an item column and a ratings column. Each row of these columns corresponds to one individual rating. If a generated user had left 10 ratings, they are in the user column 10 times. The three columns are converted to a pandas dataframe and passed to the matrix factorization function. The selected function incorporates an RBF kernel to improve accuracy. The output of this function is formatted back into a 27 by 14000 matrix, but with a rating for every location from each user. This matrix is saved as a csv file.
The final part of the recommender system is the kNN model generator. This is used
to match real users and their ratings against the generated user ratings created by matrix
factorization. To generate recommendations for a user, the system will receive a user’s
ratings history from the ratings database. For each location, the rating will be 0, -1 or
a value from 1-5, depending on whether the user has not visited a location, has visited
a location and did not leave a rating, or has visited a location and did leave a rating,
respectively. The recommender notes what locations the user has visited so they are not
recommended to them again.

```python
# Create a model for getting recommendations using ratings of
# locations user has rated
model = NearestNeighbors(n_neighbors=1, radius=0.4)
model.fit(relevant_ratings)

# Get closest matching user in ratings table to current user
neigh_ind = model.kneighbors([[user[ind] for ind in user_rated_ind]], 1, return_distance=False)
neigh_ind += 2000 * config.FEATURES_HEADER.index(context)
neigh_rat = ratings[neigh_ind]

# Find the top 5 locations that the current user has not
# yet visited
neigh_top5 = np.argsort(neigh_rat)[::-5]
neigh_top5 = np.flip(neigh_top5[0][0])
recs = []
for i in neigh_top5:
    if user[i] == 0.0:
        recs.append(i)
```

Listing 4.6: Snippet of the user recommendation generator function.

A kNN model is created using ratings from the predicted user ratings
table. The generated user that a real user most closely matches is then
identified, and recommendations are ranked according to the generated
user’s ratings.

The recommender loads the generated ratings table from a csv file, and selects only the
predicted ratings of users whose type matches the type of attraction the user is looking
to be recommended. The system then extracts the columns of ratings from this reduced
dataset that correspond to the locations the real user has left ratings for. This ensures
that the user recommendations are only based off of their provided recommendations.

A kNN model with a k value of 1 is trained using the selected generated ratings. The generated user that the real user most closely matches in terms of ratings can then be identified using this model. The generated user’s location ratings are ordered from highest to lowest, and those that the real user has already visited are filtered out. The remaining 5 highest rated attractions are then returned from the system as the user’s recommendations. A snippet of this part of the recommender is displayed in Listing 4.6.

### 4.4 Ratings Database

The database for this project is a Google Sheets spreadsheet, hosted on Google’s servers. The spreadsheet contains two sheets, one for new user ratings and one for platform interaction history.

The ratings sheet contains 21 columns, one for each location and one for user IDs. As new users are identified by the platform, their user ID is added to a new row in the users column. If a user is identified at an attraction, a -1 is placed in the cell corresponding to that user’s ID and the attraction they were identified at. If the user leaves a rating for the attraction, the -1 is replaced by the rating that they provide. Ratings from this sheet are used by the recommender system to generate recommendations for users.

The platform interactions sheet tracks every time a user interacts with the platform to leave a rating. It contains three columns: user ID, location, and time of interaction. The data from this sheet is not used by any other part of the platform. It is intended for use by platform operators to allow them to analyse how people are interacting with the platform and attractions.

```python
# POSTs user rating to Sheets
def post_rating_Sheets(data):
    url_data = config.DB_URL + "?type=") + data["type"]",
    + "&user=" + data["user"] + "&rating="
    + data["rating"] + "&location=" + data["location"]
    r = requests.post(url_data, data=json.dumps(data))
    print(r.status_code)
```

Listing 4.7: Server function for POSTing a new user rating to the database.

The database is updated through the Google Apps Script API. This is a JavaScript based system that hosts scripts that link to the Google Sheets database. It is triggered by

32
HTTP requests, and contains a doPost and doGet function to handle incoming POSTs and GETs respectively.

A POST will be received by the database if the recommendations server wants to update the database with new users that have been detected at a location, or upload a user’s rating for a specific location. The code for POST requests to the database is shown in Listing 4.7. These types of messages are differentiated with the inclusion of a message type identifier in the POST request.

In the former scenario, the database will receive a request containing a location ID, the number of visitors detected, and the IDs of the detected users. A -1 is added to the appropriate column and row in the database for each user if they have not previously left a rating for that location. If the user ID is not found in the database, a new row is created for it.

In the latter scenario, the database will receive a request containing the device ID, user ID and user rating. The user’s rating is added to the appropriate row and column in the database. If the user is not already in the database, a new row is created for them. A new row is also added to the platform interactions sheet. This row will include the user ID and location ID contained in the message. The time and date that the request was received at are also added to a third column in this row.

When the database receives a GET request, the request will just contain the user ID of the user that is interacting with an attraction display to receive recommendations. The database searches for the ratings associated with this user ID, and sends them back to the server. If the user ID is not found in the database, an error message is sent instead.
Chapter 5

Evaluation

5.1 Bluetooth

5.1.1 Experiment

To evaluate the performance of the Bluetooth functionality of the FiPy, Bluetooth distance tests were performed to identify the maximum range at which Bluetooth devices could be identified by the FiPy. Tests were conducted both indoors and outdoors to determine maximum connection distances in both environments. Due to Covid-19 concerns, both tests were carried out in a private residence rather than on-site at an attraction.

The test procedure included the FiPy, a laptop and console, and a smartphone with Bluetooth enabled. The FiPy was connected to the laptop and set to scan for all BLE signals nearby. If the MAC address corresponding to the smartphone was identified by the FiPy a message was printed to the laptop’s console.

The indoor tests involved placing the smartphone on a surface, and slowly increasing the distance between the smartphone and FiPy while monitoring the console messages. Once the FiPy stopped printing to the console, the distance between the FiPy and smartphone was recorded. This test was repeated five times. After results were collected, an average maximum connection distance and variance was calculated using the measurements. The outdoor test procedure was identical to this, but involved placing the smartphone on an outdoor surface instead.

5.1.2 Results

Table 5.1 contains the results of the Bluetooth distance tests. The maximum indoor distance was found to be 7.4 meters with a variance of 0.5 meters. This will give indoor attraction displays an approximately 15 meter diameter sphere in which users can be
<table>
<thead>
<tr>
<th>Test Number</th>
<th>Indoor Readings (m)</th>
<th>Outdoor Readings (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8.3</td>
<td>12.5</td>
</tr>
<tr>
<td>2</td>
<td>6.8</td>
<td>11.9</td>
</tr>
<tr>
<td>3</td>
<td>7.1</td>
<td>12.2</td>
</tr>
<tr>
<td>4</td>
<td>7.5</td>
<td>13.0</td>
</tr>
<tr>
<td>5</td>
<td>7.4</td>
<td>12.4</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>7.4 ± 0.5</strong></td>
<td><strong>12.4 ± 0.3</strong></td>
</tr>
</tbody>
</table>

Table 5.1: Results from Indoor and Outdoor Bluetooth Distance Tests

identified. The maximum outdoor connection distance was much higher at 12.4 meters, with a standard deviation of 0.3 meters. Outdoor attractions will therefore have an identification sphere of around 25 meter diameter. The indoor tests likely produced lower distances and a higher variance due to the interference of other Bluetooth devices and the increased number of surfaces for signals to reflect off of.

These results provide insights about the best locations to place attraction displays to ensure they can detect as many users at a location as possible. For indoor locations, the displays should be set up near entrances, as these are pathways that funnel all visitors though a relatively small location. The 15 meter coverage sphere should be sufficient to cover the entrances to most attractions. It is likely that an attraction display installed at an indoor location would be positioned against a wall. This would mean that only the front half of the coverage sphere would detect users. A directional BLE antenna that provides more coverage in front of the attraction displays may be a useful modification to any displays located indoors.

Outdoor displays will have a much wider coverage sphere than indoor displays. An appropriate installation point for outdoor attraction displays would be a car park or information centre if there is one available. These displays will not have the same concerns as the indoor displays regarding sections of the BLE coverage not being utilised. Due to the additional space afforded at outdoor attractions, tourists are expected to be identified in all directions surrounding an attraction display.

5.2 User Identification

5.2.1 Experiment

To test the FiPy’s ability to identify the correct user when the device enters ‘active’ mode, a user identification test was conducted. This test would assess two aspects of the identification system. Firstly, it would check whether the FiPy can reliably detect a BLE
signal belonging to the user closest to the attraction display, or if it picks up nearby users instead. Secondly, it will test whether the same device belonging to a user is identified every time, or if the system identifies a different device each time a user interacts with the display, and thus distributes a user’s ratings between all BLE devices they carry on them.

The test was conducted indoors. Due to Covid-19 concerns, the test was conducted in a private residence rather than on-site at an attraction. The FiPy was connected to a laptop and set to print out the BLE MAC address of the device it identified as closest following 5 seconds of scanning. The FiPy was positioned on a surface 1 meter above the ground in the centre of a room. To represent a user, a tester was positioned approximately 30cm from the FiPy, mimicking the expected distance of a user from the device when interacting with the display. The tester carried three BLE devices that a tourist could be expected to carry: a smartphone in their front pocket, a smartwatch on their wrist under a jumper sleeve, and Bluetooth headphones over their ears. To represent other tourists in the area, a tablet with Bluetooth enabled was positioned at least 1 meter away from the FiPy device. All other Bluetooth devices were removed from the room.

30 active mode scans were conducted during the test. After each scan, the MAC address printed to the screen was recorded, and the tablet was repositioned to simulate users moving in the background. The number of times each device was identified as the closest BLE device, and thus identified as the active user, was tallied up. The results of this test are shown below.

### 5.2.2 Results

<table>
<thead>
<tr>
<th>Device</th>
<th>Times Identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smartphone</td>
<td>2</td>
</tr>
<tr>
<td>Smartwatch</td>
<td>16</td>
</tr>
<tr>
<td>Headphones</td>
<td>12</td>
</tr>
<tr>
<td>Tablet</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2: Results from User Identification Test

Table 5.2 shows the number of times each device was identified as the closest device during the 30 runs of the identification test. The smartwatch was identified as the closest device 16 times, making it the most commonly identified device. The headphones were the next most identified, and had a count close to the smartwatch. There is a big gap between these devices and the smartphone, which was only identified as the closest device twice.
The tablet that represented the non-active user in the background was never identified as the closest device.

These results show that the FiPy is successful at identifying the correct user every time it scans for the active user. After 30 tests, the correct user was identified every time. The tablet was placed in positions all around the FiPy during the test, but was never identified as the closest device. This indicates that when determining the active user, the FiPy should identify the correct one each time, as long as other users are at least 1 meter from the attraction display. If other users are closer than this, the system may identify them as the active user incorrectly. Additional testing would need to be conducted to verify this.

The results also show that while the FiPy identifies the correct user each time it scans, it does not always detect the same device belonging to that user. This means that each time a user leaves a rating, it may be attributed to the device ID associated with any of their devices. The FiPy detected the smartwatch the majority of the time, but a test run over a longer period may show different results. There are many factors that could play into which device is identified as closest by the system. A smartphone may be identified more if the user holds it in their hand instead of leaving it in their pocket. The smartwatch may be identified less if it is covered by a different article of clothing, or may be identified more often if it is not covered at all. The headphones may also be identified a different number of times depending on whether they are playing audio or not. There is likely to be a lot of variance in these results between users.

Based on these two identified outcomes, the system is likely to identify the correct user each time they interact with the platform, but it may not detect the same device every time. This will have an effect on the personalisation of user recommendations, as the recommender system may not have access to every rating a user has provided. However, the impact of this should not be too profound. The recommender will still have access to a user’s latest rating, and the type of attraction they are interested in next. Each device a user carries will also be detected at each location they visit through background scans, so the recommender system should never recommend a location to a user that they have already visited, no matter what device they are carrying is detected when they interact with the display.
5.3 Recommender System

5.3.1 Experiment

When designing the matrix factorization system for generating the predicted user ratings table from the sparse matrix of simulated user data, a number of variants of matrix factorization were tested. This included three types of matrix factorization without kernels: standard matrix factorization, Alternating Least Squares (ALS), and Stochastic Gradient Descent (SGD). These methods differ in how they minimize the loss function. ALS alternates between minimizing for the user data and items data, while SGD minimizes for both variables but updates weights every sample instead of in groups.

Three types of kernelized matrix factorization were also tested: linear kernel, sigmoid kernel and RBF kernel. Each of these kernels use different activation functions, and typically have differing efficacy depending on the dimensional space of the input data and the application. The linear kernel works best with linear spaces, the sigmoid kernel is suited for neural networks, and the RBF kernel is commonly implemented in Support Vector Machines (SVMs).

To evaluate each of these matrix factorization types, a model of each was produced and trained using a set of training data composed of simulated user ratings. They were then tested using a test data set composed of the user training data to produce a set of predicted ratings. The Root Mean Square Error (RMSE) between the actual and predicted ratings from the test set was calculated for each model and compared. The simulated user ratings set was divided using an 80/20 split, where 80% of the dataset was used for the training data, while the remaining 20% was used for testing. The same training and test sets were used for all six tested models.

Projects that feature a kNN model will typically do a comparison of predictions for different k values to determine which gives the best predictions for the application. However, in this project, kNN is used to identify the single simulated user that a real user most closely matches, so the model included in this project will use a k value of 1. As such, the standard test for k value comparison was not conducted as part of this project.

5.3.2 Results

Table 5.3 shows the calculated RMSE for each of the tested matrix factorization models. From these results, it is clear that the version of matrix factorization that included an RBF kernel performed the best. It produced predictions with an RMSE of 0.99, while all other models had an RMSE of at least 1.3. The worst performing model was the standard implementation of matrix factorization. The kernelized models all outperformed
the models without kernels, with the sigmoid and linear models producing the same RMSE value of 1.32. Based on these results, the version of matrix factorization featuring an RBF kernel was used for the project’s recommender system.

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>1.4087</td>
</tr>
<tr>
<td>ALS</td>
<td>1.3364</td>
</tr>
<tr>
<td>SGD</td>
<td>1.3704</td>
</tr>
<tr>
<td>Linear Kernel</td>
<td>1.3176</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>1.3176</td>
</tr>
<tr>
<td><strong>RBF Kernel</strong></td>
<td><strong>0.9985</strong></td>
</tr>
</tbody>
</table>

Table 5.3: Comparison of RMSE Values from predictions made by multiple implementations of Matrix Factorization

5.4 Small Scale Deployment

5.4.1 Experiment

In order to test the communications between server and attraction displays, server and database, and general operation of the whole WhereNext platform, a small scale deployment of the project was created. The deployment consisted of three attraction displays, in the form of three FiPy boards, the recommendations server and the Google Sheets database. The three attraction displays were set up in three different rooms in a house and each represented a different location. They were set to run in standby mode, but would randomly interrupt the background scanning to transmit a user rating and request recommendations. The server was turned on to handle any incoming messages from the attraction display, as well as upload new user information to the database and download user ratings as recommendations were requested. As with the Bluetooth tests, this test was conducted in a private residence due to Covid-19 concerns.

This deployment test was designed to highlight any issues that may occur when the server is dealing with multiple attraction displays at once. The server would need to ensure all incoming messages are handled without issue. If multiple messages came at once they would be handled simultaneously, which may include simultaneous requests to the database or recommender system. The attraction displays would also need to be able to handle the increased amount of messaging, and successfully filter out any messages that were not intended for them. The randomness at which recommendations were requested would ensure that the ability for the server to handle these requests is tested in a range
of scenarios.

To represent users travelling between locations, a collection of Bluetooth devices were occasionally rotated between the rooms the attraction displays were set up in. Additionally, people with their own Bluetooth devices moving between rooms in the house acted as users with more unpredictable movement patterns as they were not scripted participants in the test.

The deployment was set to run for 30 minutes, and the consoles of the attraction displays and server were routinely monitored to ensure there were no issues. The database was also regularly checked to ensure that data was being added to it as expected. In the case where a bug or issue was identified, the deployment was shut down until the bug was resolved. The platform was then redeployed for another 30 minute period. This process repeated until the deployment ran without issue for a full 30 minute period.

5.4.2 Results

This experiment proved successful in testing the various subsystems of the project. The majority of the systems worked without issue, but the test highlighted some bugs that have since been fixed.

The server was successful at handling multiple incoming messages through the use of threads. All updates to the database were successful, whether they happened alone or simultaneously with other requests. If a user ID was identified at an attraction but did not leave a rating, they were marked in the database with a -1 for that location. If that same user ID was randomly selected to leave a rating for that location, the same cell would instead include the generated rating.

The recommender system was also able to operate successfully during the deployment. When a single recommendation request was received by the server, recommendations were generated in well below 1 second. In the occurrence where multiple requests came at once, recommendation generation was slightly slower, but still under 1 second. The recommender had no trouble adapting to the varying number of ratings available when generating recommendations. Whether recommendations were generated for a user using 1 existing rating from their history or 3, there was no discernible difference in the time taken to generate the recommendations.

The one area where a bug was identified was the attraction display communications. When an attraction display requests recommendations from the server, it remains subscribed to the ‘recommendations’ MQTT feed until a message is received. Any messages that are designated for other attraction displays are ignored. The attraction display that requested recommendations is intended to remain subscribed until it receives the message
addressed to it containing the user recommendations.

However, the attraction display would instead wait for only a single message to arrive, before unsubscribing. This meant that if recommendations intended for a different location were transmitted first, the display would not receive the recommendations it had requested. This error can only occur if multiple attraction displays are waiting for recommendations at the same time, so was not found during development of this system. This bug has since been fixed. The deployment that was run following this patch completed a 30 minute deployment without issues.

These results indicate that, for small deployments of attraction displays, the platform works well. However, it is not able to confirm that the platform will work as successfully for larger deployments. An example of this is the MQTT broker. The Adafruit IO broker allows for 60 messages to be sent every minute, which was more than enough for this deployment. For larger deployments, a custom hosted broker would likely be required.

There may also be issues with multi-threading on the server with larger deployments. The server was able to manage three message handling threads at once, but operation may slow down as more threads are added. These issues would need to be thoroughly tested to confirm whether a large scale deployment of the WhereNext platform is feasible or not.
Chapter 6

Conclusions & Future Work

6.1 Conclusions

The primary goal of this project was to design, develop and test a travel recommendations platform to simplify the process of identifying attractions of interest, and making recommendations and offering guidance to tourists that optimises both their experience and that of the platform operators. This goal was achieved through the development of an attraction display to provide tourists with recommendations based on their preferences and travel history, and a server to aid the running and management of attraction displays.

The attraction displays feature a user identification system that ensures the privacy of all tourists that are detected by the system. All user BLE MAC addresses are immediately privatized using SHA-256 encryption upon receipt, so no identifiable personal information is required from users to engage with the platform. The scanning range for this identification system has been verified as suitable for purpose, and recommendations have been made regarding where the attraction displays should be set up to achieve the best coverage.

A robust communication system was implemented on the attraction displays. The FiPys successfully transmit and receive messages over WiFi using the MQTT protocol. These communications were verified following a rigorous small-scale test deployment. All messages transmitted by the FiPy were received by the server, and the FiPy had no issues receiving messages in return. The communications remain reliable when a single FiPy is running by itself, or when multiple FiPys are communicating at the same time. The MQTT broker used as the middleman for these messages similarly transmits message with no evident issues.

The developed communication systems of the recommendation server were found to be similarly robust. The server can handle messages coming from multiple sources simulta-
neously, and is able to respond to those sources when required without issue. The implemented communication methods between the server and database are also successful. The server can handle simultaneous messages in this context too. The server communications functionality was verified through a small-scale deployment of the server and attraction displays.

The recommender system utilized by the server produces suitable recommendations for tourists. The system can take a user’s previous ratings, travel history and current context into account to ensure recommendations are appropriate. A dataset for aiding the development and testing of the recommender system was generated. A selection of matrix factorization methods were tested and compared, and the best performing one was selected for use. During the small-scale platform deployment, the recommender system generated recommendations for multiple users at a time, and adapted to users with varying amounts of information to base recommendations off of.

The server’s ratings database successfully sorts and stores all incoming user information and ratings. New user ratings are added to the appropriate columns, and new users are added to the database without issue. Any user data in the database can be identified and transmitted to the server as it is requested. All interactions with the recommender system are successfully tracked on their own sheet with an accurate timestamp. The ability for the database to remain operational during a small-scale deployment of the full platform was also verified.

Some aspects of the platform were not fully implemented, as originally planned. This is primarily as a result of a logistics issue obtaining the required hardware to implement these features. Where possible, steps have been made to ensure the implementation of these features can proceed more smoothly in the future.

In the FiPy software, any messages that should be displayed to a touchscreen are printed to a console. Comments have been left next to these messages to describe the information that should be shown on a display. Similarly, messages have been included to highlight areas where information should be read from the touchscreen display.

Neither LoRa, Sigfox nor LTE communications were tested during this project. All functions for messaging have been written in such a way so that when these methods are implemented, the appropriate client can be passed instead of the MQTT client, and only minimal modifications to the software should be required.

The power consumption of the attraction displays was not tested using the solar panel and battery intended for use with them. However, during development of the attraction display software, minimizing power consumption was a focus. Communications are only enabled when they are needed, as is the FiPy’s Bluetooth scanning functionality. The attraction displays enter a sleep mode whenever they are not scanning or communicating
with the server to further reduce power usage.

The current implementation of each of the features discussed above appear to be successful. The WhereNext platform has proven to be functional in small deployments, as shown by the small-scale deployment conducted during testing. For a larger-scale deployment, additional testing would need to be conducted. The ability for increased numbers of attraction displays to communicate with the server simultaneously must be investigated. A new MQTT broker would be required, ideally a custom one to provide platform operators with full control over its functionality. The ability of the server to handle multiple requests, with both the database and attraction displays, during a large-scale deployment would also need to be verified.

If deployed in a small-scale deployment, the current implementation of this project should improve the tourist experience by making the process of receiving recommendations simpler. The only requirement of the user to benefit from the platform is to engage with it. The platform will similarly benefit the operators that deploy it, as it gives them a new avenue for aiding tourists as they explore a region. The database also allows operators to analyse tourist patterns in an area, and make more informed infrastructure and development plans accordingly.

6.2 Future Work

This project features a lot of potential for expansion with future work. As mentioned in Chapter 2, no examples of on-site recommendation systems were found during the research phase of this project. The most clear future work areas would be implementing the features that were unable to be included in the project due to time constraints, including the display, additional communication methods and more optimal power saving. However, there are also areas for novel concepts to be investigated, such as large-scale and on-site tests, user identification, communications, the user interface, the recommender system, and the usage context.

To confirm the WhereNext platform is suitable for large-scale deployments, additional testing needs to be undertaken. Due to concerns regarding Covid-19, tests conducted as part of this project were kept to a small-scale and conducted in private residences. The Bluetooth and user identification tests could be repeated on-site at tourists attractions to get a more accurate reflection of the FiPy’s Bluetooth coverage when deployed. The methodology of these tests can also be improved by taking more readings and testing different BLE antenna types in each environment. A larger-scale test deployment of attraction displays should also be conducted, both in a test environment and deployed to tourist attractions. This would highlight any potential issues that may arise from
an increased amount of simultaneous processes and messages, more frequent calls to the database and MQTT broker, and communications over larger distances.

Beamforming, Multiple-Input Multiple-Output (MIMO), or other similar signal processing techniques could be implemented to improve the accuracy of detecting tourists. The current method of Bluetooth scanning works well for the general scanning of an area, but it is not ideal for identifying which user is closest to the display when it is active. Beamforming would allow the attraction displays to more accurately identify the position of users in a space. This would ensure that the same user ID is obtained every time a user interacts with the display. This technology would also allow for multiple attraction displays to be set up next to each other, and the correct user identified for each one. This more advanced detection technology will likely increase the power usage of the device, so this would need to be considered if it were implemented.

Attraction displays could be set up with multiple communication channels available for communications. This would produce a more robust communications system, as the FiPy would have multiple fallback communication methods should one of them fail. The use of a LoRa network would introduce further opportunities, as the same network could be shared with multiple other projects in an area, or provided as a service for hobbyists in the region. The increased number of channels through which messages can arrive could increase the power usage of the attraction displays as they have multiple clients running simultaneously. An option for reducing this effect would be to designate one communication method as the primary method, and only activating the others as they are needed. In the current implementation, all communications are initiated by the attraction displays, so the communication medium used could be dictated by them.

Current communication methods do not contain any forms of security or privatization. This choice was made so that more time could be spent focusing on the more novel aspects of the project instead of implementing and testing proven security features. The user IDs that are transmitted are privatized and cannot be traced back to users, but the messages that carry these IDs could be accessed by a third party. Communications between the server and database similarly contain no advanced security features. These issues could be addressed through the addition of standard security features contained in most devices, such as encryption using public and private key pairs. An option such as this would allow the server to regularly generate new keys for each of the attraction displays, ensuring communications security remains difficult to bypass.

The attraction display user interface could be made more accessible for more types of people. The current display is suitable for users that understand English, and have no vision impairments or physical disabilities that prevent them from using a touchscreen.

To make the displays more accessible to non-English speakers, users could be provided
with a menu of language options upon interacting with the display. This feature would also provide the platform with another data point about the user that information could be inferred from, such as that user’s country of origin. This information could be included in the recommender system to further personalise recommendations. Knowledge of tourist language composition can also benefit the platform operators. This will provide them with a rough demographics breakdown that can be used to plan events or developments. For example, if a museum wishes to introduce a guided audio tour, this data can be used to infer what languages this tour should be recorded in.

Making the platform accessible for users with physical disabilities would require more novel solutions. One option is to include a speaker and microphone, allowing users to provide ratings vocally and have recommendations provided through the speaker. A voice recognition system would be computationally demanding, so could be hosted on the recommendations server rather than the attraction displays. However this would require voice files to be sent, which are proportionally much larger than the other messages sent by the attraction displays.

Another option for accessibility would be the inclusion of a web interface accessible by scanning a QR code on the display. This would allow users to interact with the attraction displays on their smartphone, where they can use their phone’s voice assistant and speakers to provide feedback and receive recommendations respectively.

Attraction displays could also contain an emergency contact feature. In remote locations, tourists may not have coverage to send messages in the case of an emergency. The attraction displays will always be connected to the server, so can act as a reliable emergency communications vector. Depending on the location, the attraction display could contact staff available at the attraction, or alert emergency services if the attraction has no staff. The system could transmit a generic emergency alert, or it could allow the tourists to speak with emergency services if a speaker and microphone are installed at the attraction display. The additional language option would allow the system to connect the user calling in an emergency with an operator that speaks their language.

The recommender system features the most potential for expansion in future work. More readily accessible information could be considered when generating recommendations. This includes distance to attractions, opening and closing times, weather and admission costs, amongst other factors. The system could also provide the option for operators to highlight attractions they wish to push people towards, if, for example, there is an event on in the area. Attraction displays could also provide estimates of tourist numbers at attractions so that as the occupancy of a location increases, the likelihood of recommendation decreases to reduce overcrowding.

The recommender system could track what recommendations it gives to users, and
how they respond to their recommendations. This could be achieved either through explicit or implicit feedback. A user could be provided with the option to ignore certain recommendations, which could then be recommended for them less in the future. The system could implicitly track user response by recording the recommendations a user was provided, and comparing that to what attraction they were next identified at. If a user is regularly recommended outdoor attractions, for example, but is only found at indoor attractions, the system could shift to prefer recommending indoor attractions for that user.

An exciting potential for the recommender system would be to adapt it to use a federated learning architecture. This would involve the server transmitting a basic model to all attraction displays. Each display would update the model depending on how it sees users interacting with the platform. All updated models are then sent back to the server to produce a stronger base model. A federated learning system would further improve user privacy as it eliminates the need to send user IDs to the server. One of the primary issues this system would face is the loss of user travel history. Instead, a more general recommender would need to be produced. The models could still base recommendations off of ratings of the current location and user preference for the type of location they want to visit next. To update the model, the system could infer how users react to recommendations. This could be achieved by providing users with ‘more details’ options for each recommendation. Links can then be made that show, for example, that users who rated the current attraction highly, and prefer museums, showed interest in attractions A, B and C when shown a list of recommendations.

The underlying technology of this project has the potential to be used for other contexts outside of tourist attraction recommendations. By adapting the recommender system, the attraction display and server could be deployed in other management scenarios. Examples of applications include managing building occupancy, automatic allocation of offices in co-working spaces, helping users find specific products or product types in a store, or car-park management systems. In each scenario, the operator can benefit from the same advantages as highlighted in Section 6.1. A smaller-scale deployment in terms of area covered, such as in the context of a building occupancy manager, would also simplify the communication requirements of the platform, allowing it to perform even better.
Bibliography


