TECHNICAL MANUAL

Smart City

Conor Reilly - 16478326
Eoin Clayton - 16326173

Supervisor - Alistair Sutherland

1. Introduction

1.1 Motivation

The aim of the project is to identify and provide information of landmarks to users by submitting an image of the landmark into a mobile application. The overall goal of the project attempts to remove the manual process of searching for information about a landmark that is unfamiliar to the user and also provide an easier way to provide users with information regarding landmarks and popular attractions. Another feature of the application will be to provide points of interests in the form of routes for a user to take around the city they are in. It also looks to make travelling around cities easier for all people, hence the name “Smart City”.

Given the current era where smartphone devices continue to excel in popularity and processing power and the continued push for faster and more reliable information. This application looks to combine these modern advances by using all features of a smartphone. The device should be accessible to people of all ages and allow users to search for landmarks in real time which will enable users to acquire information of unfamiliar landmarks with ease.

1.2 Overview

The project is a landmark recognition mobile application that will enable the user to submit an image of a landmark via taking a picture or through their device’s storage. The application will then make an estimate via neural network of what the landmark is in the photo followed by a short information section about the landmark. The application will also allow a user to select one of many routes close to their surrounding area that send them on a guided tour of close landmarks via Google Maps. The application will be simple in design and cater to people of all ages. It will provide users in unfamiliar places with real time information.

1.3 Business Context

The product could be an independent software as a service provider. Allowing advertisers and companies to target local tourists in the area. Another option would be to include this software in a larger scale mobile application such as TripAdvisor or Snapchat. These companies currently target smartphone users who enjoy visual content as well as the social sharing side of social media.

1.4 Glossary

- React Native - React is a JavaScript library for building user interfaces such as mobile applications.
- Neural Network - A series of algorithms that attempts to identify underlying relationships in a set of data by using a process that mimics the way the human brain operates.
- Keras: An open source library for development of Neural Networks
- TensorFlow: An open source machine learning based library for dataflow programming across a range of tasks.
- Google Colab - A machine learning centred notebook that can execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs.
- Pandas - is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- Expo - An application that allows for testing React Native systems on mobile devices.
- Clarifai - Module within React Native that utilizes machine learning and data visualization

2. Research

One of the functionalities of the mobile application is based around its image classification feature. We began researching Convolutional Neural Networks to see if it was a viable option to achieve this. We used the Keras library to help develop this, we found it to be the best fit within our architecture and allowed us to use many different variables when determining our neural network.

To develop this neural network we also needed a solidified dataset, we found a dataset on Kaggle relating to landmarks however there was a serious need to redesign and restructure the dataset. Through our research and experience we chose Pandas library to help with this data management as it fit in with the previous Python specifications.

We also decided to use React Native as our main development for the application. We decided to use this framework as through our research we believe it is the strongest forerunner in the future of front-end development for mobile application. It also posed as a challenge as we had never used this type of software before.

3. General Description

3.1 System Functions

The mobile application was developed for tourists and those not readily available to do independent travelling. One functionality of the application is to identify landmark images and provide users the relevant information about the landmark that they have inputted. The user interface of the app aims to be simplistic and strives to be user-friendly for all age groups. The application includes options to take a picture directly or access the camera roll and select one from there. Once an image is selected it will be passed through the convoluted neural network and return the user with an estimate of what the landmark is.

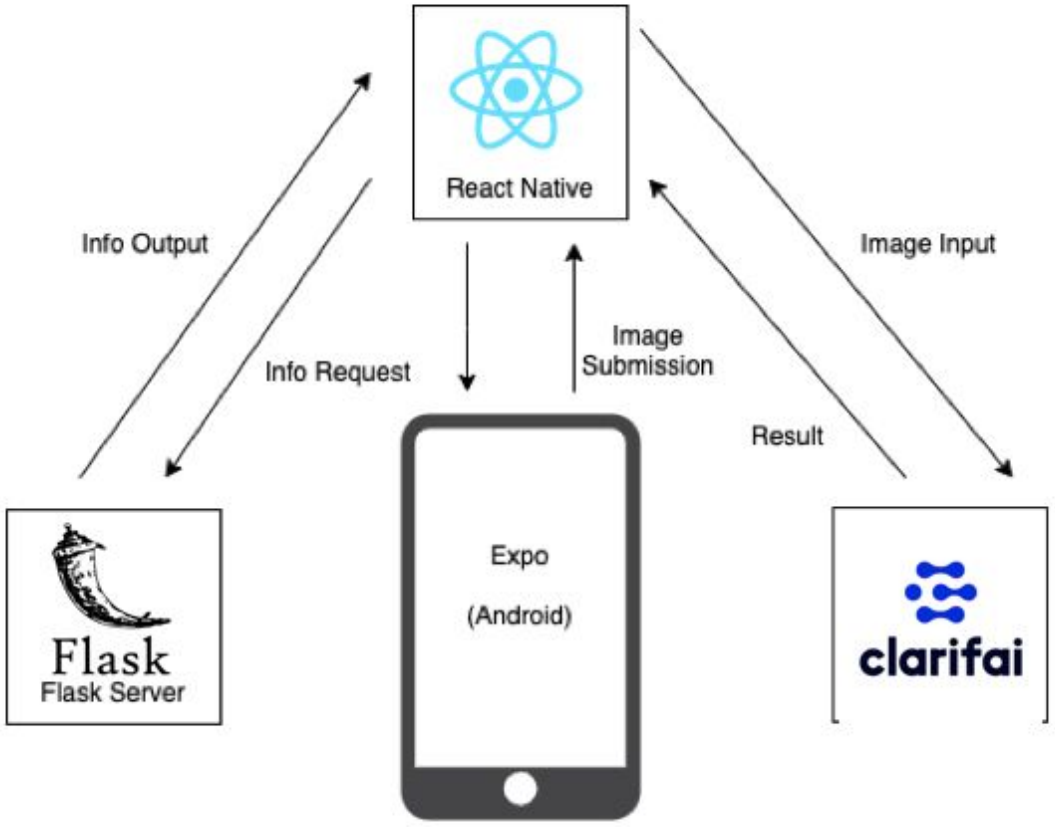
Another feature on the application will be to allow a user to select pre designed routes to explore nearby landmarks. This will link users to their google maps account and allow them to search a route that suits them or use a predesigned one by the mobile application.

3.2 User Characteristics

It is expected that a large amount of the user base of the mobile application will be those that wish to get information about a landmark that they are unfamiliar with as well as those looking to explore nearby landmarks. Another possible user may be one that has difficulty in navigating new cities and needs help to do so. With these scenarios, the application must be clear to users of all ages and must be adaptable to their surroundings. Given the features that allow a user to take a live image and use this or select a previous one from their camera roll gives the application multiple uses and does not always have to be used when directly faced with a landmark.

4. System Architecture

4.1 General System Architecture



The system is built for smartphones using Expo as a viewing platform for the application, with the use of React Native to present users the user interface. Users will require a smartphone device in order to use this mobile application. The application does not include a login base so it can be used by all people with quick access. Upon selecting an image, it communicates with a flask webserver to retrieve the related information of that landmark as well as submitting the image to a Google Colab Notebook where the most likely image will be calculated. Once the predicted landmark is sent back to the device it is displayed along with the information on the landmark, it's name and a link to a relevant website. The system also utilizes Clarifai

4.2 Architecture Overview Diagram

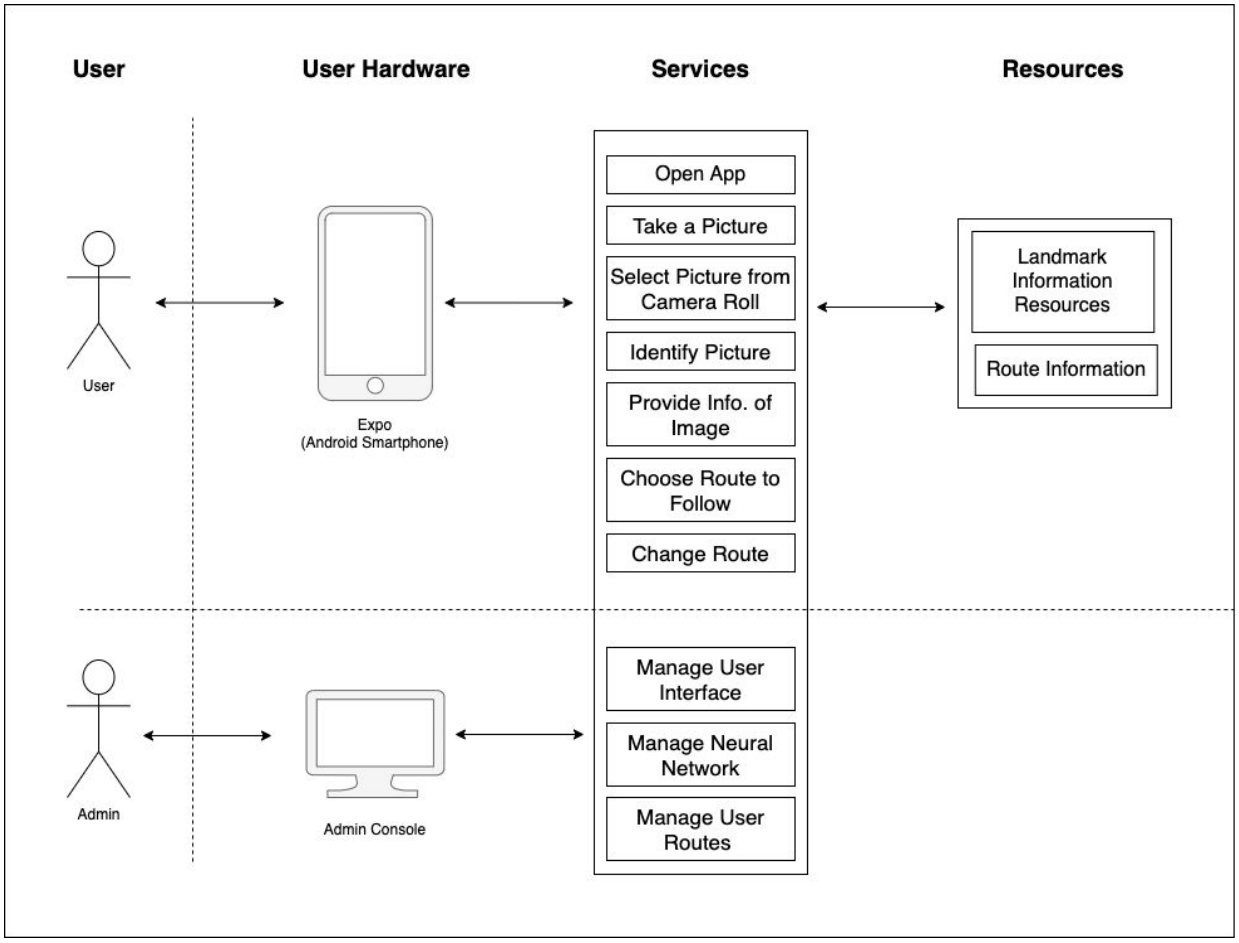


Fig 4.2.
Above illustrates an Architecture Overview Diagram for the mobile application. The purpose of this diagram is to illustrate, at a conceptual level, an understanding of the system based on its users, delivery channel, the service it provides and external resources that interact with the services inside the architecture.

User

The User is anybody who has downloaded the application onto their smartphone for getting information for a particular landmark or exploring a city they are interested in. They only have one delivery channel to gain access to the app and, that is through their mobile device as shown in the diagram above.
The Admin of the application are the users that have the authorization to make any changes to the system. This user interacts with the system through the internal network.

User Hardware

The hardware in this scenario is the user's smartphone. If the user has Expo downloaded, they can run the application on Android or iOS depending on their preference.
The administrator can use the internal network as his choice of hardware. This allows the user to connect directly to the servers, application and image classifier.

Services

1. The user must have downloaded the device correctly to open the application on a mobile device.
2. Take A Picture: This simply enables the user to take a picture using the application using their mobile device.
3. Select Picture From Camera Roll This allows the user to select a picture that they have taken using the app.
4. Identify Picture: This allows the user to submit an image that is saved from their device's storage and have the image identified.
5. Provide Info of Image: This allows the user to learn more about the landmark in their original image.
6. Choose Route to Follow: Allows a user to select a predetermined route stored in the application. This will link them directly to their google maps account and start the route.
7. Change Route: This allows the user to change the route if necessary. It is 100% customisable.
8. Management of the system is allowed only by an admin on the system. This allows them complete access to all backend devices such as the applications user interface, servers and image classifier.

4.3 Model View Controller

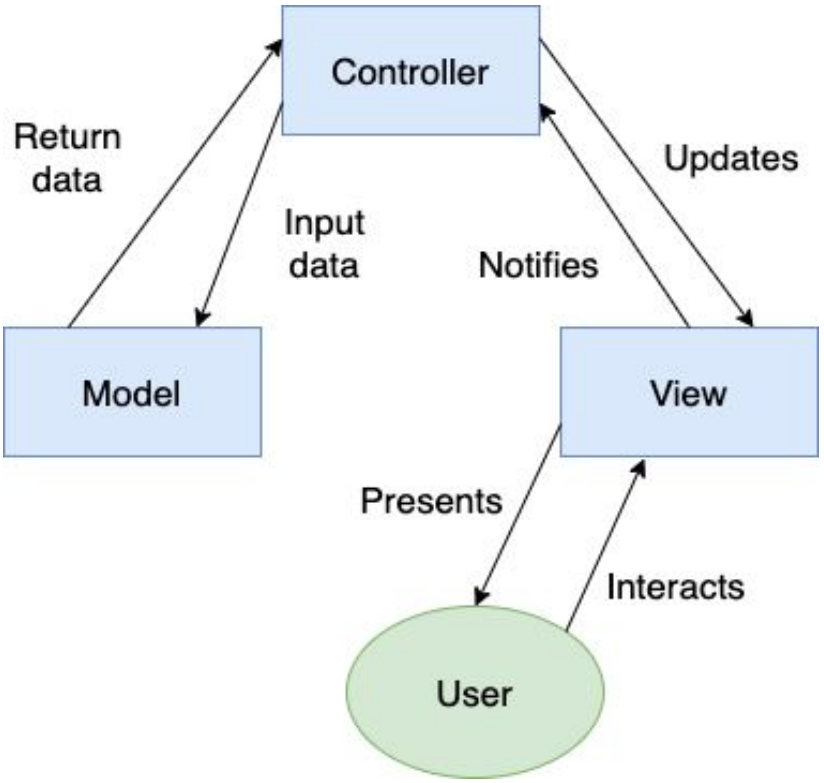


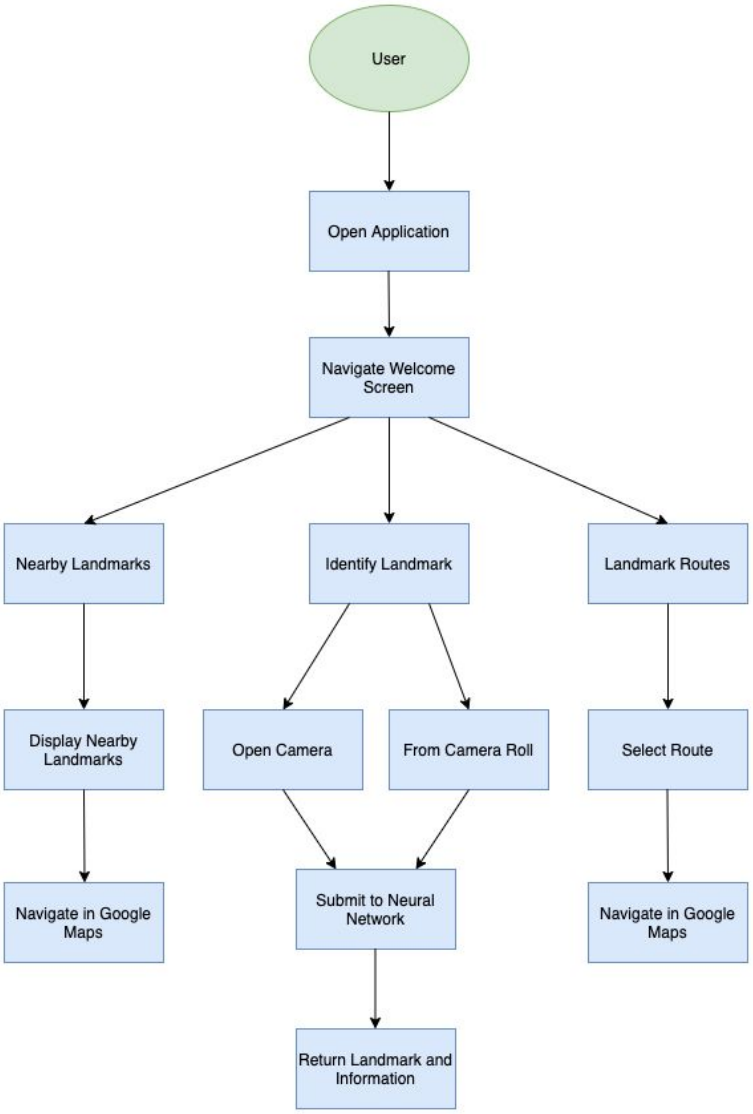
Fig 4.3 shows a Model-View-Controller design architectural pattern of the application. The purpose of a Model-View-Controller design pattern is to illustrate the separation of the modelling, the presentation and the computational actions based on the user's varied inputs.

The Model represents the underlying, logical structure of the application and system. This component model has no knowledge on the interface.

The View consists of the display of information and offers interaction to the user, it interacts with both the User and Controller directly.

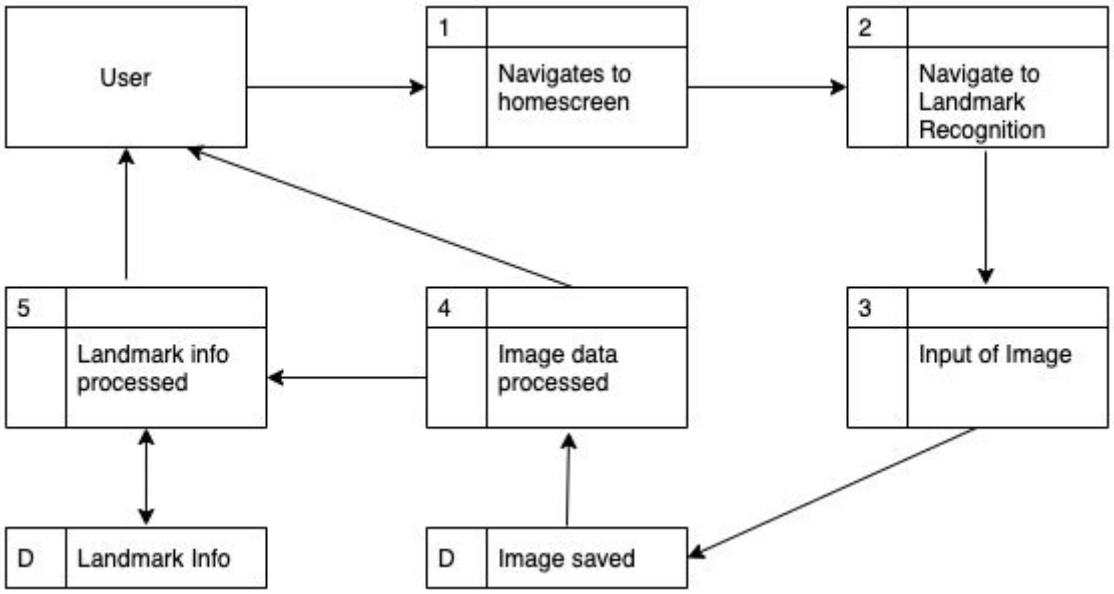
The Controller links in the model and view components of the application. This featured component reacts to the user's input from the view component and passes the output data from the model component back to the view component within the application.

5. High-Level Design



The diagram above illustrates a high-level functionality component. When a user opens the application, they are directed to a welcome screen including information on how to use the application. Following the welcome screen a user can navigate to the many different features on the application. This includes identifying a landmark via an image from camera or camera roll, seeing nearby landmarks and searching landmark routes. Each section includes an end point that is initially requested by the user whether it is returning information, a route or simply searching for a landmark.

5.1 Data Flow Diagram

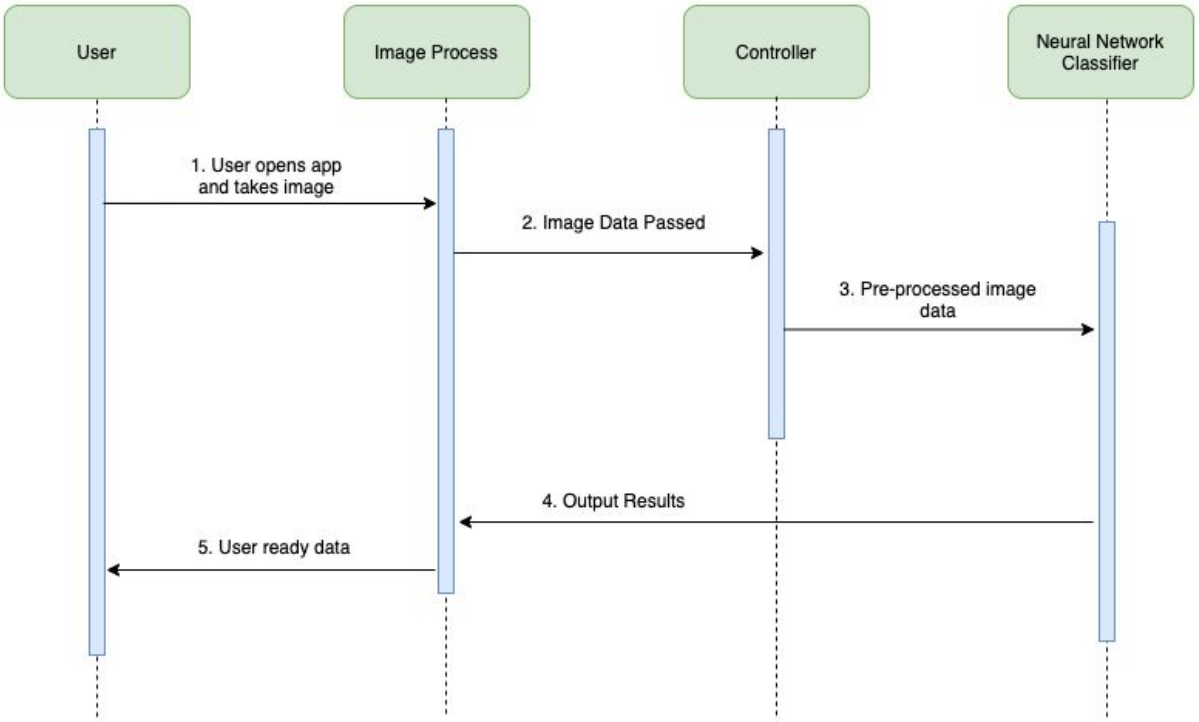


Here we have a data flow diagram, also known as DFD. Data flow diagrams are used to graphically represent the flow of data in a business information system. Data Flow Diagrams describe the many processes that are involved in systems to transfer data from the input to the file storage and reports generation. Data flow diagrams are divided into logical and physical. Here we have taken the action of the landmark recognition system where a user inputs a photo and receives information about the landmark which was selected.

1. User must open the application and navigate to the homescreen of the mobile application.
2. The user must then navigate to the landmark recognition feature within the application.
3. The user will then input the image which will be saved including it's pre processed data that will be used for calculations.
4. The image data is then processed to determine which landmark has been submitted by the user.
5. The landmark info is then processed after the landmark has been identified. All image, description, name and links to relevant websites are then displayed to the user in full.

5.2 Sequence Diagrams

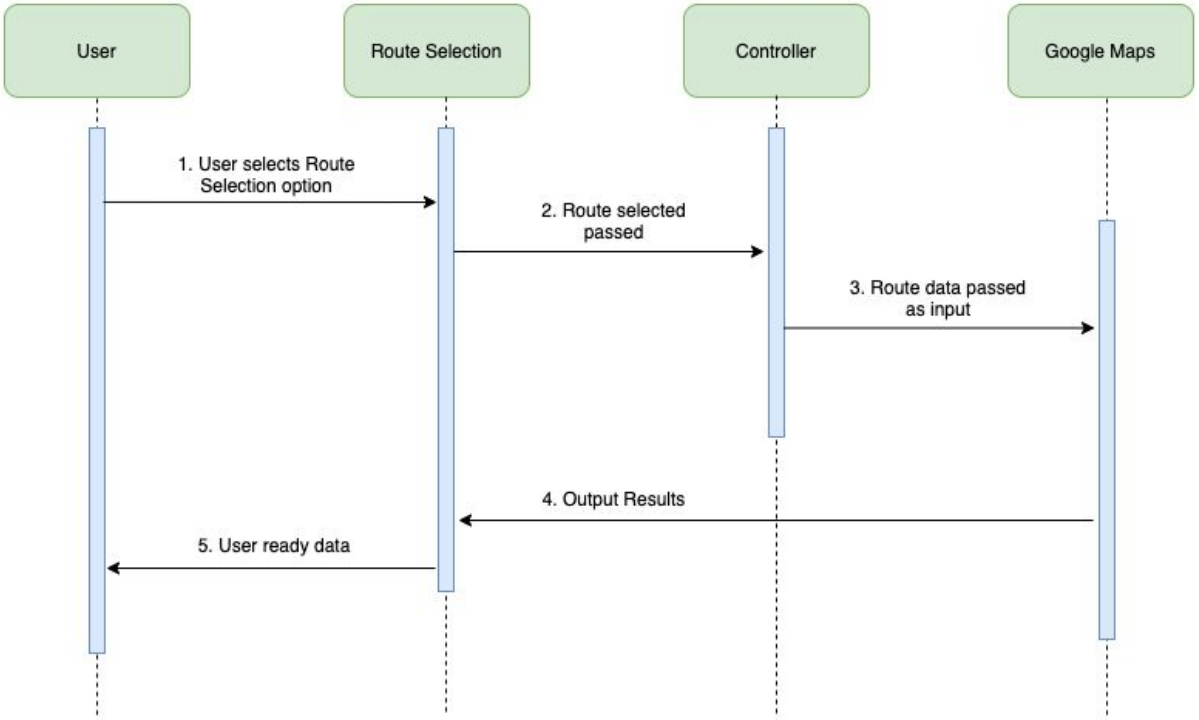
5.2.1 Image Processing Sequence



UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. Here we have a sequence diagram of the process of a user opening the application and using the “Landmark Recognition” feature.

- First the user must open the image section and either take a photo or choose one from their camera roll.
- The data relating to the image is then passed to the “Controller”
- From there the preprocessed image data is passed to the “Neural Network Classifier”
- Once the classifier is complete, it passes the output result to the application where the image was taken.
- The landmark is then displayed to the User on the application

5.2.2 Route Processing Sequence



This figure contains a sequence diagram of the process of a user opening the application and using the “Selected Routes” feature.

- First the user must open the route selection feature on the mobile application
- The data relating to the route is then passed to the “Controller”
- From there the route data is passed as an input to the Google Maps console and loaded there directly
- Once the process is complete, it passes the output result to the application where the route was selected
- The route is then displayed to the User on the application

6. Implementation

Python

1. Data Collection/Manipulation

The dataset used throughout this project was taken from Kaggle. The dataset consisted of several different types of landmark images grouped by a landmark I.D, a link to download the image and an unique ID to the image. We explored the dataset initially before downloading and found there were 1225029 rows in the dataset and 14952 classes. Since we do not have the extreme computation power resource needed for this many inputs, our supervisor suggested that it would be best to only take a subset of these classes, particularly only the most frequent landmark images. After deciding this we had to manipulate the dataset to find the most frequent classes and explore the data further. To do this we used the Pandas Library, this allowed us complete control over the dataset and manipulate it as we needed. We then needed a way to convert the image link into downloadable images, to do this we used the “import request” function.

2. Image Classifier/Neural Network

The application uses Convolutional Neural Networks to classify the input images from the user. The Keras library helped us to build and train the model for this sort of application. This allowed us to customise each of the images before entering the neural network as well as customise the neural network itself. It was decided that using a full scale pixel perfect image would require too much computation power so we decided to convert the images into more manageable pixel sizes before parsing through the neural network. Another feature we utilised was Clarifai, this allows a user to quickly convert an image into an output for a neural network.

React Native

React Native is a JavaScript library for building user interfaces such as mobile applications. We chose this to build our user interface as a challenge as we had not used this type of technology before in a large scale project. We used expo to preview the project which was very useful as hot reloading is enabled and whenever you make a change in React Native the change automatically appears on your phone. React Native provides many useful libraries which we decided to use in our app, such as the ‘expo-camera’, ‘expo-image-picker’ and ‘react-native-maps’. These three were crucial in creating the main features of the app.

‘Expo-camera’ and ‘expo-image-picker’ was used to take a picture in the app, save that picture to the camera roll and then be able to select an image from the camera roll.

‘React-native-maps’ was used to create a map view to show markers for the locations of buildings which are compatible with the app.
Using react native we were also able to create a link to google maps to display predesigned routes.

Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. We chose this service as it linked nicely with the rest of our project which was coded in Python. The Flask server was deployed to help manage the information section regarding the landmarks available within our application.

A server was set up to detect when a landmark was being searched, once an ID for that building was passed to the server, it would compile a list of the landmarks information. This included the Name, Description and relevant links to websites of interest. These details were then returned to the application and displayed for the User to see underneath an image of the landmark.

7. Problems and Resolution

Camera Functionality Issue

When operating in React Native, given that it was a new piece of software for both of us, we found doing very simple functions to be difficult. To solve our small issues we began by following a simple tutorial for building applications online. This initially helped us with our smaller problems such as design, adding images or icons and general page displays. However, the more advanced our application became, the more difficult problems arose. An issue we encountered towards the later phase of development was the accessing of the camera on smartphone devices as well as taking an image and saving it to a user's camera roll. After consulting many different resources such as the React Native Support Forum, Stack Overflow and Youtube we found a resolution that allowed us to take a picture and save it to camera roll.

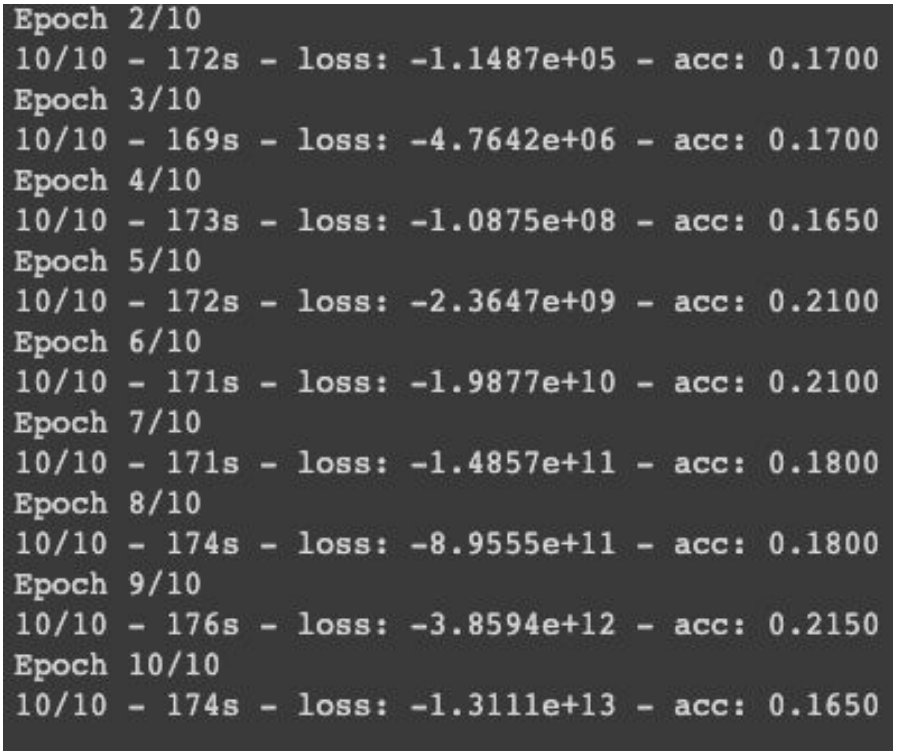
```
import * as MediaLibrary from 'expo-media-library';

MediaLibrary.saveToLibraryAsync(data.uri);
```

To resolve our issue, these two commands were essential in allowing the React Native and Expo platform access to a users Camera and Camera Roll features along with several other examples.

Neural Network Accuracy

When first attempting our neural network for image classification, we faced many different challenges. We chose Keras as our main neural network library as it works seamlessly with TensorFlow and Python. However, we then realised we needed to choose a specific type of neural network approach that suited us best. After much research we found the best suited option for image classification and applied it to our database. At first we had quite a low accuracy of only around 17-22%, this would mean that the system would only guess the correct landmark about 2/10 times.

We quickly found that a number of images within the database had been either corrupted or misdownloaded. We then had to scrape through our dataset and remove all of these images that were causing a significant issue throughout the system.



Another feature we utilised to improve our Neural Network was to deploy several image processing techniques to improve accuracy, clarity and speed when compiling. After correcting our dataset and adding several image processing techniques to our images, we significantly improved our accuracy to a point where we were happy with the results.

8. Future Work

This application has the potential to be developed further by adding more advanced features and slightly changing existing ones. We believe it has a need in today's market of social media and constant travel opportunities.

One feature we would like is to create an option for a user to "Add a Landmark" in an instance where the landmark is not listed in the application a user could enter the landmark and add details surrounding it. This would create a self updating application that gains knowledge from its users.

In future we would also like to increase the number of landmarks the system can recognise. Given our small computation power and limited resources we were only able to identify 10 landmarks. However with further research and development we could increase this number drastically. We would also like to link our Neural Network to our application in a more sophisticated way, the use of Clarifai is helpful but does not allow complete customisation.

9. Conclusion

In conclusion, we found this project interesting, challenging and rewarding throughout. Given the circumstances surrounding our university and on global scale, we utilised every type of technology at our disposal. We learned many new skills in regards to app development, user interface, neural networks and web server development. Each had their own challenges linked to them that we eventually found solutions for.

I believe if we had more opportunities to meet in person and program together we may have developed a greater understanding of each person's work throughout. We found the separation and restriction on meetings to be challenging when tackling things like debugging or error handling. We understand this is a unique situation and we look forward to the possibility of continuing the development of this application.