



# **Tourist Assistant in Dublin using chatbot Final Project Report**

**DT228  
BSc (Ord) in Computer Science**

**Jieun OH  
Mr. Michael Collins**

School of Computing  
Dublin Institute of Technology

**11/04/2022**

# Abstract

Ireland has unique characteristics in terms of Tourism, such as pubs and St. Patrick's Day. Especially Dublin is commonly known for Tourism and many tourists are visiting this city. Also, in terms of the total income of Ireland, Tourism accounts for a significant portion. Therefore, we can say that Tourism plays an important role in Dublin.

Due to COVID-19 from 2020, the number of travellers who visit Dublin has decreased compared to before. However, after a long period of lockdown, the tourism regulations have been easing these days. Therefore, it is expected that Tourism will be revitalized again in Dublin.

As explained earlier, the purpose of this project is to revitalize Tourism in Dublin. The idea mainly focusses on making travel to Dublin easier for the user. To do this, various datasets will be used by requesting to the other data APIs. Also, I implement a chatbot to talk with users interactively by training the network model.

As a result, I will suggest a helpful tourism assistant mobile application for travels who want, plan, or already do a trip to Dublin.

# Declaration

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

Signed:

*Tieun OH*

---

Jieun OH

11/04/2022

# Acknowledgements

While working on this project and dissertation, I have received a great deal of help from grateful people.

First, I would like to thank Professor Emma Murphy, who led the 'Individual Project' module this semester. She gave us lots of assistance and information on the project.

In addition, I could not have completed this dissertation without the support of my supervisor, Professor Michael Collins. He always supported me when I was stuck in difficulties and gave me helpful advice to carry out this project.

To all the colleagues in TU Dublin who have gone through this journey together.

Finally, I would like to thank my parents and friends in South Korea, who always try to do their best to support me not only financially but also emotionally.

# Table of Contents

<b>1. Introduction .....</b>	<b>7</b>
<b>1.1 Overview and Background .....</b>	<b>7</b>
<b>1.2 Project Objectives .....</b>	<b>7</b>
<b>1.3 Project Challenges.....</b>	<b>7</b>
<b>1.4 Structure of the document .....</b>	<b>8</b>
<b>2. Research.....</b>	<b>9</b>
<b>2.1 Introduction .....</b>	<b>9</b>
<b>2.2 Dublin Tourist Research.....</b>	<b>9</b>
<b>2.3 Existing System Research .....</b>	<b>12</b>
<b>2.4 Overview of the technologies .....</b>	<b>14</b>
<b>2.4.1 Development Environment .....</b>	<b>14</b>
<b>2.4.2 Data API.....</b>	<b>15</b>
<b>2.4.3 Chatbot.....</b>	<b>15</b>
<b>3. Design.....</b>	<b>17</b>
<b>3.1 Introduction .....</b>	<b>17</b>
<b>3.2 Design Methodology.....</b>	<b>17</b>
<b>3.3 Use Cases.....</b>	<b>18</b>
<b>3.4 Architecture .....</b>	<b>19</b>
<b>3.5 Entity Relational Diagram.....</b>	<b>20</b>
<b>3.6 User Interface Prototype .....</b>	<b>21</b>
<b>4. Development.....</b>	<b>27</b>
<b>4.1 Introduction .....</b>	<b>27</b>
<b>4.2 Using Data API.....</b>	<b>27</b>
<b>4.3 Developing the Layout.....</b>	<b>29</b>
<b>4.4 Developing the Features.....</b>	<b>33</b>
<b>5. System Validation .....</b>	<b>46</b>
<b>5.1 Introduction .....</b>	<b>46</b>
<b>5.2 Testing .....</b>	<b>46</b>
<b>5.3 Demonstration .....</b>	<b>47</b>
<b>6. Project Plan and Future Work .....</b>	<b>54</b>
<b>6.1 Introduction .....</b>	<b>54</b>
<b>6.2 Project Timeline .....</b>	<b>54</b>
<b>6.3 SMART Goals .....</b>	<b>55</b>
<b>6.3.1 Specific .....</b>	<b>55</b>

6.3.2 Measurable .....	56
6.3.3 Achievable.....	56
6.3.4 Realistic .....	56
6.3.5 Time-based.....	56
6.4 Future Work .....	57
7. Conclusion .....	58
8. References.....	59
9. Appendix .....	62

## **1. Introduction**

### **1.1 Overview and Background**

Dublin is known as one of Ireland's important tourist cities although it has slowed down recently due to COVID-19. Tourism earns a large portion of the total income in Ireland. Therefore, I would like to suggest the application that helps to activate tourism in Dublin using a chatbot assistant.

### **1.2 Project Objectives**

The project objective is to develop a tourist assistant system in Dublin with a chatbot. There are a lot of tourists visiting Dublin every year. Tourists can be an important role in revitalizing the local economy. Therefore, Ireland needs to increase these tourists. The main goal of this project is to simplify Dublin tourism through a chatbot, and various datasets related to Dublin.

### **1.3 Project Challenges**

These are the challenges for this project.

- How to collect vast amounts of data related to Dublin tourism?
- How to connect Data API and mobile application?
- How to train chatbot for the smooth conversation?
- How to make intuitive interface for the user?

## 1.4 Structure of the document

- Introduction
  - In this chapter, I will introduce the overall project.
  - It includes background, objectives, and challenges of the project.
- Research
  - In this chapter, I will provide the detail of the research for the project.
  - It includes research about the topic, and technologies related to the project.
- Design
  - In this chapter, I will discuss the design methodology and diagrams for the project.
  - The screenshots of each design will be attached.
- Development
  - In this chapter, I will discuss how Data API was used and how to implement layouts and features of the project.
  - The screenshots of the code will be attached.
- System Validation
  - In this chapter, I will discuss the testing of the project.
  - The demonstration of the application will be followed.
- Project Plan and Future Work
  - In this chapter, I will discuss the project plan and work to do in terms of the deadline and objectives of the project.



## 2. Research

### 2.1 Introduction

This chapter provides the research related to the project. I will discuss the research on why Dublin tourism is important and overview the technologies to implement this system.

### 2.2 Dublin Tourist Research



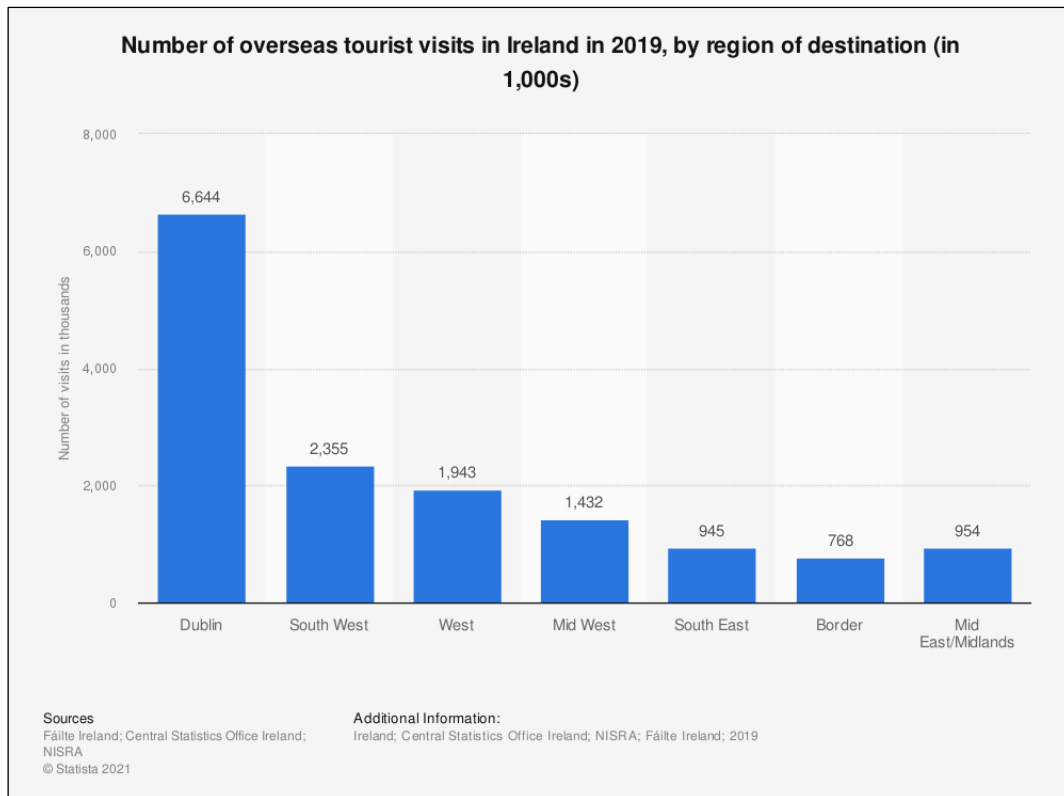
Figure 2.2.1 Rank of the best cities for a getaway

Dublin is the capital city of Ireland. As shown in the Figure above, this city has been selected as the 7th best city for a unique getaway [1]. The reason is that Dublin has a lot of history, attractions, and drinks to enjoy.



*Figure 2.2.2 Contribution of Tourism in Ireland*

As you can see from the Figure 2.2.2, before the covid-19 pandemic, tourism contributed over 10% of the employment in Ireland [2]. We can say that tourism has been one of the biggest industries in Ireland.



*Figure 2.2.3 Number of overseas visits in Ireland*

The Figure 2.2.3 shows the number of tourist visits in Ireland. As you can see, around 6.6 million people visited Dublin in a year for travel [3]. It accounts for about 66% of all tourists in Ireland. To say in one word, improving tourism in Dublin can lead to a good effect on the overall economy of Ireland.

## 2.3 Existing System Research

To differentiate this service from others, I researched services with similar concepts.

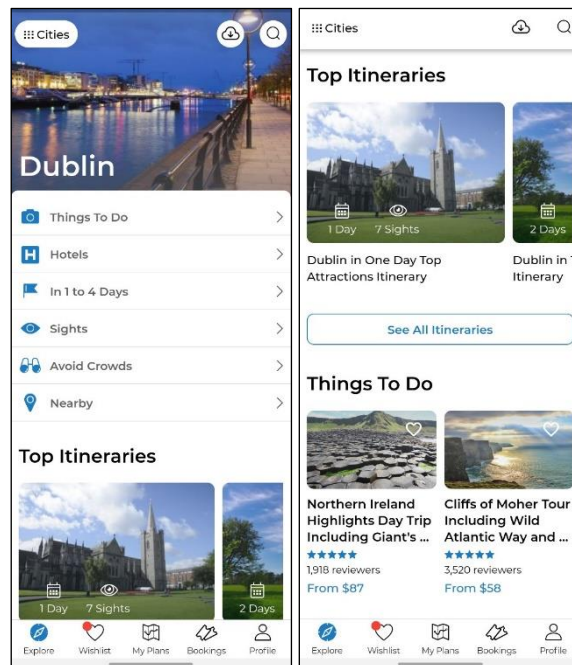


Figure 2.3.1 'Visit a city' Application

The first service is 'Visit A City'. The Figure 2.3.1. shows the interface of this application. This service provides various data such as tourist attractions, tours, or accommodations of many cities [4]. It would be certainly good to use when planning a travel plan. However, I want to make a service used after you arrive in a city. To do this, we need to provide more information such as transportation. Also, tracking the user's location is needed.

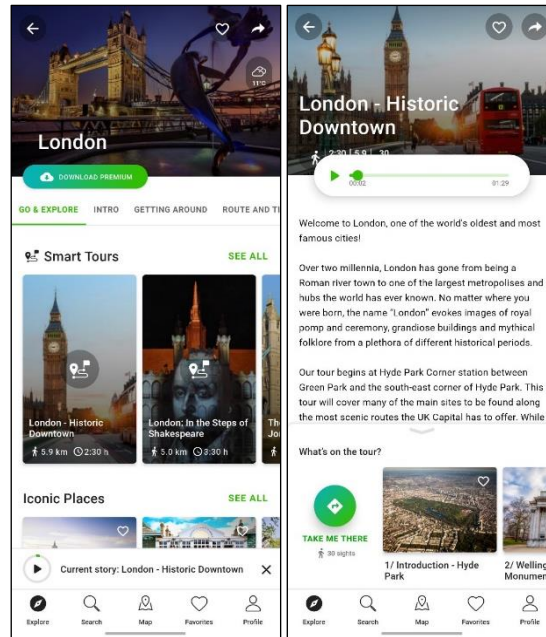
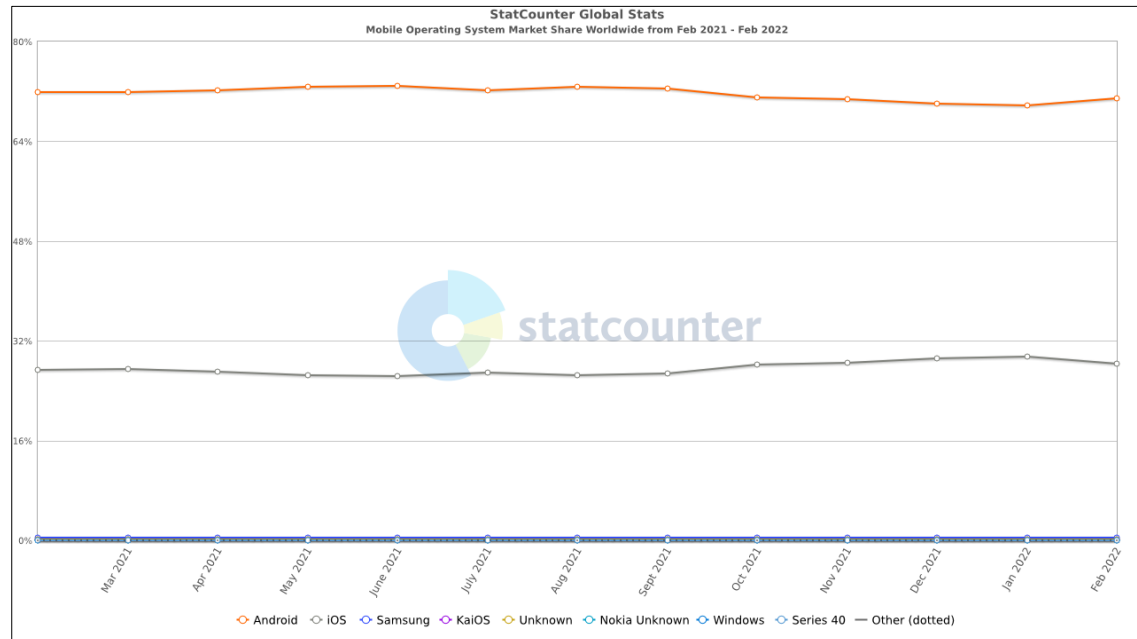


Figure 2.3.2 'SmartGuide' Application

The next service is 'SmartGuide'. The Figure 2.3.2 shows the overall interface of 'SmartGuide' application. The concept of 'SmartGuide' is walking the tourist spot with an audio guide [5]. As people usually prefer simple things, audio-based services can be a good feature. It would be more convenient to put audio into the main function, not just explain the tourist spot.

## 2.4 Overview of the technologies

### 2.4.1 Development Environment



*Figure 2.4.1. Mobile Operating System Market*

This project will aim at a mobile environment for convenient user access. According to the Figure 2.4.1 above, android accounts for 70.97% of the total mobile operating system market [6]. Therefore, the project will target android mobile devices to aim at the market.

The Android Studio can be used to develop the mobile application. Android Studio is the integrated development environment for the android application. It provides various functions that make development easier, such as auto-complete suggestions. Also, the developer can choose the programming language between Java and Kotlin what developers feel more comfortable. In my case, I had programming experience related to Java, so I choose Android Studio, especially Java for the backend, for the development environment of this project.

### 2.4.2 Data API

This project requires a vast number of datasets to provide helpful information about Dublin tourism to the user. The datasets that this project needs include place and transportation data.

#### *Google Map API and Places API*

Google Map and Places API is a service provided by Google. By using HTTP requests, the client can get information about the places [7]. It enables the client to search for a place and get the details. Whenever the service needs the place information, the client can request data to the API and get the corresponding response.

#### *Open Weather API*

Open Weather API provides the accurate weather data in an easy way. By using simple HTTP request with the latitude, longitude, and API key, anyone can access the current location's weather [8].

### 2.4.3 Chatbot

A chatbot is an Artificial Intelligence software that allows users to communicate with a computer as if talking with a human. The key point of a chatbot is to simulate a smooth conversation with the user. To implement this, a chatbot needs to understand the user's intention and formulate responses to the user's question [9]. Here, machine learning is required to recognize user intention.

### *Keras*

To make machine learning simple, Keras can be used. Keras is a neural network API written in Python. It makes the process of creation, training, and prediction of a neural network easy. The only thing that developer needs to do is to define the network model, compile that model, and train the network [10].

### *TensorFlow (Lite)*

TensorFlow is an open-source platform for numerical computation and machine learning developed by Google for machine learning. It allows to create dataflow graphs and each node in this graph means mathematical operation. The edge represents the tensor. Both node and tensor can be a Python object [11].

As Keras is just an interface, Keras needs to run on the TensorFlow server. I especially need to use TensorFlow Lite, which is for mobile devices or embedded system, because this project is based on a mobile environment.



### 3. Design

#### 3.1 Introduction

This chapter presents the details of the design for this project. I will discuss the design methodology that was chosen for the service. The system architecture, use-case, database structure, and user interface prototype will be followed.

#### 3.2 Design Methodology

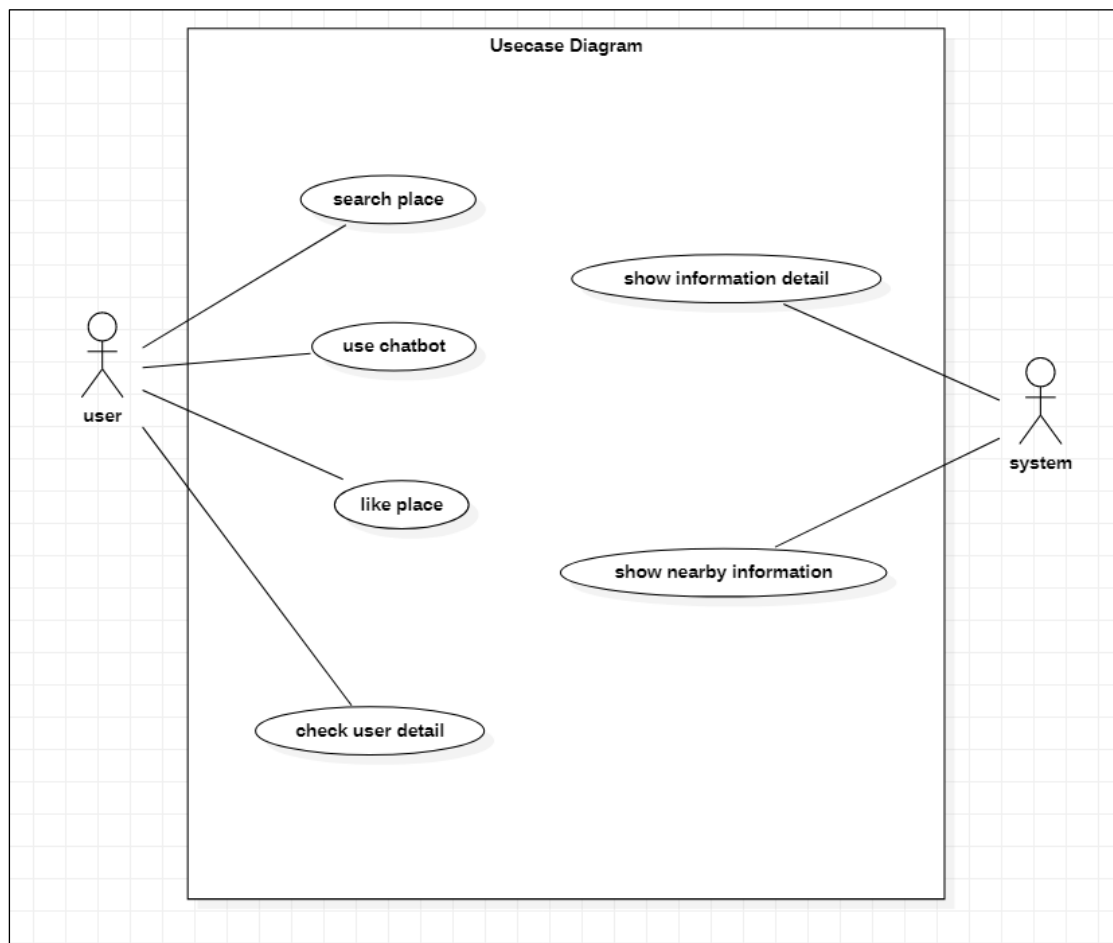


*Figure 3.2.1 Agile Methodology*

For the design methodology of this project, I adopted an Agile Methodology. The Figure above shows the overall process of the Agile Methodology. The Agile Methodology refers to software development methodology based on iterative development. This methodology breaks the project into several manageable phases to make it effective and perform it repeatedly [12].

The reason why I chose this methodology is because of its flexibility. Even if changes occur during the project, I can handle them flexibly. Also, I can break down the project schedule into weekly sprints. By setting sprint goals and achieving them every week, I can manage the project schedule more effectively. During the weekly meetings with the supervisor, he will check the results of these sprints and give feedback.

### 3.3 Use Cases



*Figure 3.3.1 Use Case Diagram*

The Figure 3.3.1 shows the use case diagram for the application. This project needs two actors: the user and the system.

The user can search the place and transportation through the application. They are also able to use the chatbot function by text. If they are authenticated users, they can press the 'like' button and check their detail.

In the point of the system, they can show the information details of the place or the transportation. Also, they can show the nearby information to the user.

### 3.4 Architecture

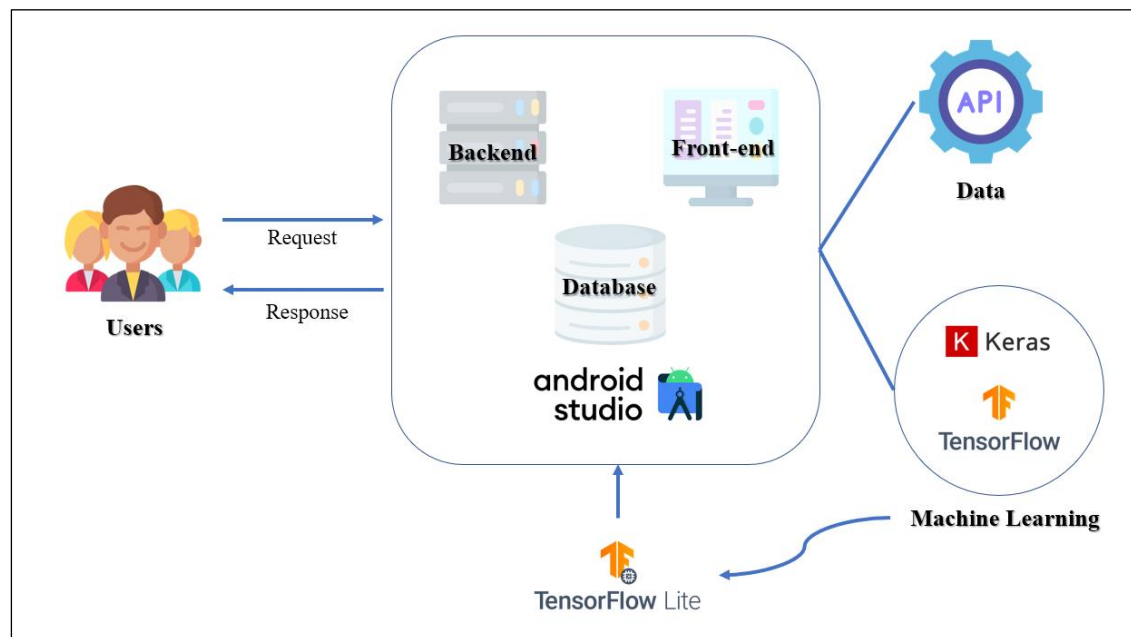


Figure 3.4.1 System Architecture

The Figure 3.4.1 above shows the overall architecture of this project. This project will utilize android studio for development. XML for the front-end, Java for the backend, and SQLite for the database.

I will use API, such as Google Places API, to get data from the organization. Also, Keras and TensorFlow will be used by Python to implement machine learning for a chatbot of the project. Then, to utilize this network model in Android Studio Java environment, I will convert TensorFlow model to TensorFlow Lite model, which can be used in Android Studio.

The user can send a request through the android application, such as searching a restaurant. Then, the application processes a request and sends back a proper response. The user can show this response through the application interface.

### 3.5 Entity Relational Diagram

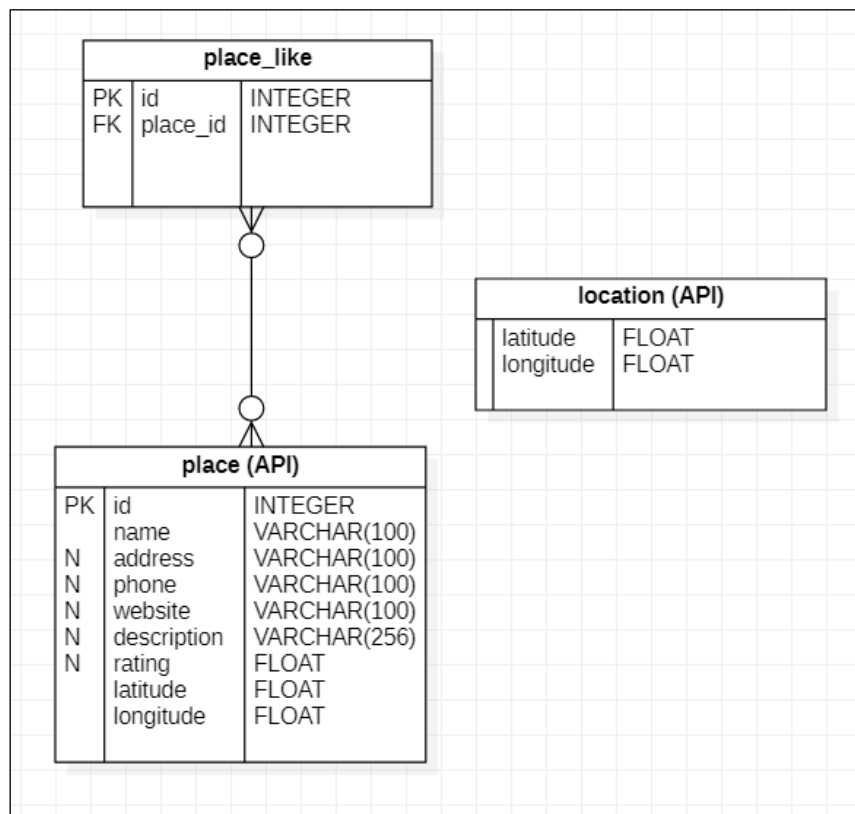


Figure 3.5.1 Entity Relational Diagram

The Figure 3.5.1 shows the database structure for the project. As our project focuses on providing information to the user, various kinds of data are needed.

The main table is a place and location, but I will import the data from Google Maps API and Places API. Therefore, substantial database table is not required. The 'place' data can be a restaurant, pub, or tourist attraction. It includes the detailed information about each place. The 'location' data has the current user's location to show the nearby places. If the user presses the 'like' buttons on specific place, the database registers this detail on the 'place\_like' table with unique place\_id.

### 3.6 User Interface Prototype

The Figure 3.6.1 below shows the user interface of the project. I implemented the system prototype using an interface designing tool named 'Figma'.

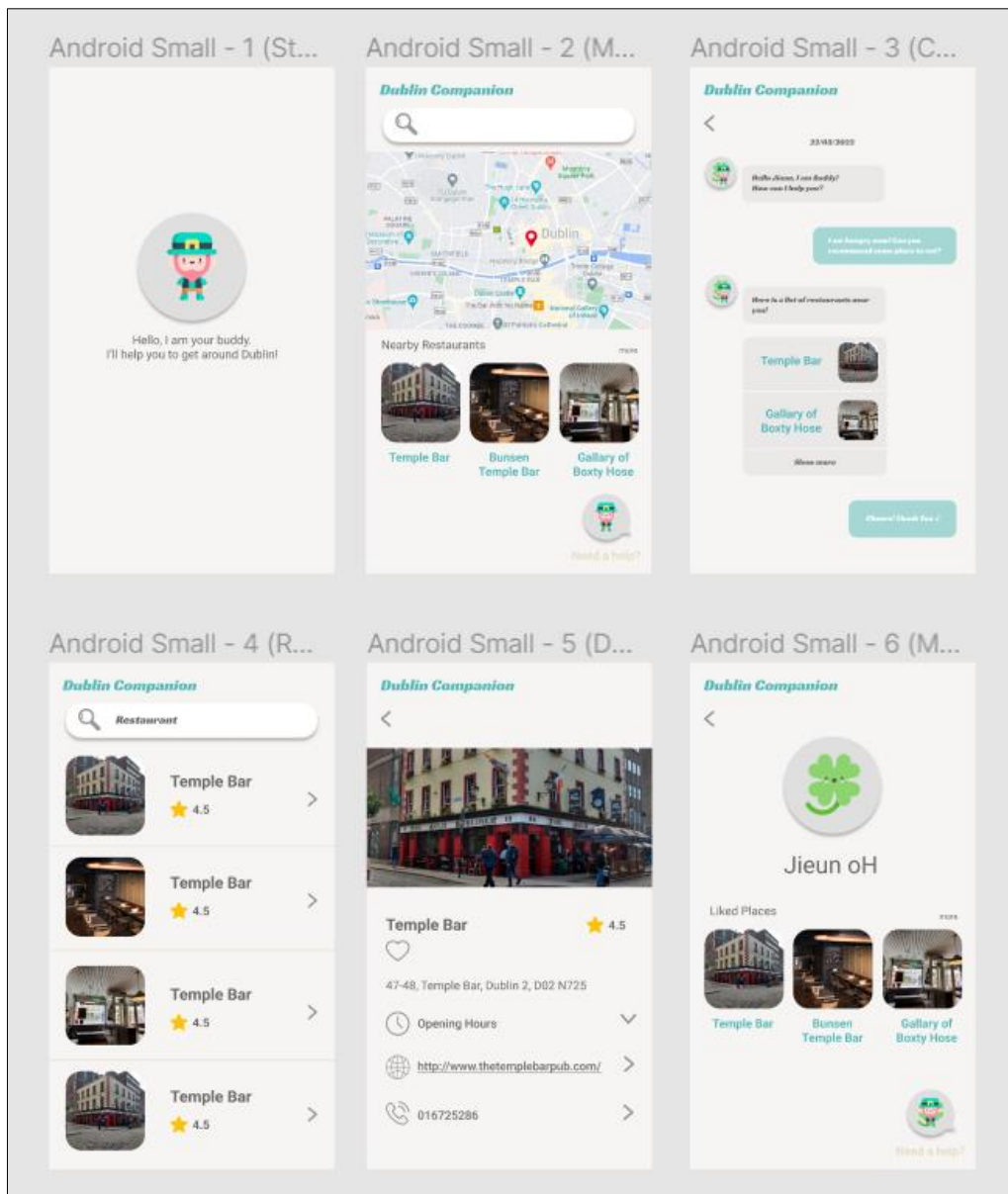


Figure 3.6.1 User Interface

## Main page

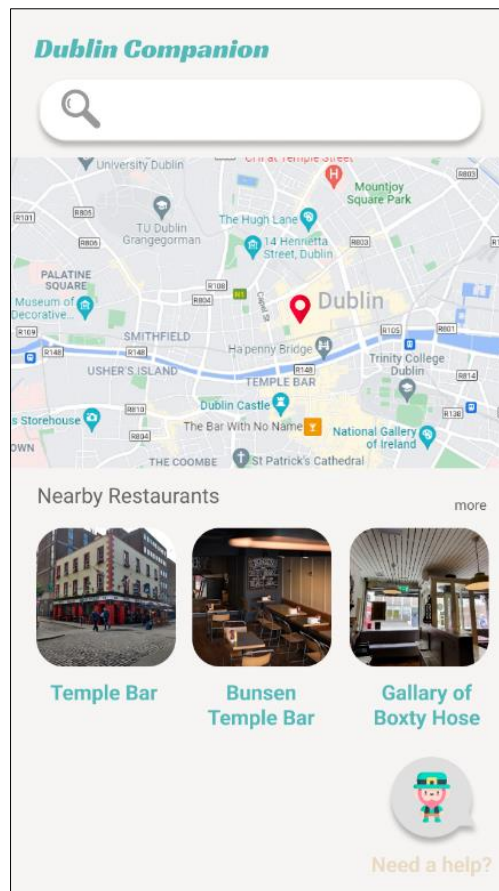


Figure 3.6.2 Main Page

The Figure 3.6.2 shows the default main screen when the user accesses the application at first. The user can search the places they want to know through the search box. They can see the map based on their current location. Also, the application suggests nearby places.

## Chat page

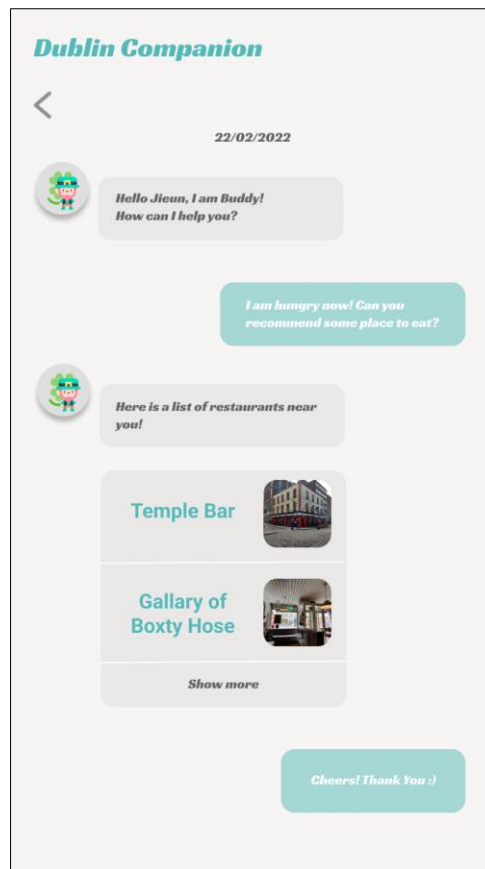
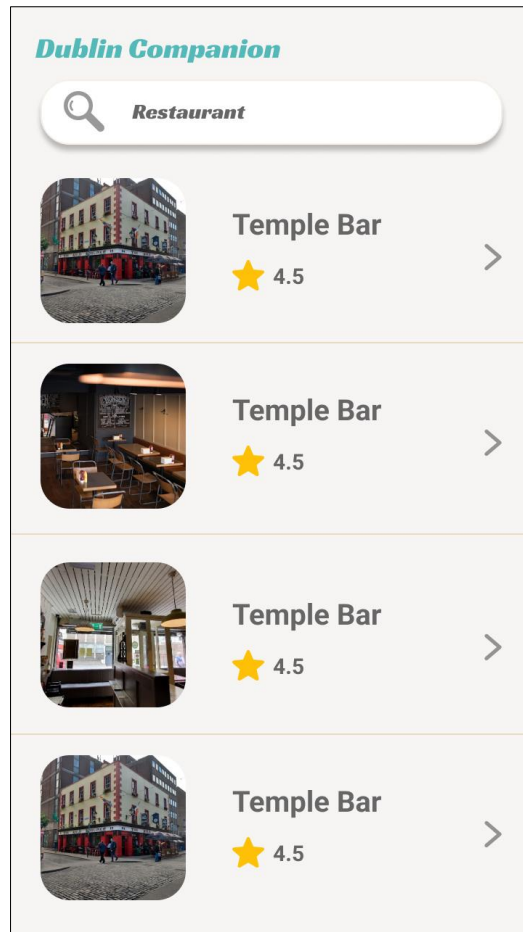


Figure 3.6.3 Chat Page

The Figure 3.6.3 above shows the chat page based on the chatbot function. When the user asks a question about Dublin, the system answers it. Also, the chatbot provides the information related to the user's question.

## *Search Result page*



*Figure 3.6.4 Search Result Page*

The above Figure 3.6.4 shows a page about the result list searched by the user. The system shows the list of the places related to the search keyword. Also, it shows the picture and some details of them. If you click each item of the list, you can move to the detail page of it.



## Detail page

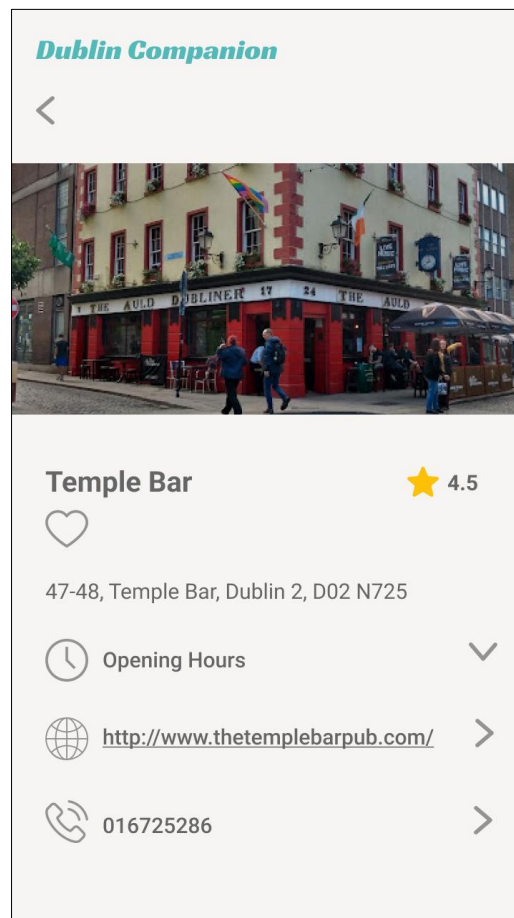
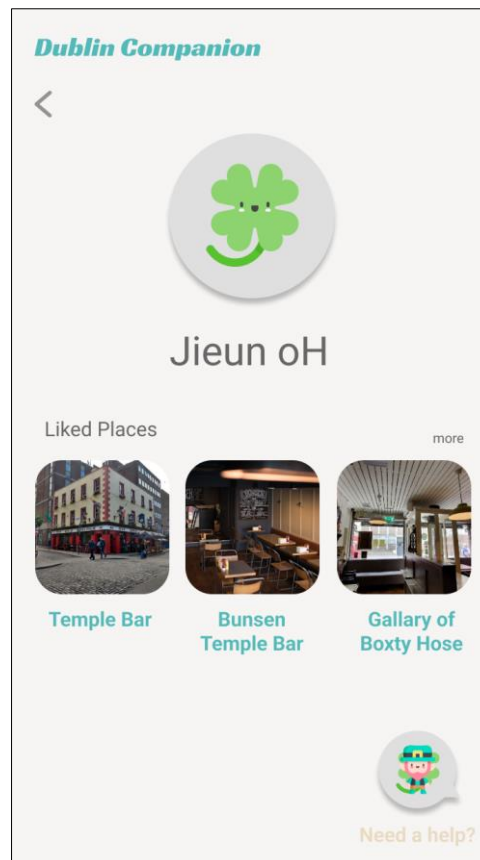


Figure 3.6.5 Detail Page

The Figure 3.6.5 shows the detail page of each place. It contains the overall information of the place, such as name, address, or opening hours. If the user clicks the information of the place, they can see more detail. For example, if the user clicks 'Opening Hours', business hours of the week will be displayed.

## *My page*



*Figure 3.6.6 My Page*

The Figure 3.6.6 represents my page of the system. This page shows the personal information of the user and the places that the user liked.

## 4. Development

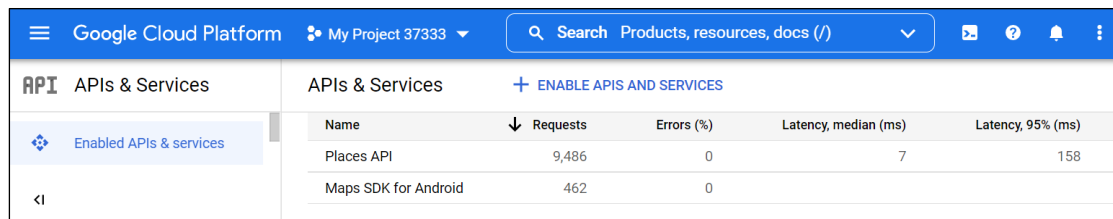
### 4.1 Introduction

This section provides the development process for the project by using code snippets and screenshots of the application. It contains the content of data APIs, databases, and a chatbot.

### 4.2 Using Data API

To provide the tourist information to the user, I need to use various tourist information based on the current user's location.

Through the previous technological research, I chose to use some of the Google APIs. It included the Maps and Places API as below.



APIs & Services		+ ENABLE APIS AND SERVICES			
Name	↓ Requests	Errors (%)	Latency, median (ms)	Latency, 95% (ms)	
Places API	9,486	0	7	158	
Maps SDK for Android	462	0			

Figure 4.2.1 Google APIs

Before utilizing these APIs, I should implement them on the Gradle of the application first and synchronize it.

```
androidTestImplementation 'androidx.test.ext:junit:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
implementation 'com.google.android.gms:play-services-maps:18.0.2'
implementation 'com.google.android.gms:play-services-location:19.0.1'
implementation 'com.google.android.libraries.places:places:2.6.0'
implementation 'com.google.android.material:material:1.7.0-alpha01'
implementation 'com.squareup.okhttp3:okhttp:3.12.1'
implementation 'com.squareup.picasso:picasso:2.71828'
```

Figure 4.2.2 Gradle (app)

Then I need to issue my credential key to use Google Maps and Places APIs.

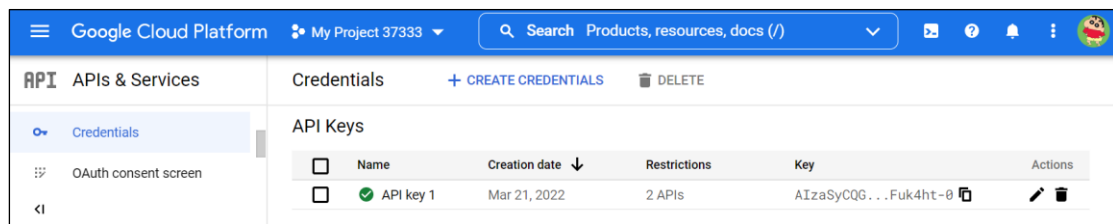


Figure 4.2.3 Google API Key

I declare this key as a variable in the strings.xml file to use it several times. The user has its own API key, so be careful not to expose it.

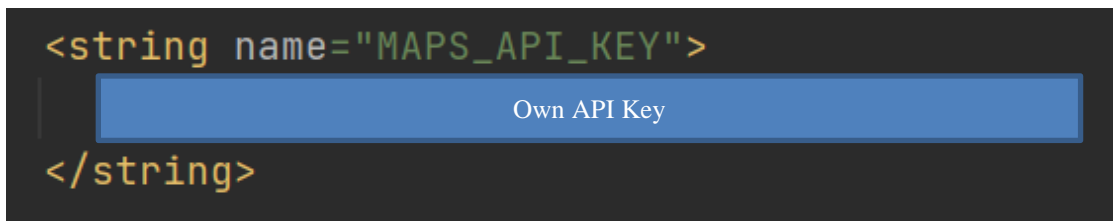


Figure 4.2.4 values/String.html

I can call the API in json format with the API key. Then, the API server returns the results array shown in the picture below. I can process this json format and use it in my application.



Figure 4.2.5 API response

### 4.3 Developing the Layout

I have discussed the User Interface of this service in the Design chapter. When I developed the layout of the application, I have done it based on this UI Design prototype. Therefore, I will briefly explain some layouts without repetition of the previous section.

First, I used two types of views when I implemented the list. One of the Views is 'ListView'. It is a type of View group, and a list of data can be displayed vertically. On the More Restaurant Page and Search Result Page, which are mentioned detailed in the next section, the vertical list will be returned. Therefore, I used the 'ListView' on this layout like below code.

```
<ListView
    android:id="@+id/list"
    android:layout_width="327dp"
    android:layout_height="456dp"
    android:layout_alignParentStart="true"
    android:layout_centerInParent="true"
    android:layout_marginStart="42dp"
    android:layout_marginTop="83dp">
</ListView>
```

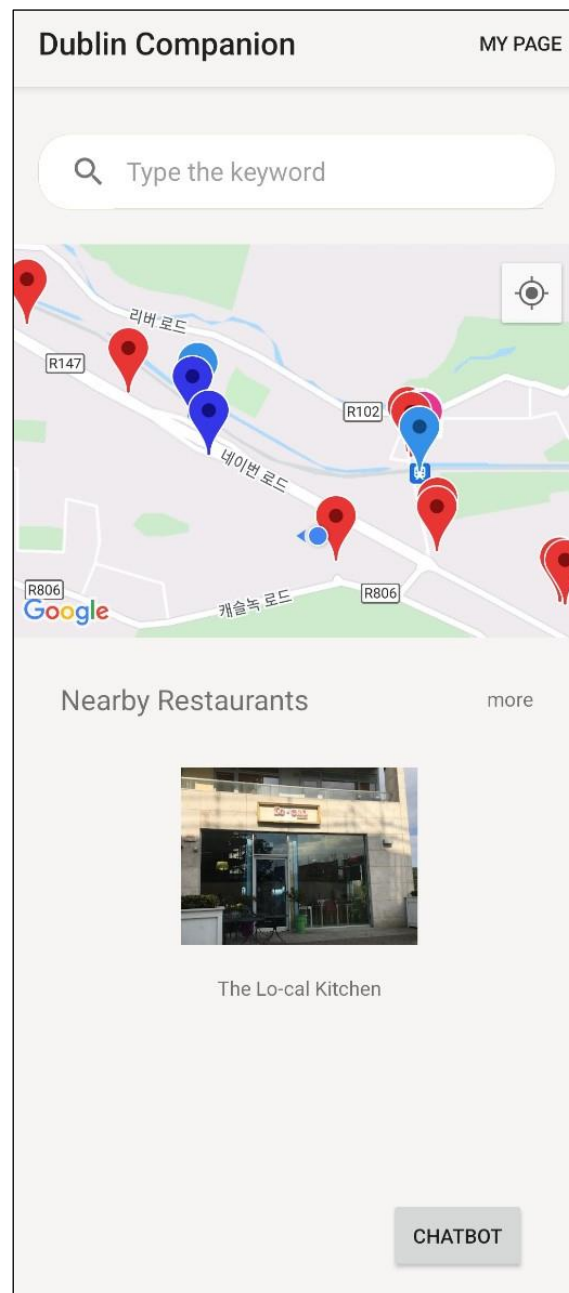
*Code 4.3.1 ListView*

In addition, each row must include the store's name, rating, and photo information. Therefore, I used GridLayout like Code 4.3.2 to set each row item. After that, I connected data received from API and layout using an adapter called 'ListViewAdapter'.

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
```

*Code 4.3.2 ListView Row Item*

On the other hand, on the Main Page, I could not use 'ListView' because this page was designed to use a horizontal list to show nearby restaurants. You can see the Figure 4.3.1.



*Figure 4.3.1 Main Page*

Therefore, to implement a horizontal one, I especially used another View named 'RecyclerView' on this page. As the name suggests, View is recycled and used when scrolling is activated on the service. It can further improve the performance of the data loading [13]. By using 'RecyclerView', I can construct a horizontal list of the nearby restaurant as Code 4.3.3.

```

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="521dp"
    android:layout_height="222dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginEnd="-3dp"
    android:layout_marginBottom="142dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

```

*Code 4.3.3 RecyclerView*

Also, to include multiple information on each row, I needed to implement a layout for 'RecyclerView' item. Only the Image and name of the restaurant are required on each row, so I used ConstraintLayout like Code 4.3.4 to place the name of the restaurant under the image.

```

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <ImageView
        android:id="@+id/image"
        android:layout_width="150dp"
        android:layout_height="150dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/image" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

*Code 4.3.4 RecyclerView Row Item*

To implement the chatbot page, I used the sample messenger layouts [14]. These layouts are divided into three types of xml file: default messenger screen, user message, and chatbot message. Each user message and chatbot message layout uses linear layout to arrange image and message horizontally and card view layout to display them into a card as code 4.3.5. Then, by distinguishing between a user and a chatbot, I can set the text of each message card view.

```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="start"
    android:layout_margin="5dp"
    android:elevation="8dp"
    app:cardCornerRadius="8dp">
```

*Code 4.3.5 CardView*

To put a map screen on the Main Page as figure 4.3.1, 'fragment' was used. By using the 'fragment' layout like below code, I can configure several screens for one activity.

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="0dp"
    android:layout_marginTop="100dp"
    tools:context=".FirstActivity" />
```

*Code 4.3.5 fragment*



## 4.4 Developing the Features

### *Main page*

The Main Page is the default screen of the application. The main page contains the map, the restaurants near the user, and the search box.

First, the map is displayed based on the current user's location. To get this location, I need to get the user's permission [15].

```
private void getLocationPermission() {  
    if (ContextCompat.checkSelfPermission(this, getApplicationContext(),  
        android.Manifest.permission.ACCESS_FINE_LOCATION)  
        == PackageManager.PERMISSION_GRANTED) {  
        locationPermissionGranted = true;  
    } else {  
        ActivityCompat.requestPermissions( activity: this,  
            new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},  
            PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);  
    }  
}
```

*Code 4.4.1 getLocationPermission()*

The method above called 'getLocationPermission' checks the location permission and requests it if it's not allowed. I can call this method on 'onMapReady' callback method. Also, I need to use Google Places API to get the user's location.

To utilize this API, I also need to initialize and declare the related objects, such as 'fusedLocationProviderClient'. After that, by using the fused location provider that I declared before in the method called 'getDeviceLocation', I can find the user's last known location, especially the latitude and longitude. I use this last known location as the user's current location and mark this position on the map.

```

private void getDeviceLocation() {
    try {
        if (locationPermissionGranted) {
            Task<Location> locationResult = fusedLocationProviderClient.getLastLocation();
            locationResult.addOnCompleteListener( activity: this, new OnCompleteListener<Location>() {
                @Override
                public void onComplete(@NonNull Task<Location> task) {
                    if (task.isSuccessful()) {
                        // Set the map's camera position to the current location of the device.
                        lastKnownLocation = task.getResult();
                        if (lastKnownLocation != null) {
                            map.moveCamera(CameraUpdateFactory.newLatLngZoom(
                                new LatLng(lastKnownLocation.getLatitude(),
                                    lastKnownLocation.getLongitude()), DEFAULT_ZOOM));
                            getNearBy(lastKnownLocation);
                        }
                    }
                }
            });
        }
    }
}

```

*Code 4.4.2 getDeviceLocation()*

Not only the user's current position but I also classify the places through markers of various colours. For example, the red marker refers to the restaurant and the green one refers to the tourist attraction. To implement this, I make the asynchronous class named 'NearPlace'. This class sends the requests to the Google Places API on 'doInBackground' method as Code 4.4.3, then organizes the responses from API on 'onPostExecute' method as Code 4.4.4.

```

@Override
protected String[] doInBackground(String... params) {

    OkHttpClient client = new OkHttpClient().newBuilder()
        .build();
    Request request = new Request.Builder()
        .url("https://maps.googleapis.com/maps/api/place/nearbysearch/json?location="
            + params[0] + "," + params[1] + "&type=" + params[3] +
            "&rankby=distance&key=" + params[2])
        .method( method: "GET", body: null)
        .build();
    try {
        Response response = client.newCall(request).execute();
        return new String[] {response.body().string(), params[3]};
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}

```

*Code 4.4.3 doInBackground()*

```

for (int i = 0; i < length; i++) {
    JSONObject resultObject = result.getJSONObject(i);
    String id = resultObject.getString( name: "place_id");
    String name = resultObject.getString( name: "name");
    String photoId = "none";
    if (!resultObject.isNull( name: "photos"))
        photoId = resultObject.optJSONArray( name: "photos")
            .getJSONObject( index: 0).getString( name: "photo_reference");
    JSONObject location = resultObject.getJSONObject("geometry").getJSONObject("location");
    String latitude = location.getString( name: "lat");
    String longitude = location.getString( name: "lng");

    idList.add(id);
    nameList.add(name);
    photoIdList.add(photoId);
    latList.add(latitude);
    lngList.add(longitude);
    typeList.add(type);
}

```

*Code 4.4.4 onPostExecute()*

To provide the nearby restaurant's information to the user, I also use the 'NearPlace' class I previously mentioned. As the 'nearbysearch' of Google Places API does not provide the photo data, I need to request one more API related to the place's photo. Therefore, I implement the asynchronous 'PlacePhoto' class and pass the 'photo\_reference' parameter that I get from the 'NearPlace' class.

```

public static class placePhoto extends AsyncTask<String, Void, Bitmap> {
    @Override
    protected Bitmap doInBackground(String... params) {
        OkHttpClient client = new OkHttpClient().newBuilder().build();
        Request request = new Request.Builder()
            .url("https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photo_reference="
                + params[0] + "&key=" + params[1])
            .method( method: "GET", body: null)
            .build();

        try {
            Response response = client.newCall(request).execute();
            return BitmapFactory.decodeStream(response.body().byteStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(Bitmap response) {
        super.onPostExecute(response);
        try {
        } catch (Exception e) {
        }
    }
}

```

*Code 4.4.5 'placePhoto' class*

Lastly, users can search the places using the search box on the main page. If the user types the keyword and submits it, this searched word is stored in the bundle and sent to the new activity related to the search result. I implemented it on ‘setOnQueryTextListener’ of the SearchView item as below code.

```
SearchView searchView = (SearchView) findViewById(R.id.search_view);
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    // submit search button
    @Override
    public boolean onQueryTextSubmit(String query) {
        // send data to SearchResult activity
        Intent intent = new Intent( packageContext: FirstActivity.this, SearchResult.class);

        Bundle b = new Bundle();
        b.putString("keyword", query); // searched keyword
        intent.putExtras(b);

        startActivity(intent);

        return true;
    }

    @Override
    public boolean onQueryTextChange(String newText) { return false; }
});
```

*Code 4.4.6 setQueryTextListener()*

## Chatbot Page

When the user clicks the 'CHATBOT' button, the application switches to the chatbot screen. On this page, a user can communicate with a chatbot. I used 'Keras' and 'TensorFlow' with Python to implement this function and converted it to 'TensorFlow Lite' file to use it on Android Studio environment.

First, I made test data to train the network model [16]. As shown in the code below, there are 9 kinds of tags classified according to the user's intention. Each tag has its patterns and responses. Therefore, when a user speaks a specific pattern, the application can provide a response corresponding to the tag.

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi there", "How are you?", "Is anyone there?", "Hello", "Good day", "Good Morning", "Good Afternoon", "What's up?"],
      "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?", "Hello, How are you?"],
      "context": [""]
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
      "responses": ["See you!", "Have a nice day", "Bye!", "See you soon"],
      "context": [""]
    },
    {
      "tag": "thanks",
      "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me", "Cheers", "Thanks a million"],
      "responses": ["Happy to help!", "Any time!", "My pleasure", "You're welcome", "No problem!"],
      "context": [""]
    },
    {
      "tag": "noanswer",
      "patterns": [],
      "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
      "context": [""]
    }
  ]
}
```

Code 4.4.7 intents.json

Then, I created a Sequential Model with three layers linearly like code 4.4.8.

```
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))
```

Code 4.4.8 Create Sequential Model

After that, I trained this sequential model with the 'fit' function by using user patterns as the input data and intents as the label value as code 4.4.9.

```
model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
```

*Code 4.4.9 fit the model*

Lastly, I saved this model and converted it to the TensorFlow Lite file to use it on Android Studio.

```
# Save the model
model.save('ex_model')

# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model("./ex_model") # path to the SavedModel directory
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```

*Code 4.4.10 convert the model to the TensorFlow Lite*

On Android Studio, I need to predict the user intention based on this TensorFlow Lite model. To do this, I first needed to map the TensorFlow Lite model on the memory with the function called 'loadModelFile' shown below [17].

```
/** Memory-map the model file in Assets. */
private MappedByteBuffer loadModelFile(Activity activity, String modelPath) throws IOException {
    AssetFileDescriptor fileDescriptor = activity.getAssets().openFd(modelPath);
    FileInputStream inputStream = new FileInputStream(fileDescriptor.getFileDescriptor());
    FileChannel fileChannel = inputStream.getChannel();
    long startOffset = fileDescriptor.getStartOffset();
    long declaredLength = fileDescriptor.getDeclaredLength();
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
}
```

*Code 4.4.11 'loadModelFile' function*

If the user sends the message, the application starts to tokenize the sentence, stem the word, and make BoW (Bag of Words) array with 'input' function as below.

```
// tokenize the sentence
while (tokenizer.hasMoreTokens())
    sentence_word.add(tokenizer.nextToken());

// stem the word
Stemmer stemmer = new Stemmer();
for (int i = 0; i < sentence_word.size(); i++){
    for (int j = 0; j < sentence_word.get(i).length(); j++){
        stemmer.add(sentence_word.get(i).charAt(j));
        stemmer.stem();
        stemmed_word.add(stemmer.toString().toLowerCase());
    }
}

// make Bag of Words array
float [] bag = new float [words.size()];
for (int i = 0; i < bag.length; i ++){
    bag[i] = 0;
}
for (int i = 0; i < stemmed_word.size(); i++) {
    for (int j = 0; j < words.size(); j++) {
        if (stemmed_word.get(i).equals(words.get(j)))
            bag[j] = 1;
    }
}
```

Code 4.4.12 'input' function

Then, through the function shown below, the application classifies the user intention and returns the output array shown in code 4.4.13. I used 0.25 as a threshold value for classification and adopted the highest value if multiple values are classified.

```
float [][] output = new float [1][classes.length - 1];
tf_lite.run(bag, output);
double ERROR_THRESHOLD = 0.25;
int index = 0;
for (int i = 0; i < classes.length - 1; i++) {
    if (output[0][i] > ERROR_THRESHOLD) {
        if (output[0][i] > output[0][index])
            index = i;
    }
}
tag.add(classes[index]);
```

Code 4.4.13 'classify' function

Lastly, the application can determine the response corresponding to the user's pattern with the user's intention obtained from the 'classify' function.

```
try {
    obj = new JSONObject(json);
    JSONArray intents = obj.getJSONArray( name: "intents");
    for (int i = 0; i < intents.length(); i++) {
        JSONObject element = intents.getJSONObject(i);
        if (element.getString( name: "tag").equals(tag.get(0))) {
            JSONArray result = element.getJSONArray( name: "responses");
            response = result.getString(random.nextInt(result.length()));
            if (element.has( name: "context"))
                context = element.getString( name: "context");
        }
    }
} catch (JSONException e) {
    e.printStackTrace();
}
```

*Code 4.4.14 'output' function*

After the application has prepared an answer to the user's message, it can send the corresponding response to the user. They can move to the other page as needed. For example, if a user wants to search the place, the application requests the keyword to the user. Also, if the user wants information about a nearby restaurant, the screen moves to the more restaurant page. I implemented this code by dividing it according to the user's intention pattern as code 4.4.15.

```
switch (intention) {
    case "search":
        if (isKeyword != 0) {
            isKeyword = 0;
            Intent intent = new Intent( packageContext: Chatbot.this, SearchResult.class);
            Bundle b = new Bundle();
            b.putString("keyword", userMsg); // searched keyword
            intent.putExtras(b);
            startActivity(intent);
        }
        break;
    case "like": {
        Intent intent = new Intent( packageContext: Chatbot.this, MyPageActivity.class);
        startActivity(intent);
        break;
    }
}
```

*Code 4.4.15 Based on user's intention pattern*



### *More Restaurant Page*

If the user clicks the 'more' text on the Main Page, the application moves to the More Restaurant Page. This page shows the list of restaurants near the user. Therefore, I used the 'nearplace' class in the same way as showing the list of nearby restaurants on the Main Page. The difference between the main page is that this page requires a rating for each restaurant. Also, to show a list of restaurants, I used ListView on the layout. Therefore, I needed to implement an Adapter called 'ListAdapter' and override the 'getView' method as below.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View row = convertView;
    LayoutInflater inflater = context.getLayoutInflater();
    if (convertView == null)
        row = inflater.inflate(R.layout.row_item, null, attachToRoot: true);

    TextView near_name = (TextView) row.findViewById(R.id.name);
    TextView near_rating = (TextView) row.findViewById(R.id.rating);
    ImageView near_image = (ImageView) row.findViewById(R.id.image);

    near_name.setText(name.get(position));
    near_rating.setText(rating.get(position));
    near_image.setImageBitmap(image.get(position));

    return row;
}
```

*Code 4.4.16 getView()*

### *Search Result Page*

When the user searches the tourist place, the application returns the Search Result Page. First, this page gets the keyword from the previous activity.

```
Bundle b = getIntent().getExtras();  
String keyword = b.getString(key: "keyword");
```

*Code 4.4.17 get keyword*

Then, I executed the 'searchPlace' class to search the places by the searched word. This class is almost like the 'NearPlace' class mentioned before but differs in that it calls other Google Places API related to the search. It can be shown in code 4.4.18.

```
@Override  
protected String doInBackground(String... params) {  
  
    OkHttpClient client = new OkHttpClient().newBuilder()  
        .build();  
    Request request = new Request.Builder()  
        .url("https://maps.googleapis.com/maps/api/place/textsearch/json?query=" +  
            params[0] + "&key=" + params[1])  
        .method("GET", body: null)  
        .build();  
    try {  
        Response response = client.newCall(request).execute();  
        return response.body().string();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
    return null;  
}
```

*Code 4.4.18 doInBackground()*

## Detail Page

If the user clicks each item on the list of the places, it leads to the Detail Page. This page shows detailed information about the selected place, such as an address, website, or phone number. Also, if the users click the website or phone number, they can visit the website or call a place.

To get the detail of them, I used the 'placeDetail' class. This class is also like the 'NearPlace' or 'SearchPlace' class, but it calls the different Places, called 'detail', API on `doInBackground()` method as below.

```
@Override
protected String doInBackground(String... params) {

    OkHttpClient client = new OkHttpClient().newBuilder()
        .build();
    Request request = new Request.Builder()
        .url("https://maps.googleapis.com/maps/api/place/details/json?place_id=" + params[0]
            + "&fields=name,formatted_phone_number,formatted_address,rating,website,opening_hours&key="
            + params[1])
        .method("GET", body: null)
        .build();
    try {
        Response response = client.newCall(request).execute();
        Log.d("tag: \"response\", response.body().toString());
        return response.body().string();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

Code 4.4.19 `doInBackground()`

Also, the user can like the place by clicking the heart icon on the page. If the user clicks the empty heart icon, the application stores this information on the 'Likes' table in the database. When the user cancels it, the column will be deleted from the table. To implement this, I made the 'DatabaseHelper' class as code 4.4.20. The constructor of this class calls the method named 'databaseCheck' to check whether it is already exists or not. If the database named 'Likes' already exists, then the application updates them. If not, they create the new database.

```

public DatabaseHelper(Context context)
{
    super(context, TABLE_NAME, factory: null, DATABASE_VERSION);
    DB_PATH = "/data/data/" + context.getPackageName() + "/databases/";
    this.mContext = context;
    databaseCheck();
}

```

*Code 4.4.20 'DatabaseHelper' Class Constructor*

The class named 'DatabaseManager' executes the proper query, such as select, insert, or delete. The code below shows the one of the functions to select 'Like' information on specific place.

```

public Cursor getLike(String place_id)
{
    return myDatabase.query(DATABASE_TABLE, new String[]{"Place_id"},
        selection: "Place_id"+"=?",
        new String[] {place_id},
        groupBy: null,
        having: null,
        orderBy: null
    );
}

```

*Code 4.4.21 Example Query of the DatabaseManager*

## *My Page*

Lastly, if the user clicks 'MY PAGE' text on the menu bar, it will connect to the My Page. This page simply shows the places that the user liked before. To implement this, I create the 'DatabaseManager' object like code 4.4.22 and call the 'getAllLike' method on 'onCreate()' method as code 4.4.23.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my_page);

    DatabaseManager dbManager = new DatabaseManager(context: MyPageActivity.this);
    try {
        dbManager.open();
    } catch (SQLException throwables) {
        throwables.printStackTrace();
    }
    Cursor cursor = dbManager.getAllLike();
}
```

*Code 4.4.22 Create DatabaseManager object*

```
public Cursor getAllLike()
{
    return myDatabase.query(DATABASE_TABLE, new String[]{"Place_id", "Image", "Name"},
        selection: null,
        selectionArgs: null,
        groupBy: null,
        having: null,
        orderBy: null);
}
```

*Code 4.4.23 getAllLike()*

## 5. System Validation

### 5.1 Introduction

In this section, I will discuss System Validation. It includes the system testing and evaluation of the overall project.

### 5.2 Testing

I conducted black box testing to check if the project was executed normally. The table below shows the test plan that I performed.

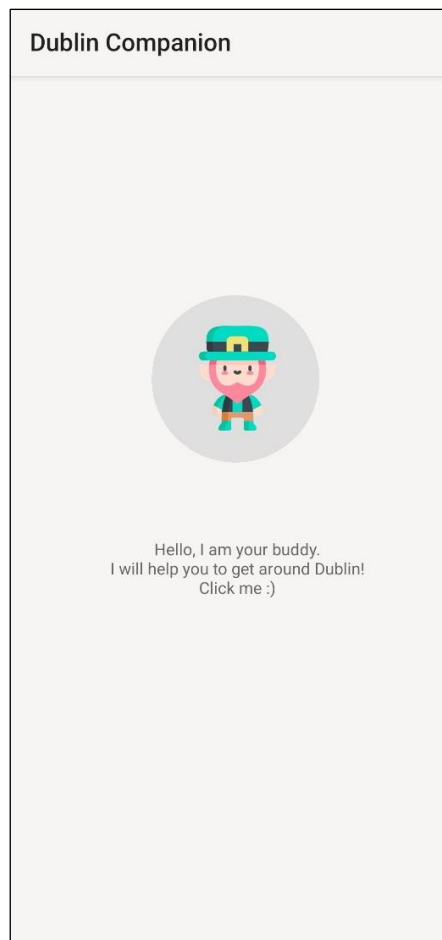
*Table 5.2.1 Testing Plan*

Test Number	Test Description	Pass / Fail
<b>1</b>	Can the user see the current location on the map?	Pass
<b>2</b>	Can the user see the travel attractions on the map based on the current user location?	Pass
<b>3</b>	Are nearby restaurants based on the current user location recommended?	Pass
<b>4</b>	Does the application go to the detail page if the user clicks on an item?	Pass
<b>5</b>	Does the chatbot answer the user's questions appropriately?	Pass
<b>6</b>	Does the chatbot switch to the required screen depending on the situation?	Pass
<b>7</b>	Can the users see more nearby restaurants list if they press the 'More' button?	Pass
<b>8</b>	Are the details of each restaurant displayed on the detail page?	Pass
<b>9</b>	Does it connect to the phone connection or website if the user clicks on the phone number and website?	Pass
<b>10</b>	Can the user press "Like" on the tourist attraction on detail page?	Pass
<b>11</b>	Can the user see the tourist attractions on my page where they pressed 'Like'?	Pass

### 5.3 Demonstration

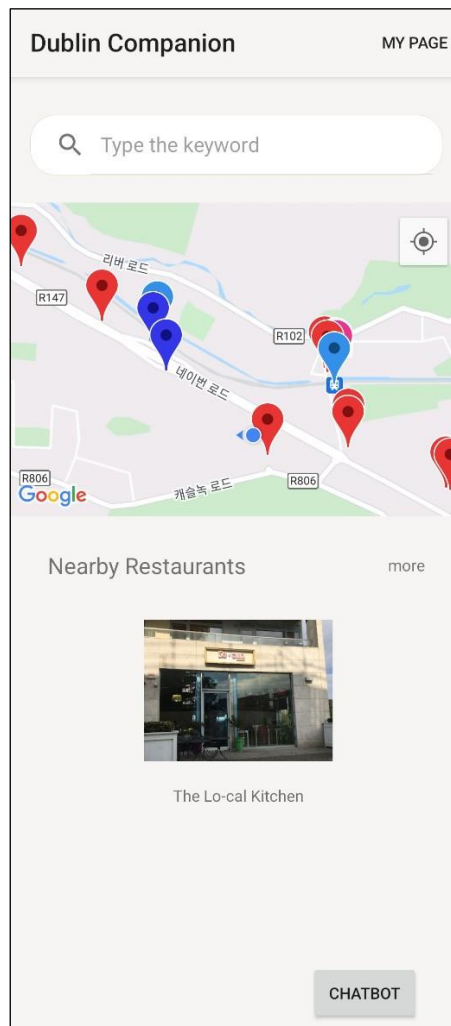
To demonstrate the features that I implemented, I will show some screenshots of the application.

First, when users enter the service, they can see the screen as figure shown below. This page shows the concept of our service briefly.



*Figure 5.3.1 First Page*

If the user clicks the button in the leprechaun character, the main page will be displayed like figure 5.3.2.

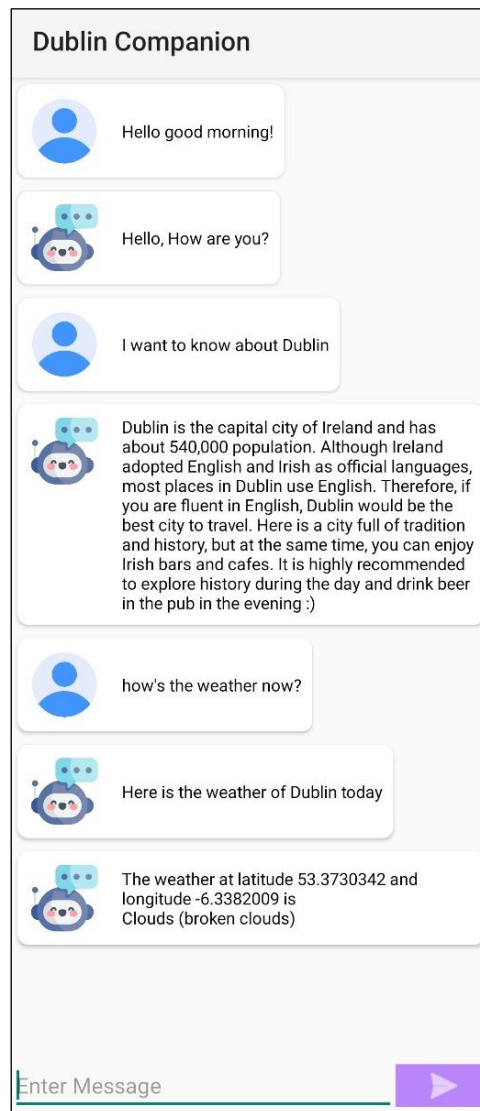


*Figure 5.3.2 Main Page*

On this page, the user can see the map based on their current location. The map shows restaurants, bars, bus stops, train stops, and tourist attractions in different colours. Also, they can see the nearby restaurants in order of distance.



If the users click the 'CHATBOT' button, they can access a chatbot page like feature below.



*Figure 5.3.3 Chat Page*

On a chatbot page, users can ask questions and receive answers. For example, if the user questions the information about Dublin, the chatbot responds with details about Dublin. In addition to this, the user can ask about the weather, nearby restaurants based on their current location, and liked tourist attractions. Also, the chatbot provides the search function with the searched keyword.

By clicking 'More' button, the user can check more nearby restaurants as screenshot below.

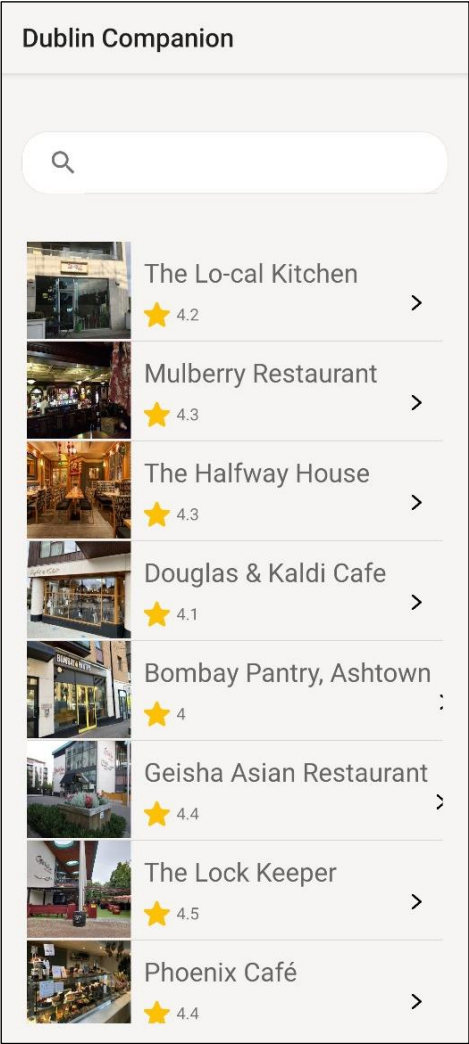
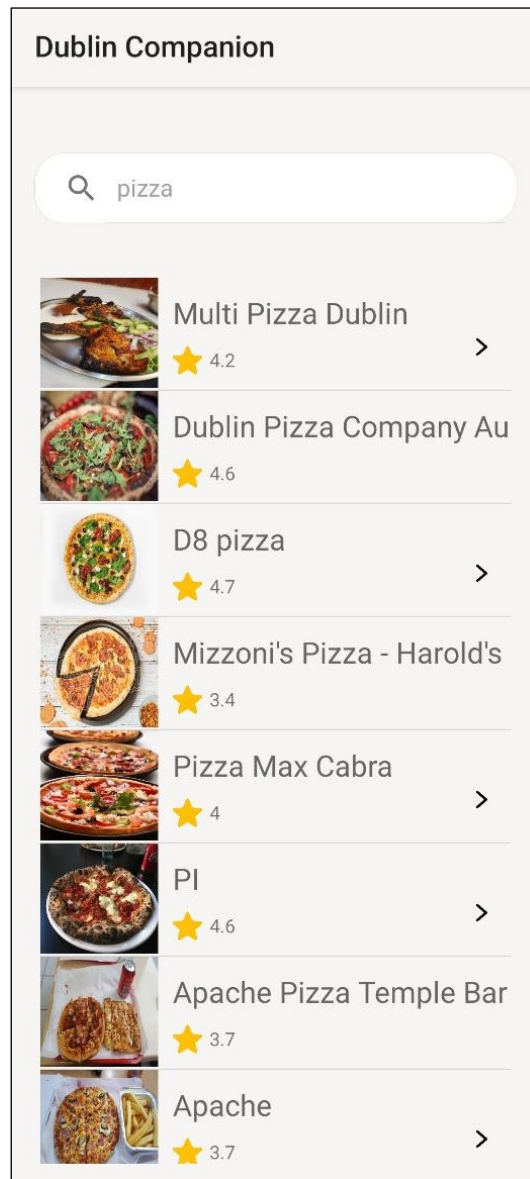


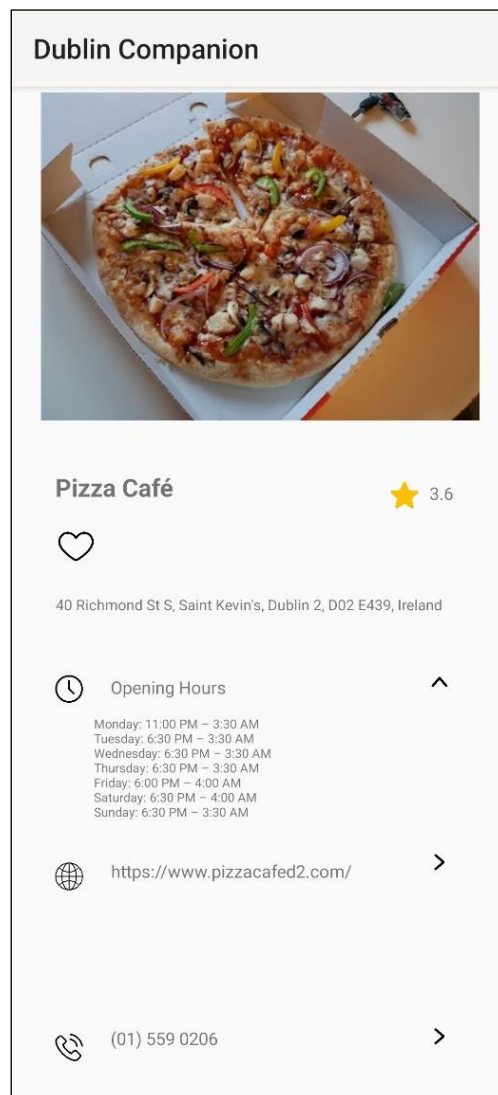
Figure 5.3.4 More Nearby Restaurant

If they want to search some places with text keyword, they can use the search box. Then, the search result will be returned like figure 5.3.5. This figure shows the result screen when the user searches for 'Pizza'.



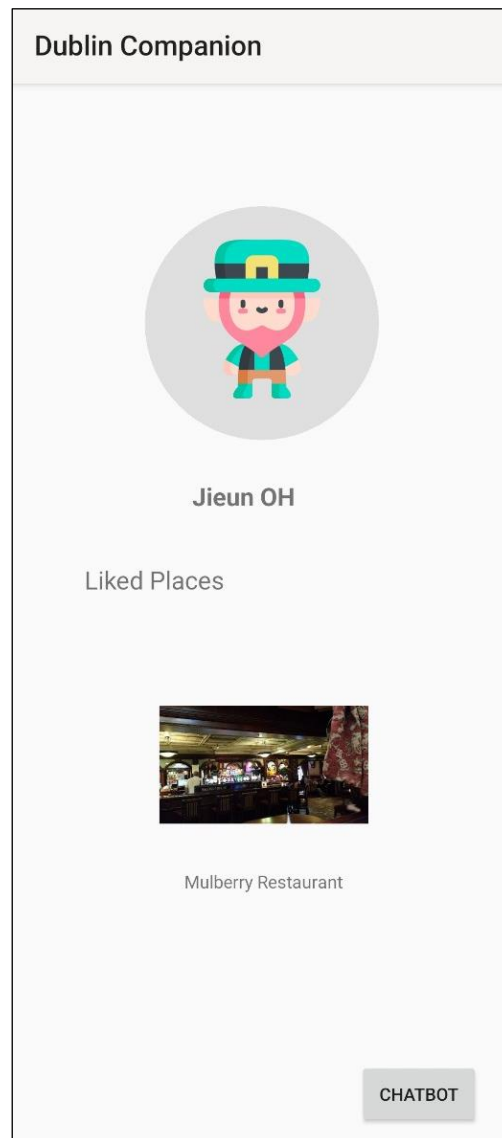
*Figure 5.3.5 Search Result*

If the user clicks the marker or each item of the list, it leads to the detail page of the corresponding place. Detail page shows the name, photo, address, opening hours, website, phone, and ranking information like figure below.



*Figure 5.3.6 Detail Page*

The user can mark this place by pressing 'Like' button. The users can check the list they pressed 'Like' on my page as shown in the picture below.



*Figure 5.3.7 My Page*

## 6. Project Plan and Future Work

### 6.1 Introduction

In this section, I will discuss the plan and future work for the project. It includes a timeline of each task and whether this plan was satisfied the S.M.A.R.T goals. Lastly, future work of the project will be followed.

### 6.2 Project Timeline

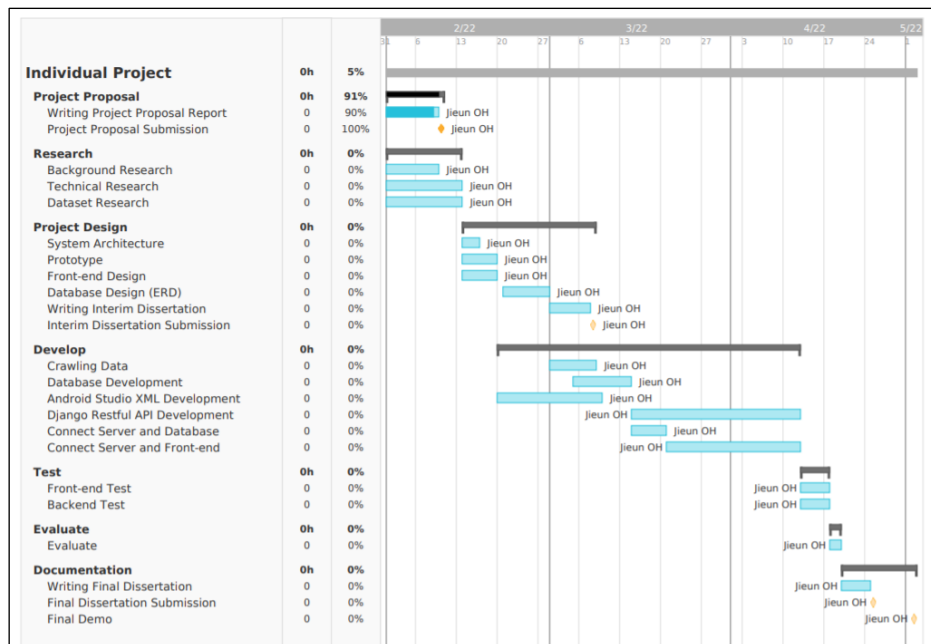


Figure 6.2.1 Timeline of the project

The Figure 6.2.1 above includes the timeline of the project. I used a project management software called 'Trello' to manage the project schedule. The timeline started on the 7th of February and will finish on the 2nd of May. Each task of the project was divided into weeks. Each week has its task to be done and goals.

## 6.3 SMART Goals

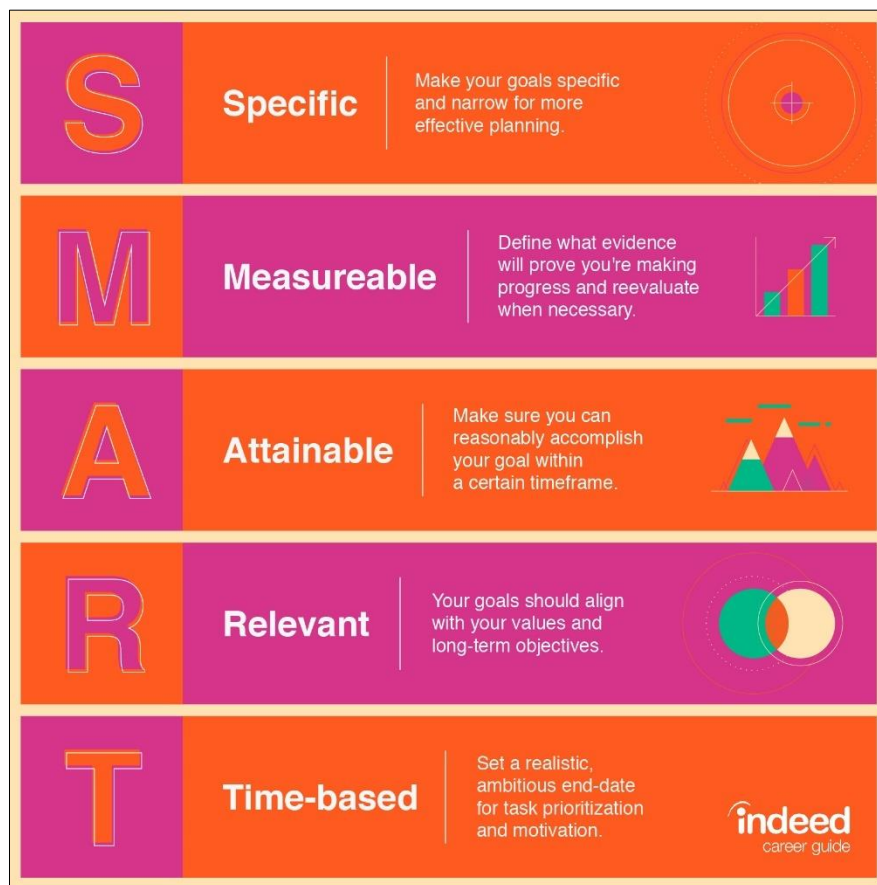


Figure 6.3.1 SMART goals

The Figure 6.3.1 shows the concept of the SMART goals. SMART goals stand for Specific, Measurable, Achievable, Realistic, and Time-based. It helps to guide goal setting and achieve the goals. When I plan this project, I tried to set goals that meet the SMART principle.

### 6.3.1 Specific

This project meets the 'Specific' objectives. Based on the research and design process, I have determined what I specifically want to implement through the project. Therefore, the goal of this project is clear and specific.

### **6.3.2 Measurable**

The project satisfies the 'Measurable' objectives. First, I can measure each goal based on the deadline. Also, I can measure it through the output of each task. For example, output of the API will be followed finishing the backend development. Through the deadline and output, I can measure the project whether it is successful or not.

### **6.3.3 Achievable**

The project meets the 'Achievable' objectives. Although I carry out the project alone, I will carry out this project for a semester. Also, I adjusted the scale of the project that is proper to do alone.

### **6.3.4 Realistic**

This project satisfies the 'Realistic' objectives of S.M.A.R.T. The reason is closely related to the 'Achievable' objectives. As I have enough time and the appropriate size of the work, this project goal is realistic to achieve.

### **6.3.5 Time-based**

I can say this project meets the 'Time-based' objectives. The reason is that the deadline for the project is set at the end of April. Also, I have performed schedule management based on each task. Therefore, there will be no delay in the project schedule.



## 6.4 Future Work

To make a more complete service, huge amounts of data and advanced learning for a chatbot are required. Currently, this application provides services only to Dublin, but it will be better to expand to a wider area. It can be overall Ireland or Europe. Also, through more advanced machine learning, I can improve chatbots smarter to identify the user's intention and provide accurate information.

As this service requests a lot of data at once, especially on the Main Page, it takes time to load them. The slow response of the service makes the user feel uncomfortable using the application. Therefore, for a better user experience, improving the performance of the data load is needed. Minimizing API calls or creating a database for the service instead of utilizing Google API can be a good solution.

## 7. Conclusion

Through the research, I found out that tourism plays a big role in Ireland, especially in Dublin. To revive Tourism, which has been reduced due to the pandemic, this application was designed in the beginning.

In consideration of time limitations and flexibility of the project plan, I adopted Agile Methodology. With this methodology, it was possible to perform the project more efficiently by dividing the task by the weekly sprint unit.

Considering the usability of travellers who do not carry devices other than mobile phones, Android applications seemed most suitable for this project. To get the multiple data related to the tourist attraction based on the user's location, I chose to use Google Maps API and Places API. Also, it seemed most suitable to use TensorFlow lite to train a chatbot.

As I had never studied machine learning before, it seemed impossible to implement a perfect chatbot. However, it was a new but beneficial challenge for me this semester.

In conclusion, this application has potential to activate the Dublin tourism again by providing detailed information related to the tourism and useful chatbot.

## 8. References

- [1] Damian Porter Affiliate Partnership Manager, Porter D, Manager AP. The most unique getaways around the world [Internet]. William Russell. 2022 [cited 2022Mar8]. Available from: <https://www.william-russell.com/blog/most-unique-getaways-around-world/>
- [2] Published by Statista Research Department, 5 N. Travel and tourism: Employment contribution ireland 2020 [Internet]. Statista. 2021 [cited 2022Mar8]. Available from: <https://www.statista.com/statistics/942013/travel-and-tourism-employment-contribution-ireland/>
- [3] Published by Statista Research Department, 4 N. Overseas tourist visits by region Ireland 2019 [Internet]. Statista. 2021 [cited 2022Mar8]. Available from: <https://www.statista.com/statistics/660138/number-of-overseas-visits-to-ireland-by-region/>
- [4] Visit a city – apps on google play [Internet]. Google. Google; [cited 2022Mar8]. Available from: [https://play.google.com/store/apps/details?id=com.visitacity.visitacityapp&hl=en\\_IE&gl=US](https://play.google.com/store/apps/details?id=com.visitacity.visitacityapp&hl=en_IE&gl=US)
- [5] SmartGuide – your personal travel audio guide – apps on google play [Internet]. Google. Google; [cited 2022Mar8]. Available from: [https://play.google.com/store/apps/details?id=org.smart\\_guide.smartguide.T\\_00007&hl=en\\_IE&gl=US](https://play.google.com/store/apps/details?id=org.smart_guide.smartguide.T_00007&hl=en_IE&gl=US)
- [6] Mobile Operating System Market Share Worldwide [Internet]. StatCounter Global Stats. [cited 2022Mar8]. Available from: <https://gs.statcounter.com/os-market-share/mobile/worldwide>
- [7] Bean J. How to use the google places API for location analysis and more [Internet]. Medium. Towards Data Science; 2021 [cited 2022Mar8]. Available from: <https://towardsdatascience.com/how-to-use-the-google-places-api-for-location-analysis-and-more-17e48f8f25b1>

- [8] OpenWeatherMap.org. Current weather data [Internet]. OpenWeatherMap. [cited 2022Apr11]. Available from: <https://openweathermap.org/current>
- [9] Chatbot: What is a chatbot? why are chatbots important? [Internet]. Expert.ai. 2022 [cited 2022Mar8]. Available from: <https://www.expert.ai/blog/chatbot/>
- [10] Team K. Keras documentation: About keras [Internet]. Keras. [cited 2022Mar8]. Available from: <https://keras.io/about/#:~:text=Keras%20is%20a%20deep%20learning,key%20to%20doing%20good%20research.>
- [11] Yegulalp S. What is TensorFlow? The Machine Learning Library explained [Internet]. InfoWorld. InfoWorld; 2019 [cited 2022Mar8]. Available from: <https://www.infoworld.com/article/3278008/what-is-TensorFlow-the-machine-learning-library-explained.html>
- [12] Agile Project Management [Internet]. Agile Project Management - A Beginner's Guide | Adobe Workfront. [cited 2022Mar8]. Available from: <https://www.workfront.com/project-management/methodologies/agile>
- [13] Create dynamic lists with RecyclerView; android developers [Internet]. Android Developers. [cited 2022Mar20]. Available from: <https://developer.android.com/guide/topics/ui/layout/recyclerview>
- [14] How to create a chatbot in Android with brainshop API? [Internet]. GeeksforGeeks. 2021 [cited 2022Apr5]. Available from: <https://www.geeksforgeeks.org/how-to-create-a-chatbot-in-android-with-brainshop-api/>
- [15] Select Current Place and Show Details on a Map [Internet]. Google. Google; [cited 2022Mar20]. Available from: <https://developers.google.com/maps/documentation/android-sdk/current-place-tutorial>
- [16] Baranovskij A. Build it yourself-chatbot API with Keras/TensorFlow Model [Internet]. Medium. Towards Data Science; 2019 [cited 2022Apr10]. Available from:

<https://towardsdatascience.com/build-it-yourself-chatbot-api-with-keras-TensorFlow-model-f6d75ce957a5>

[17] Moroney L. Using TensorFlow lite on Android [Internet]. The TensorFlow Blog. [cited 2022Apr10]. Available from: <https://blog.tensorflow.org/2018/03/using-TensorFlow-lite-on-android.html>

## 9. Appendix

### *Code for Creating Sequential Model for Chatbot (Python)*

```
# things we need for NLP
import nltk
from nltk.stem.porter import PorterStemmer

# things we need for Tensorflow
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import SGD
import pandas as pd
import pickle
import random

# import our chat-bot intents file
import json
with open('intents.json') as json_data:
    intents = json.load(json_data)

stemmer = PorterStemmer()

words = []
classes = []
documents = []
ignore_words = ['?']

# loop through each sentence in our intents patterns
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        w = nltk.word_tokenize(pattern)
        # add to our words list
        words.extend(w)
        # add to documents in our corpus
        documents.append((w, intent['tag']))
```

```

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# stem and lower each word and remove duplicates
words = [stemmer.stem(w.lower()) for w in words if w not in
ignore_words]
words = sorted(list(set(words)))

# sort classes
classes = sorted(list(set(classes)))

# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)

# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # stem each word - create base word, in attempt to represent
related words
    pattern_words = [stemmer.stem(word.lower()) for word in
pattern_words]
    # create our bag of words array with 1, if word match found in
current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each
pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])

```

```

# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)

# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])

# Create model - 3 layers. First layer 128 neurons, second layer 64
neurons and 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),),
activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))

# Compile model. Stochastic gradient descent with Nesterov
accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])

# Fit the model
model.fit(np.array(train_x), np.array(train_y), epochs=200,
batch_size=5, verbose=1)

import tensorflow as tf

# Save the model
model.save('ex_model')

# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model("./ex_model") #
path to the SavedModel directory

```



```
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```