**Dublin Business School**

**Module B8IT120**

**Mobile Application Development**

**Continuous Assessment Report**

|  |  |
| --- | --- |
| **Name:** | Dominic Dolan |
| **Student Number:** | 10369237 |
| **Module Leader:** | Darren Redmond |
| **Module Code:** | B8IT120 |

# Introduction

The app that was built was an unofficial Dublin bikes app. It was developed using a native approach rather than hybrid because that generally leads to a better user experience and I have more experience making such an app. The app uses JCDecaux’s API to retrieve the required data. The app makes use of several of googles play services APIs such as Google maps, Google places and Google location services. It also makes use of Google’s firebase service for signing in and holding data for each user.

## Objectives

The target audience for the app is mostly people who live and/or work in Dublin and often need to use the Dublin Bikes service.

When people use this app, they are likely to be on the move so the app is designed to get the required information on to the screen quickly. For this reason, there is one main activity that can do everything required of the app. Secondly, the main app screen is designed to make good use of space. This is done by having no action bar to give more space to the map and having the movable info panel which can be moved out of the way as needed.

# App Description

## UI Design

The main part of the app contains a map fragment and an info panel which can be moved further into view by the user. This means the user can decide whether they want to see a lot of the map or a lot of the info panel or a little bit of both. There is also a navigation drawer which can be accessed by swiping in from the left or by tapping the menu icon in the top left corner

### Map Fragment

The map fragment is for holding the map view of the appropriate area and all the necessary markers that show the locations of the bike stations. The markers of the bike stations can show a number in the marker, this number refers to either the available bikes or the available spaces at each station. This is to achieve the objective of getting the necessary information in front of the user quickly.

### Info Panel

The info panel contains info about the selected marker, it can perform a search, it can show the users favourites list and it can show stations near the user’s location. The info panel can be moved vertically on the screen so that the user can decide how much or how little of the info panel they want to see.

## Back-end and APIs

To get this app working with all its functionality, various APIs were required. The most important of which was the Dublin Bikes API. This API call downloaded a JSON file from JCDecaux, it then needed to be parsed into a list of objects.

The various google maps services were used quite extensively. The basic Google maps service needed to be used for creating the map fragment and placing all the necessary markers on it. The Google location service needed to be used for properly getting access to the user’s location. The Google places service needed to be used for performing a Google Maps search and finding the coordinates of addresses.

Firebase was used to allow the user to log in to the app and have their information persistent across devices. The firebase service keeps track of any stations that the user marks as one of their favourites and it also keeps track of the journeys that the user has been on.

A SQLite database was set up so that the information downloaded from the APIs would be persistent even if the user goes offline or loses signal.

## App structure

The app is made up of 5 different activities. The primary one, and the most complex one, is the main activity. This is the activity which holds the map fragment and all the content that caused the user to use this app.

The second activity and likely the first activity that the user will see is the login activity. This activity is shown to the user the first time they start the app. The user can then either register, sign in or skip registration. If the user skips then they will not be asked to sign in on start up the next time they use the app, however, if they want to sign in later they can do from various other areas of the app.

The favourites list and the journeys list are the next two activities which both work in a very similar way. They both use list views and list adapters to display the necessary data from firebase. The favourites list allows the user to remove a favourite from their list. These activities cannot be opened if the user is not logged in.

The last activity is a settings activity. This activity is based off the default android settings views. It contains settings for things like being able to login, turning off the menu icon and display name.

# Search Algorithm

The search algorithm is one of the most advanced parts of this app. The search was designed to be as fast as possible although if I had more time I can think of ways to make it faster.

The user must type in a word that contains 4 characters or more before the search activates. The first thing that occurs when a search string is sent is that the string is simply matched with the station names using the string’s built in contains method.

The next thing that needs to be done is a little more complicated. When a user types in a place name that is not a match to a station name then the app should instead show stations close to that place. For example, if the user types in ‘dame’ then the app should show all the bike stations that are in and around Dame street.

The first stage of doing this is to make use of Google maps autocomplete API and Google Places service. First, the text string is sent off to Google’s autocomplete, this sends back a list of suggestions for places that the text string corresponds to. This app will then take the first suggestion from that list. Next, the app needs to get the latitude and longitude coordinates of that first suggestion. Unfortunately, the autocomplete call only returns autocomplete objects which do not actually contain the coordinates for some reason. Because of this another call needs to be made to get the coordinates of that suggestion.

Once the latitude and longitude have been obtained, the next step is to find the bike stations that are close to this location. This procedure needs to be as fast as possible to avoid any noticeable lag for the user. The StationSearch contains two extra lists to hold bike stations, one of them holds the list of bike stations in order of latitude and the other holds them in order of longitude. Using a modified version of Arrays.binarySearch(), two bike stations are found, one which is closest in terms of latitude and another which is closest in terms of longitude. It is likely that both these stations are actually very far away from the location of interest, they are just close either vertically or horizontally. It is a simple matter to get many of the stations that are close in latitude just by getting the values near the closest station, after some trial and error the 20 closest stations in latitude were found, the same was done for longitude. These two lists were then cross referenced with each other two find the stations that are close in terms of both latitude and longitude. In general this will result in 3 or 4 stations which is perfect for what is needed.

# Weekly Logbook

## Week 3

* The overall design of the app was conceptualized.

## Week 4

* Some basic feasibility testing was done to figure out how best to use the API calls and the map fragment

## Week 5

* Methods for parsing and saving the JSON for the Dublin bikes API call was created
* The main map view was implemented with markers showing the location of each of the bike stations

## Week 6

* The google maps markers were made into custom markers so that text could be displayed on top of them. This procedure was moved to an Async task.
* The basic view for the info panel was set up and its vertical movement functionality was set up.
* An SQLite database was set up to allow for persistent data.
* The search functionality was set up in a way that matched the search string to the station names

## Week 7

* Firebase functionality was set up and with that a login screen and a way of saving journeys and favourites
* The search functionality was updated such that it uses google maps search to find bike stations close to the area being searched for.

## Week 8

* The navigation drawer was set up.
* The extra activities were set up such as the settings activity, the favourites activity and the journeys activity