SOFTWARE REQUIREMENTS SPECIFICATION

for

COMP30830: Group 14 Dublin Bikes

Prepared by: Group 14

April 5, 2022

Contents

1 Introduction

This document contains the software requirements specification for Group 14's project on module COMP30830 regarding the Dublin Bikes Web Application. Group 14 consists of Zaur Gouliev, Victoria Keane, Rhys O' Dowd. Any questions regarding this SRS can be directed at Zaur Gouliev. The GitHub repo for this project can be found on: https://github.com/gouliev/comp30830 It is a brief document outlining the specification and requirements for building the application. Extensive and further detail on the process and more can be found in the accompanying document Dublin Bikes Report.

This document in essence is a high level overview of what is needed to build the app and what functionality is planned to be built. The main purpose of this application is to allow for predicted travel and planning using Dublin Bikes, these bikes allow for users to find a station with available occupancy. It uses a data model to determine the occupancy.

Thus document does not aim to go into detail on the workflow or methodologies, rather so give the reader/user an example of how we envision to build the web application. It is subject to change as we go begin production and any changes or features that may needed to be added or removed will be documented in the Dublin Bikes Report as as aforementioned. This document does however go into a detail overview of the functional requirements, system requirements and the proposed interface for the application.

1.1 Document Conventions, Terminologies and Definitions

The following terminologies are used in this document:

- `Flask`: Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file

- `AWS: EC2`: Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. EC2 offers many options that enable you to build and run virtually any application

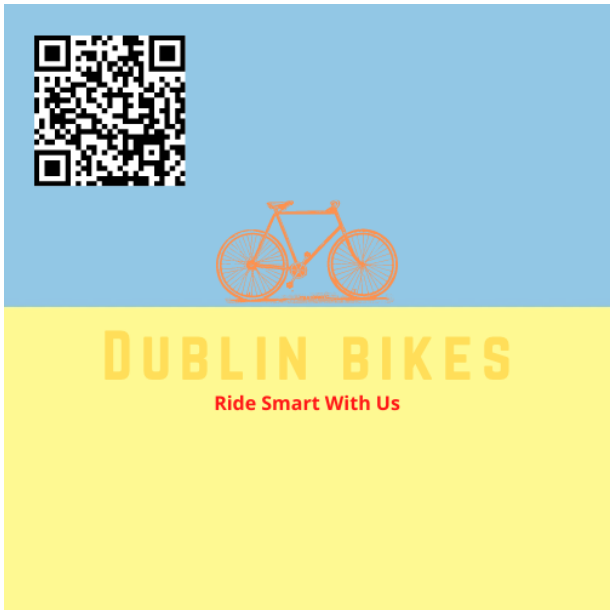

Figure 1.1: Dublin Bikes - Group 14 (QR links to GitHub repo)

- `RDS`: Amazon Relational Database Service (RDS) is a managed SQL database service provided by Amazon Web Services (AWS).

- `APIs`: An API (Application Programming Interface) is a set of functions that allows applications to access data and interact with external software components, operating systems, or microservices. We are using various APIs such as Google Maps, OpenWeather and JCD (Bikes).

- `Git`: Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code. This was used for our project.

- `Scrum`: Scrum is a framework of rules, roles, events, and artifacts used to implement Agile projects. It is an iterative approach, consisting of sprints that typically only last one to four weeks. This approach ensures that your team delivers a version of the product regularly.

- `Geolocation`: Geolocation refers to the use of location technologies such as GPS or IP addresses to identify and track the whereabouts of connected electronic devices.

1.2 Scope

This Dublin Bikes Application is a web application built on flask. A user is a person who wishes to find an available bike to rent on the Dublin Bikes scheme and is looking for either the closest station or a station near the user that has a bike available. The Dublin Bikes is primarily available in the Dublin City Centre region which is where the most amount of stations are concenrated, in and around the radius of the General Post Office (GPO).

The bike works by capturing data that is being pulled from multiple APIs and fed into an RDS database. This database is then analysed to create a predictive model which in turn then creates a way of predicting occupancy rates and also takes into account the weather (also data pulled from an API).

When the user goes onto the website, a geolocation snapshot takes place and it tells them where they are, and based on their current location maps a radius around them to determine the closest station and the occupancy (bikes) in the station.

The user then can proceed to go the station to book the bike and we can consider this a successful usage of the web app. This app, then essentially, becomes a utility web application for predictive planning of using Dublin Bikes.

The web application is not only free to use but it can be accessed by anyone with access to the website. There is no preconditions in terms of system or device and practically any device with access to the web can load the web application. We cannot guarantee the best quality of user experience on mobile devices due to the small screen, but the team (Group 14) do recommend devices larger than 10inches, i.e., iPads, Tablets, Laptops and etc are used to access the web app application. The main reason for this is because the map size is better viewed and a better experience is found. However, this web app can still be used in smaller devices and we have test on Apple and Android phone devices.

2 Overall Description

2.1 Product Perspective

The product is available to any user who has access to the internet on their device. The product does not store any data and no data is taken from the user. The below is an infrastructure design map and a user workflow that was followed to create this application. These were given/taken from the COMP30830 internal Brightspace lectures (10.1).

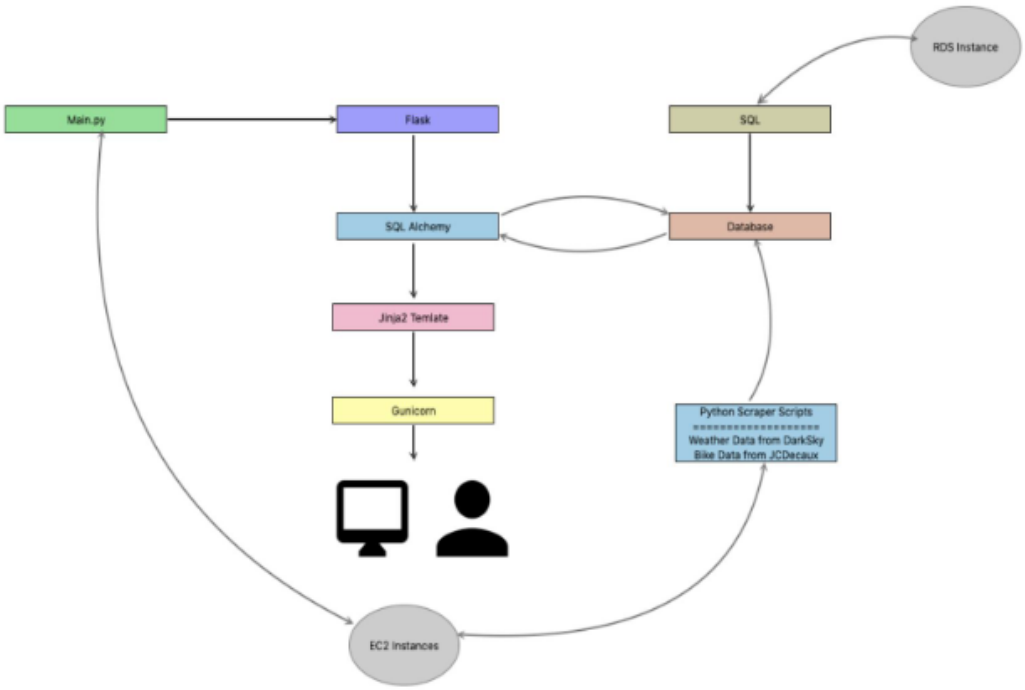

Figure 2.1: User Flow Source: COMP30830 10.1

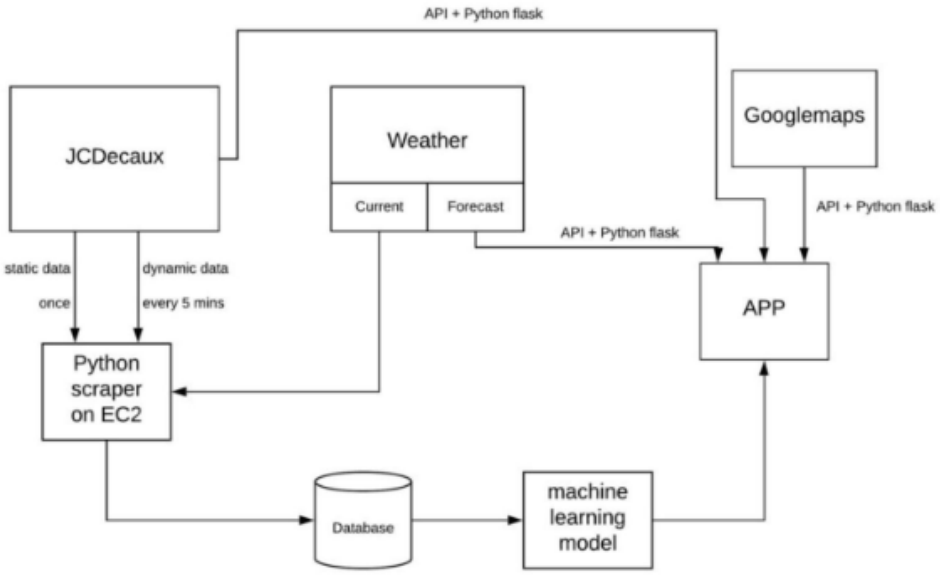

Figure 2.2: Infrastructure of application. Source: COMP30830 10.1

2.2 User Classes and Characteristics

I this application, there is only 2 types of users. These users consist of anyone who is using the web application, example, a prospective individual trying to book a bike in a station and determining the availability. The second and final user is the creators of the application who have the ability to change/add functions. The administrators (Group 14) will not make regular changes and are simply there to ensure the app runs and suffers from no down age or model failures which could result in poor and wrong information being displayed. To avoid this, simply ensuring the data is being pulled correctly everyday, synced and backed up is done on the AWS/EC2 side.

2.3 Product Functions

There are 4 main functions of this application. Further functions can be added, depending on how the sprints go and if we deem anything extra can be implemented. These additional details and info can be found in the main Dublin Bikes Report document accompanying this SRS.

- `Load Website Map`: Upon clicking the link, the website should load and upon loading should display all the correct information. This information should consist

of a map, with pins indicating high occupancy, low occupancy and also search boxes to input current location vs location going to. We have created it so the current location is automatically captured.

- `Select location`: A user should be able to select a location of where they would like to go. Upon inputting this location, the user will be prompted with a list of bikes stations in the radius of where they currently are of where they want to go. This populates the distance and travel time as well as bike occupancy rates using pins to determine severity of busyness.

- `Select station`: A user should be able to select a station, and when selecting the station it should indicate how many bikes are predicted to be there. This should give some graph or probability metric.

- `See weather analytics`: A user should also be able to see a display on the screen of weather and temperature. These can of course affect the possible level of bikes in a station

2.4 Operating Environment

The website will be operate in any mobile or computer operating environment that has GPS capabilities. It is not a prerequisite for the application, however since this app utilizes geo-tracking and location services, this means that a to gain full functionality of the application, GPS should be working, it will need these functions to be on the device as well as permissions enabled.

The GPS itself is managed within the mobile device itself, this is a hardware feature which the underlying operating system has control over and we simply utilize this technology.

2.5 Design

The design feature aims to be simple and minimalist, the approach for design is to have the most accessible and user-friendly approach that can enable users to quickly find and determine where to get a bike. The mockups are as follows and are subject to change as the project goes into sprints.

Figure 2.3: Main Page

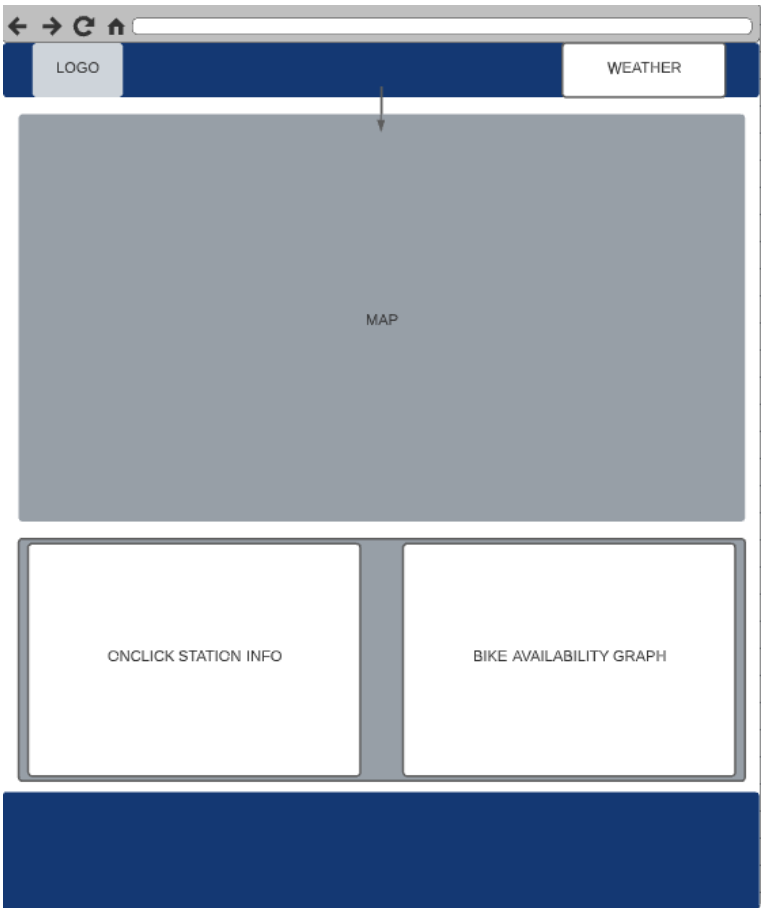

Figure 2.4: Select bike

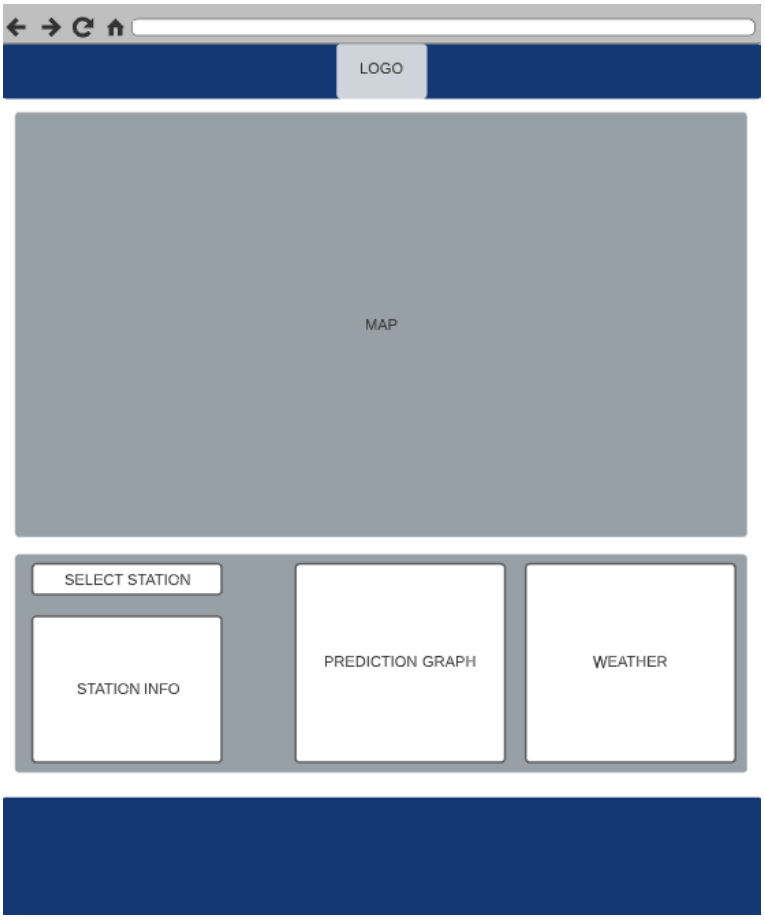

Figure 2.5: Display info

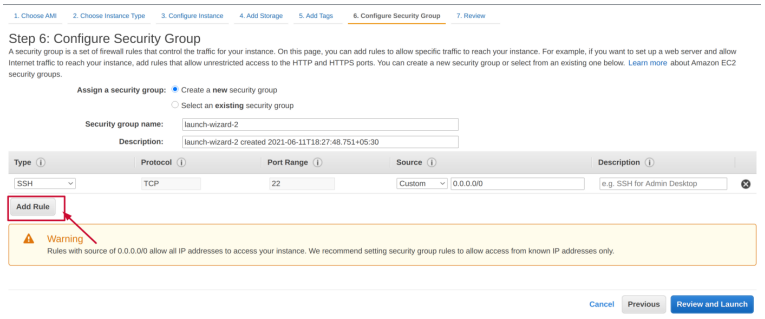

Figure 2.6: Security Groups using EC2

2.6 Security

Security of the website is maintained by ensuring all code is written in a manner where none of the API keys or keys to access anything internal is available. Since we rely on AWS, the security is dependent on AWS's EC2 instance being up to date to protect against DDoS attacks and etc. Since no data is stored on the website in terms of users, this adds a layer of safety for the users.

3 System Features

3.1 Functional Requirements

User Class 1 - The User
ID: FR1
TITLE: Open the web application
DESC: A user should be able to open the web application on their device loading up the map and functions on the map (pins and so on)
RAT: In order load/view the interface

User Class 1 - The User
ID: FR2
TITLE: Select a station
DESC: A user should be able to select a station
RAT: In order to become an active node in the ecosystem

User Class 1 - The User
ID: FR3
TITLE: View availability after selection
DESC: A user should be able to view available bikes and weather information regarding a specific selected station
RAT: In order to make a decision based on the occupancy level

User Class 1 - The User
ID: FR4
TITLE: Display analytics once selected
DESC: A user should be able to view analytical info based on the station, determining the likelihood of availability. This is in the format of graphs **RAT:** In order for better decision making and planning

User Class 1 - The User
ID: FR5
TITLE: Select a destination to travel to
DESC: A user should be able to select a destination they would like to go to and map out the duration of time it takes to there using the map interface.

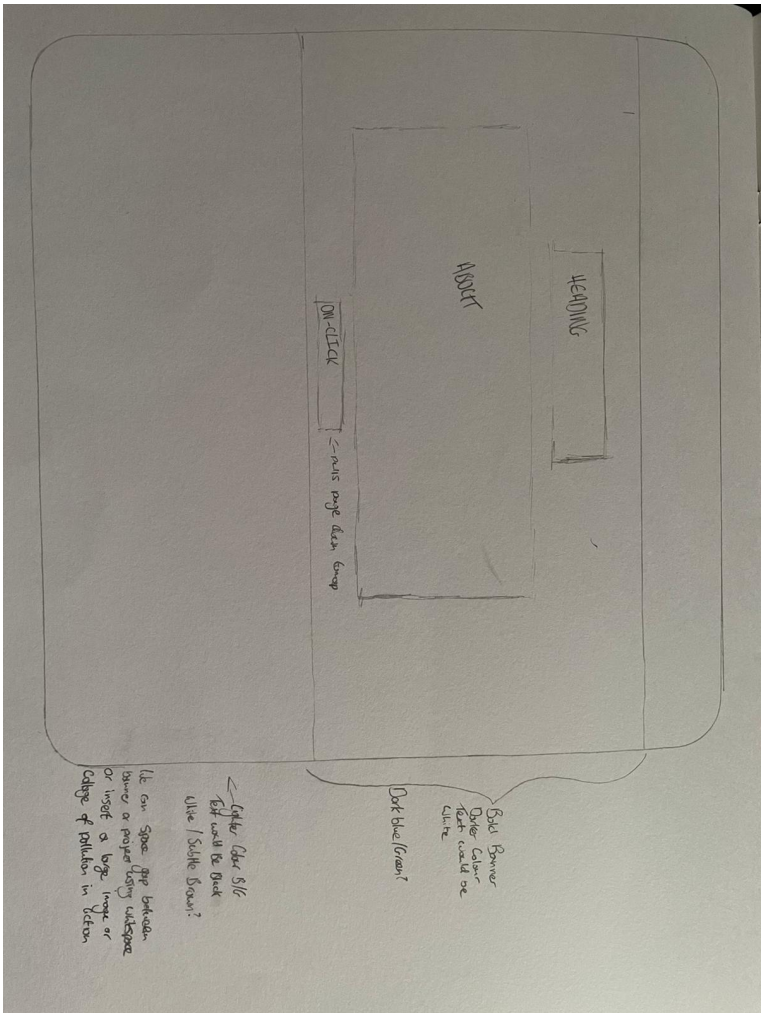

Figure 3.1: Design 1.0

RAT: In order to get a better decision on if they would like to pick up a bike on a station that on the way to the route

3.2 User interfaces

The following user interfaces of were designed prior to building the application. They can be considered as the design of choice, but the final choice is to be made during the second sprint and will be reported in a separate document. The interface is as follows:

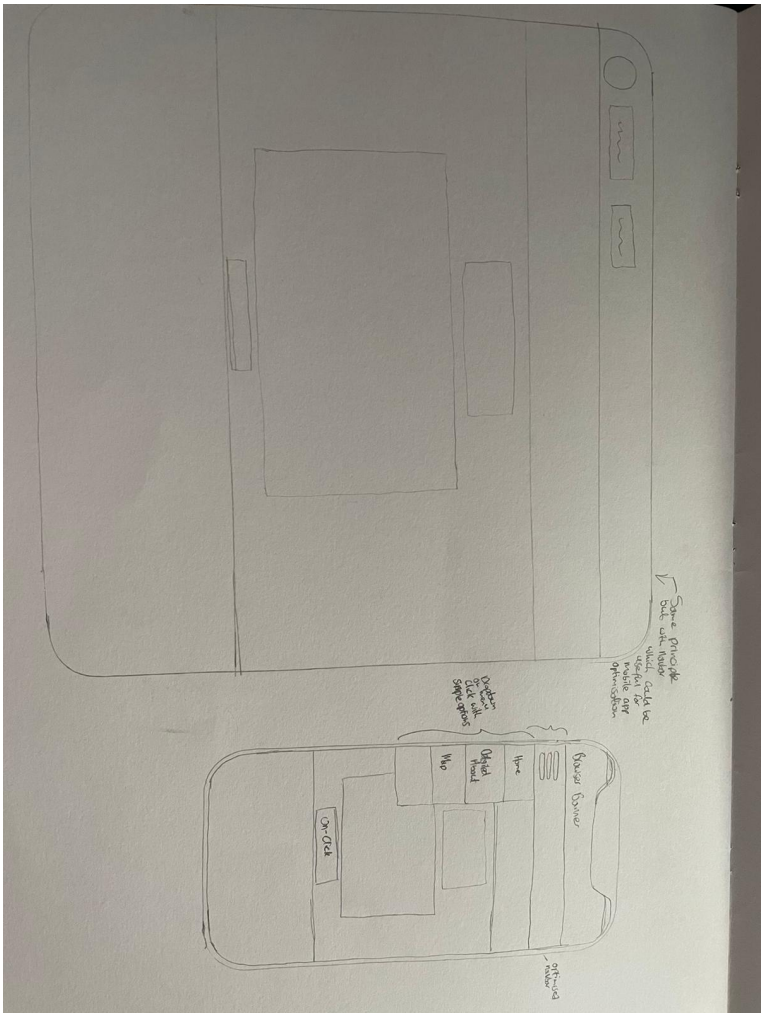

Figure 3.2: Design 1.1

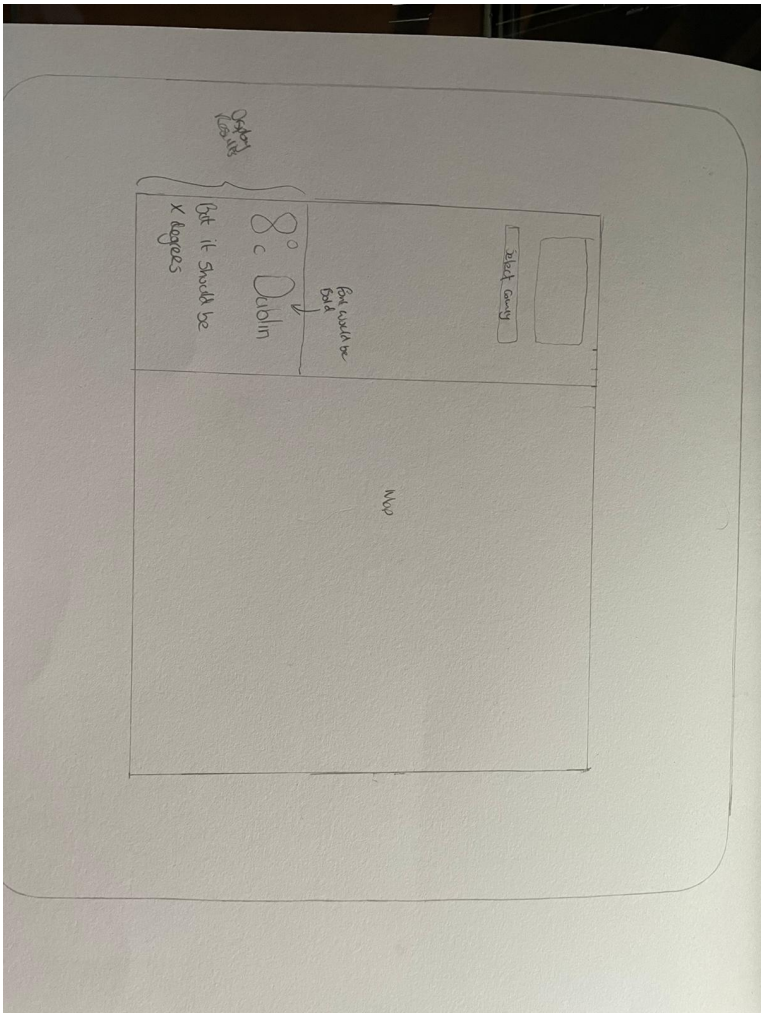

Figure 3.3: Design 1.2

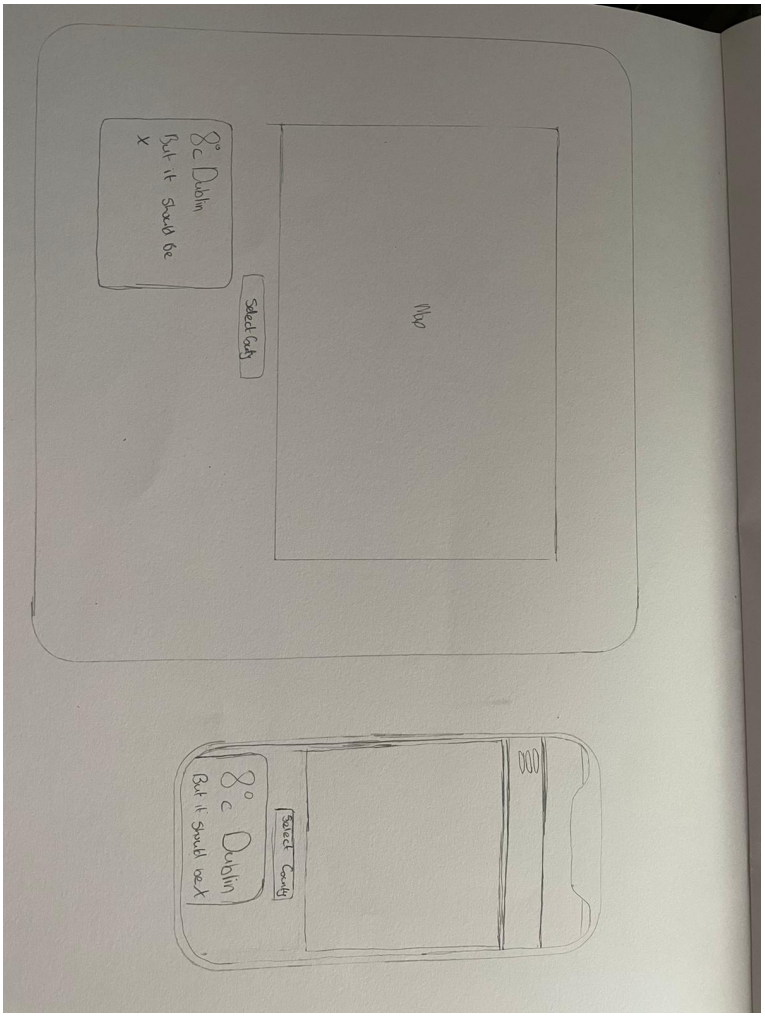

Figure 3.4: Design 1.3

4 References

4.1 References

Dr. Aonghus Lawlor **COMP30830-Software Engineering (Conv)-2021/22 2022** [*UCD Brightspace]*.
`https://brightspace.ucd.ie/d2l/le/content/158044/Home`