COMP30830
Software Engineering

Group 20 :
Conor Kennedy - 16722649
Daniel Howes - 13391711
Mark Kavanagh - 19205515

Product Owner:
Karl Roe

Table of Contents

1. Administration

Github Repository Link:

Repository is private, Aonghus has been added as a collaborator
(https://github.com/Conor-kennedy/Software-Engineering.git)

EC2 Address of live website:

http://52.211.204.133:5000/map

2. Project Overview

2.1 Introduction

The prioritisation and use of public transport is ever increasing in cities and is vital in order to control the increasing levels of congestion in cities like Dublin. Dublin is consistently ranked in the top 20 most congested cities in the world due to the fact the city was not built in mind to support the level of cars that we have in this day and age. Over the last decade, a lot of consideration has been made for more and safer cycle routes in Dublin which presents a service like Dublinbikes to take off and become part of a new, greener culture within the city.

The Dublin bikes scheme is a self-service bike rental system open to any member of the public aged 14 years and over. There are 110 stands dispersed around the city. Individuals can access the bikes by purchasing either a three-day-ticket or annual card.

The goal of this assignment was to develop a web application to display occupancy and weather information for the Dublin Bikes by retrieving data from the JCDecaux API. The project was conducted in a team of three by employing the Scrum methodology. This document gives an overview of the necessary work processes that were required for the completion of this project in terms of a technical aspect and the project management side to it.

2.2 General Overview

Section 1 provides a general outline of the app's features, intended functionality and user audience.

Section 2 describes the technical design of the application in order for it to fulfill the planned requirements.

Section 3 discusses the different parts that relate to the prediction model used to forecast the number of bikes available at a station.

Section 4 talks about the prediction modelling used with some sample tests on the data.

In Section 5, the different design decisions for the application are looked at.

Section 6 outlines all our Sprints throughout the project. The section includes all our meeting logs, stand up logs, product backlogs, burndown charts, feature descriptions, sprint reviews and our issues and resolutions.

Section 7 gives a concise conclusion to the project with several observations made by each team member.

Section 8 shows the contribution of each group member.

2.3 The objectives of the App

- Provide users of the Dublin bikes scheme with up-to-date information on the weather as well as the availability of bikes at different stations.
- The application will also allow users to plan their trips with machine learning algorithms to predict the number of bikes that will be available at a given station for sometime in the future.
- Be a trusted source of information.
- Provide up-to-date information regarding the dynamic data - in particular the number of bike stands and bikes available at a given station.
- Be a useful resource for planning journeys with Dublin Bikes.
- Easy-to-use and user friendly design.
- Be accessible on any browser.

2.4 The intended users

Potential users of the app are anyone in the general public who is using the bike scheme whether they are regular or occasional users. With the location of the stations being relatively central in Dublin, we expect to see students and those working around the city centre to be frequent users of the app so that they can plan their daily journeys.

2.5 App Functionalities

The following is a general description of the main functions that the application will provide:

- Provide the user with up-to-date information on each bike station available. This includes bike availability, number of stands available at each station, time the information was last updated and the station name.
- Display of the current weather information for the Dublin area including temperature and wind speed.
- Function that employs machine learning to make a prediction on the number of bikes that will be available at a given station.
- Display the station information on a map to give the users a visual representation of the locations.
- Show the cycle routes from one station to another which the user preselects.

2.6 Structure of the App

The application features three pages, ‘Home’, ‘Station Information’ and an ‘About’ page. The website loads up on the home page when the user enters the URL to see the app title, navigation bar, journey planner, an interactive map and a prediction section. The ‘Station Information’ page offers the user with another display of the current information available for each station and finally the About page was included to give information about the app’s use cases.

2.7 Features

The following is a general overview of the app's features. A more detailed description is available in the next section.

- Displaying markers on the map that represent the Dublin Bike stations.
- Show information such as bike availability, number of stands available, last updated and whether or not the station is open.
- Show the user cycle routes from one station to another.
- Display occupancy charts for each station.
- Make predictions for the number of bikes available at a future point in time.
- Showing the current weather for Dublin.
- User friendly design.

3. Architecture

This section will give an overview of the system architecture, outlining the different parts and how they work together to make a functional application. Each part can be considered in terms of their overall role and can be broken down in terms of data extraction, data storage, data modeling and application display.

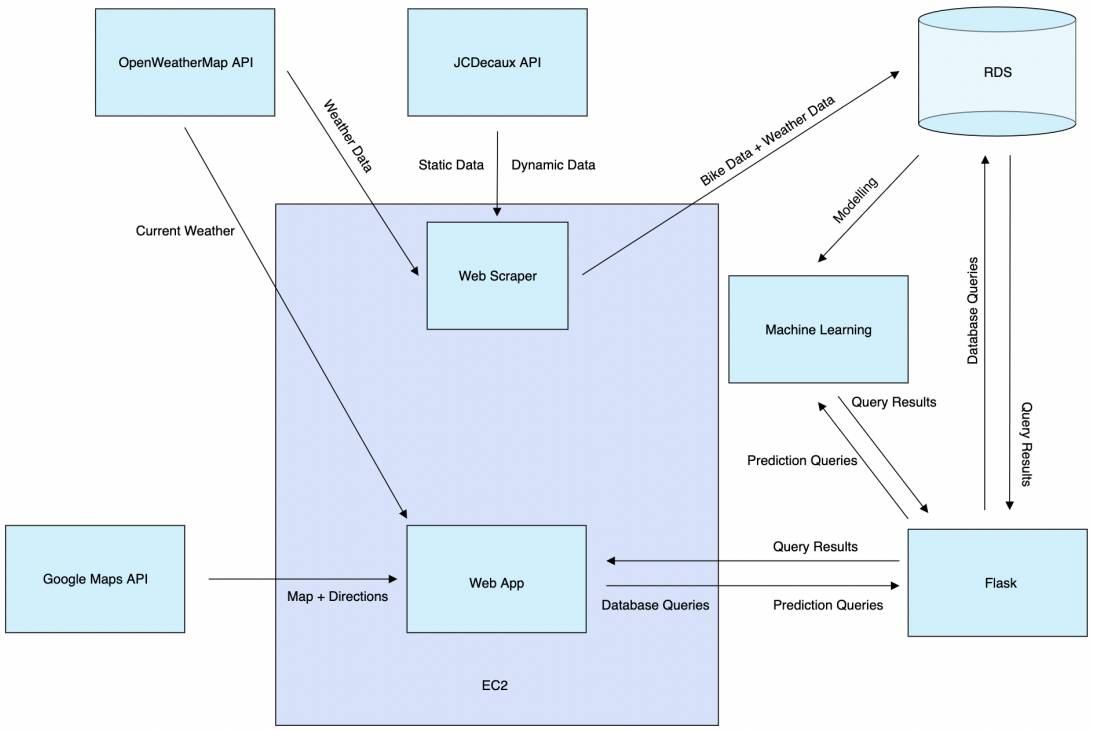
3.1 Explain the overall web application with Diagrams

The web app needs both weather information and information about each bike station. In order to get up to date data regarding both of these details, two web scrapers are used to receive data from the JCDecaux and OpenWeatherMAP APIs which are inserted into three separate tables within a database on an instance of Amazon RDS, where it can be accessed remotely.

Data from this database is displayed in the web app which shows the user the current weather status and the most recent information regarding the availability of bikes at each station. The Python Flask application acts as an intermediary between the HTML files and the database, which sends queries to the database based on user interaction with the webpage. Flask receives the results from these queries which are then passed to the HTML files in JSON format, where they are further manipulated and displayed on screen.

An Amazon EC2 instance runs both web scrapers at regular intervals using the Crontab command. The Bikes_API_Request.py runs every five minutes, receiving data from the JCDecaux API and inserting it into the Bikes table while the Weather_API_Request.py file runs every thirty minutes, receiving data from the OpenWeatherMap API and inserting it into the Weather table. The EC2 instance is also used to host the web application using the App.py Flask file. The OpenWeatherMap API is also used to receive live updates of weather information directly to the application.

In order to make predictions for bike availability, the Index.html file sends input values to the Python Flask application based on the client interaction, which then uses Python pickle and joblib to make a prediction. The results from this prediction are then sent back to the Index.html file where the results are displayed on the web page.



3.2 What technologies did you use?

The web application was developed using the Python Flask framework in conjunction with Jinja 2 templating engine. Using these two tools, HTML files are loaded to the web browser depending on the URL. Dynamic URLs are used in order to pass variables from the HTML files to Flask functions, which are used to dictate the queries sent to the database or open a pickle file for station predictions.

The data received from each API is saved in a MySQL relational database which is stored on the Amazon RDS web service. This database is accessed remotely by the Python Flask file which uses the MySQL Connector driver to connect to the database and make queries. The results of each query are converted to JSON format and passed to the HTML file where Javascript is used to either display or mathematically manipulate the data.

Javascript, in conjunction with HTML and CSS, is used to display or manipulate data on each webpage. Upon loading the main page, a Javascript callback function is called to make a Google Maps API request, which will show a map on the homepage if successful. Javascript is used to perform other dynamic tasks upon loading the webpage including creating map markers at each station location, showing user location, displaying current weather information and creating a functional user interface in the form of dropdown menus and buttons. Javascript functions make it possible for users to interact with the application and generate responses by way of clicking markers and buttons, injecting values into dynamic URLs to obtain a response from Flask.

A number of Javascript APIs are used for added functionality including Chartist.js, Datepicker, Google Maps Javascript API and HTML Geolocation API. Upon clicking a station marker, two charts are created on the webpage showing the average bikes available per day and average bikes available per hour on a specific day, using the Chartist.js API to display this data. Datepicker is used to create a calendar for the date and time selection in the machine learning section. The Google Maps API is possibly the most important API used as it allows users to view a map with station locations and show directions between different stations. The HTML Geolocation API adds a user's location to this map by getting the geolocation from their web browser.

3.3 What is the Functionality of the App?

Loading the homepage will automatically execute a Google Maps API request and query the Station table in order to get the location of each bike station. A second query is also made to the Bikes table, which returns the most recent row of dynamic data for each station. This same query is made when loading the ‘Station Information’ page, which displays the results in a table. The number of available bikes for each station is identified and used to determine the color of each marker shown on the map. Once successfully loaded, clickable markers will appear on the map at the location of each station. Markers are stored in an array in order to access each one individually throughout the code.

A request to the OpenWeatherMap API is also made in order to get the most recent weather information for the Dublin area, which is displayed on the web page.

In addition to station coordinates, user location is displayed on the map in the form of a blue marker, with a circle indicating a radius of 500m (please note for the purpose of the assignment, a set location in the city centre has been set regardless of the user’s location) . A request is made to the HTML Geolocation API in order to get a user’s geocoordinates, which are used to create a Google Maps marker. It is possible for the user to see the station nearest to them by clicking the ‘Show Closest Station’ button, which calculates the

distance between the user's location and each station resulting in a selection of the station marker with the shortest distance Initially the application used location data to display the user's location, however for the testing of this application we thought it would be more practical to hard code coordinates within Dublin City center.

Clicking a marker zooms in on the map and pans to that marker, while a pop up window appears showing the most recent information for this station. This information is received using a separate fetch request to load all data from that selected station in the Bikes table. This request is made instead of using the Bikes table data when the page is initially loaded, since a fresh request at the time of clicking guarantees the most recent data appearing.

The data from this request is also used to calculate the average available bikes per day and the average available bikes per hour of each day. This information is shown in two charts that appear on the side of the webpage when a station marker is clicked. Each chart shows the average amount of bikes available per day and the average amount of bikes available per hour for a selected day. Upon loading the webpage, the selected day is set to the current day but can be changed using the dropdown menu.

In order to get directions between two stations, a user must select a start station and a destination station from the two dropdown menus in the Route Information section. The starting station is automatically set to any station that has had its marker clicked while the destination station must be manually selected from the dropdown list. Clicking the 'Show Route' button will execute a request to the Google Maps Javascript API using the geo coordinates of the selected stations and will display directions between both stations while hiding the rest of the station markers upon successful execution. Clicking this button again will deactivate the directions information and redisplay markers and the selected station pop up window.

3.4 What tests did you perform on the App?

Tests were run for each element throughout the development of this application. The Javascript console was used to print the value of variables passed between Flask and Javascript in order to troubleshoot different parts of code.

3.5 Where does the data come from and how does it flow through the App?

Raw data is taken from both the JCDecaux API and the OpenWeatherMaps API in order to get both station and weather information respectively. After being sorted, this data is stored in the 'dbbikes' database on RDS which consists of three tables, Station, Bikes and Weather.

Station data is routinely scraped from the API with the data being sorted into dynamic and static. Static data includes station address, number of bike stands, station number, banking availability and geo coordinates while the dynamic information consists of available bikes, available bike stands, update time and whether the station is open or closed. After the very first request, static data is discarded since it remains the same for each request while dynamic data is input into the Bikes table. Queries are sent to both tables using the Python Flask application, which also reformats the results into JSON and passes this data to the Javascript within the HTML documents.

Weather data is inserted into the Weather table which was originally going to be implemented in the machine learning algorithm, however this feature was later discarded and so the data in this table is not used. Using Javascript, the weather data displayed on the web page is the result of a direct request to the OpenWeatherMap API.

4. Analytics

4.1 Prediction modelling

The following were the steps carried out in order to provide predictions:

1. Fetching data from the database.

The combination of weather data and bike data provided unsatisfactory results and hence the decision was made to drop the weather data for the prediction modelling. This could have been down to the fact that we did not scrape enough weather data to begin with which is discussed further in the retrospective section.

2. Cleaning & Preparing the data

In order to make use of the data retrieved, it was necessary to first change the columns to their suitable data types and scale certain features. The column that displayed when the station had been last updated was changed from unix timestamp into date-time. From this, the day and hour were extracted to create two extra columns.

3. Training the model

Training the model was carried out by using a linear regression model and Random Forest Regression model.

4. Evaluation

The models' performance was based on the MAE and RMSE indicators.

5. Saving the model and deployment

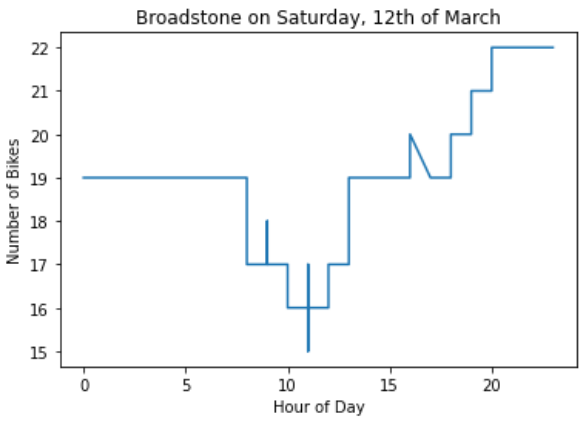
The model was saved to a separate pickle file for each station.

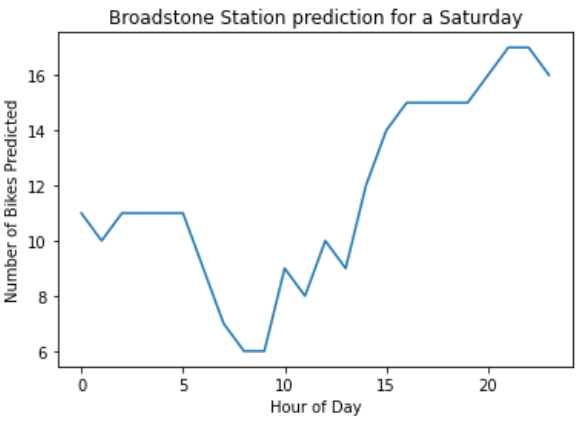
4.2 The types of models used

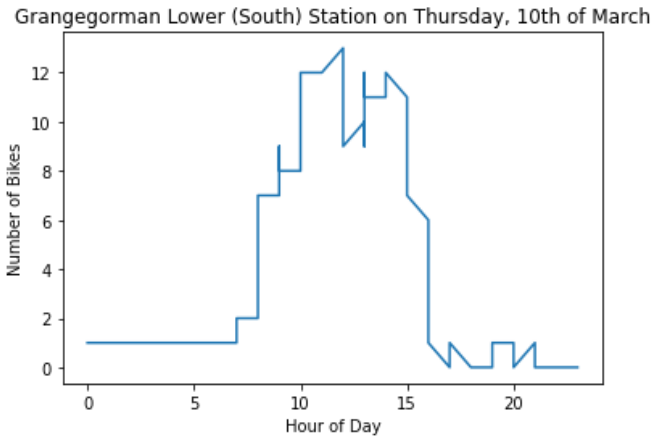
For the predictions, the models assessed were the Linear Regression Model and Random Forest Regressor by comparing their Mean Absolute Error and Root Mean Squared Error. The Random Forest Regressor performed better on both metrics and hence was the model chosen.

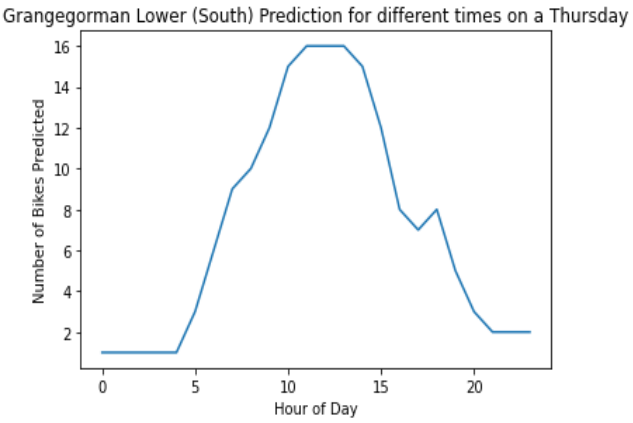
<u>Further testing of the model:</u>

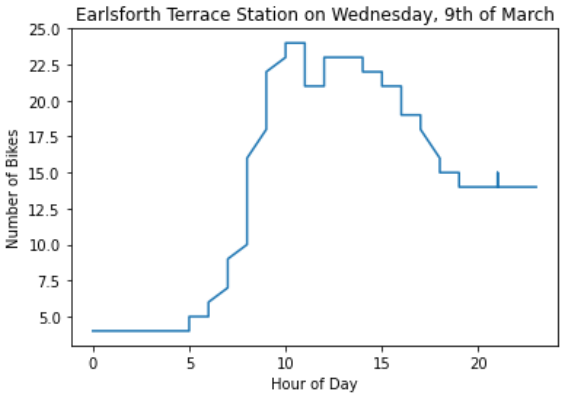
In order to see the reliability of the model, different stations on different days were looked at to view the number of bikes that were available throughout the day and what the model would have predicted for this day. The following are some of the examples:

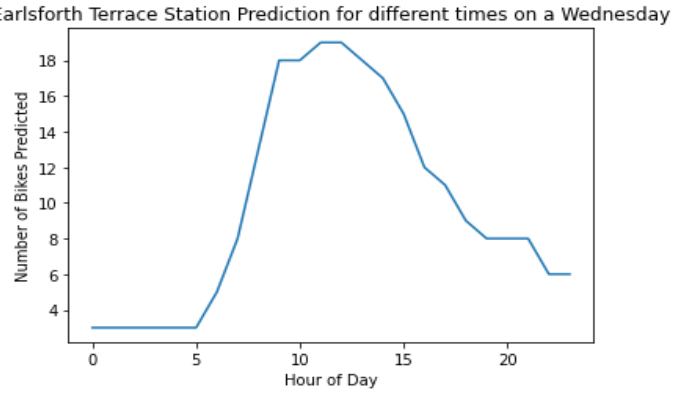












Overall, it appears that the model performs relatively well with a similar graph to each of the actual number of bikes that was available for each of the days. The model however does not appear to be working perfectly due to the underestimates that have been made in bike availability for the Broadstone and Earlsfort Terrace stations which is unsurprising given that the MAE for the model was approximately 7.5. This could be improved by the inclusion of weather data in the future and training the model over larger data sets.

4.3 Correlation in the data

There was no correlation between the weather data and bike data. This could be largely down to the fact that people use the bikes for short distances and thus weather is not a huge factor. The decision was made to only work on bike data for the predictive section.

	Available_Bikes	Available_Stands	hour_of_day	day_of_week
Available_Bikes	1.000000	-0.687031	-0.014788	0.009022
Available_Stands	-0.687031	1.000000	0.011464	-0.004490
hour_of_day	-0.014788	0.011464	1.000000	-0.000485
day_of_week	0.009022	-0.004490	-0.000485	1.000000

4.4 The user implementing the model

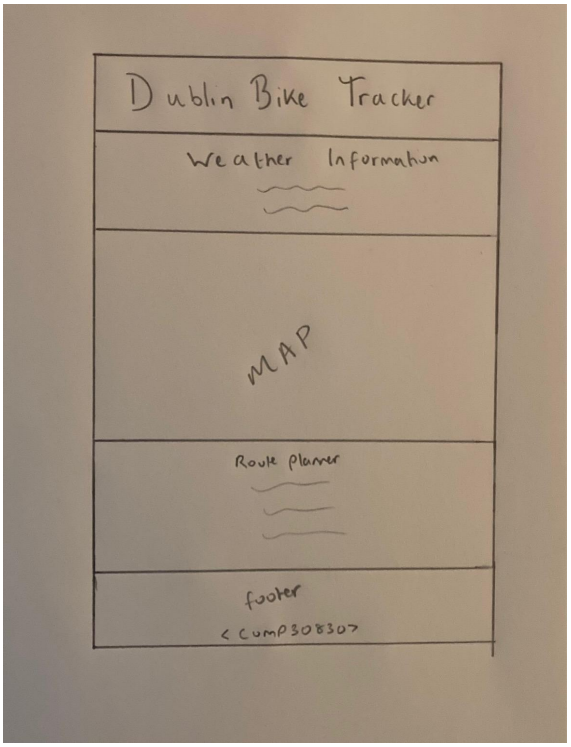
The user selects the station they want to check for followed by a time and date in the future which has been implemented by JavaScript using DatePicker. The day and hour is extracted from this and used as the input data for the station's associated pickle file which then generates the prediction.

5. Design

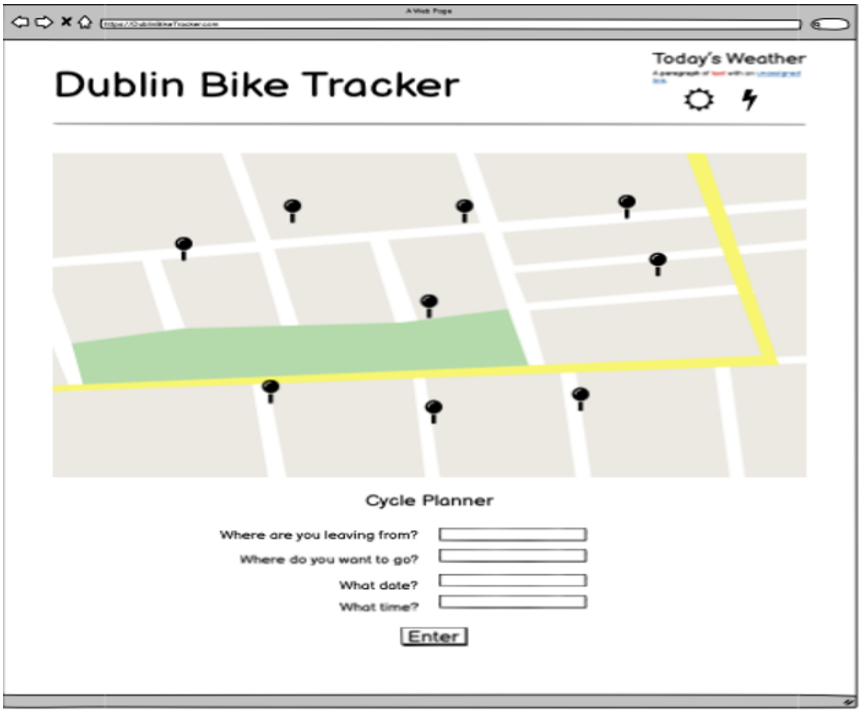
5.1 Wireframes

The following shows the changes in design over the course of the project:

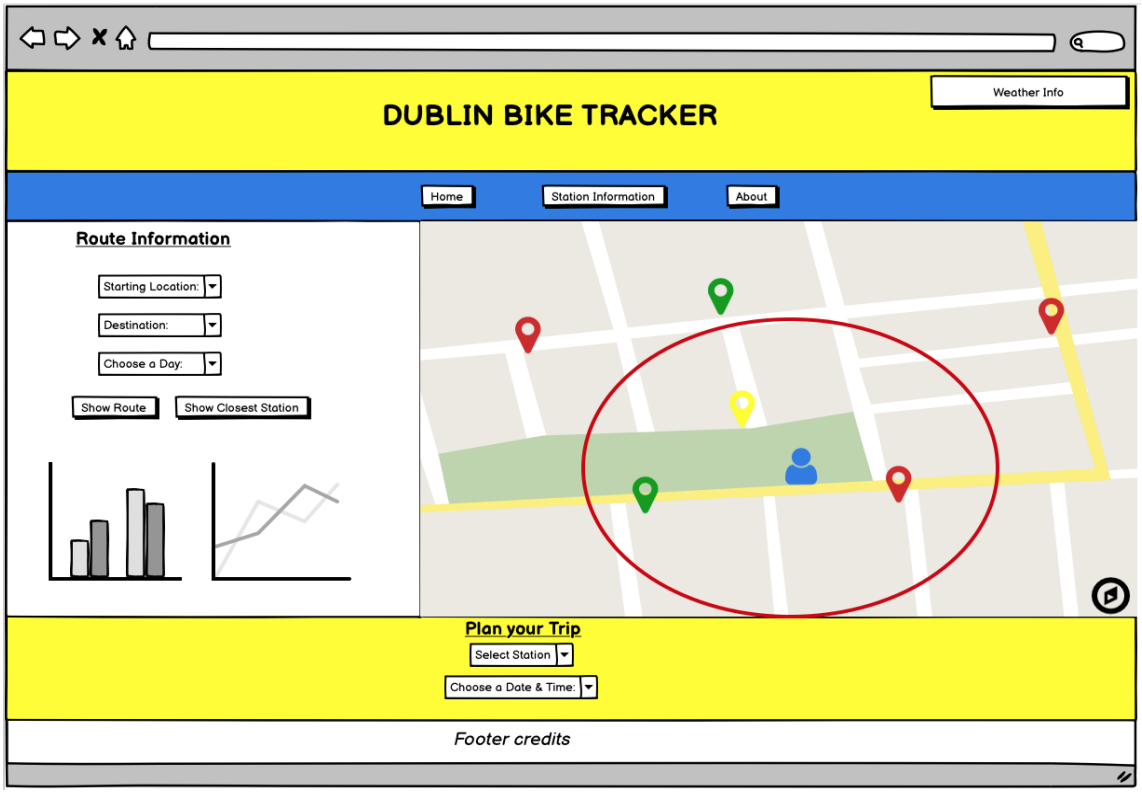
Wireframe no. 1:



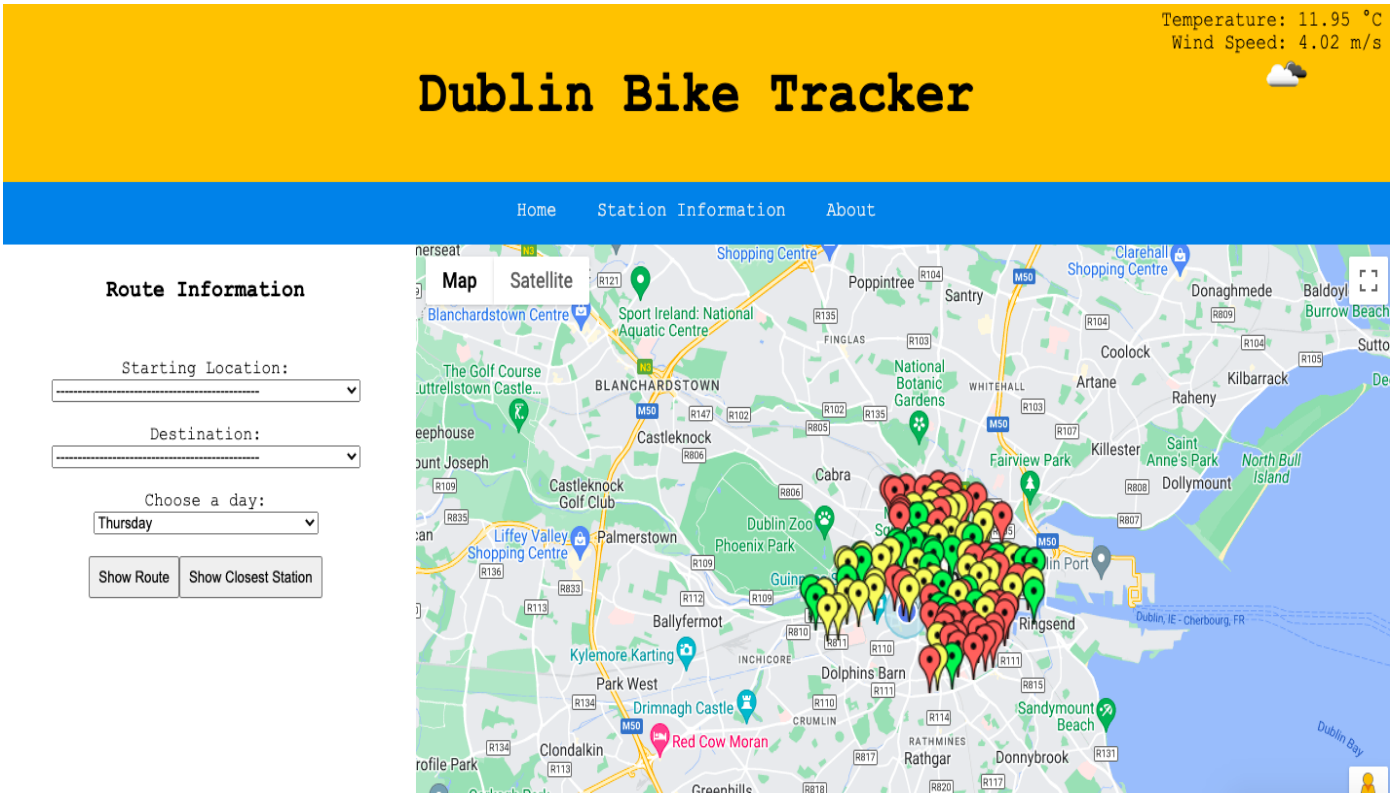
Wireframe no. 2:



Wireframe no. 3:



Final Output:



As we progressed through the project, exploring different features to be implemented, we decided to refine our initial mock up. The product owner was consulted on a number of occasions in reference to design decisions. The following is a list of the changes that were made and reasoning:

- The first wireframe paid too much attention to the weather and took away from the application's main purpose regarding the bike data. The decision was made to move the weather to the top right hand corner of the screen.
- After completing the design of the second wireframe, it was obvious that the map took up too much space on the screen in a way that looked improper. Even with resizing the map div, there was no apparent solution due to the extensive white space that appeared on either side. Wireframe 3 shows our solution to this by moving the route planner section to the left-hand side of the page layout. We felt this was the best design decision to move forward with. The second wireframe was missing a navigation bar which was also added in.

5.2 Discussing how does the app works

The central focus of this application is the map. Upon loading the webpage, the user will see a map centred around Dublin City center with colored markers representing each bike station. The color of a marker indicates the number of available bikes at that location at the time of loading the web page, red for less than ten bikes, yellow for between ten and twenty bikes and green for over twenty bikes. Each marker is clickable, where a pop up window appears, showing the name of the station, the number of available bikes, number of available stations, amount of time since the last update and operational status.

After clicking a station marker, two charts appear in the 'Route Information' section, showing statistics for the selected station. The bar chart displays the average number of available bikes per day while the line chart shows the average number of bikes available per hour for a selected day. A day can be selected using the dropdown menu and the current day is automatically selected at the time of loading the web page. After selecting a new day, the user must click on the station marker again for the charts to reload with the correct information for that day.

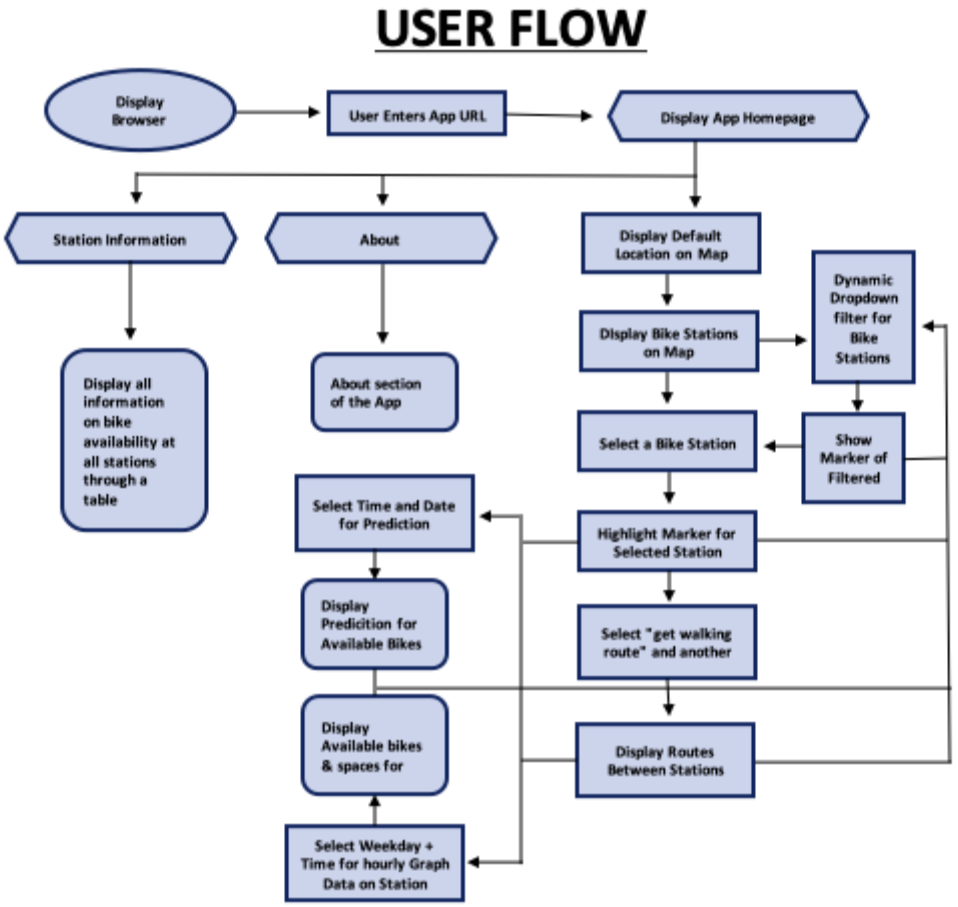
The Route Information section also has an option to display the directions between two different stations. Two dropdown menus are used to select both the start and destination stations, with the start station dropdown automatically selecting the clicked station. Once two stations are chosen, the user must click the 'Show Route' button for directions to appear on the map. Clicking on the button again will remove the directions and return markers to the map.

Clicking the 'Show Closest Station' will select the nearest station in relation to the user's location and show the pop up window displaying the most recent information for that station.

The 'Plan Your Trip' section allows users to estimate the number of bikes available at a station at a given time and date. A station is selected using a dropdown menu while a calendar is displayed where a time and date can be chosen. Clicking 'Apply' will result in an estimate being shown for the amount of available bikes at this station during this time.

A navigation bar at the top of each page allows users to easily navigate between each page. Clicking on a navigation button redirects the user to either the homepage, the 'Station Information' page or the 'About' page.

5.3 The user flow of the application



User Flow Diagram

5.4 The features relating to interactivity, real time, predictions etc

Clicking a station marker will query the database for the most recent data for that station. New data is inserted into the database every five minutes meaning that the user will see data that is no more than five minutes old at the time of clicking a marker. While the application guarantees the most recent station data is available to the user, it is dependent on how often the JCDecaux API updates, which can be at irregular intervals for each station. For clarity, the amount of time that has passed since a station's last update is also listed in the pop up window.

Upon loading the map, markers are designated a color based on the number of available bikes at that station. These colors do not change until the page is reloaded and so the colors may not be representative of the actual number of available bikes after a period of time without refreshing the page.
The app also displays the latest weather information at the top right corner of the page. This information is received from the OpenWeatherMap API which is updated hourly, meaning the weather data displayed is no more than one hour old at the time of loading the web page.

5.5 How have the features evolved?

Initially the application consisted of a map using clickable station markers to display information for each station. After discussing with our product owner, we decided to add more depth to this feature by

implementing colors for individual stations based on bike availability. We thought that using a visual aid when broadly viewing the map would make it easier for the user to identify viable stations without having to click each marker. In addition to a color, the number of available bikes was added as a label for that station, further aiding the user. However, after another discussion with our product owner, we decided it was an unnecessary feature and looked unprofessional (see Appendix C). The colors were deemed sufficient.

The next iteration of the app introduced a dropdown menu that allowed users to scroll through stations and choose one to view information about. This worked in tandem with each marker, selecting a station from the dropdown menu would click that station's marker and show information about that station. Clicking a marker would also change the dropdown selection to that station. This feature was dropped in favour of adding the directions information and a button to select the station nearest the user. The 'Show Closest Station' has a similar functionality to the first dropdown menu in that once it is pressed, it will click a marker of the desired station (in this case the closest).

The 'Station Information' page was added in order to provide a broad overview of every station in one table. From this page, the user has the ability to search for a station using the search bar. A refresh button was added at a later stage since the page only displays the most recent data for each station at the time of loading.

The first implementation of displaying a user's location on the map had used actual geo coordinates from the HTML Geolocation API, however after discussing with our product owner, we decided to use a simulated location instead. The reasoning behind this is that since a lot of the development of this app was done outside of Dublin, a user location in Dublin City center would give a more realistic indication of how the app worked.

5.6 How do the features interact with each other?

The application is centered around a map and a lot of functionality largely depends on a user's interaction with station markers. While each marker is clickable, with its own individual popup window, only one marker can be selected at a time. This yields a less cluttered map and less work for the user if they wish to compare different stations. Clicking on a station will also focus the map around that location, making it easier for the user to navigate around the map, while the 'Start Station' dropdown will automatically select the clicked station, should the user wish to view directions from that station. Station information is provided once a marker is clicked and charts are generated based on the selected marker.

In order for the 'Show Closest Station' function to work, a user's location needs to be taken into account. The distance between the user and each station is calculated and the nearest station's marker is clicked once the button is pressed. Clicking another marker will deselect the nearest station and show information for the new station.

It is not possible to click markers when directions information is shown on the map. Once two stations are selected from the dropdown menus and the 'Show Route' button is pressed, all markers and popup windows will be hidden and directions between both stations will be displayed. Once the 'Show Route' button is clicked again, the directions will disappear and the markers will be reshown, with the previously selected marker showing station information.

Charts are displayed depending on both the selected station and the selected day. The day can be selected using the dropdown menu and graphs will be displayed based on the day chosen at the time of clicking a

marker. This means that in order to view the chart for a different day at the same station, the station marker must be re-clicked after the dropdown value is changed.

6. Process

6.1 Sprint Planning Meetings

The first Tuesday of every Sprint cycle, we met as a group to discuss and plan what we need to do for that Sprint. The following is a list of resources used over the course of the 10 weeks:

- Tasks were outlined on our team Trello board which was useful to see whether or not a task was yet to be completed for each sprint (see Appendix A).
- Slack was used for communication amongst team members (see Appendix B). Again, good for collaboration amongst team members and seeing if any group members needed help with any material.
- A google drive was set up to upload the project management documents for team members to work on weekly.
- A Github repository was created and all members of the team were made collaborators.

All official meetings with the Product Owner were inputted into our Google calendars. Stand-up meetings were more informal and decided on a week to week basis depending on our personal schedules.

6.2 Sprints

6.2.1 Sprint 1

Scrum-Master for Sprint 1
Conor Kennedy

Sprint 1 Goal

- Start scraping data from the weather and Dublin Bikes API and store in AWS Database.

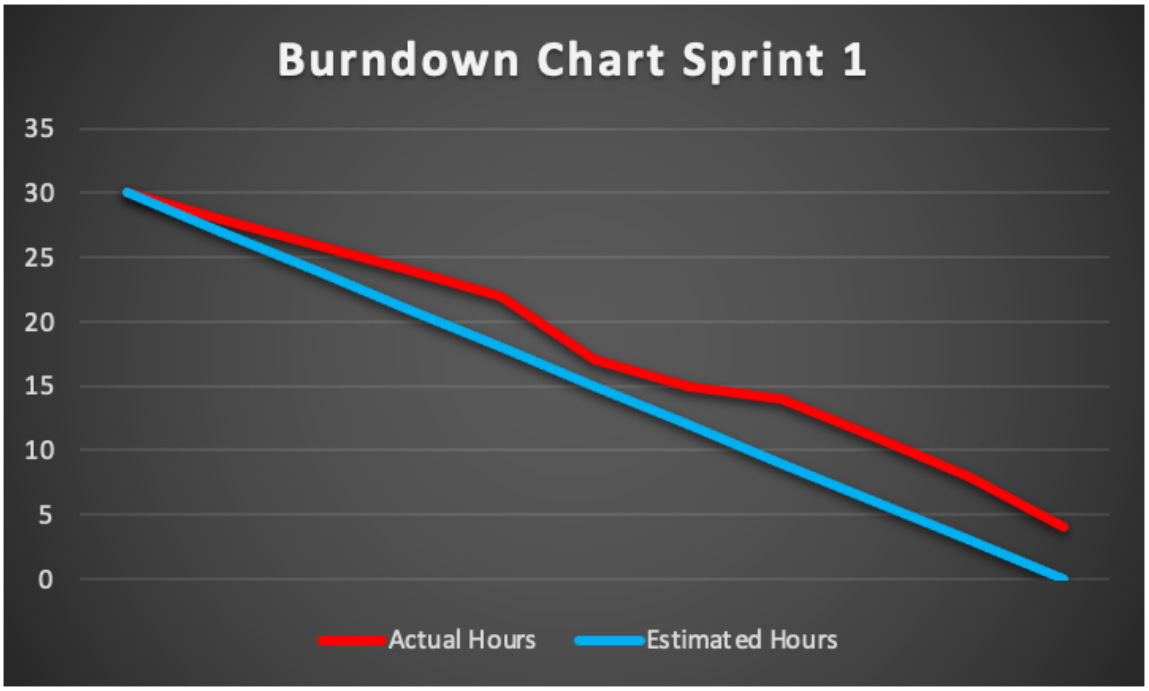
Sprint 1 Planning/Backlog

- Set up Project Management/Communication
- Set up API's to scrape information from Dublin Bikes and Weather Apps
- Filter what information is needed and what is not
- Create static and dynamic tables for the information
- Store the information in the AWS Database
- Mock-Up of what the Software will look like

Task Breakup Sprint 1
All members collaborated throughout each task.

Burndown Chart for Sprint 1 (Hours per Day)

Tasks	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Set up Project Management/Communication	2				3					
Code API's to scrape info from Dublin Bikes and Weather Apps		2	2	2						
Filter through what information is needed and what is not					2					
Create Static and Dynamic Tables for the information						2				
Store the information in an AWS Database								3	3	4
Mock-Up of the Software							1			



Meetings with the Product Owner:

10/02/22 09.00-11.00 Meeting 1 – B106 B108 CSI, UCD. All Members in Attendance.
We as a team presented:

- Can see the data on jupyter notebooks from Dublin Bikes

- Data has been parsed to filter out what is not needed
- Whatsapp and Google drive set up for communication & project management
- Goal to get AWS Database working

Karl's comments:

- Make a private Github repository where we are all collaborators
- Set up Trello & Slack for project management & communication. Google Drive is fine.
- Start doing stand ups each week on the project
- Create Backlog with user stories
- Look into location of user on a map compared to nearby bike stations
- Special feature-Integrating a telegram bot that provides the user with notifications on specific bike stations at specific time stamps

Progress for next week:

- Mock-Up of software
- Look into API's for weather apps (sensor derived)
- Continue trying to link Data with Database

<u>17/02/2022 09.00-11.00 Meeting 2 – B106 B108 CSI, UCD. All Members in Attendance.</u>

We as a team presented:

- The scraper is injecting data into the database successfully from Dublin Bikes and Weather App
- The database is working and collecting the data

Karl's Comments:

- Is there any/much redundant data? Data has been filtered to what we need
- Any data we can change into specific tables? We have static and dynamic tables sorted
- Make sure our static data is only updated once a day
- Don't accumulate data, override it.
- Store data as JSON
- User Interface-Cycle planner priority over weather information. Give more space to it
- Weather API-Don't need to capture forecast data, only real time. Store in separate table

Progress for next week:

- Complete sprint review and retrospective.
- Make sure databases are routinely updating and refreshing.
- Look into Crontab guru to code a schedule-in-for collecting data every N minutes.

<u>Team Stand Ups:</u>

<u>08/02/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.</u>

- Ran through the project scope and its entirety
- Made short term plan on how to approach it
- Start by getting the Dublin Bikes API working on Jupyter Notebooks and go from there

<u>11/02/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.</u>

- Re-think our project management and communication
- Set up Slack and Trello
- Look at API for weather apps
- Start constructing a mock up for the end product

<u>15/02/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.</u>

- Continue work on trying to connect Scraper to the AWS database
- Finish the Mock Up of the software

<u>18/02/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.</u>

- Complete Sprint 1 review and Retrospective
- Upload what we have done so far to github

6.2.2 Sprint 2

Scrum-Master for Sprint 2
Daniel Howes

Sprint 2 Goal

- Minimal viable product
- Get google maps working on the app with pin drops for bike stations with extra information inserted on the clickable pins

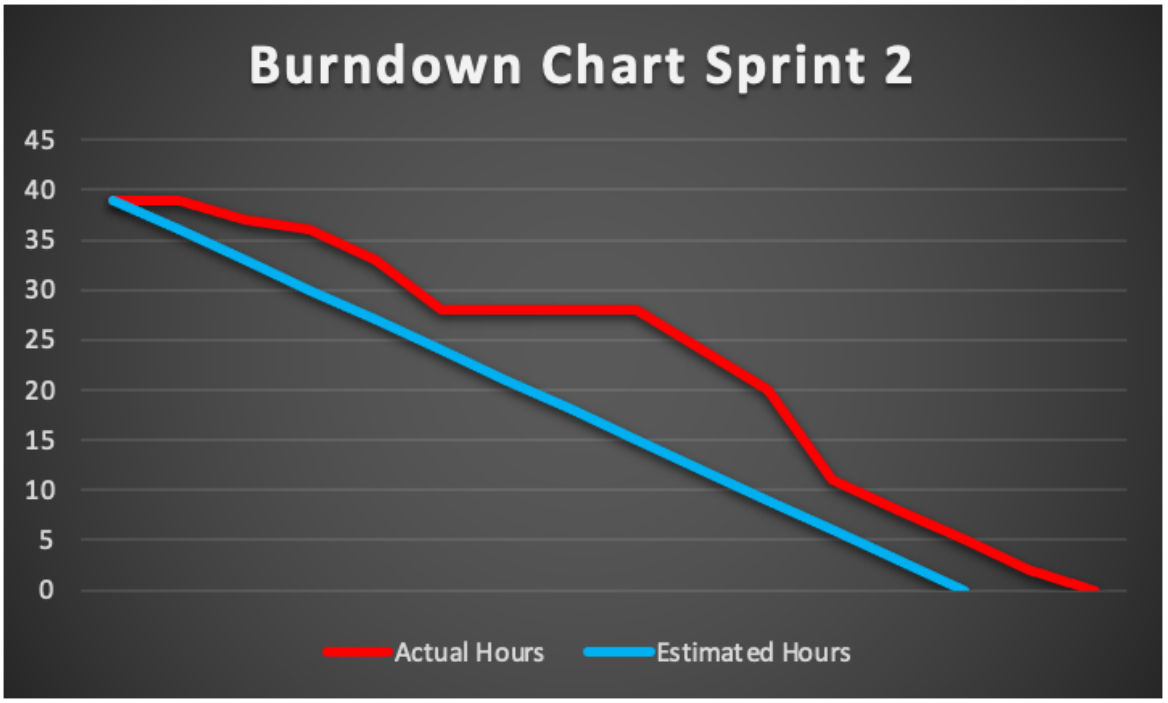
Sprint 2 Planning/Backlog

- Make sure we are scraping data from our API's and injecting into our database
- Remove any redundant data
- Make sure we are overriding data not accumulating it.
- Using Flask, create queries for the database to inject its results into the HTML file for the Dublin bikes google maps.
- Using JavaScript create a html page that shows all our Dublin bike station data and allows the user to interact with it with on click functionality for the station markers.

Task Breakup Sprint 2
All members collaborated throughout each task.

Burndown Chart for Sprint 2 (Hours per Day)

Tasks	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Scraping Data from API's + Injecting into Database		2	1												
Remove Redundant Data				1											
Data being Overridden not Accumulating					1										
Flask - Queries for Database to inject results into HTML				2	2				3	4	4				
JavaScript - Show Dublin Bike Stations on Map					2						5	3	3		
JavaScript - Users able to Interact on Map - On Click Functionality														3	2



Meetings with Product Owner:

24/02/22 09.00-11.00 Meeting 1 – B106 B108 CSI, UCD. All Members in Attendance.

We as a team presented:

- Double checked what we did for sprint 1 was correct
- Presented the project management side of the project
- Looked for guidance on what to approach next

Karl's comments:

- Are we utilising/displaying the data that's being scraped. We are utilising the data just not displaying it yet.

Progress for next week:

- Start developing features to be used
- Look into google maps using static data to make pin drops
    - On click functionality for the markers (live data inserted)
    - Dropdown menus for locations

03/03/2022 09.00-11.00 Meeting 2 – B106 B108 CSI, UCD. All Members in Attendance.

We as a team presented:

- No real progress or updates to present as workload for the course in other subjects was very demanding this week.

Karl's Comments:

- Acknowledges the overload in work and proposes that Sprint 2 is extended to the 10th of March to accommodate our workload.

Progress for next week:

- Make sure our API Scrapers are running all day
- Flask front end. Where are we on that? Working on it, progress has been made but not near completion yet.
- Get google maps working on the app with markers for stations displaying relevant real-time information.
- If we have the time, generate some charts using Chartist.

Team Stand Ups:

25/02/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Mapped out tasks to do for this sprint.
- Make sure Data is scraping and Injecting into database

28/02/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Data is scraping 24/7
- Make sure our data is overriding and not accumulating
- Look at flask

04/03/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Meeting Cancelled due to other coursework priorities

07/03/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Get flask working
- Start on java script to get bike stations positioned on maps

11/03/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Add user interactivity to bike stations on map highlighting station information

6.2.3 Sprint 3

Scrum-Master for Sprint 3

Mark Kavanagh

Sprint 3 Goal

- Machine Learning Models
  i) Show predictions of the best bike stations to get a bike from at certain times in the day
  ii) Show predictions for the best bike stations to get a bike from during specific weather forecasts.
- User friendly interface.
  i) Dropdown menu for user to search stations and see availability

Sprint 3 Planning/Backlog

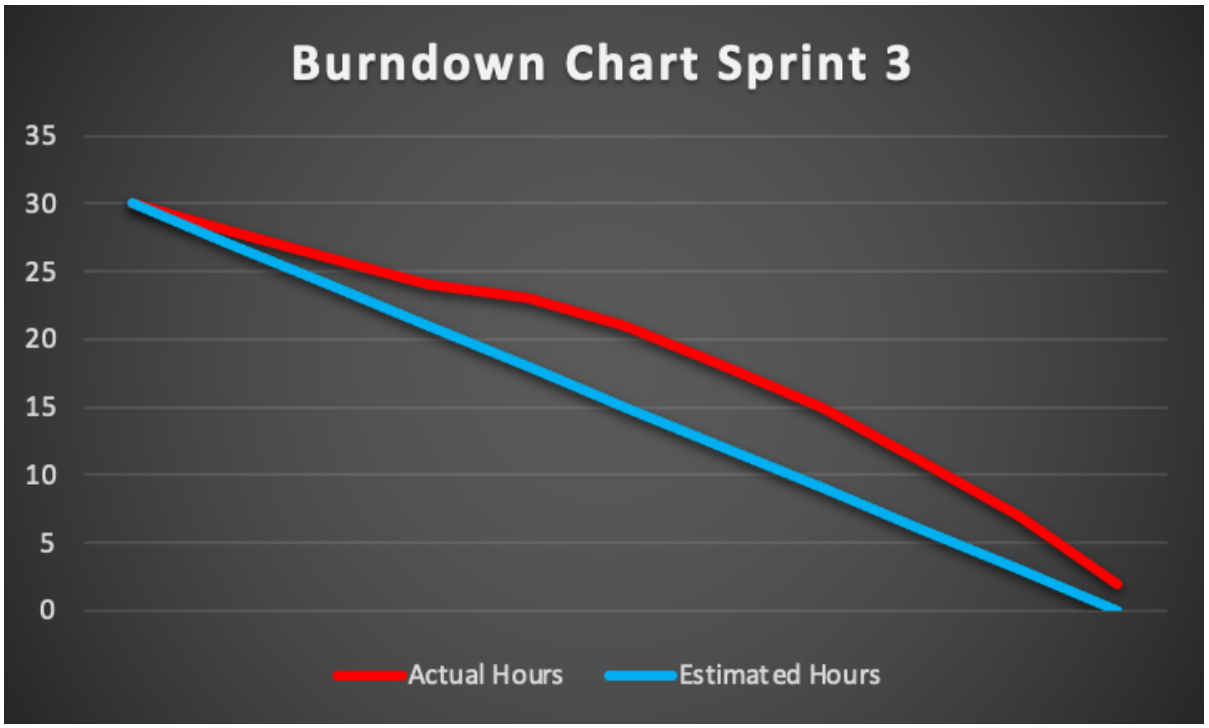
- Charts and graphs using Chartist displaying bike station averages over specific days and hours.
- Linear Regression Model
- Create a dynamic select dropdown from the database to show station names on specific days of the week and at specific hours.

Task Breakup Sprint 3

All members collaborated throughout each task.

Burndown Chart for Sprint 3 (Hours per Day)

Tasks	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Google Maps Markers to show bike stations	*2*	*2*								*2*
Dynamic Select Dropdown menu			*2*	*1*				*3*	*3*	
Rechange how static data is passed					*2*	*3*	*3*			
Generate graphs using Chartist								*1*	*1*	*3*
Machine Learning										



Meetings with Product Owner:

24/03/22 09.00-11.00 Meeting 5 – B106 B108 CSI, UCD. All Members in Attendance.

We as a team presented:

What the google maps looks like

Karl's comments:

- Have we used chartist.js to graph the average bikes available on certain hours/days?
  Have not got to this part yet. Asked for some guidance around going about it.
- Suggested was to use a separate query for the on-click functionality to do this
- Do we have a user input to select specific stations?

We need to work on a dynamic select dropdown from the database to show station names on specific days of the week and at specific hours.

- For our machine learning model, basic linear regression will suffice as a prediction for bikes at specific stations.

Have a good solid robust way to process the data and generate an output for it

Jot down how you progressed with this process

- Take the emphasis away from the map being the focal point.

It needs to supplement a user's location on the map with what stations are nearby.

Progress for next week:

- Create some charts and graphs using Chartist displaying bike station averages day/hour.
- Dynamic select dropdown for user
- Work on our basic linear regression
- Refined mock up of what our user interface should look like.

<u>31/03/2022 09.00-11.00 Meeting 6 – B106 B108 CSI, UCD. All Members in Attendance.</u>

We as a team presented:

- We presented our updated version of the map
    - On-Click functionality of stations
    - Dropdown Menu to choose specific stations
        - ➢ Presents graphs of bikes available and updates regularly
            - Per hour on the current day
            - Days per week
            - Can change to check previous days availability
- Our charts/live data/on-click functionality/dropdown functionality are all working.

Karl's Comments:

- Markers on map need a bit of work to present itself better
- Where are we at with Machine Learning? No time last week to do it, started it yesterday.
- One basic linear regression will suffice, minimal work needed on the accuracy, more emphasis on describing our process of how we went about the regression
- Write this process up in the sprint section
- When writing up sprints, decide between group members on the major components of the app and describe the processes of how you went about them.

Progress for next week:

- Machine Learning Prediction Model done
- Traffic light system or colour coordination to describe markers on the map
- Simulated geo location on map
- Structure around the webpage. CSS, header, fonts etc
- Add weather information. A lot of it done, just need to add to webpage
- Planning a route to a specific station if time allows it.

<u>Team Stand Ups:</u>

<u>18/03/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.</u>

- Get google maps on the app showing all markers for bike stations mostly done just a few tweaks needed
- API's constantly scraping and gathering info constantly throughout the day

<u>23/04/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.</u>

- Work on dropdown menu

- Start working on graphs using chartist having difficulty with this, can't get charts working properly using the static data in MySql database, data may need to be changed around

25/03/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Get dropdown menu working in next day or two
- Have to change how static data is passed from python flask file to the html file. Data changed to a global variable instead of a promise function so can be used anywhere in the script for different functions.

01/04/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Working on Machine Learning Model

6.2.4 Sprint 4

Scrum-Master for Sprint 4

Dan Howes

Sprint 4 Goal

- Refine User Friendly Interface
    - Dynamic dropdown menu functionality
    - Colour coded markers for bike stations
    - Geo Location on map
    - Refine CSS of the app
    - Add weather information
    - Route Planner
- Machine Learning Model
- Complete Group Write up of Project

Sprint 4 Planning/Backlog

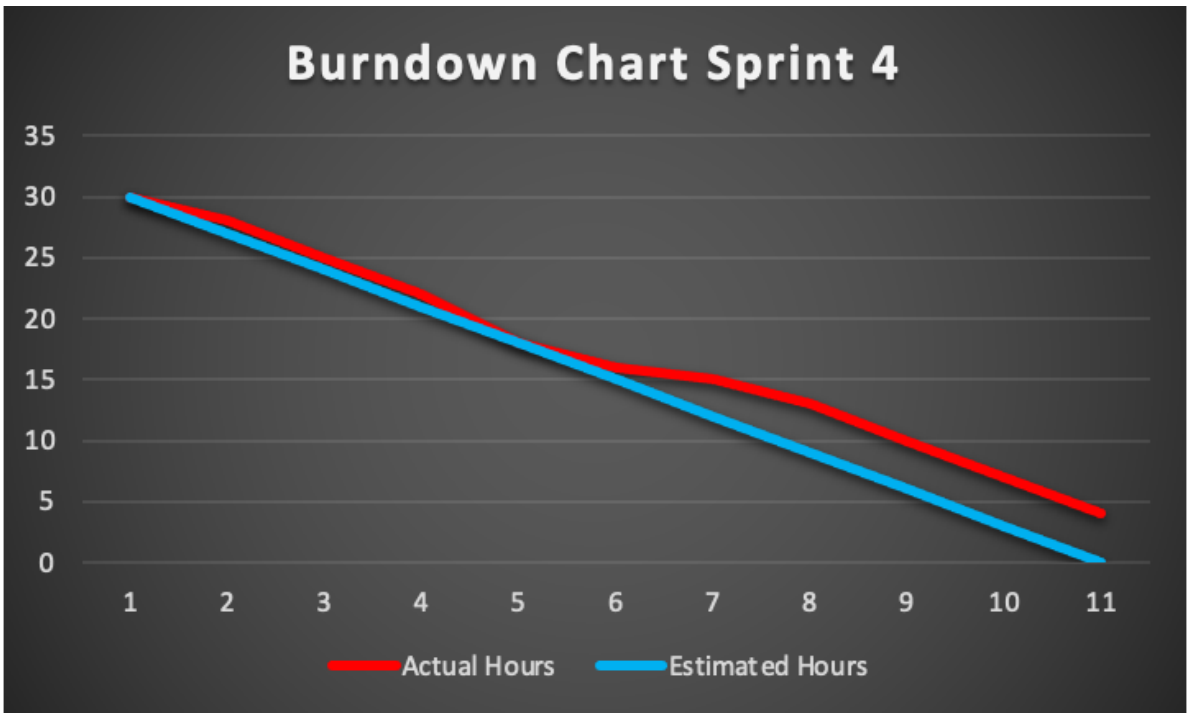
- Create a dynamic drop down menu that allows user to search for specific bike stations and plan a route
- Create a dynamic drop down menu that allows a user to see a chart presenting information on availability of bikes at a certain station, on a specific day at designated hours.
- Add a geolocation of a user onto the map
- Structure around CSS of app.
- Colour Coding markers for bike stations to show different ranges of availability of bikes.

Task Breakup Sprint 4

All members collaborated throughout each task.

Burndown Chart for Sprint 4

Tasks	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Dynamic Select Dropdown menu functionality	*2*	*1*								
Colour coded markers for bike station		*2*								
Refine CSS of the app					*2*					
Geo Location on Map				*2*						
Add weather Information			*1*							
Route Planner			*2*	*2*						
Machine Learning Model						*3*				
Complete Group Write up							*2*	*3*	*3*	*3*



Meetings with Product Owner:

07/04/22 09.00-11.00 Meeting 7 – B106 B108 CSI, UCD. All Members in Attendance.

We as a team presented:

- Presented Karl with the latest features from last week. Happy where we are overall.

Karl's comments:

- Where are we at with Machine Learning? We are nearly there, a few little kinks to work out.
    - Karl aiding and guiding us in this process at this meeting.

Progress for next week:

- CSS clean up for the app
- Prediction Model integrated and deployed on EC2
- Trello has all tasks by team in it, just needs to be tidied up.
- Write up the group report.
- *If you can* (*not necessary*)-Have the app suggest stations nearby to the user?

14/04/2022 09.00-11.00 Meeting 8 – B106 B108 CSI, UCD. All Members in Attendance.

We as a team presented:

- Presented the finished product to Karl

Karl's Comments:

- How are you getting on with the report? Going fine, will be done on time.
- Comment on functions in your code to show what you're doing.
- Have you deployed the product yet? Asked guidance on how to deploy EC2.
- App is very good, functional and user friendly. We have had 100% attendance at all meetings, we have been very receptive to feedback and guidance and very thorough on the project management side throughout. This app is a very good foundation to build upon going forward.

Team Stand Ups:

05/04/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Work on dropdown functionality a lot more user friendly
- Colour code the markers on the map – Colours inserted. Works but needs bit of refining to look better
- Add weather information to app

08/04/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Generate a route planner for user route planner implemented, uses most efficient route to get there, no time to distance feature in it though. Not enough time
- Add Geolocation to map -User is seen on map with a radius around the location
- Tidy up the CSS on the app

11/0342022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- Finish off the Machine Learning prediction model for the app
- Clean up Trello for the write up

13/04/2022 09.00-09.30 – B106 B108 CSI, UCD. All Members in Attendance.

- How are we doing on the group project write up?

6.3 Sprint Reviews

6.3.1 Sprint 1 Review

Accomplished:

- Created Trello & Slack for Project Management & Communication.
    - Need to utilise Slack for communication better. Still using whatsapp.
- Backlog and User Stories completed
- Initial Mock Up of the software drawn up
- Dublin Bikes and Weather App API are scraping information
- The relevant data is being stored in An AWS Database
- The Database is constantly being updated and collecting data
- V1 of the software has been uploaded to GitHub

Sprint 1 Retrospective:

What is working?

- API Scraper
- Data being stored in Database

What is not working?

- No client side up and running yet
- Using Flask.
    - Still trying to code it correctly

6.3.2 Sprint 2 Review

Accomplished:

- Data scraping and injecting into database successfully
- Data is successfully overriding and updating instead of accumulating
- Flask is injecting our results into our HTML page
- Map with bike stations is working, routinely updating and presenting on-click functionality

Sprint 2 Retrospective:

What is working?

- Google Maps with on-click functionality for information on bike stations

What is not working?

- App is not user-friendly yet
- Chartist is not running yet for station data

6.3.3 Sprint 3 Review

Accomplished:

- User Interface is working a lot better
    - Dropdown menu where user can click on specific stations and shows them on the map
    - Can click on the stations in the dropdown to show charts for bike availability
    - Charts show hours per day and days per week availability
    - Users can also choose previous days of availability
- On-click functionality for markers also updates regularly so data is constantly up to date.

Sprint 3 Retrospective:

What is working?

- Google Maps with on-click functionality for information on bike stations
- Google Maps with dropdown functionality for information on bike stations

What is not working?

- Machine Learning Prediction Model
- User geo Location on map
- Colour system for map markers

6.3.4 Sprint 4 Review

Accomplished:

- Bike Stations are colour coded
- Dropdown functionality is a lot neater and user friendly
- User geolocation working on map
- Route Planner for user to get to and from different bike stations
- Weather information is shown on app
- Machine learning for bike predictions at all stations

Sprint 4 Retrospective:

What is working?

- User Interface is working a lot better and more user friendly
- Route Planner for user to get to and from different bike stations
- Machine learning for bike predictions at all stations

What is not working?

- Machine Learning for predicting bike availability with the weather.
- Route Planner – No estimated time of arrival

6.4 Setbacks

Setbacks	**Solutions**
Initial Github Repository wouldn't work properly for all team members when they were pushing/pulling files.	The initial repository was deleted and remade another one.
Team couldn’t connect to the AWS Database in the early weeks of the project to scrape the necessary bike and weather data.	Figured out the UCD Wireless network was the issue. Disconnected from UCD Wireless solely using eduroam whilst on campus. There were no issues since.
Multiple instances where our code in JavaScript, Flask, Notebooks etc came back with errors	Researching solutions around coding problems using online resources & collaboration with other team members.

As a group, we didn't encounter any major setbacks throughout the project thankfully. Most were solved on the spot with a bit of research or receiving some guidance from our project owner.

7. Retrospective / Future Work

What can be improved?

- Adding the number of bikes available to the marker in a way that looks professional.
- Implement an improved prediction model to assess the impact of the weather conditions on bike availability at specific stations.
- Have a feature where a user can click on a weather icon which presents the weather forecast for the coming week.
- When the user is viewing directions between two different stations, a problem occurs if the user clicks 'Show Closest Station'. When this happens, the pop up window for the nearest station shows on screen, despite markers being hidden, and will not close automatically after the user hides directions information and clicks another marker.
- Queries to the database take a number of seconds to return results, which is the reason both the homepage and the 'Station Information' page take a long time to load. Finding a way to optimize database request and response times would be a priority if we had more time and resources.

Thoughts and observations

Overall the team is very happy with what we have managed to put together over the last 8 - 10 weeks. The application works well and includes all features that we envisioned when starting out. It was interesting to see the work involved in making a website, particularly the back-end development which was something none of the team members had done before.

What worked well? What does the team want to continue to use?

- Collaborating via Trello, Slack etc was useful throughout the duration of the project.
- The weekly meetings with the product owner assessing the progress that had been made and giving any guidance when needed.
- Overall cohesion of the team - Conor and Daniel leading the coding side of the project and Mark leading the project management side of things.

8. Sign off

8.1 Overall Contributions by team

As mentioned above, Dan and Conor led in the coding aspects of this project while Mark led the project management side of it. Conor and Dan would provide input and help with the project management aspects and Mark would provide input and help with the coding aspects from time to time. It is acknowledged by all members that this was a cohesive team effort to complete this project.

Dan contributed **35%** of the overall project.

Conor contributed **35%** of the overall project.

Mark contributed **30%** of the overall project.

Signed

Dan Howes	*Conor Kennedy*	*Mark Kavanagh*
Dan Howes	Conor Kennedy	Mark Kavanagh

9. Appendices

Appendix A

Image 1. Sprint 1 Plan and Outcome

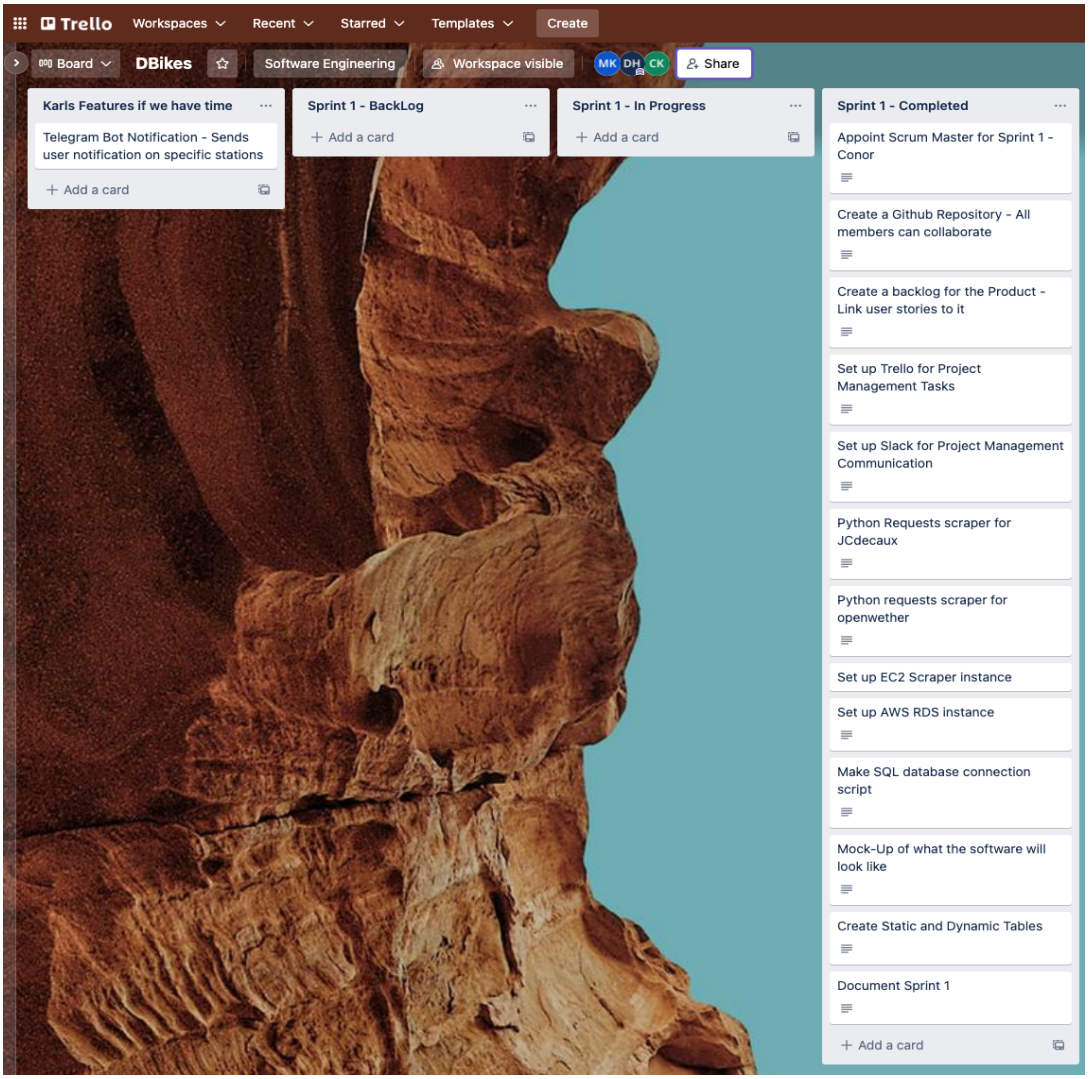


Image 2. Sprint 2 Plan and Outcome

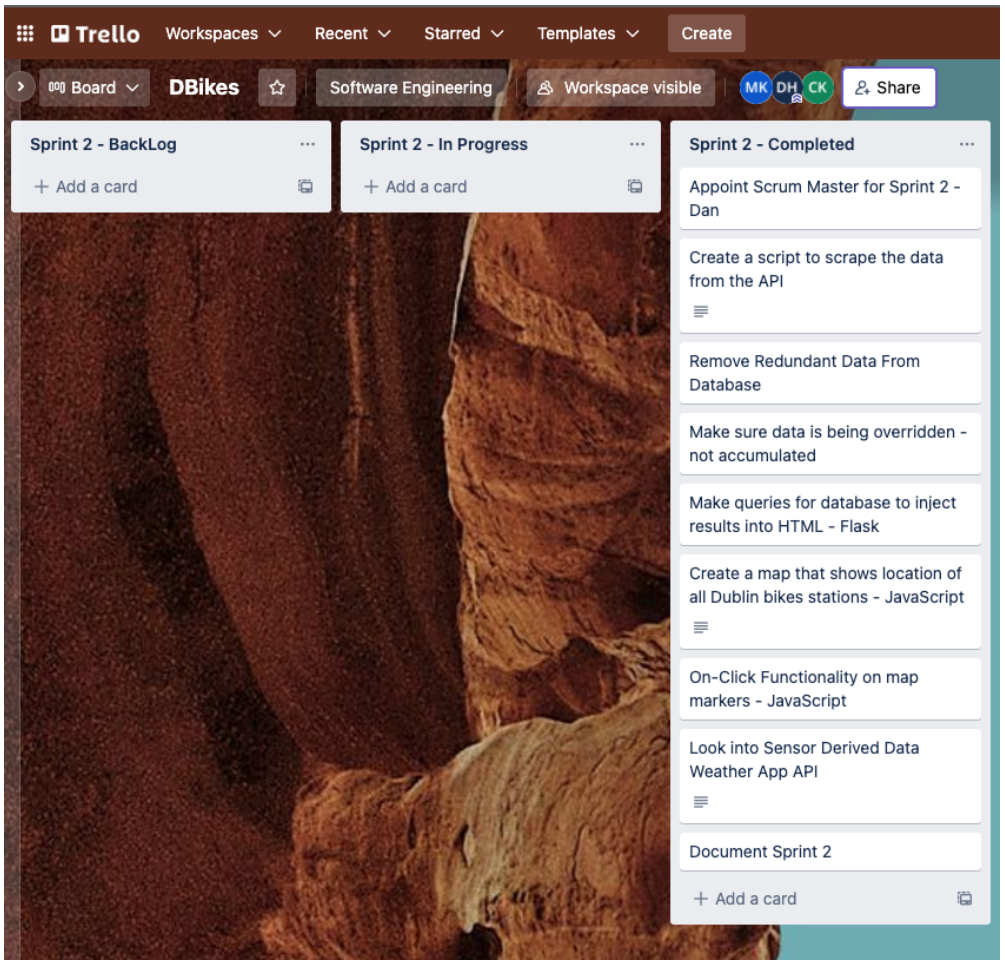


Image 3. Sprint 3 Plan and Outcome

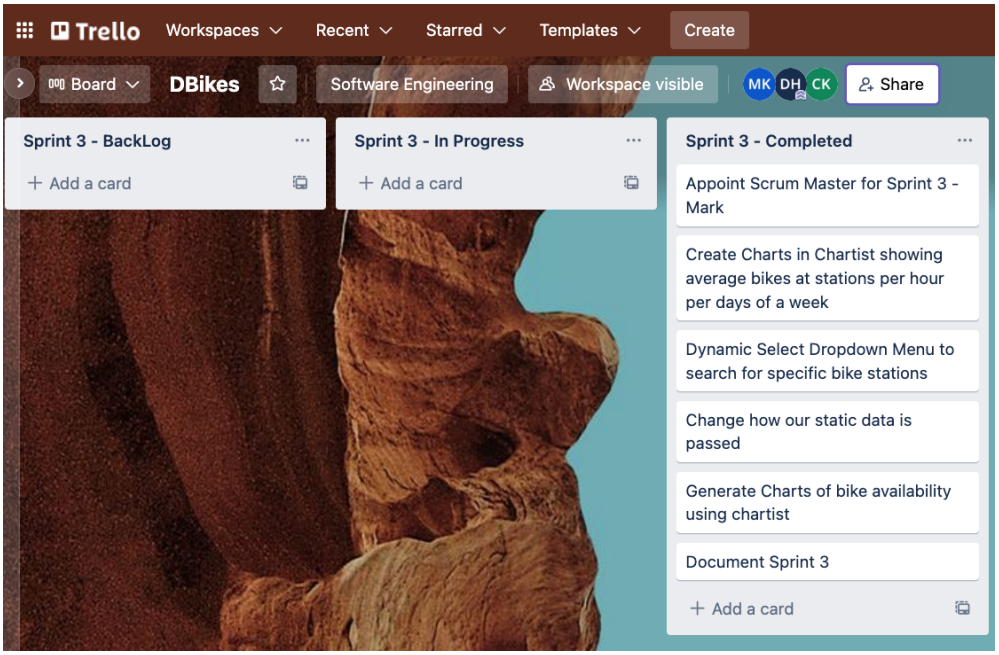
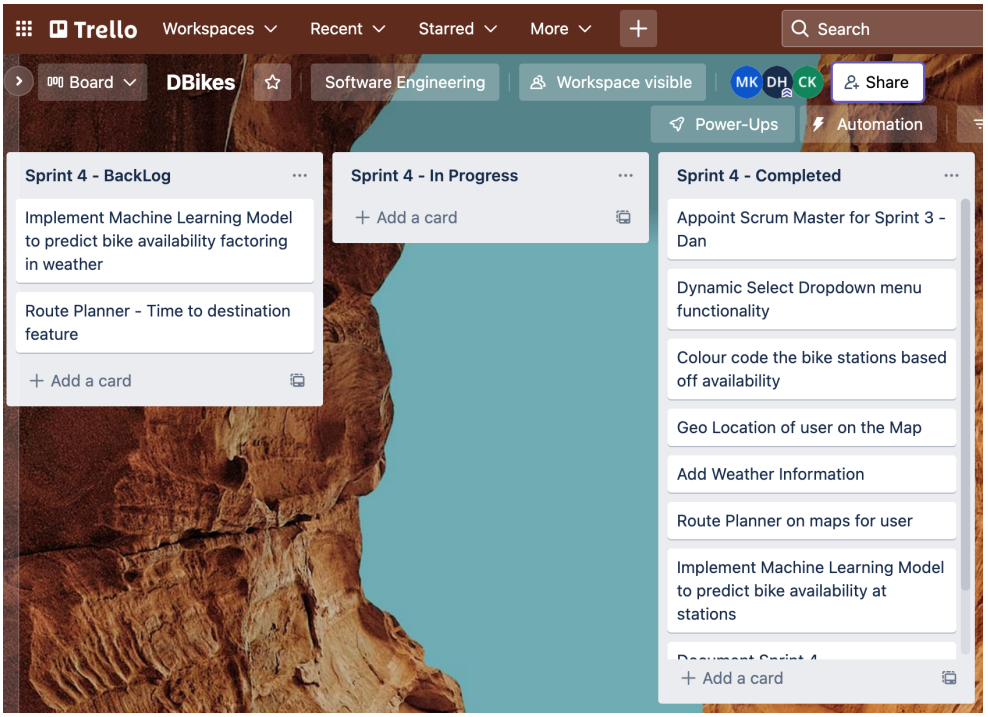


Image 4. Sprint 4 Plan and Outcome



Appendix B

Messages in the Slack group

daniel howes 7:48 PM
Uploaded V2 to GitHub with charts representing station data and the start of a user interface

Sunday, April 3rd

Conor Kennedy 4:22 PM
Uploaded About & Table page to GitHub. Still working on the modelling

Tuesday, April 5th

daniel howes 11:40 AM
Uploaded V3 to GitHub with coloured markers based on number of bikes upon loading the page and the option to show directions between two stations. Gonna try get geolocation done over the next two days.

Wednesday, April 6th

Mark Kavanagh 10:53 AM
Before next weeks meeting we need to :
-Have a retrospective of the first sprint
-Have our databases routinely updating and refreshing
-Weather API is working
->Dont capture forecast, only capture real time data and store in own tables
If we can have :
Everything up and running on our EC2 instance

Use Crontab guru for running our python code what ever set minutes we agree on

Monday, February 21st

daniel howes 4:00 PM
Just pushed the first versions of the code to GitHub. There are two files, Bikes_API_Request.py and Weather_API_Request.py, each one scraping from their respective API and inserting the relevant data to the database. Some small amendments will need to be made to the code:

- While loops need to be deleted so the files can be run with Crontab in the terminal.
- Bikes API Request code needs to be changed so that it only writes to the 'Station' database only once.
- Commented out code should be deleted for readability (Maybe put in a separate 'query' file with a list of commands that print results for database queries? Not necessary but might be nice to have.)

Appendix C