

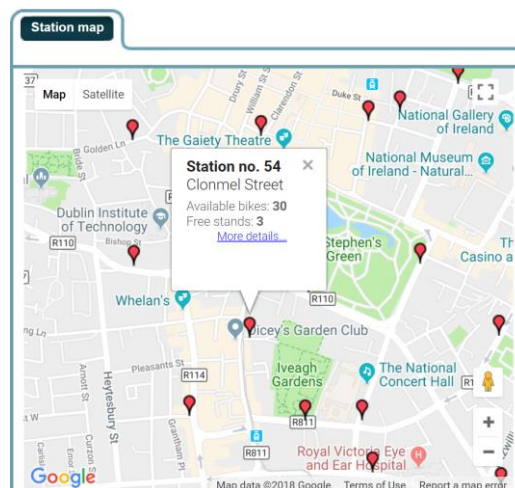
Project Report

Overview

Before we get started, we should introduce the dream team. Our group consists of Conor Lawlor (17203166), Emmet Tracey (17201032) and Daniel O'Byrne(17205389) . We are three bike riding enthusiasts from Dublin.



We were excited to get started on this project. All three of us have used the DublinBikes service. Based on our experiences, we have an acute appreciation of how frustrating their website is. The website is an unloved, dated mess. They provide a map showing all of the bike stations but it isn't very interactive. Clicking one of the station markers brings up the below screen:



The number of bikes and stands available is listed. Apart from this, the website doesn't offer much else. There is another page that has a list of all stations but this isn't interactive. All in all, it provides little help to the user. The website is clunky and awkward to navigate. What is so frustrating about this website is that JCDecaux provide a mountain of information on historical usage trends. This could be easily used to enhance the user's experience.

There is no valid explanation for why dublinbikes does not provide a better information service to users. They should provide charts detailing average usage on specific days/times

so that users can better plan their trips. Users could then make informed decisions when choosing where and when to rent a bike from. From our own experience, it was extremely frustrating to go to a station and find out there were no free bikes/stands. Our goal was to take the JCDecaux data and use it to try and bring DublinBikes up to current standards.

Modern websites espouse a minimalist, sleek style. www.dublinbikes.ie is a hark back to the mid-2000's in terms of layout and design. While offering increased functionality, we also wanted to improve the format in which the information is displayed. We focused on providing an interactive and elegant alternative to the dublinbikes website.

This project was split into three separate two-week sprints. For each sprint, a different one of us assumed the role of Scrum Master. This person was in charge of organising the stand-ups, maintaining the backlog, etc. We provide a summation of each sprint below, written by the respective Scrum Master.

We kept minutes for each daily stand-up. We tried to have one in person on each day of the sprint. We will provide a breakdown of each sprint below to show how our project progressed.

We have left the minutes from the stand-up and our sprint notes as they were. We were tempted to edit them to make them a bit more coherent but felt it was better to keep them as they were, as they capture our thought processes and frustrations.

To plan this project we used Google spreadsheets, Google Docs and Slack. We created a Trello account but we didn't really use it as we didn't feel it was necessary. The daily 'Taco' emails were also incredibly annoying. Instead we created a detailed product backlog (See appendix) and sprint backlogs on Google spreadsheets, which acted as our to-do list.

Our product owner's name was Karl and he provided us with guidance on how he wanted the end product to look and feel throughout the project.

Our project is now being hosted live on our EC2 instance at the following address:
<http://18.221.37.101:5000/>.



Our esteemed leader, Karl Roe

Sprints

For this section, each scrum master will detail the progress of the project during their sprint. We include notes on Sprint Planning, Daily Stand-ups, Sprint Reviews and Sprint Retrospectives. As most of our stand-ups were in person, the scrum master took minutes and posted them to slack. The few stand-ups we did have over slack are evident through separate entries for each team member.

The minutes of our stand-ups are thorough. In the interest of brevity, we have decided to include them as an appendix to this report rather than in the middle of each sprint. They provide a detailed breakdown of our progress each day and highlight all the problems we encountered and how we resolved them. For a full overview of how our project went, we recommend that you read them, although be warned that they are long.

Sprint 1 - Scrum Master: Conor

As the only member of the team who didn't decide to go away for a period of the first sprint, I was assigned as Scrum Master. The first day or two of the sprint involved a lot of planning and getting to grips with different technologies such as Trello, GitHub and Slack. We used Trello to set up our goals for the project but later decided to move the planning to a Google Docs Spreadsheet as it was more familiar to us. We also felt it was more amenable to how we wanted to organise our project as. We could have our product backlog there, produce our burn-down charts using Google Docs, and have our sprint planning notes and sprint review notes there as well. Google Docs was also useful as it allowed Karl to view our plans for the project. We found Slack to be very useful for maintaining a record of our daily stand-up meetings, and it also has a wide range of emojis, which the two lads really enjoyed. We were already quite familiar with GitHub from previous assignments, although having to share one repository between us all added a new dynamic where we were always checking if someone was working on the same thing for fear of overwriting someone's hard work.

Our product owner Karl was extremely helpful from the very start of the project and his advice allowed us to set out our goals for the sprint. It was his opinion that getting our data scraper set up was of utmost importance as the more data we could gather, the better. We spent quite some time planning how we wanted our product to look, what type of features we wanted to include etc. We were able to include all of this in our backlog on the Google Docs Spreadsheet.

Once we got to grips with how the project would work from an organisational point of view, we got going on the project itself in earnest. Our first sprint was a learning period, where we got a feel for how real software engineering projects work and how the agile/scrum techniques work.

Sprint Planning Meeting Notes(12/3/18)

- At the start of the meeting, we had trouble determining what features should be included in the product backlog

- We spent a good while discussing the features we wanted to incorporate in our application
- We met for 2 hours and discussed what we wanted the application to look like and what should be available to the user
- We used some (crude) drawings to try and visualise the end product
- Once we had decided on what we wanted the project to do, we began drafting the product backlog
- We moved features up and down the product backlog as we argued over their importance
- Finally, we settled on the features and split them up by sprint
- When the product backlog was finalised we started planning our first sprint
- I was chosen as Scrum Master based on the fact I hadn't plans to go away during the two week break and, therefore, I led the discussion
- We discussed each feature in the sprint and tried to estimate the expected time to implement it
- We drew up a project tracking spreadsheet to keep track of our progress

At the end of the planning meeting, the following goals for Sprint 1 were chosen;

- Have the scraper set up and running to get dynamic data on station occupancy
- Have the Google Maps API up and running with the station locations marked (using static data from DB)
- Create the flask app using CookieCutter
- Have the project hosted on our EC2
- Have our AWS Database set-up

At the end of the planning meeting, we had a rough idea of how we wanted the web app to work and look (see rough sketch below)



The following were the main ideas we had for the web app;

- Website should have a map with all stations marked on it.
- Markers should be colour-coded to show the current occupancy (e.g. Green = lots of available bikes, Red = few available bikes)
- When a user clicks on the station marker they should see how many free spaces there are, the hourly trend for the day and the current weather

- There should be an option to look at trends for each day of the week (possibly show the weather for the chosen day?)
- There should include links to DB website and MetEireann

Sprint 1 Backlog

Sprint 1 - Backlog

PROJECT DETAILS																	
FEATURE	STATUS	PRIORITY	ASSIGNEE	ESTIMATED HOURS	12/3	13/3	14/3	15/3	16/3	17/3	18/3	19/3	20/3	21/3	22/3	23/3	ACTUAL HOURS
Mine dynamic station occupancy data from JCDecaux	Complete	High	Emmet Tracey/ Daniel O'Byrne	5	3	2											5
Data from JCDecaux should populate a csv file continuously	Complete	High	Emmet Tracey/ Daniel O'Byrne	4	1.5	4											5.5
Push data from csv to AWS DB	In Progress	Medium	Emmet Tracey	9	2	1											3
Create staticData table on AWS DB	Complete	High	Emmet Tracey/ Daniel O'Byrne	3		1	2.5										3.5
Populate staticData table on AWS DB	Complete	High	Emmet Tracey/ Daniel O'Byrne	3			2										2
Create dynamicData table on AWS DB	Complete	High	Emmet Tracey/ Daniel O'Byrne	2			1										1
Populate dynamicData table on AWS DB	In Progress	High	Emmet Tracey/ Daniel O'Byrne	3			1										1
Create Flask application to create Google map template	Complete	High	Conor Lavelle/ Daniel O'Byrne	4	2			8									10
Link flask application and SQL database to show bike stations on Google Map	Complete	High	Conor Lavelle/ Emmet Tracey/ Daniel O'Byrne	6										4	11	4	19
Create link between EC2 and RDS	Complete	Low	Conor Lavelle	1		2											2
Host project on EC2	Not Yet Started	Medium	Conor Lavelle	3													0
TOTAL HOURS				43	8.5	10	6.5	8						4	11	4	52

Above you can see a breakdown of how long we estimated each feature to take and how long it actually took. In general, we underestimated the length of time each feature would take, although we managed to get all bar one of the tasks completed which is something we were happy with.

Daily Stand-Ups

As aforementioned, the minutes for all of our stand-ups are available in the appendix.

Sprint Review

Overall we are pretty happy with how the first Sprint went. We achieved most of our targets, apart from getting the project on our EC2 instance, although that wasn't very high on our list of priorities anyway as we felt it wouldn't be too hard to achieve later in the project. With regards to the specific targets, we wanted to have our scraper set up and running and the map showing in our browser with a marker on the location of each bike station, which we managed to achieve, although not without some hardship.

Dan and Emmet managed to get our SQL queries working in Python using SQLAlchemy, although we had difficulty running these Python scripts on the instance. I spent quite some time trying to get our EC2 instance to write to our RDS instance, which involved installing every possible SQL module for Anaconda. Eventually it came together and with the work Dan and Emmet had put in, we managed to get our data scraping into a CSV file on our EC2 instance as well as into tables on our RDS instance. The method we used to get our scraper program running continuously was the 'nohup' method which seemed to be quite temperamental, as it stopped running which led us to miss out on almost a weeks worth of data.

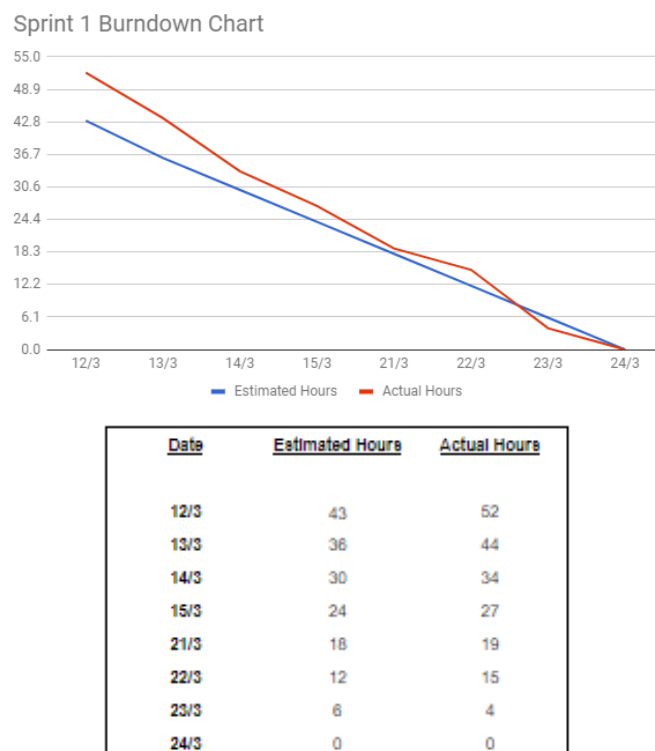
For the map, we got the markers working, although not in the manner we initially envisaged. Myself and Dan had some trouble getting this to work. We spent some time trying to use PHP, but unfortunately this was in vain, due it not being compatible with our project set up. We tried using various different methods, however, in the end we managed to do it using a mixture of the Flask 'render_template' method and Jinja.

With regards to the hours, it was very hard to estimate how long each feature would take. In the end, our burn-down chart actually looked quite good, although this was more down to luck than good judgement of our abilities to get the various tasks done in a certain timeframe.

Sprint Retrospective

From a process point of view, our first sprint couldn't have gone much better. We had quite a few frustrations, but overall we met all our targets with the exception of getting the web app running on our EC2 instance, something that, in hindsight, should never have been in our first sprint backlog. We also gradually got used to how the agile process works. Initially our logging of hours in the burndown chart was a bit haphazard, and often we didn't enter details until a day or two after. However, as the sprint went on, we got better at this and our daily stand-up meetings helped with this too as it forced us to keep each other informed of where we were at. This was especially useful when we were working separately (at home) rather than together in UCD. Of the various things we learned during the first sprint, I think that we all agree that we were more productive when working in close proximity to each other rather than working from home. Even though we may have been looking at different features, we were able to quickly bounce ideas off each other and ask for help as well.

Burndown Charts



Our burndown chart for the first sprint looks very good and, in terms of estimated hours and actual hours, we kept quite close to the mark, bar linking our flask application and SQL database to show bike stations on Google Map, which took three times as long as we had planned. Of course, this can't be taken too seriously, as our estimations were really guesses

as we had no real experience of this previously. That said, it was a good experience and we felt that it might allow us make realistic estimations for the following sprints.

Sprint 2 - Scrum Master: Emmet

We were happy with the progress that we had made during the first sprint. We had a good chat with Karl and he told us that he was happy with our progress. He also told us what he was expecting from the second sprint. I felt we had learned a lot about the Agile process during the first sprint. We were more comfortable with the different aspects of the process, stand-ups had become a part of our daily routine. Based on what Karl said during the Sprint 1 Review, coupled with our own thoughts from the Sprint 1 Retrospective, we began outlining our plan for Sprint 2. From talking with Karl, he mentioned that the focus for this sprint should be on functionality. Once we had the basic functionality in, we could then look at aesthetics.

Conor was a more than competent scrum master, but I felt that I was capable of greater things. I felt that I could lead this team to create the greatest interactive application for a bike sharing system the world had ever seen. Clearly, my head was swollen from all the red wine I'd been drinking while off gallivanting in Paris.

Sprint Planning Meeting Notes(March 26th/27th)

- Firstly, we had to take a snapshot of current EC2 instance and transfer it to another, in order to avoid losing our data
- We discussed with Karl the features he wanted. He proposed walking routes, heat maps and a couple other features we should strive towards
- At the start of this sprint our infoboxes appeared for each station, but the relevant information was not being pulled
- We had to get live data for each station appearing, which would be pulled directly from JCDecaux
- Once that was finished we had to create a dropdown menu that would have an option for each station. This would perform no action until we had created relevant chart functions
- We would then work on creating functions that would extract relevant information to create charts for weekly and daily information
- We wanted these charts to appear when the user clicks the relevant station on the map or, alternatively, if the station is chosen from the dropdown menu
- Once we had the charts working as required, we would begin to work on incorporating the weather into our map
- Finally, depending on the time frame, we would look at the extra features specified by the product owner regarding walking routes to bike stations etc.
- These extra features were low priority and if time was tight for the sprint, could be worked on further in the next sprint

Sprint 2 Backlog

After the planning meeting, we came together and created our Sprint 2 backlog. We had a long discussion over the level of importance for each feature. We ended up with the below backlog for the features we hoped to implement during the sprint.

FEATURE	STATUS	PRIORITY	ASSIGNEE	ESTIMATED HOURS	PROJECT DETAILS												ACTUAL HOURS
					27/3	28/3	29/3	30/3	31/3	1/4	2/4	3/4	4/4	5/4	6/4		
Create a snapshot of the existing RDS	Complete	High	Conor Lavelle Emmet Tracey	1	0.5		0.5									1	
Restore snapshot on a new instance	Complete	High	Conor Lavelle Emmet Tracey	3	1	3	1									5	
Show station real-time occupancy when clicked. Get an infobox showing for each station (name, address, bikes available, stands available)	Complete	High	Conor Lavelle Emmet Tracey Daniel O'Byrne	5		3		2								5	
--- Use escape from API	Complete	High	Conor Lavelle Emmet Tracey Daniel O'Byrne	3				3								3	
--- Pull information from RDS if API goes down	Not Yet Started	Medium		2												0	
Put in dropdown option for each station - User should be able to select a specific station and receive information on past trends and future predictions	In Progress	High	Conor Lavelle Emmet Tracey Daniel O'Byrne	4									4			4	
--- Query DB and retrieve information in a format that can be transformed into a Google chart	In Progress	High	Conor Lavelle Emmet Tracey Daniel O'Byrne	8							7		8	3	3	21	
--- Display chart showing weekly breakdown of data from RDS per station	In Progress	High	Conor Lavelle Emmet Tracey Daniel O'Byrne	6							6					6	
--- Display chart showing daily breakdown of data from RDS per station	In Progress	High	Conor Lavelle Emmet Tracey Daniel O'Byrne	3							2					2	
--- User should be able to select a station and enter a future date and time - will receive an estimate on availability using historical trends	Not Yet Started	Medium		3												0	
Using a user's location, app should provide a walking route to the selected station. Requires Google API	Not Yet Started	Medium		4												0	
Pull in weather forecast for Dublin	Not Yet Started	Low		2												0	
Station marker on map should show colour to indicate occupancy level	Not Yet Started	Low		3												0	
Change colour on markers to reflect occupancy	In Progress	Low	Conor Lavelle	2				2								2	
TOTAL HOURS				49	1.5	6	1.5	7	0	0	13	8	12	3	3	49	

Above you can see a breakdown of how long we estimated each feature to take and how long it actually took. We were a little over ambitious in this sprint as five of the features were never even started, let alone finished.

Daily stand-ups

As aforementioned, the minutes for all of our stand-ups are available in the appendix.

Sprint Review

The infoboxes caused a lot of frustration at the end of Sprint 1. However, a slightly different approach was found and we got them to work very early in this sprint. A lot of the rest of Sprint 2 was frustrating though. We had been having problems with our instance and this came to a head at the start of the sprint. We were using an AWS Educate account but we received emails that we were approaching our maximum limits. We switched to a regular account and restored our RDS on this. The next day we were cut off from our original instance.

Our biggest frustration was to do with getting the charts. We were able to query the DB and retrieve the relevant information. However, we couldn't find a way to access this data. Reading the stand-up minutes over these days, it is evident how much time we spent trying to crack this problem. Over 30 man hours in total. We met with Karl and he provided us with some comments and advice but unfortunately we couldn't crack the charts during this sprint.

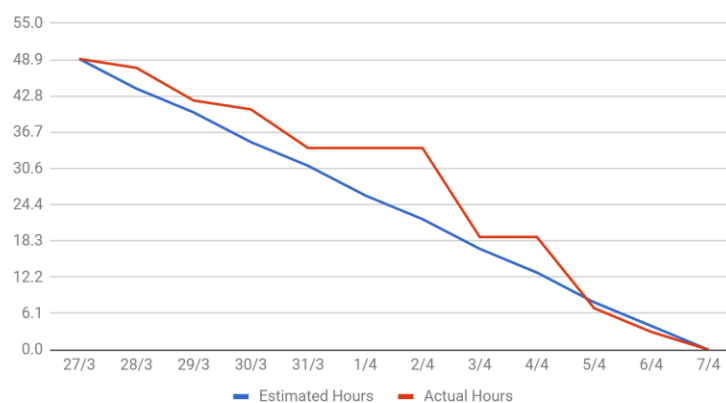
Sprint Retrospective

In terms of deliverable code, Sprint 2 was frustrating. However, from a project process side it was successful. The stand-ups were great as they allowed us to be constantly up to date on what each member of the team was working on. We also used the sprint backlog frequently to choose what features to focus on. Logging the hours each day was also satisfying as we could see the progress we were making. We were familiar with the Agile process and so

found it easier to adhere to its methodologies during the second sprint. Our estimations for the time it would take to implement each feature was less accurate in this sprint. The main reason for this was our issue with the charts ended up absorbing heaps of our time. During the Sprint 2 Retrospective we discussed this difficulty we were having with estimating the time to allocate for each feature. We had read that this is a difficult aspect of the Agile process and we had learned first-hand that sometimes the estimations can be way off. We agreed that we should factor in some extra time for features that involved code/approaches we weren't familiar with. For example, we had no experience using app.route and only limited experience with JSON data. We realised that we underestimated how long it would take to get our head around these concepts and brought this knowledge into Sprint 3.

Burndown Charts

Sprint 2 Burndown Chart



Date	Estimated Hours	Actual Hours
27/3	49	49
28/3	44	48
29/3	40	42
30/3	35	41
31/3	31	34
1/4	26	34
2/4	22	34
3/4	17	19
4/4	13	19
5/4	8	7
6/4	4	3
7/4	0	0

In an odd coincidence we ended up spending exactly the same amount of hours as estimated. However, this does not mean we are Agile experts who could predict the time taken to implement each feature perfectly. Looking at the Sprint 2 Backlog, you can see that we had 5 features that were still 'In Progress' at the end of the sprint and another 5 that were 'Not Yet Started'. We were way off in terms of how many features we would be able to implement in this sprint (mainly due to the ludicrously long time we spent on the charts). We gave ourselves too much to do in this sprint. Although our total estimated hours was accurate, we implemented significantly less than we had hoped.

Sprint 3 - Scrum Master: Dan

So far we had been pleased with how the project had been progressing. Karl also seemed very happy with our progress so we continued into Sprint 3 with confidence that we could produce a great app. The functionality of our website was close to what we had intended for the application, so we knew appearance would be a priority in this Sprint. At this stage, we were also very comfortable with the agile processes involved in the project. We have made a routine of taking detailed daily stand-up notes, which I aimed to continue through the next two weeks.

Thus far, Conor and Emmet had done a stellar job in captaining the ship towards achieving our sprint goals. However, as Vannesa Williams belted out in her 1991 smash hit, I felt, in terms of scrum masters, we had "Save(d) The Best For Last".

Sprint 3 Planning Meeting (April 10th)

- In Sprint 2 review, we spent a large chunk of time during that sprint trying to figure out a way to extract the relevant numerical data to populate the chart. On the morning of the planning meeting, we finally had the breakthrough to retrieve this information. A great way to start the sprint.
- Karl informed us that he was happy with the functionality of our application and we now needed to focus on the aesthetics of the page. We discussed some of our target features and possible bonus features we could implement, depending on time constraints.
- Karl suggested having the station and day dropdowns located above the map, with divs populated with the relevant charts sliding up to populate the space under the map. We agreed that we would try to implement it and decide how it looked and make a decision after.
- Karl gave us good advice regarding adding features that a user would be amused by, so the page wasn't all about functionality but could bring a smile to the face of a user. We agreed that we wanted to implement some form of gif or other feature that would satisfy this requirement. The loading images for the overall page and the charts would be the place to implement this.
- We agreed that below the map would be populated with three divs. The left and right divs would be where the charts would populate, while the middle would show the live data for the chosen station.
- We discussed our plan for incorporating the effect of the weather on bike availability. We planned to take historical data from a selection of stations, analyse availability on good/bad weather days, and infer statistics from the results. We could then warn the user about the effect of the weather, as learned by our model
- We also discussed our database and EC2 instance problems with Karl again. The memory on our instance was being rapidly used up. We decided over this sprint that we could potentially optimise the data by deleting rows in the database and this would also shorten the response time for the charts to load, as less data would be queried.

Sprint 3 Backlog

After the planning meeting with Karl, we further discussed the level of priority of each feature and finalised this list in our Sprint 3 backlog. We also estimated the number of hours we thought implementing these features would take.

Sprint 3 - Backlog		PROJECT DETAILS														
FEATURE	STATUS	PRIORITY	ASSIGNEE	ESTIMATED HOURS	10/4	11/4	12/4	13/4	14/4	15/4	16/4	17/4	18/4	19/4	20/4	ACTUAL HOURS
Put in dropdown option for each station - User should be able to select a specific station and receive information on past trends	Complete (Started in Sprint 2)	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	3	3											3
On clicking a button within the infocases on the map - query DB and retrieve information in a format that can be transformed into a Google chart	Complete (Started in Sprint 2)	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	2	4											4
Display chart showing weekly breakdown of data from RDS per station	Complete (Started in Sprint 2)	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	2		2										2
Display chart showing daily breakdown of data from RDS per station	Complete (Started in Sprint 2)	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	1		1										1
Create a formal plan of what we want the website to look like. We will then use CSS to apply this plan	Complete (Started in Sprint 2)	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	0.5	1											1
Use JS, CSS and HTML to carry out our design plan	In Progress	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	25	2	3	2	5	10		2	15	3	5	3	50
Clean and comment all of our code	Complete	High	Conor Lawlor Enmet Tracey Daniel O'Byrne	2								1	1	1	1	4
Create appropriate tests for our code	In Progress	High	Enmet Tracey	3								1	0.5		1	2.5
Host project on EC2	Completed	Medium	Conor Lawlor	2					2							2
Pull information from RDS if API goes down	In Progress	Medium	Conor Lawlor	2								1				1
The dropdown should be alphabetically ordered so the user can easily find their required station	Complete (Started in Sprint 2)	Medium	Conor Lawlor Enmet Tracey Daniel O'Byrne	2		1										1
Change colour on markers to reflect occupancy	Complete	Medium	Conor Lawlor	2			1									1
Pull in live weather for Dublin	Complete	Medium	Conor Lawlor	3				1	1							2
Populate a div with a 5 day forecast pulled from an API, so users can plan a trip	Complete	Medium	Conor Lawlor Enmet Tracey	2								1		1		2
Create a prediction model that analyses bike occupancy in times of good or bad weather. We will then provide the user with an estimate of available bikes, based on our model	Complete	Medium	Conor Lawlor Enmet Tracey Daniel O'Byrne	5	1							3	1			5
Using a user's location, app should provide a walking route to the selected station. Requires Google API	Not Yet Started	Low		4			1									1
TOTAL HOURS				60.5	11	7	5	6	12	8	2	22	5.5	7	5	82.5

Although the planning meeting focused mainly on the design of the app, we still had some function features we needed to implement. Some of these features started in Sprint 2, however we did not have time to complete them. Thus they were set as high priority.

Daily stand-ups

As aforementioned, the minutes for all of our stand-ups are available in the appendix.

Sprint Review

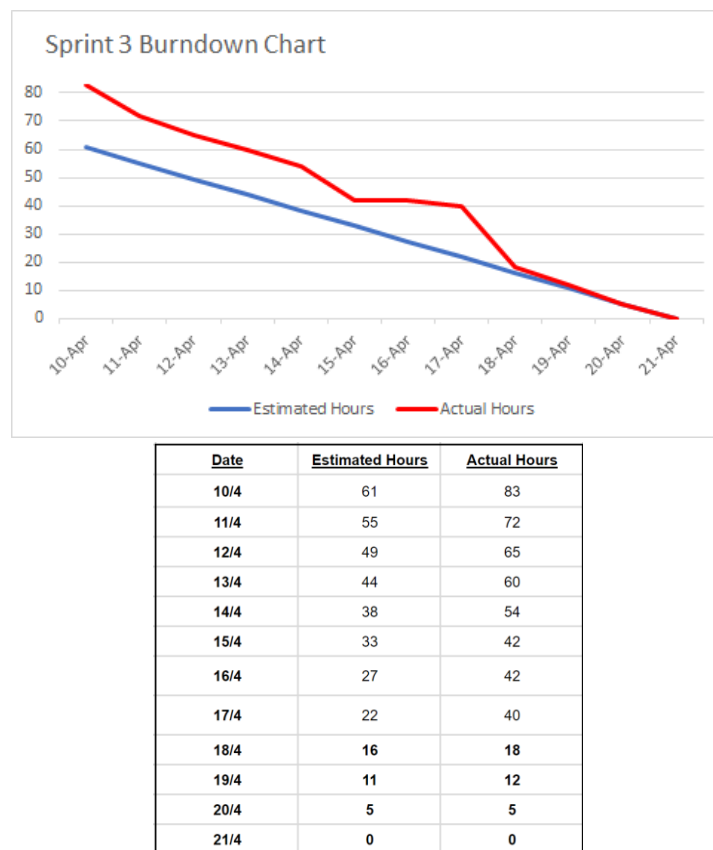
The final sprint was very rewarding as we witnessed our application come together into what we envisaged it to be. From our first meeting together of understanding how to properly set up a database and brainstorming ideas for the functionality, to now, the process has been a fruitful one. One which we feel will stand to us come the summer project.

Entering into the third sprint, we felt that the functionality of the page was mostly complete. We thought that the appearance of the page would be our main priority. However, as the sprint planning meeting showed, there were many functional requirements of the page that were incomplete. Thus, we spent significantly more hours on the project in this sprint than the previous two.

The sprint began with a significant breakthrough. We figured out how to extract the relevant information from the dictionary returned by the Python file, which would populate the charts. This was a major stumbling block in Sprint 2, one that prevented us from reaching some of our Sprint 2 goals. When this was implemented we began designing our app and implementing it. I felt we worked great as a team in this sprint as all three of us rotated tasks. This rotation happened between working on certain functionality such as dropdown interaction with the map, and implementing design features using CSS. We also used the skills we had gained in our Data Analytics module in creating a model to predict bike availability on rainy days.

There were a number of features that we did not get to implement in this sprint however. We felt this was mostly due to some major delays we experienced implementing certain features in the previous two sprints. Originally we had planned that if the API were to go down, the map would be created using data stored on our database. However as we looked to implement that during this sprint, we realised that we would have to rewrite much of the Javascript and merge the static and dynamic databases. We felt at this late stage it was not worthwhile or possible to meet this requirement, which is a shame. There were also a few small features we would have liked to implement such as using the user's current location to find a route to a station. However, this along with a couple more potential features were low priority for us.

Burndown Charts



As mentioned, we expected a significant increase in the number of estimated hours for this sprint. Consequently, our actual hours were greater than expected. This was mainly due to the fact that we spent 50 hours on implementing our design plan, when we had estimated 25.

Sprint Retrospective

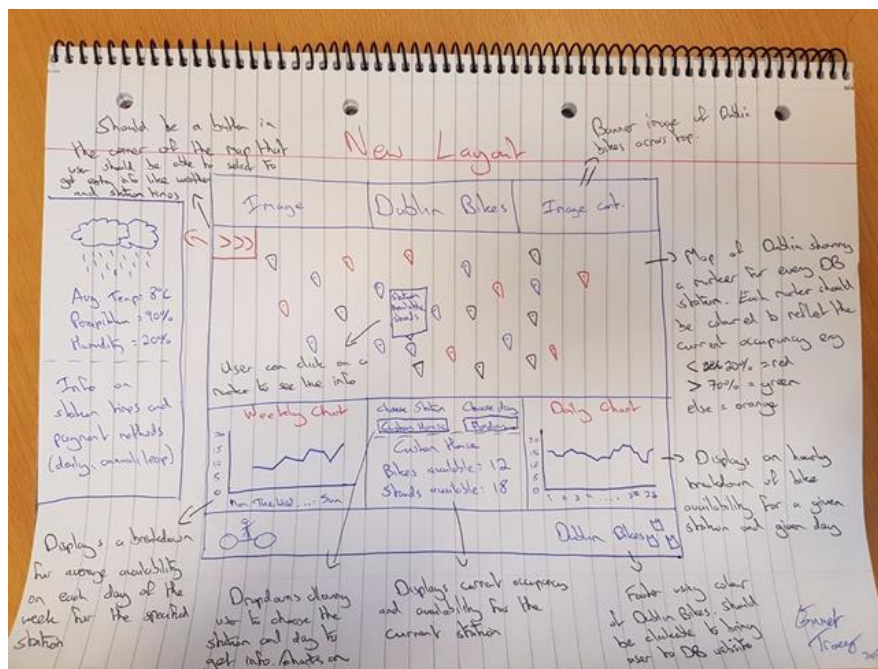
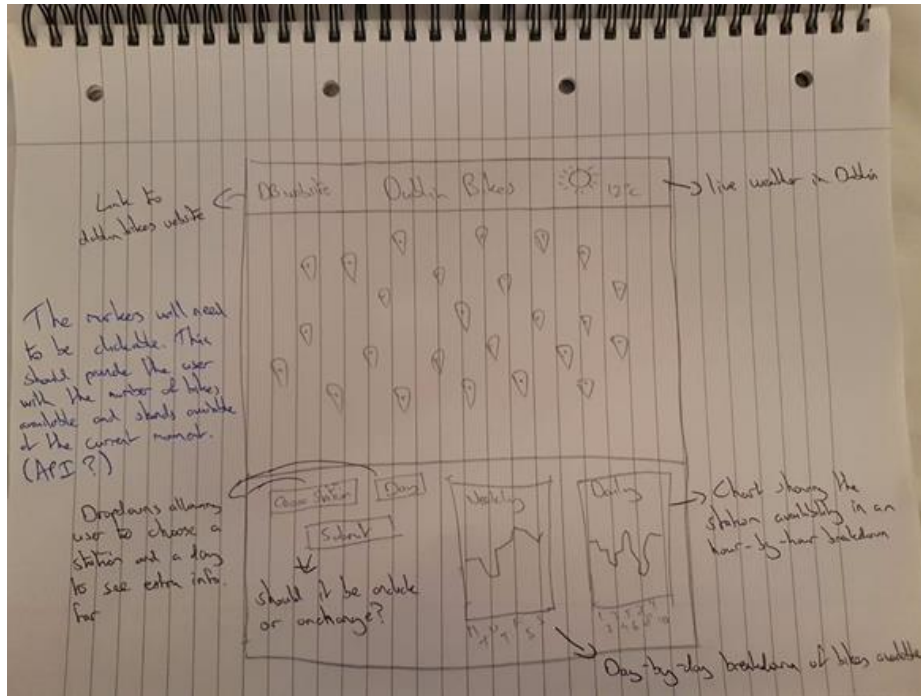
By the end of this sprint I felt we were all very comfortable with the Agile process and could fully envisage its benefits in the workplace. I found the sprint review (for Sprint 2) and the planning meeting (for Sprint 3) especially useful as it gave us a guide to the progress we had made and the progress that was required for the coming weeks. It also gave us the

opportunity to reflect on what features we didn't implement, so we could re-adjust our expectations, as we had to do for this sprint.

The daily stand-ups were very beneficial for the continuous tracking of these expectations. The fact that we had a daily stand-up in person almost every day, for the last six weeks, showed our commitment to this project and its process. I felt it was also testament to how we worked as a team.

Evolution of Design

We went through a few different design concepts before settling on what we have now. While debating how we wanted the application to look we found it helpful to draw up some mock interfaces. These provided a nice visualisation of how we wanted the app to look. This made it easier to write the HTML/CSS code. We drew the first design sketch during the first sprint planning meeting. We drew up the more detailed, second sketch near the start of Sprint 3. We fleshed it out more and used it to try and capture the final design.



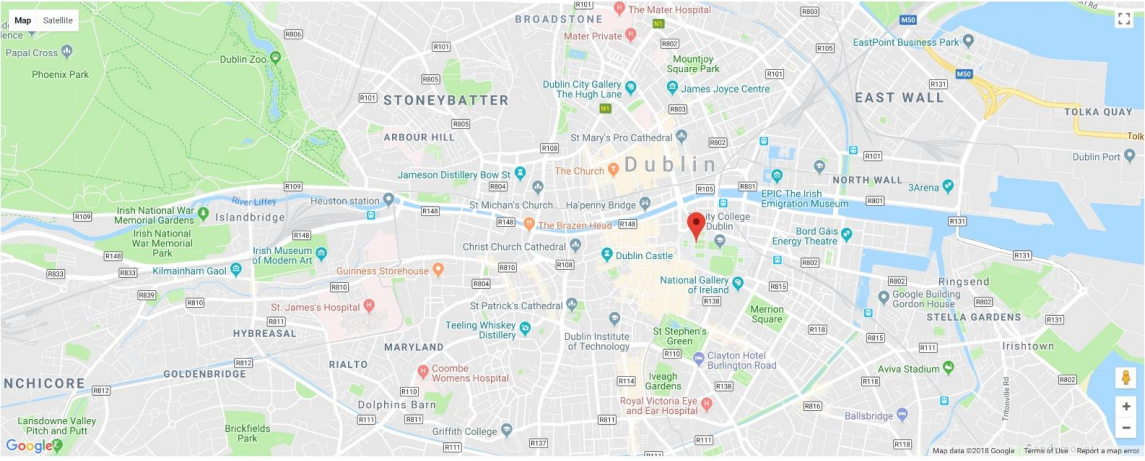
We had numerous iterations of our project as we attempted to put the above designs into practice. Some screenshots of the user interfaces development are provided below:

UCD Connect x oleathic/Group20.COM x Upgrading to Newer Rel... x Dublin Bike Planner x

127.0.0.1:5000

Apps Myzone Facebook Gmail: Email from G... UCD Connect Title vision

DUBLIN BIKES



Map Satellite

BROADSTONE

STONEYBATTER

ARBOUR HILL

DUBLIN

EAST WALL

TOLKA QUAY

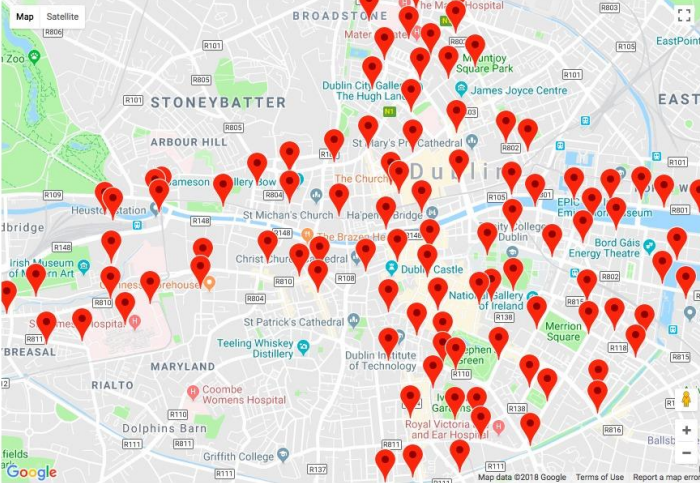
Map data ©2018 Google Terms of Use Report a map error

Dublin Bike Planner x flask render_template variable x Quickstart — Flask Document... x The Flask Mega-Tutorial Part... x Conchur

localhost:5000

Apps Home - Google Pla... Inbox (1,518) - olea... Sky Sports | Sport... Fálite go Facebook... Tech News, Review... Amazon.co.uk: Low... CS Conv Plex Programming NETGEAR ReadyNAS

DUBLIN BIKES



Map Satellite

BROADSTONE

STONEYBATTER

ARBOUR HILL

DUBLIN

EAST WALL

Map data ©2018 Google Terms of Use Report a map error

Elements Console Sources Network Performance

Network

localhost:5000

(index) x

Serving from the file system? Add your files... more never show X

```

68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
var markers = new google.maps.Marker({
  position: {lat: 53.340927, lng: -6.262501},
  draggable: false,
  animation: google.maps.Animation.DROP,
  map: map
});

var markers = new google.maps.Marker({
  position: {lat: 53.356769, lng: -6.26814},
  draggable: false,
  animation: google.maps.Animation.DROP,
  map: map
});

var markers = new google.maps.Marker({
  position: {lat: 53.351182, lng: -6.269859},
  draggable: false,
  animation: google.maps.Animation.DROP,
  map: map
});

var markers = new google.maps.Marker({
  position: {lat: 53.346874, lng: -6.272976},
  draggable: false,
  animation: google.maps.Animation.DROP,
  map: map
});

var markers = new google.maps.Marker({
  position: {lat: 53.380662, lng: -6.260177},
  draggable: false,
  animation: google.maps.Animation.DROP,
  map: map
});

```

Line 87, Column 4

Call Stack

Not paused

Breakpoints

No breakpoints

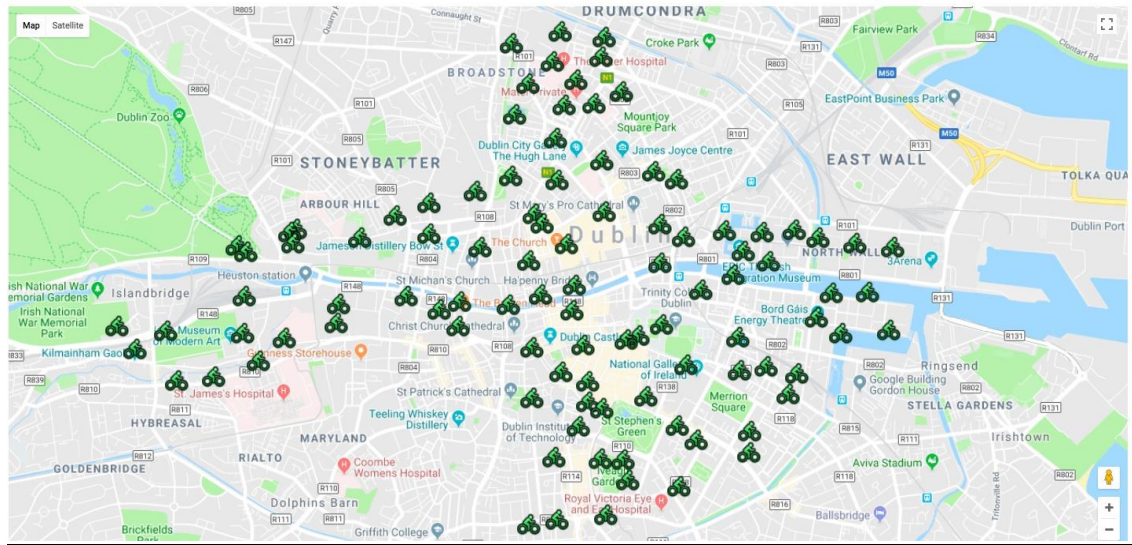
XHR/fetch Breakpoints

DOM Breakpoints

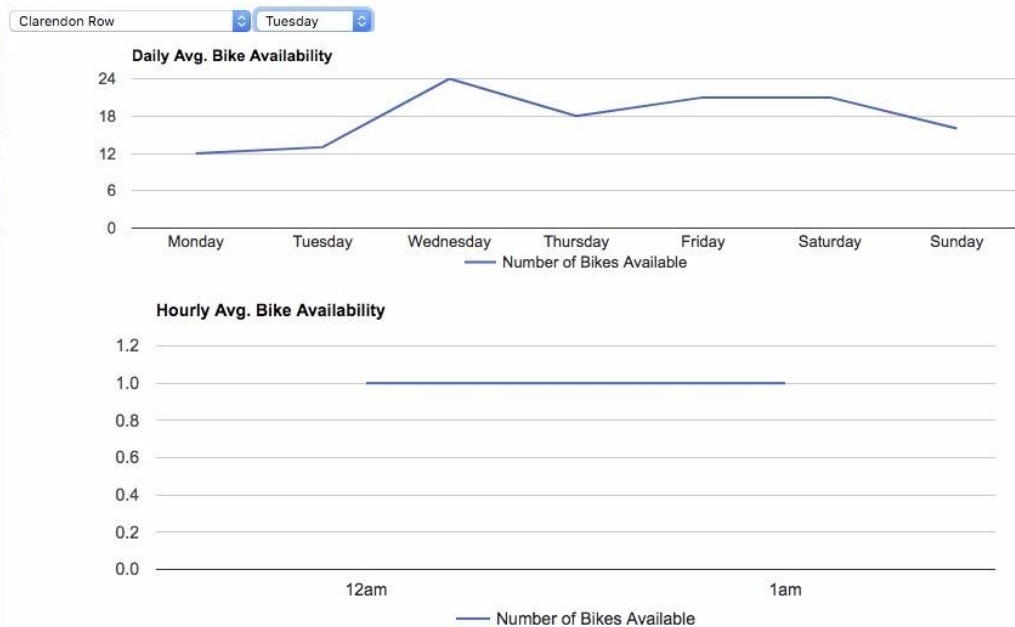
Global Listeners

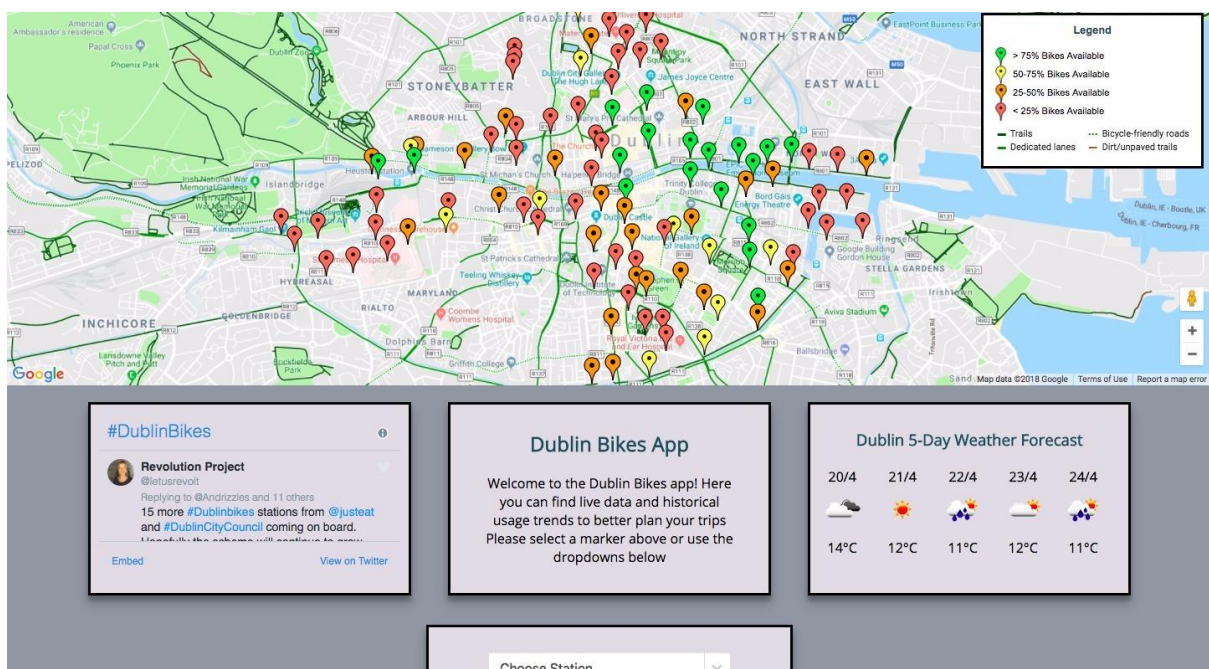
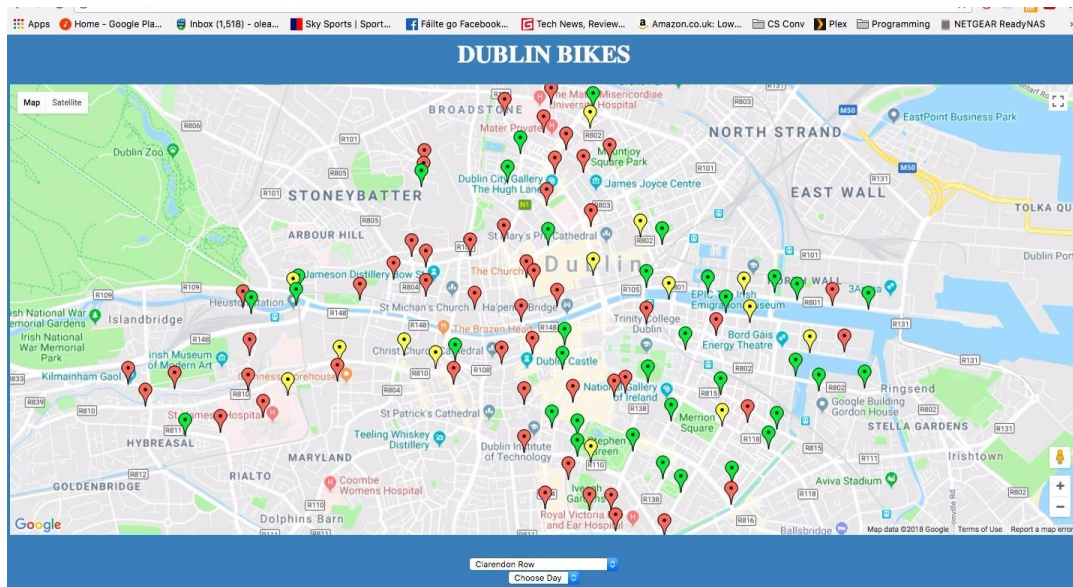


DUBLIN BIKES



Information for:76



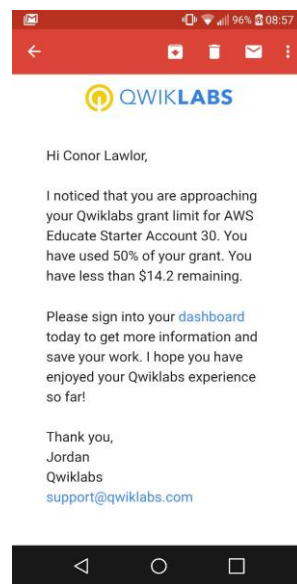


Problems Encountered/Solutions

Over the six weeks of the project we encountered numerous problems. This was the biggest coding project any of us had tackled so we ran into a heap of issues. A lot of what we needed to use was new to us so there was many hours spent trying to understand how it worked. It was also difficult to work out how exactly Python, Javascript, HTML, CSS and SQL all interacted with each other. Our stand-up minutes in the appendix contain a log of pretty much every problem we faced and how we overcame it. In this section we will discuss some of the main problems instead of rehashing everything.

Instance

We had problems with our instance from the start. We had all used an AWS instance for the previous Soft Eng projects so we thought it would be fine. Unfortunately, we ran into multiple issues. We were originally using Conor's instance which was on an AWS Educate account. We soon started getting emails from QwikLabs that we were approaching our usage limits. We had the scraper file running on this instance and also the RDS so we were concerned that we wouldn't be able to continue scraping data or that we would be locked out of our RDS and lose our existing data. We decided to create a snapshot of the existing RDS and restore it on a different instance. We decided to use Emmet's instance for this. We successfully restored the snapshot on this instance and everything seemed to be fine. However, Emmet's instance was supporting Amazon Linux while Conor's had been Ubuntu. We soon noticed that our programs were failing as SQL wasn't working properly on Emmet's Linux instance. After a few hours of research and installing heaps of different packages we decided it would be better to cut our losses and use a third instance. We launched a new Ubuntu instance on Emmet's. Thankfully we were able to get SQL working on this one, although it took quite a while as we never kept a log of all modules installed on the first instance. We restored the snapshot of our DB as well, and we were soon up and running again. This whole time we had kept the original instance running. The day after we got the third instance working we found out we had been locked out of the original instance for breaching the usage limits (we had used up our allotted QwikLabs credit).



Even with the new instance we kept having problems. We realised that the scraper file, running with the 'nohup' command, was stopping by itself, so we were no longer constantly getting data. After some research it seemed to be that the instance was out of space and could no longer run the scraping file. We deleted every unnecessary file and program we could find but this only created a tiny amount of extra space. We spoke to Karl and others about this and they didn't really understand why all of the available space on the instance was being used up. We decided to change our scraper to run every hour instead of every five minutes to save space, and then for the final job of hosting the project, Conor set up a new AWS instance (not on the educate system), which we have had no issues with thankfully.

We spent a long time researching the problems with our instances and working to ensure that our data was safe. This was not a problem we had been expecting to face at the outset and at times it was very annoying. We didn't seem to be doing anything different to other teams but we had a whole host of problems they weren't facing.

Charts

After our issues with the instance, our biggest headache was getting the information from our RDS to be displayed in a chart. We started working on this on Friday, March 30th and we only cracked it on Wednesday, April 11th. It was, by far, the feature we spent the longest time working on and just about the most frustrating experience of our cushy lives.

We enjoyed some early success with the charts. We were able to work out how to query the RDS and retrieve information. When we just hard coded a station and day into the function we could generate the wanted results and populate the charts. We ran into problems once we tried to make these charts dynamic and update to reflect a user's choice.

We spent hours cooped up in study rooms together trying to solve this problem. We had been able to use app.route to call a function from the app. This then queried the DB and retrieved the information for whatever station was requested. Our problem was with the format that the data came back in. It seemed to be a dictionary with only one key and a value made up of an array of arrays. No matter what we tried we could not find a way to access the data in this format in order to pass it into our drawChart function. We ended up spending about 10 hours each looking at different approaches and couldn't solve it. We tried retrieving the information a different way, using jsonify and pretty much everything else we could find on Stack Overflow but to no avail. No matter what approach we took the data always returned as dictionary with one key. The corresponding values consisted of an array of arrays and we were unable to find a way to access this data and pass it into the generate charts function.

After much trial and error we were finally able to solve this problem. We realised that in order to get it to work we would need to use objects. We amended the queries to return an array of dictionaries as objects. We then had to insert the key in order to return the value. These numerical values were then used as the data to create the chart. It was disheartening how long it took us to solve the problem. We can only imagine how amazing our app would have been if we could have put this time to another use.

It is also worth pointing out that even when we got our charts working, the browser console showed an error each time the specific methods were called, despite there being no actual issues with the functionality of the charts of the web app as a whole.

Markers

We spent a long time trying to populate our map with the markers for each station using PHP. After speaking to Karl during one of the meetings we were told this wasn't a desirable approach and so went back to the drawing board. Conor was the person working on the markers predominantly and he spent a good while looking up how to incorporate them. Initially, the markers were created using a static table from our RDS instance, however, this was going to make it hard to show live information in our info windows associated with the markers. Eventually the issue was solved by using the JCDecaux API file to get the latitude and longitude for each station and a Jinja for loop.

Dropdown

Adding in dropdown lists of all the bike stations operated by DublinBikes seemed like a pretty straightforward task at the start. However, considering we had plans for a lot of functionality to work in tandem with these dropdowns, it proved a much more trialsome task. Firstly we wanted to set up a dropdown for the list of stations. Straightaway, we had an issue reading in the list of station names and having them populate the option tags in our select menus. Initially we tried using the `render_template` method from our flask app to do this by piping in the station data from our static table. Although this method worked, it wasn't suitable for the extra functions we wanted our station dropdown to do. After some trial and error, we ended up creating an array containing the station name and number which was created when the JCDecaux data gets pulled in. This worked quite well and we created a dictionary for each station, with the number as key and name as value. Through this method we could order the dropdown in alphabetical order.

We planned on having a dropdown list for the days of the week. This was easy to implement initially, but we would later have difficulty linking this to the first dropdown station so that the daily dropdown could implement a chart for the selected station and day. Once we had the dropdowns in place, the next task was to get them to load the charts. As mentioned above, we had a lot of difficulty in doing this although once we managed to get the correct data to create the charts pulled, the stations dropdown was able to create the weekly chart without much extra effort. The real difficulty lay in getting the station number from the stations dropdown pushed to the daily dropdown so that it could call a daily chart showing the hour by hour average for the same station on the chosen day. To do this, we ended up creating a global variable which contained the station number of the station selected in the stations dropdown. Through this we could call a weekly and daily chart for our selected station, however, the loading of the charts was especially slow. There wasn't much we could do about this initially, but we did see an issue in that when a new station was selected, the daily chart for the previous station would still be shown until a day was selected. To tackle this issue, Dan and Emmet set it up so that the current day was loaded for the newly selected station.

The next issue we needed to deal with was that when a user clicks the more information button in the info window on the station marker that, along with loading the charts, the station

shown in the dropdown would change to mirror that of the one chosen on the map. Between the three of us, we probably spent a total of 4 or 5 hours on this. In the end the solution was actually quite straightforward. We simply created a function which would match the station number of the info window to that of the dropdown list. We may have sorted this issue out quicker if we had realised that part of our problem was the function we used to sort the stations in the dropdown in alphabetical order. All part of the learning process, one could say.

Another issue we had with the dropdowns, yes another one, was that we wanted to show the current information on the chosen station in an information box below. This seemed straightforward and we thought we had it figured out quite quickly, as all we needed to do was add the station's available bikes and stands etc. into the array of dictionaries we had created to populate the dropdown. This worked, but we soon realised that the information passed to this information box differed if you used the map marker. Conor figured out that the reason for this was that the value assigned to the option value (which was being used to call the function to populate this information box) was just the value within the stations array, and not the actual station number. After an hour or two of playing around, Conor managed to sort this by adding the station number as an id for each option in the dropdown list.

The final piece in the dropdown puzzle involved trying to link the stations to our map markers and their associated info windows. Conor spent several hours researching different methods to achieve this but had no success. We then decided to change tack, as although we wanted the dropdown list to show the location of the selected station on the map, we didn't necessarily need it to populate the info window, as this information was already being shown below the dropdowns. Instead, Conor managed to get the map to zoom to the location of the selected station. This was actually relatively straightforward, and a good work around to get to the same end goal.

Features/Functionality

We won't discuss all of the features that our app has. A lot of them are obvious. We will discuss some of the more nuanced features. We will also explain how the app works and why we designed it in this way. One of our main focuses was interactivity. We wanted each aspect of the app to be connected and communicate with each other.

Database

- We created 2 tables on RDS
- The first table, called StaticData, contained the following information:
 - Station Number
 - Station Name
 - Station Address
 - Station Latitude
 - Station Longitude
- We originally used this table to place the markers on the map. During the project we realised that it would be easier to use the data from the API and so retired this table
- We had a second table called DynamicData, which contained the following information:
 - Station Number
 - Station Status
 - No. of Bikes Available
 - No. of Stands Available
 - Timestamp

Scraper

- We set up a file to scrape data from JCDecaux every 5 minutes
- This information was then stored in the DynamicData table
- We used this information to generate our charts. When users request historical trends on a particular station the DynamicData table is queried
- We created a file called dynamicDataScraper.py that performed this scrape. It pulled the information needed to populate the columns in the DynamicData table
- We used a 'nohup' command and ran the dynamicDataScraper.py file on our EC2 instance so that it would run interminably
- As mentioned earlier we ran into a lot of problems with the scraper. For some reason our dynamicDataScraper.py file would stop running and we would be no longer collecting data
- We eventually decided to change the scraper to gather data every hour instead. This change worked and the program stopped abruptly ending

Map

- The map takes up a large proportion of our website. We wanted the map to be big enough so that users could easily find their desired station.
- The map is overlaid with bicycle layer. This shows the location of dedicated cycling lanes and bike friendly roads
- The map has a legend on the right to assist users

Markers

- The markers on the map are colour-coded to provide the user with a quick overview of bike availability in each station
- The markers are interactive. Selecting one causes a pop-up box to appear displaying the station name, number of bikes available and number of stands available
- The pop-up box also provides information on using your Leap Card and whether credit/debit cards are accepted at the terminal
- The pop-up box has a button labelled 'More Information' that a user can click. This causes multiple actions:
 - A weekly chart displays in the bottom left corner of the page. This shows the average bike availability at the respective station for each day of the week
 - The 'Choose Station' dropdown changes to show the respective station name. The 'Choose Day' dropdown changes to the current day
 - The number of bikes and stands available at the respective station is displayed below the dropdowns
 - An hourly chart displays in the bottom right corner of the page. This shows the average bike availability at the respective station for each hour of the day chosen that the station is open (The bikes can be rented from 5am to 11.30pm)

Twitter

- We have embedded a Twitter feed below the map on the left. This shows any tweets with the hashtag #DublinBikes in order of recency

Weather

- The weather is displayed below the map on the right. This provides a live forecast of the weather in Dublin city for the next 5 days. This is pulled from OpenWeatherMap and updates each time the page is loaded so the forecast is as accurate as possible
- We chose to only include the date, icon and temperature. We felt that this was necessary as users are most likely to be interested in whether it is forecast to rain or not as cycling in the rain is bleak

Charts

- The weekly chart shows the average number of bikes available at the chosen station for each day of the week. We query the DB for all the available information on the chosen station. The average number of bikes available at each station on each day is then calculated. This information is then used to populate the charts.
- The hourly chart shows the average number of bikes available at the chosen station for each hour on the chosen day of the week. We query the DB for all the available information on the chosen station on the chosen day. The average number of bikes available at each station on this day for each hour is then calculated. This information is then used to populate the charts.
- The charts have two lines. The blue line plots the average number of bikes available. The red line plots the average number of bikes our model predicts will be available if it is raining

- A gif of a bike cycling displays as the charts are loading. We included this as we wanted the users to know that their request for information had been acknowledged and that the charts were just waiting for the data to return

Dropdowns

- The app contains two dropdowns, a 'Choose Station' dropdown and a 'Choose Day' dropdown
- The 'Choose Station' dropdown contain the names of all stations operated by DublinBikes and, as with the more information button in the marker info window, selecting a station will populate a weekly chart and a daily chart (which shows the current days hourly breakdown by default).
- Selecting a station will also populate the information box below the dropdowns, showing the number of bikes and stands available etc., at the respective station.
- Selecting a station also causes the map to zoom in on the location of the chosen station.

Prediction model

- The charts contain two data lines. One is the average of the number of bikes for a particular station, on a particular day/hour. The other is this figure adjusted for the results of our model.
- Our model predicts that on average, there are 4% more bikes available at a station, when there is rain.
- To come to this prediction, we read in the weather data given by the product owner. We then used data scraped from the API between the 13th to the 27th of March, which we had saved to a CSV file. There were three days missing from our data so we could compare eleven days worth of data.
- From these eleven days, we had 228,000 lines worth of data, as it scraped every 5 minutes. For the sake of this model we felt it unnecessary to include all of these lines. We only considered every 20th line, so in total we had 11,400 lines.
- Having made these adjustments, we merged the two dataframes on the equality of the date and the hour as the weather data was provided once every hour.
- We then trained a linear regression model on this one dataframe, yielding our prediction.

Optimisations

In Sprint 1 as we were figuring out how to use the bike information in our application, we thought we would pull from the database that was continuously updating. However we soon realised this may not be the most efficient method. From a users point of view, it would take longer to retrieve live information from a dynamically populating database, than just pulling straight from the source API. However, at the time we thought that we could use this method as a back up if the API ever went down. We continued to create our application and thought we could revisit this optimisation technique in Sprint 3. Conor began to look at implementing this, however he soon realised it would take a lot more work than expected. The way our Javascript functions were set up to create the map did not match the data we were pushing into the Static and Dynamic tables on our instance. If we were to fully implement this optimisation, we would have had to merge these databases, and rewrite the majority of our Javascript/HTML. As we only approached the problem late in the process, we felt it was not possible to achieve. This reliance on the API is an area of weakness in our application, and this lack of robustness is something that is something we would like to avoid with future projects, whether or not they need to pull live information.

When we did get the historical data charts working, they took longer than we had imagined to populate. Therefore we looked into optimising them to return faster to the user. Firstly, we did not consider any data between 11pm and 5am, as bikes are not available to rent in this time period. This extra information is useless to the user and it was only a cause for slowing down our SQL queries and loading of the charts. When the relevant information did come back from these queries, we also optimised the format in which they were used in the chart. As the numbers had to be rounded, we did this in the Javascript rather than within the SQL query itself. This made loading the charts marginally quicker compared to what we originally had, although it isn't much of an optimisation.

Future

There are many features we wish we could have implemented if we'd had more time. Unfortunately we sunk so much time during Sprint 2 into the charts that we didn't have as long as we would have liked to spend on jazzing up the application and adding increased functionality. Of all the words of mice and men, the saddest are, "it might have been."

One of the main features that we would have liked to include was to use a user's geolocation in order to provide them with a walking route to the chosen station. Although we spent some time looking into this during Sprint 3, we were under a bit of time pressure and decided it was better to focus on what we had already, instead of trying to implement something new. It is also worth noting that there would have been difficulties implementing this on the EC2 instance as the website would not be following HTTPS, and therefore web browsers would direct users away from the website as it wouldn't be considered secure. Apparently there is a work around for this, although to sink all those extra hours into this additional feature could have been detrimental to the rest of the project.

If we had more time we also would have improved the charts. We only had 4/5 weeks of bike usage data available to use. The averages we provide for weekly/daily breakdown may not be too accurate as there simply hasn't been enough data recorded to produce a proper average. If we kept the scraper and app running for a few more months we would be happy that the figures provided for average bikes available for each station would be closer to reality. At the moment, our charts are susceptible to outliers. Over time our charts would have regressed to the mean and the average figures provided would have been accurate. As we have only been scraping data for the last few weeks our data is also vulnerable to seasonal change. The weather has been particularly bad this spring which means that usage levels are likely to have been lower than normal. The weather is likely to improve in the coming months which is likely to see a rise in the number of users as the fair weather cyclists emerge from their hibernation.

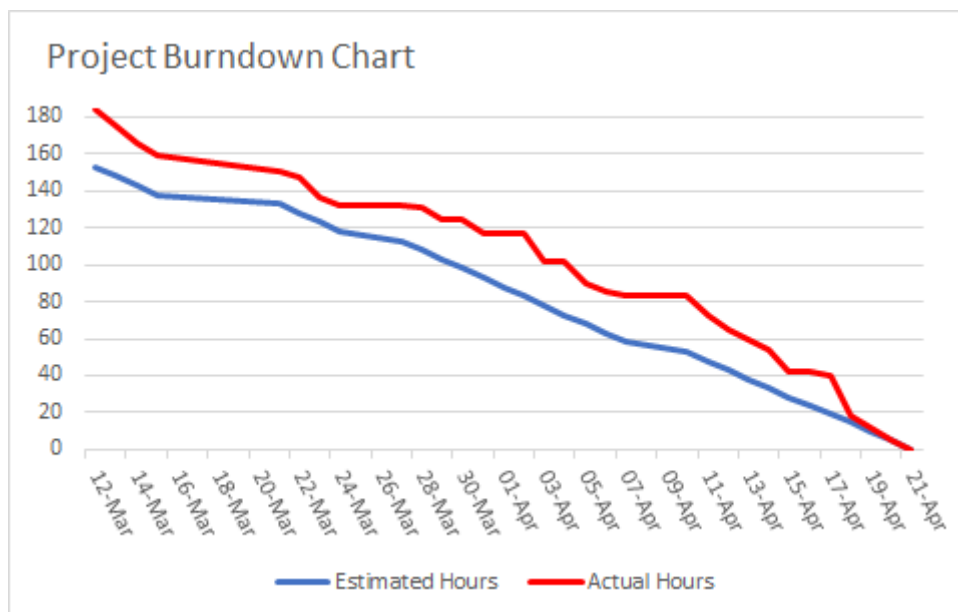
We also would have liked to improve our prediction model. We only had a limited amount of weather data and bike usage data that overlapped. To carry out this task, we used historical data supplied to us by the product owner and tried to match it up with our own bike usage data. As time was of the essence and our product owner wasn't very confident that the model would be of great use, we built a simple linear regression model which predicted an overall increase of 4% of bikes available at a station when it was raining. With more time we could have incorporated more overlapping bike and weather data to produce a more accurate model, which we could then have used to allow users to select a station and day and predict the number of bikes available based on the future weather prediction, rather than our basic model which simply shows a 4% increase in bikes available if rain is predicted.

Project Review

We all really enjoyed working on this project. It was 6 long weeks of hard work and we had some seriously frustrating times. However, at the end of it all we had a working application. It was a great experience as it was the first time that we had produced something that could actually be used. It was very satisfying to be able to implement an improvement to the Dublin Bikes service ourselves.

We also enjoyed using the Agile process. During the lectures it was hard to get an appreciation of how adapting this process can improve a project. We learned first-hand that following this process provides increased flexibility and cohesion. We talk about what we learned and our own experiences in the personal journal.

Below we present the burndown chart for the overall project. As can be seen, our recurring problem was a tendency to underestimate how long it would take to implement some of the features. Although we took longer than envisaged at the outset, we got there in the end.



Resources

Github Repository - https://github.com/oleathlc/Group20_COMP30670_Project.git

EC2 Instance Hosting App - <http://18.221.37.101:5000/>

Slack - <https://group20comp30670.slack.com/messages/C9KBCFLBX/>

Google Docs (Product Backlog, Sprint Backlogs) -

<https://docs.google.com/spreadsheets/d/1n0zkihRx9VL7nQH1jayKkCZNTQyZJdyp6hp4dZrIA64/edit?usp=sharing>

Appendix

- Product Backlog

PROJECT DETAILS						
FEATURE	STATUS	PRIORITY	START DATE	END DATE	ASSIGNEE	
SPRINT 1			12/3/18	23/3/18	Conor Lawlor	
Mine dynamic station occupancy data from JCDecaux	Complete	High	12/03/2018	13/03/2018	Emmet Tracey/ Daniel O'Byrne	
Data from JCDecaux should populate a csv file continuously	Complete	High	12/03/2018	13/03/2018	Emmet Tracey/ Daniel O'Byrne	
Push data from csv to AWS DB	In Progress	Medium	12/03/2018	13/03/2018	Emmet Tracey	
Create staticData table on AWS DB	Complete	High	13/03/2018	14/03/2018	Emmet Tracey	
Populate staticData table on AWS DB	Complete	High	14/03/2018	14/03/2018	Emmet Tracey/ Daniel O'Byrne	
Create dynamicData table on AWS DB	Complete	High	14/03/2018	14/03/2018	Emmet Tracey/ Daniel O'Byrne	
Populate dynamicData table on AWS DB	In Progress	High	14/03/2018		Emmet Tracey/ Daniel O'Byrne	
Create Flask application to create Google map template	Complete	High	12/03/2018	15/03/2018	Conor Lawlor/ Daniel O'Byrne	
Link flask application and SQL database to show bike stations on Google Map	Complete	High	21/03/2018	23/03/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Create link between EC2 and RDS	Complete	Low	13/03/2018	13/03/2018	Conor Lawlor	
Host project on EC2	Not Yet Started	Medium	n/a	n/a	Conor Lawlor	
SPRINT 2			27/03/2018	07/04/2018	Emmet Tracey	
Create a snapshot of the existing RDS	Complete	High	27/03/2018	29/03/2018	Conor Lawlor/ Emmet Tracey	
Restore snapshot on a new instance	Complete	High	27/03/2018	29/03/2018	Conor Lawlor/ Emmet Tracey	
Show station real-time occupancy when clicked. Get an infobox showing for each station (name, address, bikes available, stands available)	Complete	High	28/03/2018	30/03/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
---- Use scrape from API	Complete	High	30/03/2018	30/03/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
---- Pull information from RDS if API goes down	Not Yet Started	Medium	n/a	n/a		
Put in dropdown option for each station - User should be able to select a specific station and receive information on past trends and future predictions	In Progress	High	04/04/2018	04/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
---- Query DB and retrieve information in a format that can be transformed into a Google chart	In Progress	High	02/04/2018	06/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
---- Display chart showing weekly breakdown of data from RDS per station	In Progress	High	02/04/2018	02/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
---- Display chart showing daily breakdown of data from RDS per station	In Progress	High	02/04/2018	02/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
---- User should be able to select a station and enter a future date and time - will receive an estimate on availability using historical trends	Not Yet Started	Medium	n/a	n/a		
Using a user's location, app should provide a walking route to the selected station. Requires Google API	Not Yet Started	Medium	n/a	n/a		
Pull in weather forecast for Dublin	Not Yet Started	Low	n/a	n/a		
Station marker on map should show colour to indicate occupancy level	Not Yet Started	Low	n/a	n/a		
Change colour on markers to reflect occupancy	In Progress	Low	n/a	n/a	Conor Lawlor	
			10/4/2018	20/4/2018	Daniel O'Byrne	
Put in dropdown option for each station - User should be able to select a specific station and receive information on past trends	Complete (Started in Sprint 2)	High	10/04/2018	10/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
On clicking a button within the infoboxes on the map - query DB and retrieve information in a format that can be transformed into a Google chart	Complete (Started in Sprint 2)	High	10/04/2018	10/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Display chart showing weekly breakdown of data from RDS per station	Complete (Started in Sprint 2)	High	11/04/2018	11/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Display chart showing daily breakdown of data from RDS per station	Complete (Started in Sprint 2)	High	11/04/2018	11/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Create a formal plan of what we want the website to look like. We will then use CSS to apply this plan	Complete (Started in Sprint 2)	High	10/04/2018	10/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Use JS, CSS and HTML to carry out our design plan	Complete	High	10/04/2018	20/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Clean and comment all of our code	Complete	High	17/04/2018	20/04/2018	Conor Lawlor/ Daniel O'Byrne	
Create appropriate tests for our code	Complete	High	17/04/2018	20/04/2018	Emmet Tracey	
Host project on EC2	Completed	Medium	14/04/2018	14/04/2018	Conor Lawlor	
Pull information from RDS if API goes down	Complete	Medium	17/04/2018	17/04/2018	Conor Lawlor	
The dropdown should be alphabetically ordered so the user can easily find their required station	Complete (Started in Sprint 2)	Medium	11/04/2018	11/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Change colour on markers to reflect occupancy	Complete	Medium	12/04/2018	12/04/2018	Conor Lawlor	
Pull in live weather for Dublin	Complete	Medium	12/04/2018	13/04/2018	Conor Lawlor	
Populate a div with a 5 day forecast pulled from an API, so users can plan a trip	Complete	Medium	17/04/2018	19/04/2018	Conor Lawlor/ Emmet Tracey	
Create a prediction model that analyses bike occupancy in times of good or bad weather. We will then provide the user with an estimate of available bikes, based on our model	Complete	Medium	17/04/2018	19/04/2018	Conor Lawlor/ Emmet Tracey/ Daniel O'Byrne	
Using a user's location, app should provide a walking route to the selected station. Requires Google API	Not Yet Started	Low	n/a	n/a		

Stand-up Minutes

Minutes for our daily stand-ups are presented below in chronological order.

Sprint Planning Meeting Minutes (12/3/18 11:26am)

Discussed what we want to include in the first sprint. Product features are included in the Sprint 1 sheet on our shared Google Docs Sheet.

By the end of Sprint 1 we want to have:

- The scraper set up and running to get dynamic data on station occupancy
- The Google Maps API up and running with the station locations marked (using static data from DB)
- Flask app created using CookieCutter
- Project hosted on our EC2
- AWS Database set-up

We assigned priority to each feature for this sprint, assigned tasks to each team member, and we estimated the hours for each feature.

General Plans for the Project (12/3/18 12:12pm)

- Website should have a map with all stations marked on it.
- Markers should be colour-coded to show the current occupancy (e.g. Green = lots of free spaces, Red = very little free spaces)
- When a user clicks on the station marker they should see how many free spaces there are, the hourly trend for the day and the current weather
- There should be an option to look at trends for each day of the week (possibly show the weather for the chosen day?)
- There should include links to DB website and MetEireann(?)

Daily stand-up in person (13/3/18 10:07am)

- Conor set-up flask app with Google API. Map is centred on the Canal Cordon. Looked into connecting the RDS and EC2 but still need to get more info on this.
- Dan and Emmet worked on the scraper which adds DB data to a .csv file. Looked into converting .csv to RDS data and how to run the scraper with crontab.
- Aims for today is to get scraper running on EC2 and investigate how we get the .csv to the database

Daily stand-up in person (14/3/18 1.06pm)

- Conor managed to connect ec2 and rds after several hours of frustration. Ran the scraper written by Dan and Emmet on the ec2.
- Emmet and Dan created a scraper for the dynamic data which is working fine. Using nohup command it will run continuously on the ec2 instance.
- Started writing Python code to connect to the RDS.
- Emmet started writing sql code to get the static data into tables on the RDS.
- Today we aim to get the static data into the RDS, see how to get markers appear on the google map and tidy up the project tracking document

Daily stand-up via slack (15/3/18)

Daniel O'Byrne [9:40 AM]

- Yesterday myself and Emmet worked on writing the code to successfully connect to the database and to create and populate the static and dynamic tables on the ec2 rds. We ran into a number of issues largely relating to sql syntax. But a lot of googling and fiddling around, and we ran it successfully. We managed to that locally then when trying to get the dynamic data pushing to the ec2 rds, ran into trouble relating to modules imported and couldn't quite figure it out
- Today I'll be assisting Conor in looking at ways to get the static data working with the HTML in the flask app and possibly researching how the dynamic data will work with it

Conor [9:42 AM]

- Yesterday I spent more time looking at the GoogleMaps API for flask and how to place markers on it representing the stations. I also tried to figure out the issues with the dynamic scraper not working. I got that issue sorted last night so now our scraper is up and running printing to both the csv file and the RDS (Note: the guys did most of that work, I just helped debug the file)
- I am hoping to try get the flask reading from the staticData table in the RDS and place markers for each station in the table. (edited)
- We also need to flesh out our project task table (on Google Docs), and fill in our hours for the last two days

Emmet [9:48 AM]

- Spent yesterday working with Dan on getting tables set up on the RDS. Once we had the staticData table created we used the scraper to pull the static data and then push this onto the database. We then created the dynamicData table and tried to adapt our dynamic scraper to populate this table on RDS.
- We ran into two main problems yesterday. The first related to SQL syntax in Python programs. Took us a good while to work out what the exact syntax we needed to create tables. Eventually we were able to get it sorted. We also had issues with white space and apostrophes in the JSON file causing problems with populating the tables. Once we solved this problem (by encasing values with multiple words in single quotes and using .replace() to remove apostrophes) we had a program running on Eclipse that would scrape the JSON data every 5 minutes and populate the dynamicData table on RDS.
- The second problem was that the program we had written didn't work as it should once we moved it to the EC2 instance. Thankfully, Conor was able to solve this with a late night moment of genius
- Focus for today will be on looking into how our map will be populated by the data from RDS. Specifically, looking at how data is read from RDS
- Will also update our sprint backlog and provide some more information on the features

Conor [9:58 AM]

- Yesterday we also agreed to take tomorrow and the long weekend off as we are ahead of schedule. We may have a quick stand-up tomorrow just to record how we get on today, but we will not be working on the project again until Tuesday.
- I will also be updating the product owner (Karl) on our progress with the project and asking for any feedback he may have for us.

Daily stand-up via slack (16/3/18)

Daniel O'Byrne [1:03 PM]

- Yesterday I worked on trying to figure out how the flask app would interact with the tables on the database, first to get the static data. Myself and Conor thought that using php would be the best way to do this given we had done it before on a previous project. I tried to figure out how to use php over the instance. Xampp was how we did this originally but we tried to look at other ways. I didn't come to a conclusion after many hours of googling.

Conor [1:10 PM]

- As Dan has pointed out, we spent a lot of time trying to configure the Flask app and our RDS tables. We played around with SQLAlchemy in Python but it didn't work so we switched to trying PHP. We managed to pull info from the table using PHP, but this was only via XAMP. When I tried using Flask, they PHP would not work. From a bit of Googling, it seems that they may be incompatible. We decided to hold off on going any further until we spoke to the product owner (Karl), who will hopefully be able to help us out. Emmet is away in Paris this weekend so he will let us know how he got on yesterday when he gets back although he was also trying to help us figure out the issue. We will now break until Tuesday.

Daily stand-up via slack (20/4/18)

Conor [5:58 PM]

- As agreed, I didn't do any project work over the weekend. Today, the product owner (Karl) was in contact with me, and he informed me that he wants us to use Flask only and not PHP. With this in mind, the plan is to go back to trying to pull data from the RDS with our SQLAlchemy engine and try using jsonify (as suggested by Karl) to format that data for our HTML page associated with our Flask app. I will spend a few hours working on that this evening, and hopefully come tomorrow we will have made some progress on it.

Emmet [6:00 PM]

- I have returned safe and sound from Paris. Full to the brim of red wine
- I saw the messages from the product owner about not wanting to use PHP in the application. Will look at using flask only this evening and tomorrow morning. Try to get the map populated with the stations
- Not envisioning any problems for the time being but see how things go once I've had a look properly
- Hope you lads had a good weekend

- I was away since Thursday so haven't done any work on the project over the weekend but will be starting back at it tonight

Daniel O'Byrne [6:17 PM]

- I'm away in Mayo until tomorrow night so will hopefully pick up the progress then. Good to hear from Karl about the approach he wants us to take, given the struggles we encountered with php. Hopefully we will be able to easily apply jsonify to pull the data which should bring us close to completing our sprint one goals. Good job men!

Daily stand-up in person (22/3/18 11.09am)

- So yesterday, me and Emmet looked into getting the flask app working to push the data from our RDS to the HTML. I spent a few hours trying to work on the flask, but had to spend quite some time on getting mysqlclient added to my system. I only managed to get that sorted this morning. Emmet then had his own difficulties with flask but we managed to have them sorted by this morning. We are going to spend the next few hours trying trying to sort this out, and hopefully have it sorted for tomorrow

Daily stand-up in person (23/3/18 11.48am)

- Yesterday, the three of us spent the morning on the same task. We knew that the HTML could now read our data and we can use that data to make markers. We struggled however to get the jsonified data to work, so we left that out for the time being. We then had issues using a loop to get a marker for each station. After 4 hours, we decided to leave it. I emailed our product owner (Karl) today for some guidance, he got back but in the end I used a different resource and managed to get the markers showing last night. Today, we are going to work with the markers to try show the station name on each marker and maybe add in a few more things. Then we will have our Sprint 1 review and prepare for our Sprint 2 meeting with Karl on Tuesday

Sprint Review Notes (23/3/18 12.47pm)

- Overall we are pretty happy with how the first Sprint went. We achieved most of our targets, apart from getting the project on our EC2 instance, although that wasn't very high on our list of priorities, and it shouldn't be too hard to achieve in the next Sprint.
- With regards to the specific targets, we wanted to have our scraper set up and running and the map showing in our browser with a marker on the location of each bike stations, which we have managed to achieve
- For the scraper, we initially had our data being scraped into a CSV file on our EC2 instance, although we weren't going to be able to use this with our Flask app, so we had to use SQLAlchemy to get the data scraped into tables on our RDS instance. Dan and Emmet got this working, although we will need to figure out a way to get the data scraped into the CSV file (on our EC2) for the first 2/3 days into our SQL table on the RDS.
- For the map, we have the markers working, although not in the manner we may have initially envisaged. We spent some time trying to use PHP, but unfortunately this was

in vain, due it not being compatible with our project set up. We tried using various different methods, including jsonify, to get the data from our RDS to populate markers on our Flask app map. However, in the end we managed to do it without a specific plugin. This may be something we should check with our product owner in our meeting on Tuesday.

- With regards to the hours, it was very hard to estimate how long each feature would take. In the end, our burn down chart was actually quite good, although this may be down to luck. Hopefully we will be able to have a greater awareness of how long all other features should take for the next two Sprints.
- Sprint Review Complete
- We feel that we have a potentially shippable, rapid prototype to present to the product owner

Daily stand-up in person (26/03, 12:57pm)

- We met up in person for a quick discussion on our plan for Sprint 2. We are meeting with the product owner tomorrow morning to discuss how Sprint 1 went and see if he has any suggestions or requirements for Sprint 2

Daily stand-up in person (27/03, 12:47pm)

- We met up with the product owner today to discuss our progress. He seemed happy with how we are getting on and gave us some good advice and feedback
- The product owner also told us what he is expecting from Sprint 2 and we can use this as a backbone for our Sprint 2 Backlog
- During the meeting with the product owner we raised the issue we are having with RDS. It seems that our credits on RDS may run out and we are scared that we will lose our dynamic data. We decided to take a snapshot of the existing database and restore it on a new instance
- We are meeting this afternoon to have our Sprint planning meeting for the second sprint. We have a list of some of the features we want to implement during this sprint.
- We want to combine this with the product owner's requirements to produce a comprehensive list of features
- Dan has been working on fixing the code for the infobox windows. We had issues with the infobox not appearing for each individual station
- Emmet and Conor have been looking into restoring the database on a new instance
- Will post later with a brief description of our Sprint 2 planning meeting

Daily stand-up in person (28/03, 4:27pm)

- Conor and Emmet were able to move the project to a new EC2 instance to avoid the credit limit issue we were facing due to using AWS educate
- Dan was able to get the infobox to appear for each station
- We are having problems with the new instance. mysqldb isn't available on the Linux instance so we are going to launch a new Ubuntu one
- All 3 of use are going to work on getting the dynamic data feeding into the infobox

- If we have some more time we will look at starting the drop down for the additional detail per station
- Main focus is getting the new instance scraping properly and populating the RDS so we can keep our scraped data

Daily stand-up in person (29/03, 6:16pm)

- Yesterday we met for a few hours in the afternoon to work on our issue with the RDS
- We haven't received an email from AWS recently about our usage but we are still concerned that we may suddenly be locked out of the instance
- All three of us were working on restoring a snapshot of the database on a new instance. We originally restored it on a Linux instance but we couldn't get the scraper to write to the RDS. Had a problem with the mysqldb module and couldn't find a solution
- Decided to kill the Linux instance and launch a new Ubuntu instance. Conor and Dan are familiar with Ubuntu and had encountered this mysqldb module problem before
- Took us a good 3 hours to get the program running properly on the Ubuntu instance. We eventually were able to solve the SQL problem and we now have a backup instance running the scraper and saving to a second RDS. We are hoping to continue to use the original database but at least we now know that if we are locked out we have the data backed up and can switch to our reserve
- Conor did a lot of the grunt work for getting the scraper working on Ubuntu. Installed every package under the sun
- Dan was also working on the infoboxes. We have them popping up when each station is selected. Emmet and Dan then worked on incorporating dynamic data into the infoboxes. Spent a long time trying to determine the best way to retrieve and store the necessary data. Have a good idea now and will hopefully implement it tomorrow.
- Had a good meeting with the product owner day. He is happy with the progress so far and gave us an outline of what he is looking for by the end of Sprint 2
- Main focus should be on functionality - over the next few days we want to ensure the infobox appears for each station and displays live data. We also need to include the dropdown option for each station. Should be able to choose a station on a specific date and time and receive usage charts based off historical trends
- Product owner also raised a new, interesting feature to be implemented. User should be able to provide their location and then when they choose a station they will be provided with the walking route to it. Will need to incorporate the Google API. We will discuss this feature in more detail tomorrow morning
- Would like to take this opportunity to formally wish Daniel a (belated) happy birthday

Daily stand-up in Conor's house (30/03, 11:52am)

- We got locked out of our original instance yesterday so thankfully we had backed up our data to the new one. We are now using the new instance and monitoring the connection and usage daily
- Been working on the infoboxes this morning and have them pulling in for each station. We are currently using the API to provide the dynamic data
- After discussing with this approach with the product owner, we have decided that we should also pull the information from the RDS as a backup. If the API goes down, we will be able to pull the most recent data from the database and use this to populate the infoboxes
- Started looking into the charts for the historical trends. We are going to be working on getting the charts working for the rest of the day
- Meeting again on Monday. There will be no stand-ups on Saturday March 31st or Sunday April 1st

Daily stand-up in person (02/04, 11:05am)

- We weren't working on the project over the weekend
- We finished the infoboxes the last day and now have dynamic information pulling in for each station
- We also started looking into the charts to display the breakdown of historical trends
- The plan for today is to continue looking into the charts and hopefully be able to display some information requested by the user (daily/weekly usage)
- We are having some difficulty working out how to translate the data from the RDS to the charts

Daily stand-up in person (03/04, 12:43pm)

- We met in Conor's again yesterday and spent a good few hours working on the charts
- We were able to request data from the RDS and display it below the map using Google Charts. We have just hard coded the station and day for the time being
- During the practical we got some advice about how to adjust the code so that the charts will dynamically reflect what the user chooses. We are meeting tomorrow in the evening to try and implement this
- During the practical, we received more information about incorporating the weather into our model. We were also told about the predictive models we are expected to provide. We will discuss these more in the Sprint 3 planning meeting as for the moment we are still focused on functionality
- We spoke with the product owner today about some of the functionality and aesthetics of the app. We agreed that it would be best if there was a drop down that allowed the user to choose a station and day to display information for. We also decided that selecting one of the stations on the map should display the charts for that station
- We had some difficulties with reflecting dynamic data in the charts yesterday. Hopefully, we can make some more progress tomorrow afternoon

Daily stand-up in person (04/04, 7:23pm)

- We didn't work on the project yesterday after meeting with the product owner and having our stand-up so little to report
- We have been working on the charts again for the last few hours. Trying to get the charts to display for each station when chosen
- Still having some difficulty implementing this
- Meeting with the product owner again tomorrow morning

Daily stand-up in person (05/04, 5:01pm)

- Yesterday we spent 4 hours together in a study room trying to solve our problem with displaying the charts on the app
- Unfortunately we didn't make too much progress and are still having issues
- We have been able to add a 'more info' button to each pop-up window. When this is selected it calls a function to query the DB for the average use per day for that station. However, we can't get this information into a format that we can use to generate the Google charts
- We spent a long time trying lots of different approaches but didn't really get anywhere with it
- Met with the product owner again today to discuss how the app is coming along. He spent a long time helping us out but we still haven't solved the problem fully
- We are meeting again tomorrow and will discuss the problem together and see if we can devise any new approaches
- Been stuck on the same issue for a few days and I can tell that morale has fallen. Might bring the team out for a few pints to raise spirits

Daily stand-up in person (06/04, 8:40pm)

- Unfortunately we are still having problems with displaying the charts
- We have been able to use app.route to call a function from the website. This then queries the DB and retrieves the information for whatever station was requested. Our problem is with the format that the data comes back in. It seems to be a dictionary with only one key and a value made up of an array of arrays. No matter what we try we have not been able to find a way to access the data in this format in order to pass it into our drawChart function. We have spent about 10 hours each at this stage looking at different approaches and still haven't been able to solve it. We have tried retrieving the information a different way, using jsonify and pretty much everything else we can find on Stack Overflow but no joy
- During the stand-up today we tried implementing the code that Aonghus provided but we weren't able to get it to work. We may look at it again over the next few days
- We have another assignment due this Sunday so we aren't planning on spending too much time looking at the Dublin Bikes app over the weekend
- Meeting on Monday for our sprint review

Daily stand-up in person (09/04, 12:27pm)

- Our most pressing concern for Sprint 2 was that QwikLabs had been sending us emails that we were reaching the usage limits for our AWS Educate account
- We decided that it would be important to create a backup of our DB so that we could protect our dynamic data
- We took a snapshot of the existing DB and restored it on a new AWS instance. We ran into problems using this second instance as it was running Linux and we couldn't get SQL to work properly on it. We eventually ended up launching a third instance running Ubuntu. We were able to restore the DB on this instance and get SQL to work. A day after we had the backup instance running properly we were locked out of the original one so thankfully we had taken the steps to secure our data
- We then focused on getting the infoboxes working properly. We changed the infoboxes to be populated by JCDecaux as opposed to pulling from the StaticData table. This meant that the infoboxes were now displaying live data for each station about bikes, available and stands available. We were able to solve the issue with the infoboxes being in the wrong location by using the latitude/longitude from the API to place them
- We had changed the markers on our map to be little men on bikes instead of the default place markers Google provides. As much as we loved these little guys they were being placed nowhere near the actual station location. With a heavy heart we had to kill them all and replace them with the cold, lifeless Google markers once again. Like Icarus, they flew too close to the sun and ended up paying the ultimate price. The candle that burns twice as bright burns half as long. May God have mercy on their souls
- Swapping the instance was a bit of an inconvenience but after a few days we were able to get it up and running. Once we fixed the infoboxes we then focused on trying to implement the charts
- After some trial and error we were able to query the database and pull back specific information for each station. We could then use this to generate Google charts. Originally, we had hard coded each query
- Our biggest issue for this sprint has been with trying to amend the charts so that they reflect dynamic data. We have been unable to get the information to return in a format that we can use to generate the charts. No matter what approach we take the data always returns as dictionary with one key. The corresponding values consists of an array of arrays and we have been unable to find a way to access this data and pass it into the generate charts function. We have spent ~30 man hours trying to solve this problem during this sprint. It is extremely frustrating as we seem to be very close but can't work out the final step
- We have been able to use app.route so that when a user chooses to retrieve the weekly information for a specific station the DB is queried. The necessary information for that station is then returned

Daily stand-up in person(10/4, 10:12am) :

- As mentioned by Emmet in the Sprint 2 review, we spent roughly 30 man hours during that sprint trying to figure out a way to extract the relevant numerical data to populate the chart. This morning we finally had the breakthrough we had all been dreaming about. Against all odds we figured it out. We cracked our Enigma.
- The queries returned an array of dictionaries as objects. We then had to insert the key in order to return the value. These numerical values were then used as the data to create the chart.
- We then had our meeting with the Karl for the sprint planning. We created the plan of features we want to achieve for this sprint and are putting estimates on the number of hours required for each.
- We will work on the functionality of the drop down with the station names today. Each name should be clickable to show relevant charts, the same way the info boxes are selected.

Daily stand-up in person (11/4, 11:05am):

- Last night Conor and Emmet did work on getting the drop down menu working with the functions to create daily and hourly graphs. Emmet switched to using onchange rather than onclick which yielded successful results
- Today we will be looking at feature prioritisation as we have a considerable list of features we have talked about implementing.
- Later today and tomorrow we will start to work on the css and and general appearance of our page as thus far we have only worked on the apps functionality. Emmet is going to create a finalised drawing of what we want the page to look like and start to implement that design.

Daily stand-up in person (12/4, 10:55am)

- Today we met with Karl and discussed our progress and what our plan is for the coming week.
- He suggested using pandas to get create some sort of data analytics model in order to predict bike occupancy, as well as considering weather.
- In terms of predictions not considering weather, we had thought that because we had gathered data from the start of the project, our historical charts would work as an indication of bike availability. However, the Karl informed us that we needed to use this data to train a model which provides bike availability instead. We would then have to give some prediction of this availability at times of bad weather.
- Today Conor will work further on getting live data using jquery.
- Emmet will begin to look at creating a predictive model incorporating weather.
- I started to look at css to make the page look more attractive. I had issues for over an hour late last night as the css was not updating on the webpage. Emmet then told me the solution was to do with the caching and a hard refresh was needed. A bittersweet moment as I realised I had burned the midnight oil for no reason, as Emmet slept soundly. Today I will continue to work on the basic CSS properties for our app.

Daily stand-up in person (13/4, 1:21pm)

- This morning Conor and Emmet continued working on the css, and have made good strides in getting us closer to our target layout
- We have had to make the hourly chart load automatically to the current day, when a station is chosen. Otherwise when a new station is chosen, the hourly data from the previous station stays there.
- We feel this is a good work around, however it means that if a user chooses a station and a different day (then today), today's hourly chart will load first, then the day selected, which takes some time. This is a weakness we will have to think about further and can hopefully come up with a solution.
- For the rest of the day we will keep working on the css and the features we have prioritised. Incorporating the weather still remains an issue that we haven't quite decided how to deal with. At the moment we have a very simple prediction model but plan to look at a more useful model tomorrow.

Daily stand-up in person (14/4, 11:02am)

- Today we will continue to work on css features and hope to be close to completing our design plan
- We will work on getting loader images into the chart divs so that when a user chooses a station, a loading image/gif will appear until the charts come up
- We will also work on populating are other divs underneath the map. We plan on getting a twitter feed into the one on the left and the live info in between the two charts
- Conor will work on creating a legend for the map while myself and Emmet work on the loading images and populating the divs
- Conor will also look to host the project on ec2. Then we will be able to check how the site looks cross-platform e.g. on mobile
- We will park the weather prediction model for today as we feel we can get the design features fully implemented, and can work on weather prediction early next week.

Daily stand-up in person (16/4, 11:17am)

- After completing the majority of the design plan on Saturday, we are now satisfied regarding the functionality and appearance of the app. Thus, we are discussing what we need to prioritise in our last week of the project.
- We had previously discussed the idea of a heat map (for bike availability), when the user zoomed out a particular distance. However, we have decided to not go forward with this idea, as we feel at this stage there is enough going on and it's not a priority.
- We have scheduled to clean up our code in the coming days. At the moment there are a number of commented out attempts at implementing some extra features
- Today we will give final attempts to link the dropdown menu and info boxes as we could not figure it out on Saturday. Hopefully Karl can shed some light on this tomorrow, if we do not succeed.
- Today, we will begin the write up using the documentation we have built up over the last five weeks.

Daily stand-up in person(17/4, 10:50am)

- Yesterday we started the write up for the project and made good progress. As our daily stand - up notes have been pretty detailed, the documentation shouldn't need much more work, rather will just need to be formatted.
- This morning I revisited the problem we were having with linking the drop down menu with the info box below it. We have the info box on the map working with populating this information, but not the dropdown
- We couldn't figure it out for some time and Karl suggested to keep working on it, and we agreed given the importance of this feature
- Other than that Karl was pleased with where we were and how our documentation looked.
- For the rest of the day we will briefly look at the dropdown problem again. However, will set this aside if it doesn't work after 2 hours as we have testing and modelling to do
- Also for some reason 7am data is not working on any of the charts, we will hope to figure this out before submission

Daily stand-up in person(18/4, 12:35 am)

- Some may have thought yesterday's gruelling 12 hour shift in college, would have taken the wind out of our sails. But alas, we are as fired up as ever to continue our progress
- Yesterday, I began the modelling. It was going well until I tried to append a column onto the data frame and jupyter notebook crashed. I was trying to use our testData.csv however it had 220k rows which was far too big. This problem was encountered at the end of the night so decided to park it until today.
- Last night Emmet started the testing but had serious issues importing the app folder and we couldn't figure it out. We are going to wait to talk to Karl on Thursday about this
- Conor got the dropdown and info-boxes working perfectly last night. He also started trying to get the info boxes appearing on click of the dropdown but could quite get it.
- As mentioned in yesterday's stand-up, all of the data for 7am was not showing in the charts, but tday it has reappeared. We have no idea why, but we are delighted.
- Today Conor managed to keep only every 20th row of the testData.csv so we have roughly 11k rows in our model. Our model was very insignificant saying there are only 2.25% more bikes when it rains. Conor also started looking at robustness if the API is down but is finding it tricky so we will need to look at that further
- Today myself and Emmet both re-attacked the loader image problem. We wanted to have it so the loader image would re-appear every time the user chooses a new option on the dropdown or map. Emmet has made great progress and the solution is simpler than we both thought originally. We are over the moon. "HOORRAAAYYY" (Conor Lawlor)
- For the rest of the day we will clean up and comment all of our code as it is currently a bit all over the place. We also need to make some final touches on the layout.

Daily stand-up in person (19/4, 11:16am)

- Last night Conor added some comments to make the code clearer. I worked on some CSS and implemented CSS code for the dropdowns that Emmet had on file from another assignment, and we felt it worked well
- This morning we met with Karl and ran through final styling suggestions he had. He recommended some features we agreed would work well and plan to implement them for the day
- Emmet gave the tests another go however found difficulty in accessing the files from the GMAPviews.py. Jake also couldn't figure it out so we will have to reassess some last minute tests
- Today I will also add comments to the code adding to what Conor did, and add some of the CSS features
- Conor will also work on the CSS features that we agreed on
- Emmet will focus mostly on documentation for the day, working on the feature descriptions and functionality.

Daily standup in person (20/4, 10:34am)

- Last night Conor worked further on the appearance of the page experimenting with colours. We have been quite indecisive on this so far but have gone for a slick modern look.
- Last night I also experimented with geo location. However we decided that it was a fairly useless feature if we could not give a route to the nearest station for the user.
- Today we will put the final touches to our site and ensure we have the documentation up to date.
- Emmet is going to handle the footer and finalise it's appearance, as well as finalising the tests. Conor is going to ensure the format of the documentation is consistent and easy to read. I am going to comment on our Python code to explain to the user what each script is doing.
- It's a bittersweet moment as we close of our final standup. It's been a wild journey full of ups and downs.
- In the words of Richard Paul Evans, "It has been a mistake living my life in the past. One cannot ride a horse backwards and still hold its reins."