COMP30830: Software Engineering (Conversion)

Assignment 2

Dublin Bikes 2020

Final Report (Draft2)

Team Members:

Thomas Grogan
Kevin Mitchell
Conor Loughran

Contents: **Page**

Project Overview 4

Process 7

Data Analytics 22

Architecture 25

Design 28

Project Retrospective + Future Work 32

Overall Contribution 33

Meeting Logs 34

Standup Logs 42

Sprint Reviews and Retrospective Notes 56

Application Address: http://www.dublinbikes2020.com

Code: https://github.com/kevinmitch14/Software_Engineering

Project Overview:

Introduction:

Our application, simply named “Dublin Bikes 2020”, is available at the live address http://www.dublinbikes2020.com. The product is aimed primarily at existing customers who use the original Dublin Bikes website but find the user interface to contain redundant information. It is designed to be streamlined and to contain everything a customer needs in order to retrieve information about bicycle availability, and nothing they do not need. Customers should still refer to the original website dublinbikes.ie for more specific information and updates about the service, such as Covid-19 information updates and road safety guidelines.

The product was developed over the course of eight weeks, complying with Scrum methodology by consisting of a number of sprints, in this case four, with ten days (or two weeks) allocated to each sprint. The early stages involved scraping data that gives us weather forecast information and bicycle availability information, and storing it in a database from which the application could fetch information for the user. This was followed by the development of the user interface, and a Flask application that would direct some of this data to the user through a google map, resulting in the completion of a bare-bones version of the app at the halfway point of the project. The later stages involved cleaning up the user interface, optimising performance where possible and adding extra features that the user would find useful, such as a machine learning model that could predict how many bicycles would be available at a chosen bike station at a given time or day of the week.

This report explains the process from Tuesday the 11th of February (the date of our first project meeting) through the submission date of 19th April. Here, the justifications for including certain features, any issues that arose at any point of the project, and our documentation of our interactions related to the project will be explored. The architecture of the project, the data analytics and the design of the user interface will also be looked at in more detail. Finally, we will discuss the contributions of each group member, and any future improvements we would make to the project if we had more time to work on it.

Objectives:

The main objective of this application is to provide a streamlined user interface to a user that only requires a minimal number of features: most users of this application will want an immediate view of a google map and information about the number of available bikes or bike stands at their chosen station. The user interface would ideally integrate numerous features at the one online address, rather than the user having to visit a different URL to access a different feature provided by the application, as per

the original dublinbikes website. Therefore, we opted to provide a variety of features to the user all in the one webpage.

Target users:

This application is suitable for both regular users of Dublin Bikes and people using the service for the first time, but targets regular users more directly. Regular users want to get the most functionality out of the app with the least amount of effort and time, and this application targets these users by providing a streamlined service that offers as many features as possible without overcluttering the user interface. Due to the inclusion of certain features, such as giving the user directions to their chosen bike station, however, the app is also useful for tourists who may be familiar with similar bike-rental services provided by other cities, but who may not know their way around the city of Dublin.

What does the app do?

The app shows the user how many bikes or bike stands are available at the station of choice of the user. It shows the user an infowindow with the number of available bikes and the number of empty bikes at the station when they click on a station on the google map. The app can predict, to an acceptable degree of accuracy, how many bikes would be available at a given station at a time of day chosen by the user. It provides charts that will show the average number of available bikes at a chosen station, providing both daily and hourly data to the user. The app by default identifies the nearest station to the user, and at the click of a button can give the user directions to this station, or any station chosen from a dropdown menu. It also provides a basic weather forecast to the user. These features are all provided at one web address, so that the user does not have to travel to different URLs to access different features of the application.

Structure:

The structure of the user interface is basic but functional. There is a header that simply contains the title of the webpage “Dublin Bikes” with the google map situated on the left hand side taking up most of the page. There is a sidebar on the right hand side of the page and a footer, both of which show most of the information the page contains. The sidebar contains the dropdown menu with which the user can select a station and get directions to it. The real-time clock and weather data are also contained in the sidebar, as well as the dropdown menu from which the user can choose a time to predict the number of bikes that will be available at a given station. The footer area is where the graphs that show the hourly and daily bike data appear when a station is chosen from either the map or the dropdown menu.

Additional features

Most extra features included are modifications to the google map. These include a bike lane overlay on the google map which shows the user which streets have bicycle lanes on them, and a dark mode feature for the google map which darkens the map so that it is easier to view in the dark. The map markers are colour-coded so that the user knows immediately when they look at the map which stations have no bikes available, and which ones have a larger number of bikes available at any given time.

Process:

Here, we will go through each sprint one by one and explain our course of action, our justifications for features that we implemented, design decisions, issues that we encountered, how we conducted our standup meetings and our meetings with the product owner, Karl, and a burndown chart analysis for each sprint. For each sprint we have some retrospective thoughts on what went well during the sprint and what did not go well. Full logs of our standup meetings, full scale meetings and our review and retrospective notes are included at the end of this report.

Sprint 1

Course of action:

The first work for the project was carried out at the first group meeting. Weset up our shared GitHub repository and practiced pushing and pulling files to and from the repositoryfrom each of our machines. The trello board was set up and the first tasks for completion were added to the board. The tasks for completion in the first week of the sprint ranged from housekeeping tasks like setting up a Slack forum where any aspect of the project could be discussed remotely, to more hands-on tasks such as the setting up of the data scrapers that would retrieve bicycle data for each station (i.e. number of bikes available at each station in real time) from the JCDecaux API.

The second week of Sprint 1 was much more hands-on, involving setting up the scraper that would retrieve weather data, directing the data from each scraper to a database on RDS, developing the basic structure of the Flask application, and setting up the Google Maps API. More research on flask was required than expected, so while we carried out some of this research during sprint 1, we decided to extend this task to Sprint 2 so more research could be done. Generally, the course of action for this sprint was standard regarding the design of the project and the choices made; we took guidance from Karl and didn't take any drastic decisions or make any design choices that would stand out at this point.

Justifications and Design decisions:

There were three data scrapers set up in Sprint 1. The first was a simple static scraper that would retrieve data from each bicycle station that did not change over the course of time, such as the name and coordinates of a specific station. The second was a dynamic scraper set up to retrieve data from each station that did change, i.e. the number of bikes at a given station at any time. This was set up to retrieve data every five minutes using cron, a time-based job scheduler; originally this was done using nohup but we found cron to be more reliable. We assess all of the scraped data through MySQL Workbench as we all have experience using this particular program from the Relational Databases module last term (COMP20240).

The third was another dynamic scraper, this time to retrieve data from the Dark Sky weather application. We had been using Dark Sky for our weather information since the first sprint, long before the takeover bid by Apple at the end of March; by this point it was too late to switch to a different API for scraping weather information, but DarkSky's API will be available to use until the end of 2021, even though their android applications are being shut down in July of this year.

Issues:

The issues faced at the beginning of this project were largely trivial and due to our lack of experience working with the new technologies. There was an initial issue with timing the dynamic bike scraper to collect data every five minutes that was solved by using cron, as mentioned above.

Documentation:

As this was the first sprint and we did not have a complete plan for the entire sprint, we added to the board the first tasks we knew had to be completed, before adding more tasks to the board the following week. As seen in the below screenshots, the complete template for how tasks would be added and presented on the trello board was not yet finalised; this template was brought up to scratch by the end of Sprint 1 and then used for the remaining Sprints. The Burndown chart and Backlog table design was completed in the second week, and populated with data that covered the entire sprint.

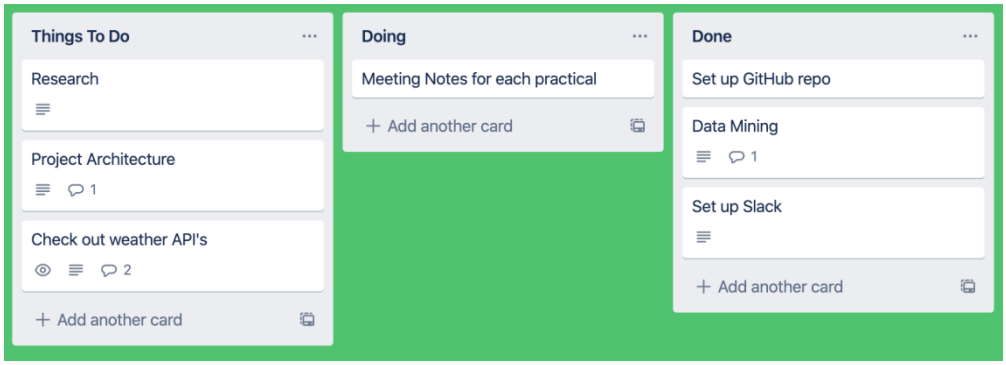


Fig 1: Trello Screenshot from Sprint 1, Day 2

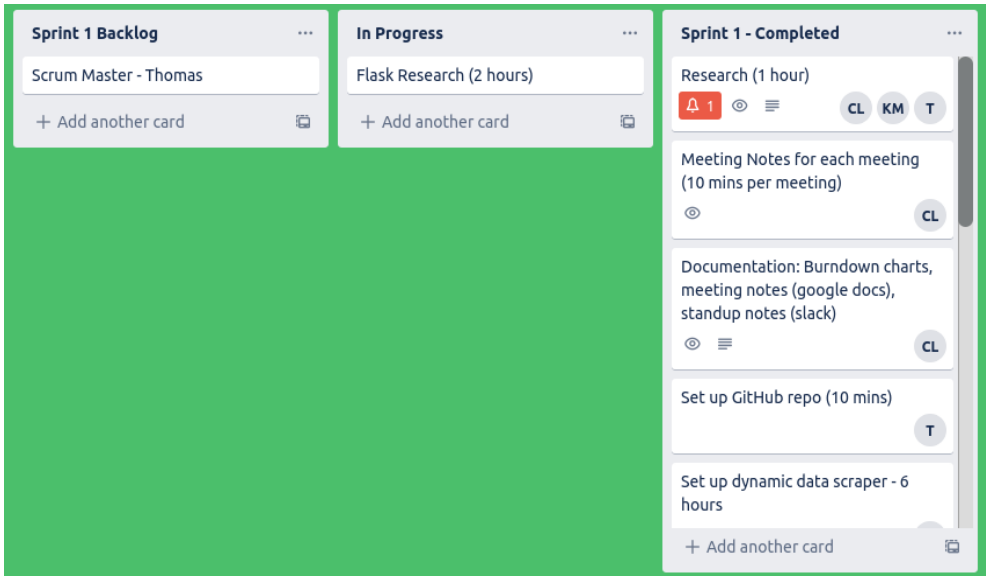


Fig 2: Trello Screenshot from Sprint 1, Day 10

Meetings and Standups:

There were no standup meetings for the first week of this sprint; we communicated though Slack to discuss any issues we faced and updated the trello board as each task was completed. We found that having the daily standups each day for the remainder of the sprint was extemely helpful and we

were better able to put the entire sprint into perspective in terms of keeping on schedule with our work.

Full meetings for the project are considered as those in which all members are present during the practical sessions which involve input from the product owner, Karl. There were four large-scale meetings during this sprint involving Karl (all during practical time) and one meeting amongst ourselves on Day 6 of the sprint to make up for lost ground from the lack of standup meetings for the first five days of the sprint.

Burndown Chart and Analysis:

The format for each Burndown chart, as defined below, is the same for all four sprints:

The ***Burndown Chart*** section lists each task and the amount of hours it should take to complete each task. Whether or not a task is completed in the amount of hours we predicted it to take does not apply here; this chart simply tells us the day that each task was worked on so that the actual remaining work can be plotted against the estimated remaining work.

The ***Backlog*** section compares the predicted hours per task to the actual hours per task, and has a value Y for yes or N for no based on whether or not the task has been completed.

Sprint 1

Burndown Chart

Task	Predicted Hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Time Remaining
Rudimentary Research	1	0	1	0	0	0	0	0	0	0	0	0
Meeting/Standup Notes	3	0	0	0	1	0	0.5	0.5	0.25	0.25	0.5	0
Github Setup	0.167	0	0.167	0	0	0	0	0	0	0	0	0
Slack Setup	0.167	0	0	0	0.167	0	0	0	0	0	0	0
Trello Setup	0.167	0	0.167	0	0	0	0	0	0	0	0	0
Weather API Research	1	0	0	0	0	0	1	0	0	0	0	0
Burndown Charts+Docs	4	0	0	0	0	0	0	0	1.5	0.5	2	0
Flask Setup (Basic)	4	0	0	0	0	0	0	4	0	0	0	0
Data Scraper (Bikes)	4	0	0	0	1	1	0	2	0	0	0	0
Data Scraper (Weather)	4	0	0	0	0	0	0	4	0	0	0	0
Google Maps API Research	0.5	0	0	0	0	0	0	0	0.5	0	0	0
Google Maps API Setup	2	0	0	0	0	0	0	0	2	0	0	0
Project Architecture	2	0	0	0	0	0	0.5	0	0	1.5	0	0
Google Maps setup in Flask app	2	0	0	0	0	0	0	0	0	0	1	1
Flask Research	2	0	0	0	0	0	0	0	0	1	1	0
Actual Hours Remaining	30	30	28.667	28.667	26.5	25.5	23.5	13	8.75	5.5	1	
Estimated Hours Remaining	30	27	24	21	18	15	12	9	6	3	0	

Backlog

Task	Complete	Predict	Actual
Meeting/Standup Notes	Y	3	3
Github Setup	Y	0.1667	0.1667
Slack Setup	Y	0.1667	0.1667
Trello Setup	Y	0.1667	0.1667
Weather API Research	Y	1	0.5
Burndown Charts+Docs	Y	4	4
Flask Setup (Basic)	Y	4	1
Data Scraper (Bikes)	Y	4	6
Data Scraper (Weather)	Y	4	4
Google Maps API Research	Y	0.5	0.5
Google Maps API Setup	Y	2	2
Project Architecture	Y	2	2
Google Maps setup in Flask app	N	2	2
Flask Research	Y	2	2
Rudimentary Research	Y	1	1
Total Hours	N	30	28

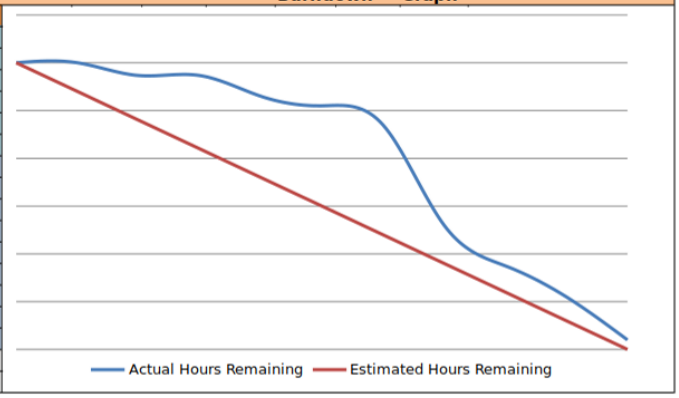


Fig 3: Burndown Graph and Backlog for Sprint 1.

We were well behind on our schedule for all of the first week: lack of daily standups and overall lack of direction in the project caused us to fall behind at an early stage. On average, the predictions for how many hours we should spend on the tasks was reasonably accurate, though some tasks took a little longer or shorter than expected. During the first week of the sprint we had a rough idea that we were going to allocate 30 hours worth of work in total to the sprint, but we did not know yet

which way those hours would be divided up. Although we mostly caught up towards the end of the sprint, we did not have the Google Maps setup in Flask application completed, as we did not succeed in getting the static information for the google map showing in the flask application by the time we had our retrospective. This was completed over the weekend between Sprint 1 and Sprint 2 so as to not eat up valuable time from the next sprint.

Sprint 1 Retrospective:

1. What went well?
Our organisation in the second week: we brought the burndown rate back to a somewhat acceptable level through better organisation and more hours contributed to the project overall. By learning to use Trello and Slack properly, we got better at organising our workload. We had our Github repository working properly; everybody was comfortable with pushing and pulling files to and from the repository. The meetings with Karl in the first week helped us get started on the project, and those of second week helped us with project management and set us up for meeting deadlines.
We completed the following during the first sprint:
- Set up data scrapers to collect data at fixed intervals (Bike (dynamic), Bike (static), Weather (dynamic))
- Set up the databases for each data scraper on RDS instance
- Scheduled dynamic scrapers with chrontab to update tables: every 5 mins for the bike dynamic scraper) and every hour for the weather scraper
- Set up basic flask application, integrated Google maps API into Flask
- Created markers with static data for the Google map
- Defined rough idea of project architecture and a mockup of the user interface

2. What didn’t go well?
Some of our predictions regarding how long certain tasks should take were off the mark; some tasks took longer than required to complete (i.e. the dynamic data scraper for bicycle data), and some took a much shorted time than predicted (i.e. the basic Flask application setup). Organisation and coherent teamwork during the first week was poor: we met at practical times but did not conduct daily standups. This contributed to a delay in work and falling behind on our schedule, resulting in the non-completion of the following tasks:
- Getting the static information for the google map showing in the Flask application
- Sufficient Flask research: we carried out the two allocated hours of research, but more was required and had to be added to the tasks for the next sprint.

Sprint 2

Course of action:

In our first meeting with Karl we discussed all of the steps to be carried out during Sprint 2 so we could plan the timing and tasks to be implemented for the entire Sprint during the first meeting. By the end of this sprint we were expected to have a bare-bones implementation of the application with basic functionality. During the first week we brushed up on our ability to use Flask with further research (a task extended from Sprint 1) and were then able to retrieve information from the database through the Flask application. A mockup of the desired front-end interface was designed. For inspiration on how to design the user interface, we researched a few services similar to that of DublinBikes offered by cities such as London, Barcelona and Los Angeles, and in quick succession

to this the bare-bones html page for the front-end was created. A user-flow diagram was attempted at this point but was not completed as the application design itself was not completed: this was pushed back to Sprint 3 as a result.

The tasks during week 2 pushed us towards having a bare-bones working version of the application: we created popup windows for each marker which included information about the number of bikes at a given station. A dropdown menu of stations was created so the user had a choice of choosing a station from the dropdown menu or clicking on a map marker; the on-click event triggers a call to the database to return information about the number of available bikes to the user interface. Weather data from the database was now displayed on the user interface via the Flask application, and towards the end of the sprint work began on a geo-location feature, which would ideally be able to identify the nearest station to the user.

We decided towards the end of Sprint 2 to extend the sprint into the study break in order to get a head start on Sprint 3, but due to circumstances outside of our control we did not complete all of these extra tasks, and instead they were pushed back to Sprint 3 which began after the two-week study period. One task which was completed was the addition of colour-coded map markers, which would give the user an estimate on the percentage of total bikes available at each station without needing to click on the map marker.

Justifications and Design decisions:

We tried to make the user interface streamlined and easy to use for a regular user who would need to know instantly information about the weather forecast and how many bikes are available at their station of choice, but that would also cater for users who may not have been sure of their location (i.e. tourists) and could be guided to their nearest station through the geo-location feature.

Initially, we had decided to get the current bike occupancy figures from the JCDecaux API each time the user chooses a station. We decided instead, however, to obtain the data from the most recent entries of the database after discussing our approach with the product owner; there was a worry that users of the application could choose to view the figures for multiple stations in quick succession, leading to a large number of requests to the API in a short amount of time, which could possibly lead to us being banned from accessing the API.

Issues:

Trivial issues:
Initially there were teething issues with Flask, but through research and online tutorials it became easier to work with. This was part of a bigger issue involving a lack of knowledge on how various parts of the entire architecture communicate with each other. This was solved over time with extra research done outside of allocated hours for the sprint.

Data issues:
We found that the most effective way to pass data from Flask to the front-end was as a JSON object, which once again involved further analysis and teething issues at the beginning as we learnt to decipher the JSON information properly, but it was something that we got used to quickly.

Code issues:
A small number of map markers were not appearing when the app was run; we found that this was due to a bug in the SQL query that retrieved station information from the database. The problem

was that we were ordering the results of the query by the most recent update time. Various stations were not updated for longer periods of time in the JCDecaux API, and therefore some rows were being returned which contained data for the same stations. It took some time to find an SQL query to fix this; the new query involved using nested SELECT statements, which was not ideal as it would really slow down the application.

Scraper issue:
On inspection in Workbench we found that the dynamic weather scraper was not populating the table every single hour. We discovered that this was due to an issue with one of the data fields from the API which contained a null value for the rain-type column which tells you, when it is raining, whether the rain is heavy or light. Whenever it wasn't raining, the API returned a null value for the entire row, rather than just for the feature of the rain-type column, meaning we were not retrieving any data at all when it was not raining. The issue was identified through running the responsible script manually outside of the application itself, and to resolve it we had to remove the the rain-type column entirely so that the only information that would be collected from that point on would be whether there is rain or not, with no information about whether the rain is heavy or light.

Documentation:

From the very beginning of this sprint we knew all the tasks that would have to be completed and added these to the backlog on Trello during the first meeting. As seen in the below screenshots, we had further developed our methodology for getting the most out of our Trello board, with information on predicted and actual hours taken to do certain tasks appearing here as well as the actual Backlog table, which we will look at soon. The Burndown chart was modified every couple of days with the latest information, before being pushed to GitHub along with the rest of the draft documentation at the end of the Sprint.

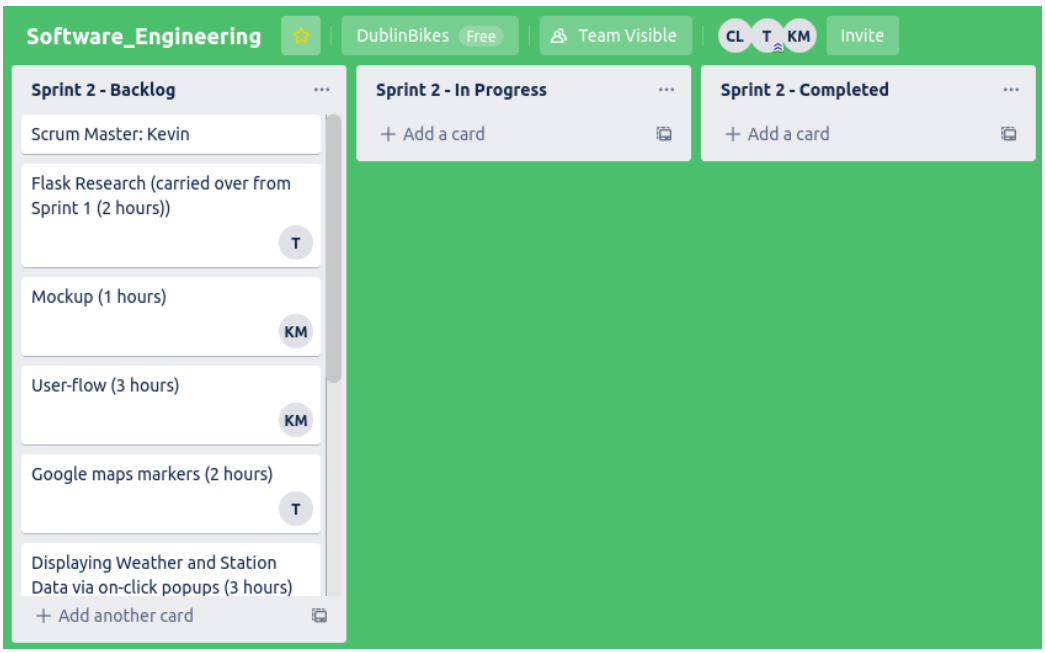


Fig 4: Trello Screenshot from Sprint 2, Day 2

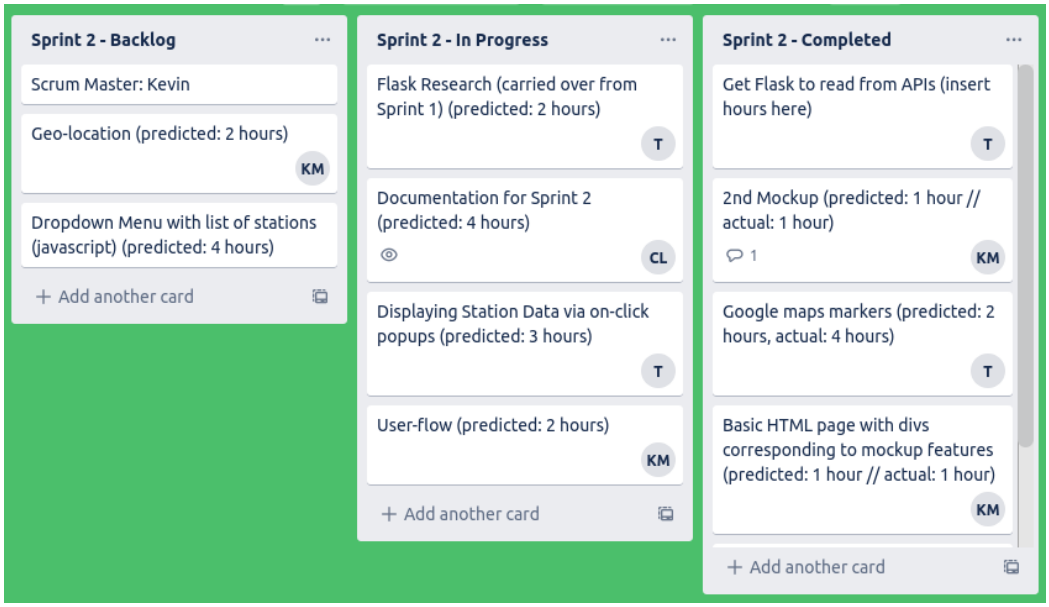


Fig 5: Trello Screenshot from Sprint 2, Day 7

Meetings and Standups:

Beginning on day 2 of the sprint we held a standup meeting every day of thesprint: these were held, when possible, in person during a full-scale meeting during the practical slots on Tuesday or Thursday mornings (the location of every individual standup is documented in the standup meeting logs at the end of this report). On days that the group did not meet in person, we carried out the standup online via Slack. Full meetings for the project (of which there were 4 during this sprint) were all held in person during the Tuesday and Thursday morning practical slots.

Burndown Chart and Analysis:

Sprint 2

Burndown Chart

	Predicted Hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Time Remaining
Flask Research	2	0	0	1	0	0	0	0	1	0	0	0
Google Map Markers Functionality	2	0	0	0	1	1	0	0	0	0	0	0
Google Map Marker Colours	1	0	0	0	0	0	0	0	0	0	1	0
2nd Mockup (Front-end interface)	1	0	1	0	0	2	0	0	0	0	0	0
Popup windows w/ Station Data	3	0	0	0	0	0	3	0	0	0	0	0
Basic Geo-location feature	2	0	0	0	0	0	0	0	0	1	1	0
Dropdown Menu with all stations	4	0	0	0	0	0	2	2	0	0	0	0
Documentation Sprint 2	4	0	0	0.5	1	1	0	0.5	0	0.5	0.5	0
Research on Similar Services	1	0	1	0	0	0	0	0	0	0	0	0
Have Flask read from APIs	2	0	2	0	0	0	0	0	0	0	0	0
Display JSON data from API on page	2	0	0	2	0	0	0	0	0	0	0	0
Basic HTML page	1	0	0	1	0	0	0	0	0	0	0	0
Display Scraped Weather Data	1	0	0	0	0	0	0	1	0	0	0	0
Actual Hours Remaining	26	26	22	17.5	15.5	13.5	8.5	5	4	2.5	0	
Estimated Hours Remaining	26	23.4	20.8	18.2	15.6	13	11.4	7.8	5.2	2.6	0	

Backlog

Task	Complete	Predict	Actual
Flask Research	Y	2	2
Google Map Markers Functionality	Y	2	4
Google Map Marker Colours	Y	1	1
2nd Mockup (Front-end interface)	Y	1	1
Popup windows w/ Station Data	Y	3	5
Basic Geo-location feature	Y	2	3
Dropdown Menu with all stations	Y	4	2
Documentation Sprint 2	Y	4	4
Research on Similar Services	Y	1	1
Have Flask read from APIs	Y	2	2
Display JSON data from API on page	Y	2	2
Basic HTML page	Y	1	1
Display Scraped Weather Data	Y	1	1
Total Hours	Y	26	29

Burndown Graph

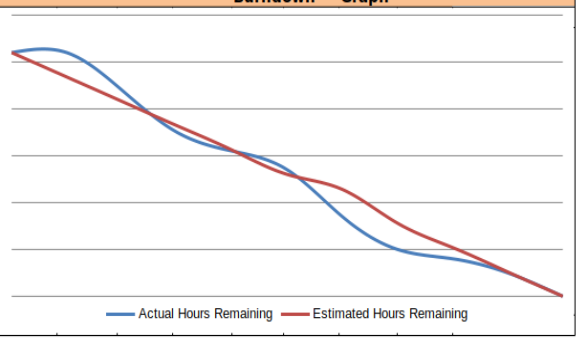


Fig 3: Burndown Graph and Backlog for Sprint 2.

We were behind on the first day due to the first meeting and sprint planning taking place on Day 2 of the Sprint during practical time. On average, the prediction for how many hours to spend on the project altogether was accurate for most tasks, but underestimated for others (it took 29 hours to complete tasks to which we had allocated 26 hours). During the midterm break, we were going to add further features to the application as part of an extension to Sprint 2, but these tasks were not completed (expect for adding the colours to the location markers based on bike availability, which appears in the above burndown chart). The remaining features were instead pushed back to Sprint 3.

Sprint 2 Retrospective:

1. What went well?
Organisation for the entirety of the sprint: we planned our time well for the most part and stayed on track with the product owner's expectations. We tried to do as much as possible early in the sprint so that we were under less pressure during the final days of the sprint.
Extra work beyond the end of the sprint led to extra work being completed on the geo-location feature and the addition of a brand new feature (the colour coding of the station markers based on bike availability).
We completed the following during the second sprint:
- Completed our research on how to use Flask
- Completed the second draft of a mockup of the desired user interface
- Created a basic HTML page with divs corresponding to the features contained in the mockup
- Successfully displayed JSON data from API on webpage
- Created Google map markers for each bike station
- Had the station data displayed through a pop-up info window triggered by an on-click event
- Created a dropdown menu with a list of all stations
- Created an information div that displays information for the station selected in the dropdown menu (separate to the onclick event for stations chosen via map markers)
- Added colours to the map markers based on bike availability (Red = low availability, Orange = medium availability, Green = high availability)
- Improved design and styling of data in the info windows for each marker.

2. What didn't go well?
Our predictions for how long certain tasks should take was understimated, meaning more hours were required during a sprint in which many external factors (i.e. pre-study break assignments and examinations in other modules) added hugely to our overall workload.
The extension of the sprint into the midterm break, while leading to the addition of one new feature (colour-coded map markers) and the development of a pre-existing feature (geo-location), was a failure overall, mostly due to circumstances outside of our control, but also partly due to a lack of clarity of purpose and less communication between team members as each of us had returned home from UCD for the midterm break. This failure led to the non-completion of the following tasks:
- Did not start any aspect of the data analytics
- Did not add icons to the weather data

The extension of Sprint 2 was a low-risk venture, so its failure did not affect the future of the project as these tasks could easily be pushed back to Sprint 3.

Sprint 3

Course of action:

Sprint 3 involved pushing the boat out in terms of what our application could offer. By this point we had an extremely basic working application, but we needed to improve the user interface to make the application more user-friendly, and add extra features to make the application more useful in general. In the first group meeting Karl emphasised the importance of Data Analytics and Visualisations as the focal points of this sprint.

In terms of features, the first week of the sprint involved creating charts that would show visualisations of the average daily amount of bikes available at any bike station, and then one to show the average hourly amount of bikes available. The geolocation feature was given further functionality so that the user could now be given directions to their nearest station. Weather icons from Dark Sky were added to the user interface, and the weather data display in the sidebar was cleaned up. We completed the user-flow diagram that was pushed forward from Sprint 2, now that we had a vision of what the fully working version of the application would look like.

During the second week week we got to work on the Data Analytics part of the project: for this sprint this involved mostly research. Further work was done on the geo-location feature so that the user could now get directions to any station chosen from the dropdown menu.

Justifications and Design decisions:

Our justification for including the optional geo-location feature, and developing it to the extent that we did, is that it will help users who are not familiar with Dublin City to find any bicycle station by name without major difficulty, continuing with our plan to make the application as useful as possible for both locals and tourists.
We also changed the colour-coded map markers design methodology. Originally, the red coloured stations presented that 0-25% bikes were available, orange meant that 25-50% of bikes were available and green showed that over 50% of bikes were available. We decided to change this because the colour red gives the impression that there are zero bikes available, and the user would have to click on the station to find out whether there were any available bikes or not. This was changed so that the colour red represents stations that have zero bikes, orange would be for stations with 1-25% available bikes, and stations with over 25% available bikes would be shown in green.

Issues:

Chart issues:
Thecode for the chart displaying hourly data contains nested loops; this would inevitably impact loading times in a negative way. An improvement to chart loading speeds was implemented by carrying out indexing in the relevant tables in the database, which allowed the SQL query fetching the information to work much faster than before.

Data Analytics:
The research into the Data Analytics side of the project took longer than expected as we had no experience with developing a machine learning model: we had only just started covering this in our Data Analytics module (COMP47350) so we were pretty much starting from scratch.

Time constraints associated with the above issues:
We were unable to find a solution other than the nested loops in the code for the charts displaying hourly data, and because of the time taken to carry out the Data Analytics research we did not get the model completed by the end of the sprint. This work was pushed forwards into the final sprint.

Documentation:

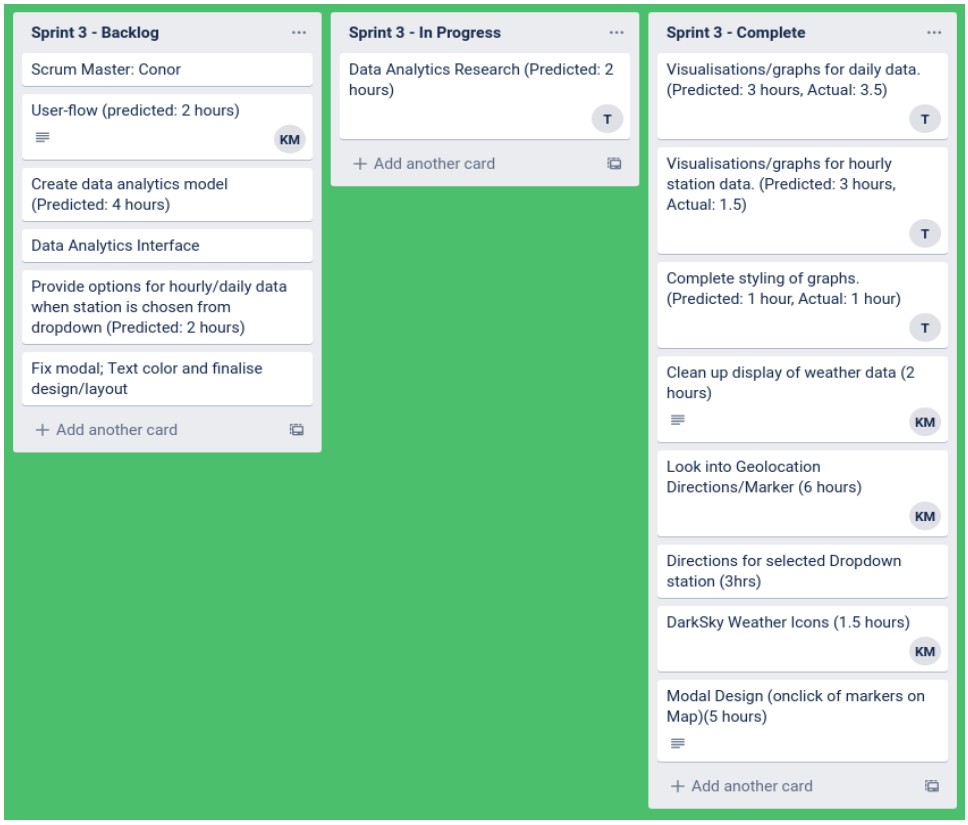


Fig 7: Trello Screenshot from Sprint 3, Day 5

As per sprint 2, the entire backlog was created at the beginning of the sprint (at the first meeting) as we knew all of the tasks we wanted to have completed by the end of the sprint.

Meetings and Standups:

Full meetings for this sprint were held remotely on Google Hangouts during practical times on Tuesday and Thursday mornings. There were only three meetings during this sprint: we decided not to carry out the fourth meeting as we only had tasks relating to data analytics remaining for this sprint by the time the meeting on day 9 of the sprint came round.

Beginning on day 2 of the sprint we held standup meetings up until day 8 of the sprint. These were either carried out on Slack, or on Google Hangouts after our meeting with Karl had ended. We did not carry out sprint meetings towards the end of the sprint due to work for other modules taking centre stage; apart from the data analytics, very little work was organised for the end of sprint 3.

Burndown Chart and Analysis:

Sprint 2

Burndown Chart

Task	Predicted Hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Time Remaining
Graphs for daily bike data	3	0	1	2	0	0	0	0	0	0	0	0
Graphs for hourly bike data	3	0	0	0	2	1	0	0	0	0	0	0
Graph design/styling	1	0	0	0	0	1	0	0	0	0	0	0
Clean up weather data display (CSS)	2	0	0	0	0	2	0	0	0	0	0	0
Geo-location marker + directions to nearest station	4	0	2	0	2	0	0	0	0	0	0	0
Directions to nearest station on dropdown	3	0	0	0	0	0	3	0	0	0	0	0
DarkSky weather icons	1	0	0	1	0	0	0	0	0	0	0	0
Modal design (onclick map markers)	5	0	0	0	5	0	0	0	0	0	0	0
Data Analytics Research	2	0	0	0	0	0	0.5	0.5	0	0.5	0.5	0
Data Analytics Interface	3	0	0	0	0	0	0	0	0	0	0	3
User-flow (carried over from Sprint 2)	2	0	0	2	0	0	0	0	0	0	0	0
Data Analytics Model	4	0	0	0	0	0	0	0	0	0	0	4
Option for displaying hourly/daily bike data	2	0	0	0	0	0	0	0	0	0	0	2
Finalise design of modal	1	0	0	0	0	0	0	0	0	0	0	1
Documentation Sprint 3	4	0	0.5	0.5	1	0	0.5	1	0	0	0.5	0
Actual Hours Remaining	40	40	36.5	31	21	17	13	11.5	11.5	11	10	
Estimated Hours Remaining	40	36	32	28	24	20	16	12	8	4	0	

Backlog

Task	Complete	Predict	Actual
Graphs for daily bike data	Y	3	3.5
Graphs for hourly bike data	Y	3	1.5
Graph design/styling	Y	1	1
Clean up weather data display (CSS)	Y	2	2
Geo-location marker + directions to nearest station	Y	4	6
Directions to nearest station on dropdown	Y	3	3
DarkSky weather icons	Y	1	1.5
Modal design (onclick map markers)	Y	5	6
Data Analytics Research	Y	2	4
Data Analytics Interface	N	3	0
User-flow (carried over from Sprint 2)	Y	2	2
Data Analytics Model	N	4	0
Option for displaying hourly/daily bike data	N	2	0
Finalise design of modal	N	1	0
Documentation Sprint 3	Y	4	6
Total Hours		40	36.5

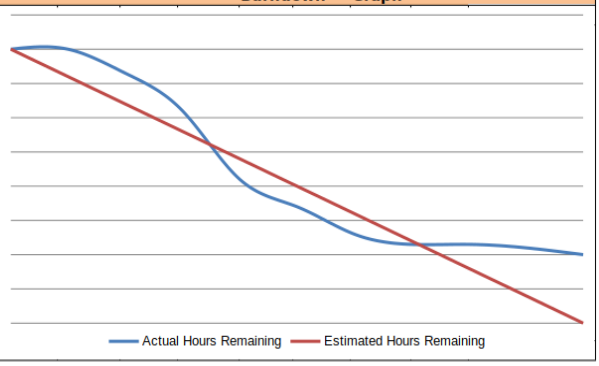


Fig 3: Burndown Graph and Backlog for Sprint 3.

We did not complete all of the planned tasks on time in this sprint. This was an ambitious sprint as we had allocated 40 hours worth of tasks to be completed, but we only completed 75% of this work, with about 10 hours' worth of work having to be pushed forward to the final sprint. There were a number of reasons for this failure to keep schedule:
- a lack of overall project organisation during the second week of the sprint
- it took us longer than envisaged to carry out the tasks that we did complete (it took us 36.5 hours to do 30 hours' worth of work, as seen in the backlog chart
- the data analytics work in this sprint was unsuccessful.

Sprint 3 Retrospective

1. What went well?

Organisation during thefirst week of the sprint: Despite having delegated more hours of work than usual to this sprint (40 hours, rather than 25-30 for the first two sprints) we were well ahead of schedule by the middle of the sprint (end of day 5).As with previous sprints, wetried to do as much as possible early in the sprint so that we would be under less pressure during the final days of the sprint.

We managed to get a lot of work done relative to previous sprints as we had carried out tasks in Sprint 3 that were that required much more effort as they were much more complex than before.

We completed the following during the third sprint:
- Completed work on geolocation – user can get directions to their nearest bike station, or any of all the bike station from their own location
- Graphs depicting both daily and hourly data showing how many bikes are at a station
- Styling of graphs
- DarkSky weather icons in user interface

- Design of onclick map markers
- Clean up of display of weather data in sidebar
- User-flow diagram for application with full functionality
- Research into Data Analytics section of project

2. What didn't go well?
As with all of the previous sprints, predictions on how long it would take to do certain tasks were underestimated: we did 37 hours of work when 40 hours were allocated, but only completed 30 hours worth of tasks, accounting for just 75% of the allocated workload. Our overall organisation of tasks on Trello was not up to the same standard as it had been in the previous sprints: the board was set up appropriately and modified up to halfway through the sprint, but there was little or no activity on the trello board during the second half of the sprint. As well as all of this, the lack of communication between team members at the end of the sprint caused by external factors meant that we would start off sprint 4 with a little bit more work than we expected.

The above issues caused us to fail to finish the following tasks for the end of sprint 3:
- Data Analytics work: no model completed
- Finalisation of text colours and text layout for onclick modal
- Dropdown menus to allow the user to pick a day and time to make a prediction (data analytics interface)

Sprint 4:

Course of action:

Sprint 4 should have only involved adding the finishing touches to the project; for us it meant both doing this and implementing the data analytics side of the project. In our first sprint 4 meeting with Karl we discussed the steps to be taken for the completion of the data analytics, and then the main objectives of the sprint were outlined: we had to deploy the application to EC2 and make the application look and feel like a finished product. We also discussed a few optional tasks such as trying to optimise peformance and separating our code by language, i.e. keeping javascript and css in separate files, and then talked about a couple of extra features that we found interesting, like bike lane overlays and dark mode on the google map, as well as an SSL cert for the geolocation feature, which would be required to access the user's location once the app was deployed to EC2.
In the early days of the sprint we completed the data analytics section of the project. Following this we started on some of the tasks discussed during the meeting; we tried to clean up the user interface as much as possible, and began cleaning up our code and adding comments to it. We then implemented some new features such as a bike lane overlay for the google map and dark mode.

The second week consisted mainly of housekeeping and setting up the project for submission. We deployed the app to our EC2 instance, which unearthed a few issues with the app that would need to be resolved before the final submission date.

Issues:

At this point in the project we were spending far more time on the project than we had originally planned. Lots of time was spent on the machine learning model and we had issues incorporating the model into the application. More details on that can be seen in the data analytics section of the report. Some extra features such as the bike lane overlays and Dark Mode took longer to implement than expected as we had to learn how to use bootstrap for the first time, which also added to the

overall project overhead. We also experienced a number of issues when we deployed (what we thought was) the completed application to EC2.

GitHub:
We were initially unable to push the model (pickle file) to GitHub as the the file is 176MB in size and the upload limit for a single file is 100MB. We solved this by using Git LFS (Large File Storage), an open source application that allows the user to upload files that are larger than the limit of 100MB. There was a risk of causing problems in the repository by using this feature but the repository still functioned as normal after the pickle file was uploaded.

Database Issues:
We've had issues with connections to our database not closing properly and then causing very poor load times. We believed that the reason for this was the large number of connections being made to the database on a regular basis.We later found out that there was an easy fix; the problem was caused by SQLAlchemy and was easy to fix once it was idenfied.

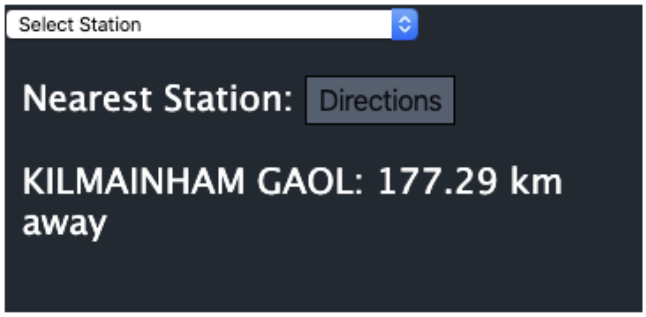
Issues with application deployment to EC2
We decided to deploy the application to EC2 using the web serverss nginx and gunicorn after some research and study of online tutorials. The configuration was confusing and time consuming. We obtained a domain name (www.dublinbikes2020.com) from godaddy.com. This did not work well initially and the website was quite unreliable; at times the website did not load at all and returned a single load error. We later discovered that we needed to change the nameservers from those of godaddy.com to those of AWS, where the application was actually running. This solved the initial performance issues.

We realised during Sprint 3 that the nested queries for obtaining dynamic bike data from the database would cause performance issues at some point. When we deployed the app to EC2 we experienced slow loading times for the map marker infowindows which showed bike occupancy data. At times this would cause the entire website to crash. This was rectified by adding a 'created at' column to the database containing the dynamic bike information; the new column would hold the date and time of a row's entry into the database. This allowed us to remove the nested loop which led to a significant performance improvement.

Bike Prediction Issues:
The performance of our bicycle prediction feature was unexpectedly slow and unreliable at times while the app is running on EC2. There are no issues with this feature when the app is run locally. We identifed the issue in the late stages of the project and fixed it to the best of our ability with the remaining project time. More detailed information on this can be seen in the Data Analytics section of the report.

Justifications:

We did not go ahead with the SSL certificate for the geolocation feature as we were originally planning. This was due to issues with the performace issues of the app, notably the bike prediction feature, when the application was deployed to the EC2 instance. Fixing this issue took precedence over implementing the SSL certificate. Instead, a hardcoded user location shows that the geolocation feature works as intended, with directions to each bike station beginning outside the computer science building in UCD.



Fig?: This shows the geolocation feature, before deployment to EC2, correctly identifying the nearest station to the computer testing the feature from its location in the southwest of Ireland.

We were also beginning to do a little bit of research on cookies: we were aware that a large cohort of the target audience of this app would be commuters who travel to and from the same stations at the same times every day, and that cookies would possibly help reduce loading times for these users as a result. We put this to the side once the other problems began to surface.

Documentation:

As per the previous sprints,we created a backlog with all the tasks we expected to have to do during the sprint. A couple of tasks were added as they appeared throughout the sprint, such as the fixing of the multitude of errors that were uncovered bythe deployment of the application on EC2.

Meetings and Standups:

As per sprint 3, the full meetings for this sprint were held remotely on Google Hangouts during practical times on Tuesday and Thursday mornings. There were four meetings in total, one during each allocated practical time.

Unlike previous sprints, we decided to have a standup meeting on the very first day of the sprint (normally we would have our first standup meeting on day 2 during the practical). This was due to the data analytics work being pushed into the beginning of this sprint so project work was done on day 1 of the sprint before we put together our backlog of tasks for the remainder of the project.

Burndown Chart and Analysis (or lack thereof):

Numerous extra tasks occurred after the deployment of the application to EC2, as there were unexpected hiccups that occurred. We had no choice but to fix them so that the application performed as it did on our local machines, so because of that, our beginning-of-sprint predictions for the amount of hours to spend on certain tasks did not include the extra tasks that had to be carried out towards the end of the sprint, as we had no knowledge they would need to be done. This made our burndown chart and trello board almost completely defunct for this sprint.

We probably spent over twice as many hours working on the project during this sprint as we had intended: during the first week, we were already spending a lot more time on the project than envisaged, and this was before the deployment of the application to EC2 which blew this out of proportion.

Sprint 4 Retrospective

1. What went well?
Work ethic: we were more focussed on the end goal in this sprint than in any of the previous sprints, and the decision to begin working on day 1 of the sprint instead of day 2 meant we were in the zone in terms of the project work before the first meeting even took place.
We completed the following during the final sprint:
- Finished the data analytics model
- Implemented the model in the front end (+ dropdown menus so user can choose day and time)
- Added finishing touches to user interface + modal (popup infowindows)
- Implemented bike lane overlay and Dark Mode on the google map
- Deployed the application to the EC2 instance
- Added comments to code
- Cleaned up code
- Improved map marker loading speed by optimising SQL queries
- Fixed bike prediction performance

2. What didn’t go well?
Time management: we spent far more time on the tasks for this sprint that originally intended
We didn’t implement the following extra features as a result:
- The implementation of the SSL certificate for the geo-location feature.
- The cookies feature

Data Analytics:

The data analytics part of this project involved the implementation of a machine learning model that would predict the number of bicycles that would be available at a given station at a certain time of the day. The code for the model is available in the notebook file Regression_Model.ipynb in our GitHub repository: https://github.com/kevinmitch14/Software_Engineering/blob/master/app/Regression_Model.ipynb.

How does your application provide predictions?

This application provides predictions using data that we've collected from the JCDecaux API to specify the number of bikes at each station, and this data is fed into a machine learning model. The target is to be able to predict how many bicycles would be available on an hourly basis, on any day of the week, at any bike station in the city.

The user simply chooses their station, day, and time from the dropdown menus in the sidebar. Choosing the station produces the two charts at the bottom of the page that show the daily and hourly data for bike availability, and then clicking 'Predict Availability' after choosing their day and time will produce a prediction of the number of bicycles that would be at that station on that day at that time.

The feature is integrated into the front end via html and javascript. The dropdown menus from which the user chooses the day and the time are initialised in html code, as is the 'Predict Availability' button. This button calls a javascript function which takes the options that the user chose in the dropdown menus and passes them to the flask application. Here, the parameters are input into a pandas dataframeand returns the prediction to the front end where it is displayed in the sidebar.

Model:

At first we attempted a linear regression model but it proved difficult to find any strong linear relationshps in the data. A test of running the data through a linear regression model resulted in an RMSE (Root mean square error: the lower the better) of 10.3 and an R2 (root squared: the higher the better) of 0.06.

Although the accuracy of the model was not the most important element of the data analytics side of the project, the predictions from the linear regression model would have been so inaccurate to the point of the feature being completely useless to the customer using the application, and therefore pointless to implement when there were other options available that we hadn't tried yet. This led to experimentation with a Random Forest model which gave us a much higher level of accuracy: TheRMSE was 5.8, the MAE (Mean Absolute Error) was 4.3 and the R2 was0.7. The Random Forest model was therefore implemented in full instead of the linear regression model. The results of the testing of both models can be seen in the images of the Jupyter Notebook on the next page.

```
In [102]: lin_reg_mod = LinearRegression()

In [103]: lin_reg_mod.fit(train_features, train_labels)

Out[103]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normal
          ize=False)

In [104]: pred = lin_reg_mod.predict(test_features)

In [105]: test_set_rmse_1 = (np.sqrt(mean_squared_error(test_labels, pred)))

          test_set_r2_1 = r2_score(test_labels, pred)

In [106]: print(test_set_rmse_1)
          print(test_set_r2_1)

          10.295692144524518
          0.06404409585946147
```

Fig ?: The results of testing the linear regression model.

```
In [97]: # Random Forest Regression Model
         rf = RandomForestRegressor()
         # Train the model on training data
         rf.fit(train_features, train_labels);

In [98]: # Use the forest's predict method on the test data
         predictions = rf.predict(test_features)
         # Calculate the absolute errors
         errors = abs(predictions - test_labels)
         # Print out the mean absolute error (mae)
         print('Mean Absolute Error:', round(np.mean(errors), 2))

         Mean Absolute Error: 4.3

In [99]: test_set_rmse = (np.sqrt(mean_squared_error(test_labels, predictions)))

         test_set_r2 = r2_score(test_labels, predictions)

In [100]: # Print the RMSE and R2.
          # We got an R2 of 0.70018 which is quite good considering the quality and lack of data.
          print(test_set_rmse)
          print(test_set_r2)

          5.814614850370847
          0.7001876657833143
```

Fig ?: The Random Forest model would yield far more accurate results.

The data was read into the notebook through .csv files containing the data we had collected up until that point. This data had to be cleaned first using pandas, and then the static and dynamic bike dataframes were combined into the one dataframe. We had to be careful to have an appropriate cutoff point with the data so that data collected after restrictive measures brought in to combat Covid-19 would not be taken into account. The weather data was imported into the notebook too but was not incorporated into the model (more detail on this can be seen below in the Issues section). Dummy variables were created for the each of the days of the week and each of the hours of the day for which the model would make predictions.

Issues:

The machine learning model uses the data we had collected up to the point where the restrictions caused by the Covid-19 outbreak skewed the data abnormally, causing much of this data to contain inaccurate information. Unfortunately, this means we only had about two and a half weeks of realistic, accurate information to work with.

We were also intending on incorporating the weather data into this model too, but there were a few reasons we decided not to in the end. Firstly, time constraints towards the end of the project were an issue, and there would've been issues being able to predict accurate hourly weather data up to a week in advance, even from an API as good as Dark Sky, due to the unpredictable nature of Ireland's temperate oceanic climate. As well as this, due to us only having two and a half weeks worth of data, we could not collect enough weather data to be able to accurately predict bike availability based on weather, when you have to take all of temperature, rainfall and wind speed into account at the same time.

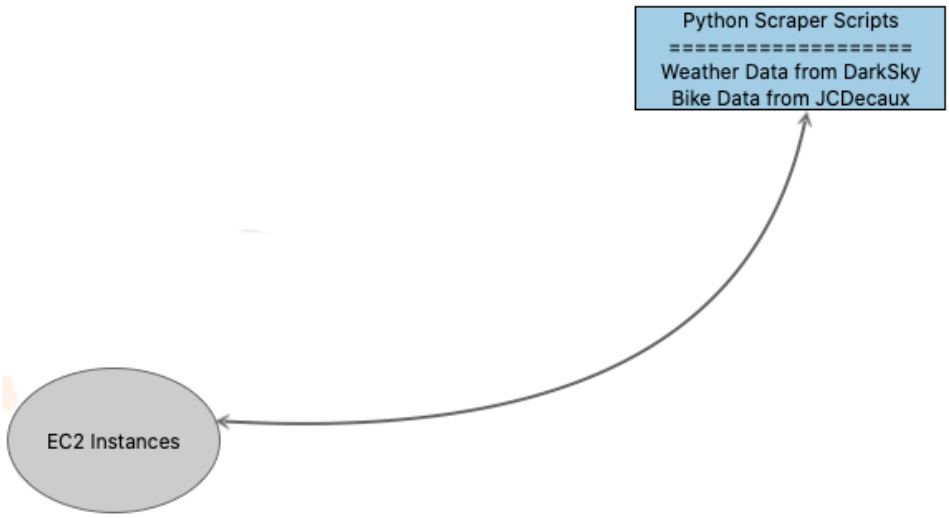
There were also issues incorporating the model into the front end. The main problem was that the model used dummy variables for the day and hour fields. We had to find a way to have the user's selection inserted into a data frame which could be passed to the model to have a prediction returned. This was not very straight forward and caused frustration at the time due to time constraints.

As mentioned previously, there were performance issues when the application was deployed to EC2.We found out from the nginx web server error logs that there were memory issues occuring whenever the model was loaded, causing slow loading times and occasionally causing the feature to freeze completely. It took many hours of research to find a solution; a post on StackOverflow recommended using joblib instead of pickle for the machine learning model. This solved all of the performance issues and the bike predictions now work seamlessly.

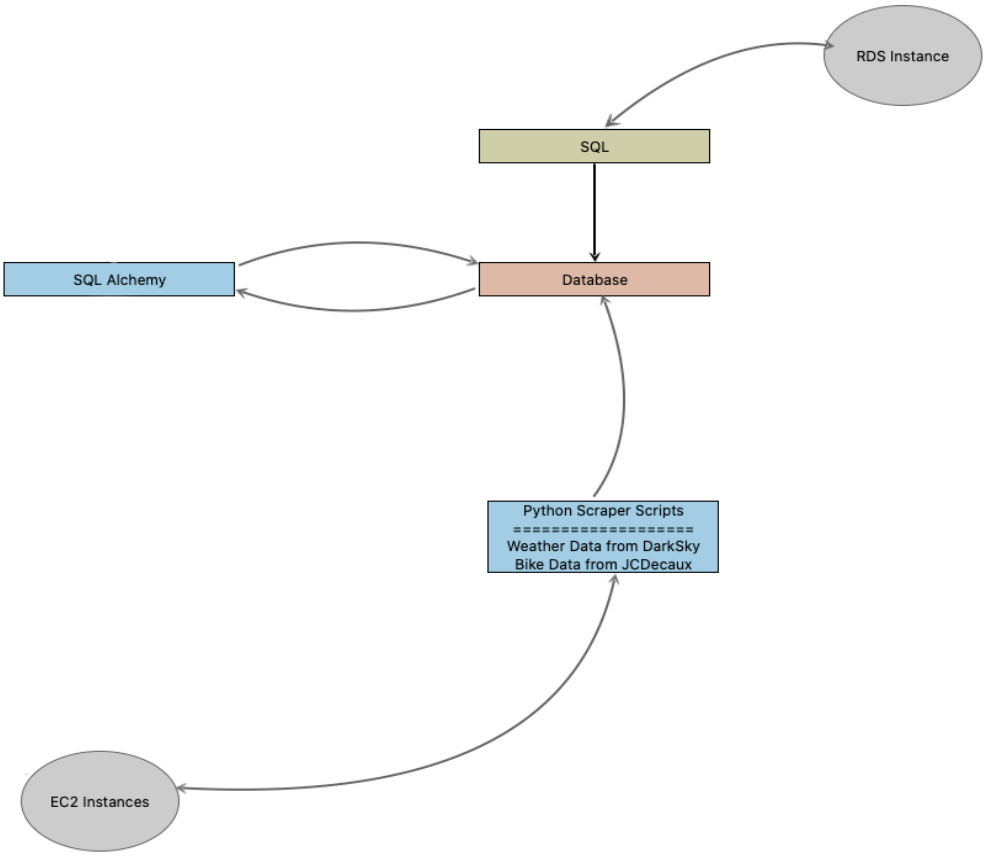
Architecture:

We began with just our EC2 instances that we set up during the practical sessions in week 2 and 3 of the module.

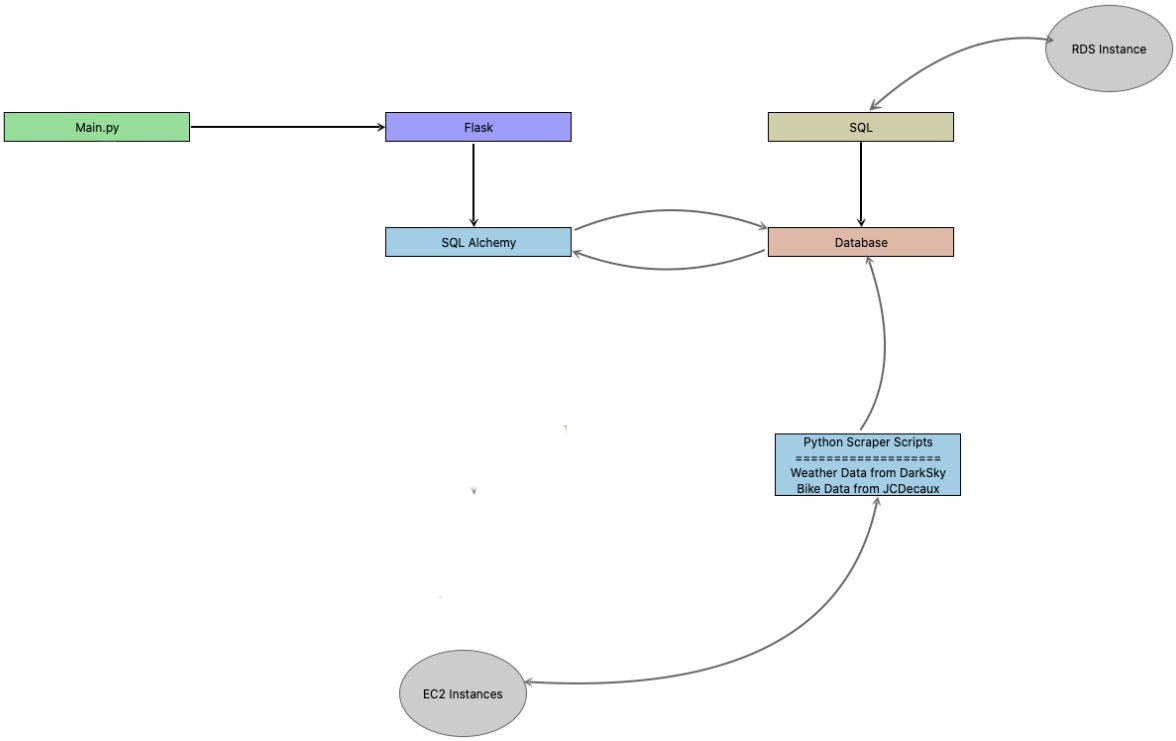
The data scrapers that would obtain data from the JCDecaux and Dark Sky APIs were the first scripts we created . One scraper was required for the weather data, all of which is dynamic. Two scrapers were required for the bike data: a static scraper that would obtain information that does not change, i.e. the bike station name, the co-ordinates of the station, and the maximum capacity of bikes it could hold. A separate scraper would obtain dynamic data, such as the number of available bikes or free stands at a station at any one time. These scrapers would run every few minutes (for the bike data, timed using crontab) and every hour (for the weather data) on an EC2 instance.



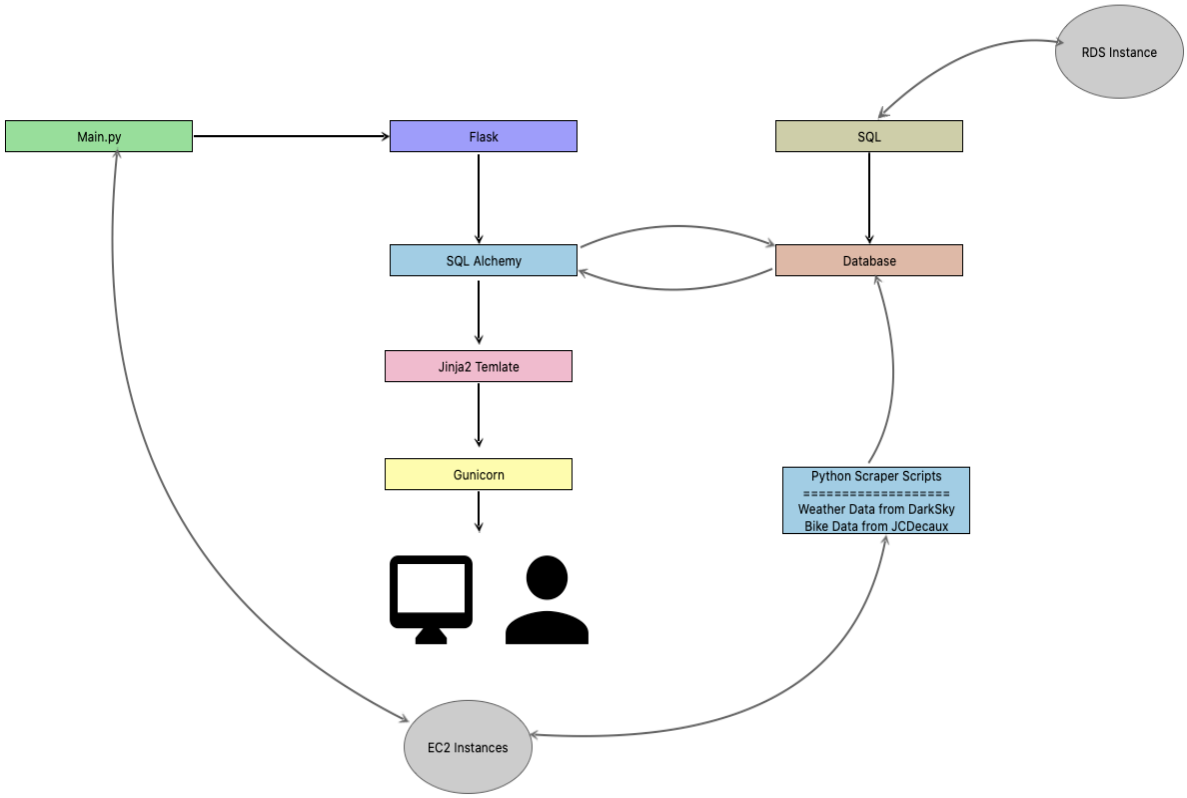
An Amazon RDS instance was set up to hold an SQL database which would store the data obtained by the scrapers. As the scrapers are python scripts, SQLAlchemy was required to faciliate communication between the scraper and the database. When this connection was made to populate the databases, the information could then be accessed from MySQL Workbench where it could be inspected.



The main.py script was then created to house the flask application. Flask also communicates with the database via SQLAlchemy as ‘main’ is a Python script. The main.py script is all-encompassing; it contains and connects everything the application needs to run: the back-end flask architecture which retrieves the data for the analytics runs within it, and the scripts containing the front-end code like html and javascript that display this data to the user.



The final step is to deploy the application to EC2 instance. This was done using Gunicorn, a web server gateway interface used to run Python web applications. A domain name from which the application would be run was then secured. This led to the completion of the system architecture which can be seen below in its entirety.



Back-end technologies:
This is a mostly Python-based web application (the data scrapers and the main application are all Python scripts).The data retrieved from the APIs is stored in SQL databases, which have to be accessed using SQLAlchemy.

Front-end technologies:
Html (with CSS for extra styling) and Javascript are the three web application-based front-end languages that give power to the user to choose the actions they want performed by the application.

Data Analytics:
The regression model for the data analytics is written in Python but stored in a Jupyter Notebook (.ipynb) file. It does not pull data directly from the database like the Flask application but instead reads it in as a dataframe, from .csv files. Within the notebook the data is cleaned using Pandas and the model implemented.

Overall Functionality:
The program ran perfectly and without any performance issues before it was deployed to the EC2 instance. In hindsight, the application should have been deployed at an earlier stage so that we would have more time to iron out any potential problems that arose from the change in environment. We did manage to fix all of these performance issues before the deadline.

Design:

The application is designed to give the user as much functionality as possible in the smallest amount of space. The map and the sidebar house all of the information the user needs when deciding on where to rent out a bicycle, with the footer housing additional information such as the prediction charts.

Interaction between the application and user: providing real-time results and predictions

The application works by presenting a google map of Dublin city to the user. The map markers show the exact location of each bike station, and the colour of the marker gives an estimate on the amount of bicycles available at that station at that time. When a map marker is clicked, the name of the station is displayed in a popup window, along with the number of available bikes and free bike stands, and an icon in the bottom right corner telling the user whether or not there is a credit card terminal for payment at their selected station. Clicking on a station on the map will also activate the chart section which appears below the map: the chart on the left will show the average number of bicycles availably daily from this station, and the chart on the right will show the average number available each hour. The google map also has options that you can toggle on or off: a Dark Mode that shows a darker version of the map for easier viewing in the dark, and a Bicycle Layer option that will show which streets on the map have bicycle lanes.

The user can also choose a station via a dropdown menu at the top of the sidebar on the right. When the user selects a station via the sidebar, the number of available bikes and free stands for that station appear in the sidebar rather than a popup window. The charts that show the average daily and hourly data will appear in the same place as before. There is an option here to select a day and a time from two dropdown menus. With this information the application will make a prediction as to how many bicycles would be available at a station chosen by the user at any given time or day of the week, which can be more useful to the user than displaying the average hourly or daily bike data on a chart. The sidebar also contains a real-time clock and a basic weather forecast for the day. An icon which shows whether it is sunny, cloudy or raining is displayed alongisde the temperature, and below this there are measurements for the wind speed and chance of rain, all of which is data retrieved from the Dark Sky API from which we scrape all of our weather data.

Design development:

The first mockup of the user interface was developed towards the end of Sprint 1. It partly fed into our final design, but was deemed too weather-centric and didn’t devote enough space to bicycle station information for the user.

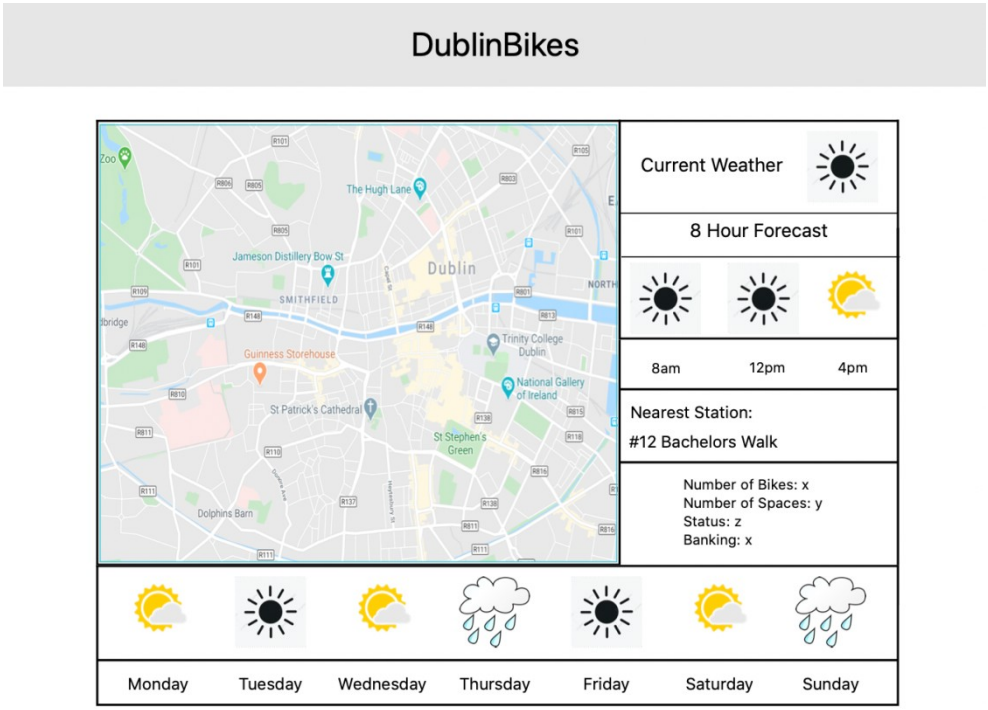


Fig ?: An early mockup of the user interface completed during the first sprint, designed on MockPlus.

The below mockup of the user interface was imagined during Sprint 2, and removed most of the weather data. This was the first mockup to indicate where the prediction charts would be.

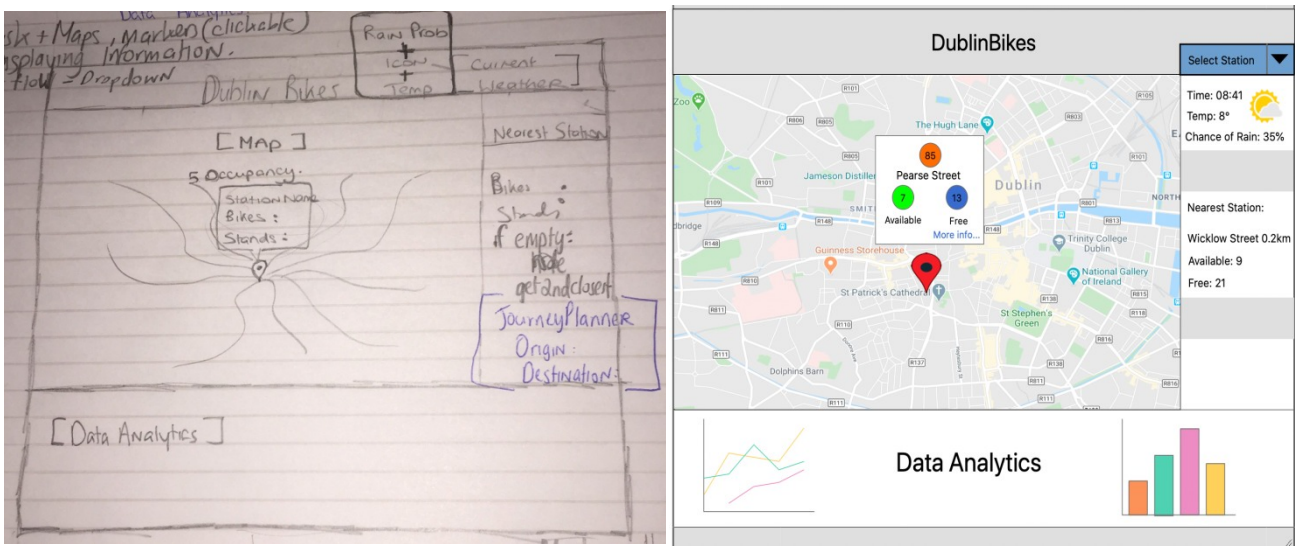


Fig ?: An early mockup sketch of the user interface completed during the second sprint, followed by a version designed on MockPlus.

The second mockup came much closer to giving an idea of what the completed user interface would look like. The completed interface is just an evolution of the second mockup, with a change in colour scheme and the extra added functionality for the bike station predictions.

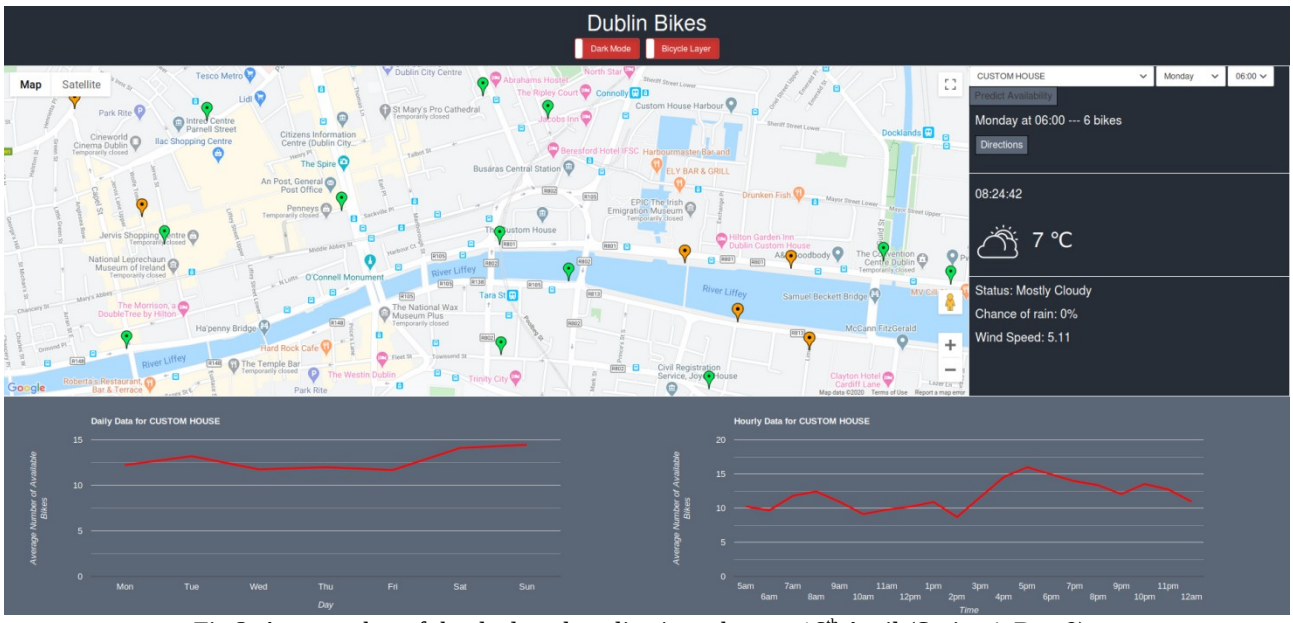


Fig ?: A screenshot of the deployed application taken on 16th April (Sprint 4, Day 9).

The above image shows what the application looked like after we had ironed out the major problems unearthed by the deployment of the application on EC2. Below, we see an image of the user interface while a number of different features are being implemented at the same time. It shows the app giving directions to the nearest station from the hardcoded user location at UCD. It shows the results of a clicked map marker which gives bike information to the user via a popup window. It also shows this information in the sidebar where the user chose the same station from the dropdown menu, and a prediction for the number of bikes that would be at that station at 10am on a Tuesday.

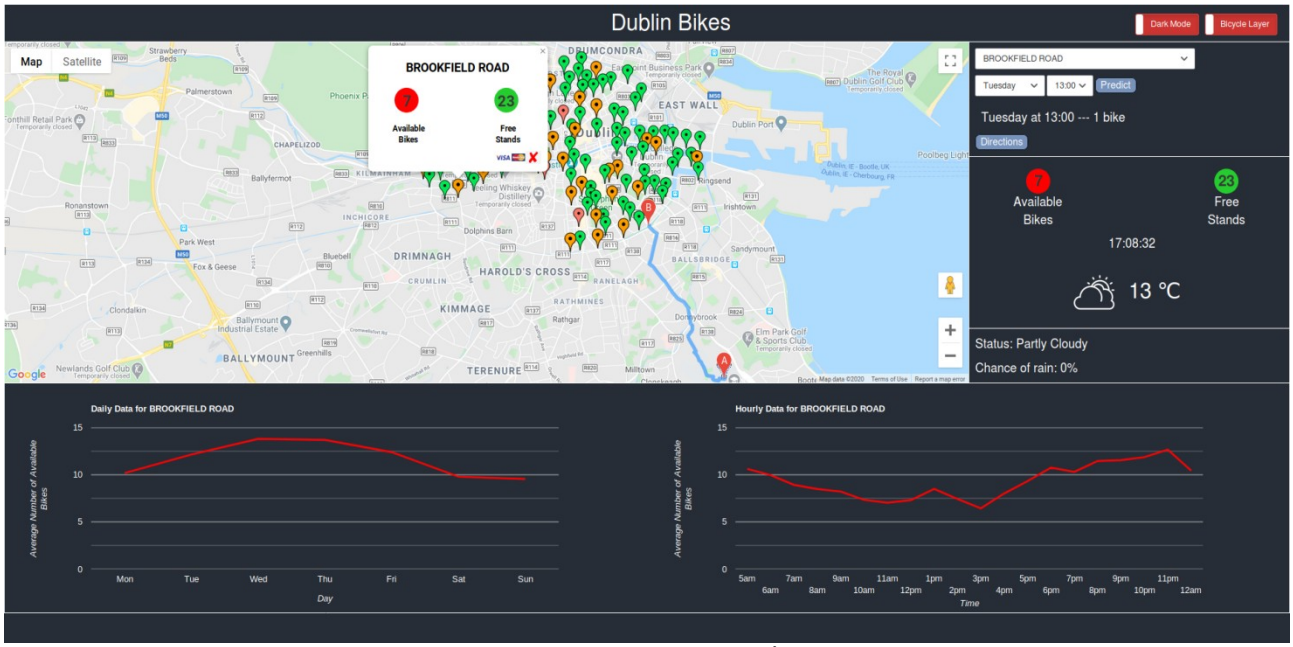
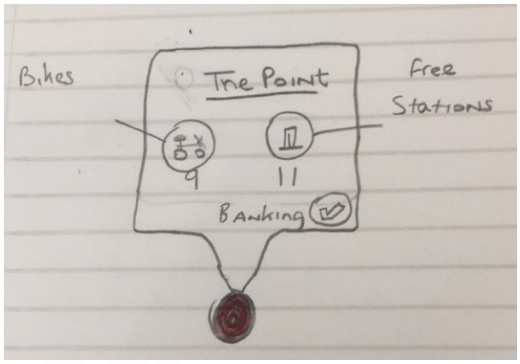


Fig ?: A screenshot of the deployed application taken on 18th April (Sprint 4, Day 11 extension).

The modal design was drawn up in sprint 2, as was the functionality of a popup window appearing in the application. The design in the finished application was not completed until sprint 3.



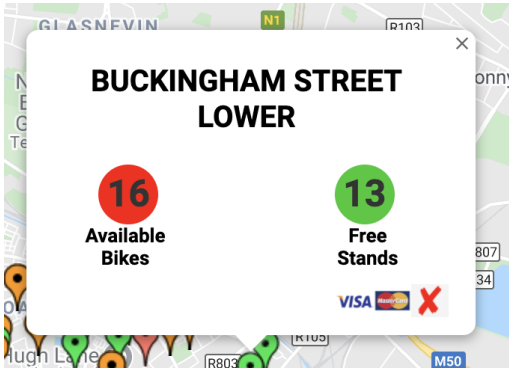


Fig ?: The modal design for the popup windows did not change from the original mockup done in Sprint 2.

Have we met the user and technical requirements?

We feel that we have met the requirements from the end user's point of view: the design is simple and only houses the information the users need without overcluttering of information. The whole page is interactive and can carry out all user commands without moving to a different URL. We feel like we have satisfied the technical reqirements for the most part, but are aware that the application can be slow at times.

Project Retrospective + Future Work:

In this section we will identify a number of areas where we think improvements could be made if we were to continue working on this project beyond the deadline, or if we could begin the project from scratch with the same timeframe as before.

Project Management:

Overall, we were satisfied that the project was conducted and managed sufficiently over the course of the eight weeks. More time and care could have been devoted to ensuring that standup meetings occurred every single day of the project, even those were none of the three members carried out any work, purely for documentation purposes. Some of our predictions for the amount of time that certain tasks would require throughout the project were wildly underestimated, especially towards the end of the project where, while we expected our workload to increase, we also expected our prediction accuracy to increase as well.

Application Performance:

As mentioned previously in the report, an extra week to work on optimising the performance of the application would have been helpful, as we could have ironed out more of the issues that arose from deploying the application on our EC2 instance at an earlier date. Had we been able to deploy the app to the instance earlier, we might have been able to have those problems fixed promptly at the time of deployment, which would have given us more time to further improve the design and possibly even add extra features.

Data Analytics:

If we had more time we would have liked to implement weather data into the machine learning model in some way. Our ability to do this in the permitted timeframe was also hampered by the restrictions put in place due to Covid-19, which left us with less data to work with. Ideally we could have had 4 or 5 weeks of bicycle and weather data to work with, rather than just 2 and a half weeks. Although we are satisfied with the performance of our Random Forest machine learning model, it would have been interesting to see what kind of predictions we could have made with more data.

Extra Features:

If we had more time we would have implemented the SSL certificate so that the geo-location feature could work in its entirety when the application was deployed to EC2. Previous testing showed that the feature worked locally, with accurate directions to the nearest station from the location of the computer running the app. We also may have added an option to toggle between displaying the daily average bike data chart and the the hourly average bike data chart. In the application both charts are displayed alongside each other as default.

Implementing cookies is something we would have done if we had more time, as many users would travel to and from the same stations every day, and we could improve loading times for their information.We had planned to do this before the deployment of the application caused problems.

Overall Contribution:

Team members: Thomas Grogan, Kevin Mitchell, Conor Loughran

Most of the code was written individually by Thomas Grogan and Kevin Mitchell, and some was written by Thomas and Kevin in collaboration; the history of commits to our GitHub repository will give most of the information about which individual programs and pieces of code were written by which member. The documentation and management of the project was conducted by Conor Loughran. Through discussion between the three members, we concluded that the overall project completion by each member of the team could be split as follows:

Conor: 30%
Kevin: 35%
Thomas: 35%

The justification for this split is that although the deliverable project (conducted mostly by Thomas and Kevin) and the project management (conducted mostly by Conor) are worth the same amount of marks, the total hours devoted to the project management were considerably fewer than those required to complete the deliverable project. The decision on which section of the project would be conducted by which members was decided during the first sprint: the original plan was for the member in charge of management and documentation to change with each sprint, but we found that the original configuration worked well for the project.

Meeting Logs:

The meetings for this project took place on Tuesday and Thursday mornings during practical time, meaning there was generally four full meeting per sprint that involved the product owner, Karl. These were carried out in person in the lab sessions, until Sprint 3, after which all meetings took place remotely on Google Hangouts. Shown here are the notes taken with the main points from every meeting for the project.

Software Engineering Project: Meetings: Sprint 1

All meetings take place during the allocated hours for the lab sessions associated with COMP30830 Software Engineering. The exception during this first sprint is Meeting 3, which was intended as the first standup meeting of the sprint, but developed into a full meeting in which the project was discussed in full and tasks were planned for the remainder of the sprint.

Meeting 1: Tuesday 11/02/2020 (Sprint Day 2 of 10)
9:00am-11am - Computer Science B106 (during practical time)
Members present: Thomas, Kevin, Conor

- First meeting: group members initialised on Saturday 08/02/2020
- A shared github repository was set up to contain files associated with the project. Test commits were pushed to the repository by everyone to ensure everything was working properly. Each member ensured that their version of Visual Studio Code had git compatibility.
- Trello board set up, updated with the first project tasks required.
- Discussed the first steps to be taken in the project: Setting up a data scraper in python to retrieve JSON data with information for each bicycle station.

Meeting 2: Thursday 13/02/2020 (Sprint Day 4 of 10)
9:00am-10:45am - Science East E1.17 (during practical time)
Members present: Thomas, Kevin, Conor

- Slack set up.
- Worked on data scraper; not yet finished
- Discussion with Karl (TA) about setting up a weather API. Further discussion included the importance of record keeping and documentation of all aspects of the project.
- Updated tasks on Trello: New tasks assigned: overall project research, research APIs for weather; learn more about scrum methodologies for record keeping. Update on existing tasks: further changes to be made to existing but incomplete data scraper.
- Burndown chart analysis: 6 hours work behind schedule.

Meeting 3: Monday 17/02/2020 (Sprint Day 6 of 10)
4:30pm-5:50pm - Health Science Building Lobby
Members present: Thomas, Kevin, Conor

- Discussed completed tasks: data scraper for bicycle data now functional and retrieving information every three minutes.
- First time looking at the design of what the user should see on the webpage: map to take up most of screen, pop-up will show bike station name, number of bikes etc. and also ideally a graph of how many bikes are predicted to be at a specific station during certain weather conditions at different parts of the day (ideally one estimate per hour from daytime hours 6am-10pm)
- Discussing whether or not the weather data is to be used for each specific bike station or just for the city as a whole
- Figuring out the best way to display weather. Current idea: displaying the general forecast on the main page and then more specific information in the pop-up when the user clicks on a certain station
- First look at system architecture: rough diagrams drawn to display what the system *should* look like
- Wrote up questions to ask Karl at Practical Session on Tuesday 18/02/2020
- Questions: ask about architecture, where is the flask application in the architecture? Best things possible to do with weather data?
- Items to add to backlog of things to complete before end of sprint:
    - Set up python script to read weather data every hour and push it to the database (RDS) and when that is working it is to be pushed to github, then pulled from github to EC2. (Estimated time: 4 hours)
    - Update all notes from meetings, begin formatting documentation and designing burndown graphs.

Meeting 4: Tuesday 18/02/2020 (Sprint Day 7 of 10)
9:10am-10:45am Computer Science B106 (during practical time)
Members present: Thomas, Kevin, Conor

- Daily standup completed during this time (notes contained in Standup_Meetings.pdf)
- Asked Karl questions regarding documentation (backlogs, burndowns etc.) as well as a problem in the development of the code for scraping weather data, which was subsequently fixed
- Still working on getting weather API up and running
- Completed basic Flask application: app is now sending a small amount of data to the front end.
- Organised time/date for Sprint 1 Review and Retrospective: 11am on Friday 21/02/2020
- Started to keep all documentation for daily standups in a separate Slack channel based on advice from Karl, full meeting documentation (such as longer meetings during practical time) to be kept separate in a Google Doc
- Burndown chart analysis: still slightly behind schedule

Meeting 5: Thursday 20/02/2020 (Sprint Day 9 of 10)
9:00am-10:45am Science East E1.17 (during practical time)
Members present: Thomas, Kevin, Conor

- Daily Standup completed during this time
- Update with Karl: showed progress on bike and weather data scrapers, confirmed working template for burndown and backlog charts.
- Bike and weather data scrapers sending info to database as normal.
- Currently working on having Google Maps show on the basic Flask application: more research on Flask required and being done during meeting time
- Burndown chart analysis: all work currently on schedule.

Meetings for Sprint 2

Meeting 1: Tuesday 25/02/2020 (Sprint Day 2 of 10)
9:00am-10:45am - Computer Science B106 (during practical time)
Members present: Thomas, Kevin, Conor

- Daily Standup completed during this time
- Meeting with Karl: Received long list of advice on what to have done for the next sprint:
- By Thursday: design for the front end: where things are actually going to be placed
- Front end interface, features for this interface, all these features should be assigned in a backlog now at the beginning of the sprint (this was carried out during this meeting (see Trello screenshots for 25/02/20)
- Nice formalised structure as to how everything is going to play out during the course of the sprint; have a nice structure on everything you do from now on
- What is required: flask app, google map on flask page, markers on every page, everything linked up to display information in the application
- Justification of things needs to be figured out before doing things: why are you doing this, why are you adding this feature etc.
- User-flow diagram: how exactly is it going to work
- There will be different use cases of this app: some will want to go on and find out what they want straight away e.g. where is the nearest bikes. Other users: won't know where they are, i.e. tourists. Explore all of these different options, the more you think about in terms of different types of users (without overcluttering the application in the process) the better; however we need to justify every feature we add from now on. (end of meeting w Karl)
- Research being carried out into websites created for other cities with similar bike-rental services to those of dublinbikes, to see how other cities are implementing this system: this is to help both with inspiration for the design and functionality of our own application.

Meeting 2: Thursday 25/02/2020 (Sprint Day 4 of 10)
9:00am-10:45am - Science East E1.16 (during practical time)
Members present: Thomas, Kevin, Conor

- Daily Standup completed during this time
- Meeting with Karl:
- Be more specific with tasks in Trello: try to break down each task into subtasks
- Interesting task: compare speed of getting data from API vs getting data from database
- Markers, dropdown, popups should definitely be done by the end of this sprint

Meeting 3: Tuesday 03/02/2020 (Sprint Day 7 of 10)
9:00am-10:45am – Computer Science B106 (during practical time)
Members present: Thomas, Kevin

- Daily Standup completed during this time (via Slack)
- Meeting with Karl:
- Timewise, we are on track for this sprint and have satisfied most of the requirements.
- (Burndown chart analysis: slightly ahead of schedule)
- Things to do: Justify why we pull data from RDS rather than JCDecaux API, Create dropdown for stations that displays station info
- (Possible extra feature to add as we are ahead of schedule: add colour coding (based on bike availability) to Google Maps markers.
- What we're doing today: Working on dropdown menu.
- Conor absent due to illness

Meeting 4: Thursday 05/03/2020 (Sprint Day 9 of 10)
9:00am-10:45am - Science East E1.17 (during practical time)
Members present: Thomas, Kevin

- Meeting with Karl:
- Everything fully on schedule, i.e. bare bones working prototype of product completed by the end of Sprint 2.
- Decision made to extend Sprint into the midterm break in order to add extra features
- Conor still absent

Meetings for Sprint 3:

Meeting 1: Tuesday 24/03/2020 (Sprint Day 2 of 10)
9:00am-10:00am - Google Hangouts (during practical time)
Members present: Thomas, Kevin, Conor

- First remote meeting via Google Hangouts
- Daily Standup (first of Sprint 3) completed during this time
- Meeting with Karl: raised two important focal points for Sprint 3: Data Analytics and Visualisation
- Reinforcement of the fact that this is not a Data Analytics module: do not spend all of your time on it: just make sure whatever you do implement is operating smoothly from the front end
- Test models (linear or regression) on Jupyter Notebook for accuracy (100% accuracy is not essential, only an evaluation required)
- Visualisations: Some features already implemented include colour-coded markers which estimate a percentage of bikes available at any given station
- Ideas to possibly implement: bike availability table - a (onclick/hover) marker that will show a visualisation of the average number of bikes available on a daily basis
- If time permits, better attention will be paid to the appearance (css)
- Possibly implement a predict section with which a user can select a day, time and station, and this info is fed to the Flask application which will return an estimate of how many bikes should be at that station at the time chosen by the customer
- Discussed housekeeping issues with Sprint 2
- Sprint 3 Implementation Plan: added new features for implementation to the project plan
- Added new tasks to the Trello board

Meeting 2: Thursday 26/03/2020 (Sprint Day 4 of 10)
9:00am-10:00am - Google Hangouts (during practical time)
Members present: Thomas, Kevin, Conor

- Daily Standup completed during this time
- Meeting with Karl:
- Charts: average daily amount of bikes for each station completed: working very slowly, but Karl confirmed it will run much faster when run on the AWS (and that its not an issue with the code itself)
- If still slow on AWS, possibly add spinner or loading icon to mouse pointer to make the application look more professional
- Analytics part: need to let the user choose the day, time, station etc.
- Since lockdown announcement bike data has not been particularly useful: ideally just use data from before then as data since announcement would be less accurate

- Map markers design consideration: red markers at stations makes it look like there are no bikes available at that station: this was changed so that red = 0% bikes available at a station and orange = 1% - 25% bikes available
- Plan for data analytics: will begin by the start of next week (Day 5 or 6 of Sprint)

Meeting 3: Tuesday 31/03/2020 (Sprint Day 7 of 10)
9:00am-9:40am - Google Hangouts (during practical time)
Members present: Thomas, Kevin, Conor

- Daily Standup completed during this time
- Meeting with Karl:
- Cleaned data should appear in table, do not take into account data collected since covid-19 lockdown as it will be inaccurate overall
- Discussed extra feature implemented (geo-location): For working with direction based stuff, when deployed on EC2 we need to have an SSL certificate installed so that the application user can provide their location to the app. If this implementation does not work a hardcoded city centre location should show that the feature is functional.
- Sidebar design: use only one font, and perhaps split up the sidebar so that there is a defined area or box for weather data to separate it from bike/station data.
- Plan for this week: mostly data analytics. Start by training a model on just the bike data, then add the weather data and see if there is a difference.
- Weather consideration: how far ahead does DarkSky go in terms of weather? E.g. if it's Tuesday and somebody wants to get a bike on Sunday. Consideration: be aware of exactly how far into the future daily/hourly data can be

There was no meeting 4 for this sprint.

Meetings for Sprint 4:

Meeting 1: Tuesday 07/04/2020 (Sprint Day 2 of 10)
9:00am-9:40am - Google Hangouts (during practical time)
Members present: Thomas, Kevin, Conor

- Daily standup completed during this time
- Data Analytics: each station has a unique dataset
- Have some sort of model as a functional (or in this case not so functional) component: we are limited by the database we have, but once we have a linear regression model thats trained and implemented that is sufficient
- At this point, whatever we have, just go with that: write about these limitations in the report
- Flask application: start integrating into the front end (+ document in report)
- Things to be done in sprint 4:

1. Make it look and feel like an actual app: the importance of having a complete and easy to use user interface cannot be stressed enough; Eliminate roughness around the edges: the more polished the better
2. Deployment of Flask app on EC2
3. Anything that wasn't completed in Sprint 3 (i.e. data analytics)

- Tasks that aren't absolutely mandatory but can be improved:
  1. Optimise performance; if queries are slow to load, try to find out why.
  Deploy on EC2 before testing performance as speed on your local machine might not accurately reflect real-world performance
  2. Try keep coding clean, i.e. javascript, css in separate files etc.
  (and don't forget to put all this stuff in report)
- Possible interesting features:
  1. Bicycle lane overlays
  2. SSL cert for geolocation feature, hardcoded location if not
- For report: Geolocation, after being deployed on EC2, would not work without an SSL cert, so therefore etc.
- Structure expected in report: not yet finalised, but best way to look at it is having an introduction, then going sprint by sprint; some kind of project retrospective at end
- Outline limitations: what you would've implemented had you more time, overall issues with project, i.e. lack of data
- Screenshots: show off every single thing you've done
- Report should be sufficiently detailed: it shouldn't read like an instruction manual, but like a chronological report following the story of the project from the beginning of the end.
- What to get started on now: Deployment, user interface, models

Meeting 2: Thursday 09/04/2020 (Sprint Day 4 of 10)
9:00am-9:40am - Google Hangouts (during practical time)
Members present: Thomas, Kevin, Conor

- Daily standup completed during this time
- Discussed progress since last meeting two days ago: Started adding comments into the code
- Issue with Github, data analytics model too big to upload (about 171MB, Github only accepts files up to 100MB)
- Accuracy assessment: best = 0.75, mean squared error = around 4
- We will try to deploy app to the EC2 instance in the coming days
- Discussed Started adding comments into the code
- Report: start with introduction/scope of project, then wait for lecture brief on how to structure report overall

Meeting 3: Tuesday 14/04/2020 (Sprint Day 7 of 10)
9:00am-9:40am - Google Hangouts (during practical time)
Members present: Kevin, Conor

- Thomas absent due to illness

- Daily standup completed during this time
- Deployed application to EC2: partly functional but there are teething issues
- Possible (or probable) cause of issues: database connection. Ensure that every connection with the database is closed off
- Confirmed that geo-location feature would not work when app deployed to EC2 due to lack of SSL certificate: will implement feature properly if we have time
- For final report: make sure to have a screenshot of the working application
- To do for final week: finish report, fix any remaining connection issues with application, clean up user interface
- Don’t forget to include info about DarkSky in report re. takeover by Apple

Meeting 4: Thursday 16/04/2020 (Sprint Day 9 of 10)
9:00am-9:40am - Google Hangouts (during practical time)
Members present: Thomas, Conor

- Kevin absent due to illness
- Daily standup completed during this time
- Most of project is complete
- Few extra jobs to do: making a few changes to the css, finish report, continue testing the application
- Performance issue with predictions on available bikes: the predictions worked fine locally, before it was deployed to the server.
- As a result we will spend remaining time trying to fix this issue, rather than implementing the SSL certificate for the geo-location feature which was originally planned. (Location hardcoded at CS building in UCD to show that the directions feature works)
- If issue cannot be fixed and the performance is bad, have the application print an error message explaining that the feature is still beta whenever it doesn’t make a prediction.
- Report: go into more detail about technical aspects of architecture, design etc
- Main objective: identify issue with slow bike number predictions
- Perhaps put in a readme.txt in the GitHub repository as an introduction

Standup Logs

Standup meetings were carried out most days during this project and consisted of a minimal number of questions to be answered by each member of the team, logging their activity for the day and whether their work caused them any problems.

Sprint 1

There were no standup meetings from days 1-5 of this sprint. The first standup, scheuled on day 6 evolved into a full team meeting, the info of which can be found in the logs for the full meetings.

Standup: Tuesday 18/02/2020: 9:30am (B106 CSI)
Sprint day 7 of 10

- What did you do yesterday?
  Thomas: Completed data scraper and began directing data to a database
  Kevin: Started work on data scraper for weather data.
  Conor: Completed meeting notes for the first three meetings (first two held at practical time, third held outside of class)

- What will you do today?
  Thomas: Learn and develop Flask application
  Kevin: Further work on data scraper to retrieve weather data
  Conor: Begin designing burndown charts and extra required documentation for current sprint

- Is there anything in your way?
  Thomas: More information needed about Flask
  Kevin: Connecting to SQL database: work in progress
  Conor: Lack of knowledge of overall scrum record-keeping methodology

Standup: Wednesday 19/02/2020: 1pm Health Science Lobby
Sprint day 8 of 10

- What did you do yesterday?
  Thomas: Developed basic flask application
  Kevin: Made connection to database, finished weather data scraper and pushed it to Github
  Conor: Rough draft on backlog and burnout charts

- What will you do today?
  Thomas: Extra work on weather scraper
  Kevin: Research Google Maps API, look at connecting data from weather API to the front-end
  Conor: Create backlog and burndown template that can be used for all sprints, populate with information from the current sprint so far
- Is there anything in your way?
  Thomas: N/A
  Kevin: N/A
  Conor: N/A

Standup: Thursday 20/02/2020 9am: Science East E1.17
Sprint Day 9 of 10

- What did you do yesterday?
  Thomas: Updated weather scraper to point to RDS, pulled update to ec2 instance and created crontab schedule for it for every hour.
  Kevin: Worked on Google Maps API
  Conor: Created backlog and burndown template, populated it with the latest data

- What will you do today?
  Thomas: Look at getting basic Google map on showing in Flask
  Kevin: Work on Google Maps API
  Conor: Update charts + documents, prepare for review + retrospective

- Is there anything in your way?
  Thomas: Time constraints due to working on other assignments
  Kevin: N/A
  Conor: N/A

Standup: Friday 20/02/2020 11am: Newman Building Lobby
Sprint Day 10 of 10

- What did you do yesterday?
  Thomas: Worked on Google maps in Flask, Flask research
  Kevin: Worked on Google Maps API
  Conor: Created backlog and burndown template, populated it with the latest data

- What will you do today?
  End of sprint: Flask research will be carried over into sprint 2
  Finalise documentation for sprint 1

Sprint 2

As per most sprints, there was no meeting on the very first day, which precedes the first full meeting with the product owner during practical time.

Standup: Tuesday 25/02/2020: 9:30am (B106 CSI)
Sprint day 2 of 10

What did you do yesterday?
Thomas: N/A
Kevin: N/A
Conor: Pushed finalised Sprint 1 Documentation to Github repository (element of Sprint 1 Backlog)

What will you do today?
Thomas: Further development of Flask application
Kevin: Research websites similar to those of other cities with bike-rental services, design layout of user interface
Conor: N/A

Is there anything in your way?
Thomas: N/A
Kevin: N/A
Conor: Time constraints due to study for examination on Thursday 27th February

Standup: Wednesday 26/02/2020: 1pm (Health Science Building Lobby)
Sprint day 3 of 10

What did you do yesterday?
Thomas: Managed to get Flask to read from API.
Kevin: Looked at other services similar to DublinBikes for design and functionality inspiration (Barcelona, London, LA). Created wireframe and second mockup of front end interface
Conor: N/A

What will you do today?
Thomas: Get JSON data that was returned to display on the page.
Kevin: Create barebones html page, with divs corresponding to the features in the mockup.
Conor: Update all documentation, i.e. standup meetings, full meetings, begin populating burndown charts and graphs for Sprint 2

Is there anything in your way?
Thomas: handling of JSON data
Kevin: Balancing other modules
Conor: Balancing other modules

Standup: Thursday 27/02/2020: 9:30am (Science East E1.17)
Sprint day 4 of 10

What did you do yesterday?
Thomas: Got JSON data that was returned to display on the page.
Kevin: Created barebones html page, with divs corresponding to the features in the mockup
Conor: Updated documentation for meetings and standups (did not succeed in the beginning of the population of burndown chart + graph)

What will you do today?
Thomas: Get map markers displaying
Kevin: Will work on dropdown menu for the bike stations through JavaScript, User-flow diagrams and dialogue, research into Geolocation and calculating distance between coordinates.
Conor: Populate burndown charts, re-organise backlog on Trello to include things we did not expect to have to do

Is there anything in your way?
Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Friday 28/02/2020: 1pm (carried out online via Slack)
Sprint day 5 of 10

What did you do yesterday?
Thomas: Establish proper connection between Flask and the databases.
Kevin: Worked on user-flow
Conor: Worked on charts and documentation, updated Trello board

What will you do today?
Thomas: Map markers for each station
Kevin: N/A
Conor: N/A

Is there anything in your way?
Thomas: Issues with returning JSON data
Kevin: N/A
Conor: N/A

Standup: Monday 02/03/2020: 6pm (carried out online via Slack)
Sprint day 6 of 10

What did you do yesterday?

Thomas: Got available bike and stands numbers to appear on infowindows for map markers.
Kevin: N/A
Conor: N/A

What will you do today?
Thomas: Finish the infowindows.
Kevin: Started work on dropdown menu
Conor: Updated backlog and burndown charts

Is there anything in your way?
Thomas: Time constraints
Kevin: N/A
Conor: Illness

Standup: Tuesday 03/03/2020: 10:45am (B106 CSI)
Sprint day 7 of 10

What did you do yesterday?
Thomas: Finished infowindows (showing available bike stands etc.)
Kevin: Worked on dropdown menu (complete, may need alteration)
Conor: Updated backlog and burndown charts

What will you do today?
Thomas: Link dropdown on-click event with database
Kevin: Displaying weather data from the sql database on the page
Conor: Further documentation work, found mistake in sprint 1 documentation that needs to be updated/changed

Is there anything in your way?
Thomas: Time constraints
Kevin: Time constraints
Conor: Illness

Standup: Wednesday 04/03/2020: 1:30pm (carried out online via Slack)
Sprint day 8 of 10

What did you do yesterday?
Thomas: Dropdown on-click events linked to database
Kevin: Display of weather data from the sql database on the page
Conor: N/A

What will you do today?
Thomas, Kevin: Extra Flask research
Conor: Further documentation work

Is there anything in your way?
Thomas: Time constraints
Kevin: Time constraints
Conor: Illness, Time constraints

Standup: Thursday 05/03/2020: 9:30am (Science East E1.17)
Sprint day 9 of 10

What did you do yesterday?
Thomas, Kevin: Finalised Flask research
Conor: N/A

What will you do today?
Thomas and Kevin: Devise list of new features to work on over mid term break
Kevin: Work on geo-location (to nearest stations)
Conor: Further documentation work

Is there anything in your way?
Thomas: Time constraints
Kevin: Time constraints
Conor: Illness, Time constraints

Standup: Friday 06/03/2020: 4:30pm (carried out online via Slack)
Sprint day 10 of 10

What did you do yesterday?
Thomas and Kevin: Created new tasks to extend Sprint 2 into midterm break
Kevin: Started work on basic geo-location feature (more functionality to be added in future)
Conor: Further documentation work

What will you do today?
Thomas: Create Google Map Marker colours which tell you the percentage of total bikes available at a station
Kevin: Finish work on basic geo-location
Conor: Final documentation work for end of Sprint

Is there anything in your way?
Thomas, Kevin, Conor: Java Programming Examination

Sprint 3

There were no sprint meetings on the first day, nor the final three days of this sprint, as mentioned in the process section of the report.

Monday 23/02/2020: No Standup Meeting

Standup: Tuesday 24/03/2020: 9:30am (Google Hangouts)
Sprint day 2 of 10

- What did you do yesterday?

Thomas: N/A
Kevin: N/A
Conor: N/A

- What will you do today?

Thomas: Begin research on charts and graphs for showing bike availability
Kevin: Work on geolocation: directions to nearest station
Conor: Finalise + push Sprint 2 Documentation to GitHub, populate backlog chart for Sprint 3

- Is there anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Wednesday 25/03/2020: 4:30pm (Slack)
Sprint day 3 of 10

- What did you do yesterday?

Thomas: Research into charts/graphs for front end
Kevin: Worked on geolocation: directions to nearest station
Conor: Finalised Sprint 2 Documentation, created Sprint 3 Backlog

- What will you do today?

Thomas: Work on chart creation using Google Charts
Kevin: Work on DarkSky weather icons
Thomas + Kevin: Change map marker design colours (see Sprint 3 Meeting 2)
Conor: Compile meeting and Standup notes for Sprint 3 so far

- Is there anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Thursday 26/03/2020: 9:30am (Google Hangouts)
Sprint day 4 of 10

- What did you do yesterday?

Thomas: Developed Chart for daily bike availability.
Kevin: DarkSky weather icons
Conor: Compiled meeting and Standup notes for Sprint 3 so far

- What will you do today?

Thomas: Begin developing chart for hourly bike availability
Kevin: Geolocation: directions to nearest station + modal design
Conor: Created burndown graph for Sprint 3 so far

- Is there anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Friday 27/03/2020: 11:30am (Slack)
Sprint day 5 of 10

- What did you do yesterday?

Thomas: Worked on chart for hourly bike availability.
Kevin: Finished work on geolocation: directions to nearest station + added real-time clock to user interface
Conor: Compiled meeting and Standup notes for Sprint 3 so far

- What will you do today?

Thomas: Finish bike availability chart, worked on overall chart design
Kevin: Add directions to nearest station functionality, clean up weather information.
Conor: N/A

- Is there anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Monday 30/03/2020: 5pm (Slack)
Sprint day 6 of 10

- What did you do yesterday (last Friday)?

Thomas: Worked on bike availability charts + design

Kevin: Added directions to nearest station functionality, cleaned up the weather information display (css).
Conor: N/A

- What will you do today?

Thomas: Begin data analytics research
Kevin: Clean up modal display, add functionality to give directions to station from dropdown menu
Conor: Updated standup notes and burndown graph

- Is there anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Monday 30/03/2020: 5pm (Slack)
Sprint day 7 of 10

- What did you do yesterday)?

Thomas: Started data analytics research
Kevin: Added directions to station from dropdown function
Conor: Updated standup notes and burndown graph

- What will you do today?

Thomas: Further data analytics research
Kevin: N/A
Conor: Further documentation work

- Is there anything in your way?

Thomas: Other modules
Kevin: Other modules
Conor: Other modules

Sprint 4

There was a standup meeting every single day during Sprint 4 for the first time over the course of the project.

Standup: Monday 06/04/2020: 5pm (Slack)
Sprint day 1 of 10

- What did you do yesterday?

Thomas: Merged bike and weather dataframes in jupyter notebook (for data analytics)
Kevin: N/A
Conor: N/A

- What will you do today?

Thomas: Try to find a machine learning model that works for our data
Kevin: Clean up display on user interface
Conor: Finalising Sprint 3 Documentation

- Is there anything in your way?

Thomas: Difficulty finding patterns/relationships in data
Kevin: N/A
Conor: Sparse information regarding end of last sprint: will consult team

Standup: Tuesday 07/04/2020: 9:30am (Google Hangouts)
Sprint day 2 of 10

- What did you do yesterday?

Thomas: Used random forest machine learning model with reasonable success
Kevin: Cleaned up user interface display
Conor: Finalising earlier documentation

- What will you do today?

Thomas: Continue to work with random forest model and see how accurate it is
Kevin: Work on bike-lane overlay and ‘Dark Mode’
Conor: Compiled Sprint 4 Meeting 1 notes

- Is there anything in your way?

Thomas: Lack of knowledge of random forest machine learning model
Kevin: Issues with database; large number of connections
Conor: N/A

Standup: Wednesday 08/04/2020: 5:30pm (Slack)
Sprint day 3 of 10

- What did you do yesterday?

Thomas: Completed Machine Learning Model (random forest)
Kevin: Fix information display
Conor: Compiled meeting notes

- What will you do today?

Thomas: Incorporate model into the front-end of the app.
Kevin: Finish bike lane overlay and dark mode. Look into the use of cookies to save user information
Conor: Begin project report

- Is there anything in your way?

Thomas: Not sure how to carry this out, requires research
Kevin: N/A
Conor: Not enough info regarding report layout, length etc.

Standup: Thursday 09/04/2020: 9:30am (Google Hangouts)
Sprint day 4 of 10

- What did you do yesterday?

Thomas: Incorporated machine learning component to front end
Kevin: Finished basic versions of bike lane overlay and dark mode.
Conor: Compiled meeting notes

- What will you do today?

Thomas: Try to push data analytics model to github (file too large)
Kevin: Fix problems with bike lane overlay and dark mode
Conor: Begin project report

- Is there anything in your way?

Thomas: Time constraints: beginning to spend much more time on project than planned
Kevin: Time constraints
Conor: Time constraints

Standup: Friday 10/04/2020: 11am (Slack)
Sprint day 5 of 10

- What did you do yesterday?

Thomas: Found out how to push large files to github using git LFS. Sorted out issues with closing connections to database
Kevin: Got bicycle layer and dark mode working properly
Conor: Started report

- What will you do today?

Thomas: Add comments to all of the code
Kevin: Have a look at cookies and how to implement them
Conor: Compile documentation for sprint 4 so far, work on report

- Is there anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Standup: Monday 13/04/2020: 5pm (Slack)
Sprint day 6 of 10

- What did you do yesterday?

Thomas: Added some comments to code
Kevin: Did some research into cookies
Conor: Worked on report

- What will you do today?

Thomas: Deploy app to EC2
Kevin: Fix html issues
Conor: Report

- Is there anything in your way?

Thomas: Learning process of app deployment
Kevin: N/A
Conor: N/A

Standup: Tuesday 14/04/2020: 9:30am (Google Hangouts)
Sprint day 7 of 10

- What did you do yesterday?

Thomas: Deployed app on EC2 using nginx and gunicorn. Secured domain name www.dublinbikes2020.comfrom godaddy.com
Kevin: Did some research into cookies
Conor: Report, meeting docs for sprint meeting 3

- What will you do today?

Thomas: Work on SQL queries to improve speed, try to sort out issues with nameservers.
Kevin:
Conor: Report

- Is there anything in your way?

Thomas: Learning process of app deployment
Kevin: N/A
Conor: N/A

Standup: Wednesday 15/04/2020: 5pm (Slack)
Sprint day 8 of 10

- What did you do yesterday?

Thomas: Set up nameservers for domain to point to AWS nameservers. Added created_at column to dynamic bike database to improve loading times from database, thereby improving map marker loading times. Added indexing to tables to increase chart loading speed

Kevin: Did some research into cookies
Conor: Report, meeting docs

- What will you do today?

Thomas: Add comments to code, clean code
Kevin: Fix modal design for popup windows on deployed app, fix wind speed + rain probability units
Conor: Report, burndown chart sprint 4

- Is there anything in your way?

Thomas: Learning process of app deployment
Kevin: N/A
Conor: N/A

Standup: Thursday 16/04/2020: 9:30am (Google Hangouts)
Sprint day 9 of 10

- What did you do yesterday?

Thomas: Cleaning up code, commenting code
Kevin: Made changes to html/css page designn, fixed modal design for popup windows on deployed app, fixed wind speed + rain probability units
Conor: Report, burndown chart

- What will you do today?

Thomas: Fix issues arising from deployment of app to EC2
Kevin: Fix directions function for station chosen from dropdown (Flask changes), add available bikes/stands information to sidebar when a station is chosen
Conor: Report, meeting docs for final meeting

- Is there any problems/anything in your way?

Thomas: N/A
Kevin: Slow loading times from database, fixed with change to an SQL query.
Conor: N/A

Standup: Friday 17/04/2020: 11am (Slack)
Sprint day 10 of 10

- What did you do yesterday?

Thomas: Changed machine learning mechanism from pickle to joblib, which improved app performance
Kevin: Added available bikes/stands information to sidebar when a station is chosen
Conor: Report, meeting notes

- What will you do today?

Thomas: Add comments to frontend code (html, javascript) and main application
Kevin: Further work on directions to bike stations function
Conor: Report

- Is there any problems/anything in your way?

Thomas: N/A
Kevin: N/A
Conor: N/A

Sprint Reviews and Retrospective Notes

These are a few short notes we took from the review and retrospective meetings at the end of each sprint. Most of this appears in a modified form in the Process section of the report, except for a few notes at the end of each retrospective, which outline a high level plan for the following sprint.

REVIEW
SPRINT 1: DAY 10 OF 10
21/02/2020

WHAT DID WE GET DONE
Set up data scrapers to collect data at fixed intervals
- Bike (dynamic), Bike (static), Weather (dynamic)
Set up the databases for each data scraper on RDS instance
- used MySQL to view the scraped data, accessed through Workbench
Scheduled dynamic scrapers with chrontab to update tables:
- every 5 mins (bike dynamic)
- every hour (weather dynamic)
Bike static – explains coordinates of each station, name of station, stuff that doesn't change etc.
Set up basic flask application, integrated Google maps API into Flask
Created markers with static data for the Google map
Defined overall project architecture + wireframe

WHAT DID WE NOT GET DONE
We did not succeed in getting the static information for the google map showing in the Flask app
Sufficient Flask research: not yet fully comfortable with Flask and how it works

BURNDOWN/BACKLOG ANALYSIS
Behind on Burndown for first week: Lack of daily standups and overall lack of direction in project caused us to fall behind schedule at an early time
On average, the prediction for how many hours to spend on the project altogether during the first sprint was reasonably accurate, even though some tasks took longer or shorter than expected
During first week we had a rough idea we were going to allocate 30 hours to the first sprint, but we did not yet know which way most of those hours would be divided up

Overall project research was an idea at the beginning of the sprint, but research sections were soon split so that pieces of research were done for specific elements of the project, such as Flask and Weather APIs.

RETROSPECTIVE
SPRINT 1: DAY 10 OF 10
21/02/2020

WHAT WORKED
Organisation in the second week: we brought the burndown rate back to an acceptable level through better organisation and more hours contributed to the project overall.
Trello and Slack: great for organisation
Github compatability: everybody comfortable with pushing and pulling files to and from a shared github repository devoted to the project
Meetings with Product owner (Karl) in the second week worked for helping us with project management and set us up for meeting deadlines.

WHAT DIDN'T WORK
Some of our predictions regarding how long certain things were going to take: some took longer than required, some shorter (i.e. setting up the basic Flask application took much less time than predicted, but further research and adding extra functionality is taking a long time and is not yet complete
Organisation and coherent teamwork during the first week: we met at practical times but did not conduct daily standups; this contributed to delay in work and falling behind in hours overall by the end of the first half of the sprint

HIGH-LEVEL PLAN FOR NEXT SPRINT
Get google map working properly
Adding to the flask frame
Begin creating interface for weather
Get what is showing on the dynamic weatherframe (current weather) showing up
(Get database linked up with the numbers (latitude and longitude))
Get map parkers linked up with the database positions)
Get basic application running.

REVIEW
SPRINT 2: DAY 10 OF 10
06/03/2020

WHAT DID WE GET DONE
- Completed our research on how to use Flask.
- Created a basic HTML page with divs corresponding to mockup features.
- Connected Flask application to database.
- Successfully displayed JSON data on webpage
- Created Google Maps markers for each bike station.
- Had station data displayed through an info window which is triggered by on-click event.
- Created a dropdown menu with a list of all stations.
- Created information div which displays data for station that is selected in dropdown.
- Added colours to map markers based on bike availability. (Red <= 25%, Orange <= 50%, Green > 50%)
- Creation of geolocation feature which calculates the nearest station to the user.
- Improved design and styling of data in marker info windows.

WHAT DID WE NOT GET DONE
- Did not get data analytics/graph section set up.
- Didn’t get to add icons, etc to weather data.
- User-flow diagram: this was attempted, but wasn’t yet possible as not enough of the application was finished. This will be pushed forward to Sprint 3.

DID ANYTHING GO WRONG?
We ran into a data scraping issue with our weather scraper; on inspection on Workbench we saw that the scraper was not populating the table for every hour. We discovered that this was due to an issue with one of the data fields from the API which contained a null value for the rain-type column which tells you whether the rain is heavy or light. Whenever it wasn’t raining the value returned was a null value, which somehow invalidated the entire rows affected by the null value so that they did not contain any data at all. The issue was identified through running the responsible script manually outside of the application itself, and it was resolved simply by removing the rain type column, so that the only information displayed in the table now relates to whether there is rain or no rain, with no information about the type of rain.

BURNDOWN/BACKLOG ANALYSIS

Behind on Burndown on first day due to first meeting and sprint planning taking place on Day 2 during practical time.
On average, the prediction for how many hours to spend on the project altogether during the first sprint was accurate for most tasks, but understimated for others (it took us 31 hours to complete work that overall we thought would take 26 hours).
This time we allocated all of our tasks and most of our predicted hours at the very beginning of the sprint, while some were added halfway through the sprint after we had more clarity on our position and progress
Due to the midterm break, we were going to add further features to the product as part of an extension of Sprint 2, but due to reasons outside of our control only one of these new ideas was completed (adding colours to location markers based on bike availability). Therefore, these extra features will be implemented during Sprint 3.

RETROSPECTIVE
SPRINT 2: DAY 10 OF 10
06/03/2020

WHAT WORKED

- Organisation for most of the sprint: We seemed to plan our time well for this sprint and stayed on track with the product owner’s expectations. We tried to do as much as possible early in the sprint so that we would be under less pressure during the final days of the sprint.
- We managed to get the majority of the tasks for this sprint completed on time. Coming into the study break, we decided to extend the sprint in order to work on some extra features. This included the geolocation feature and the colour coding of station markers based on bike availability.

WHAT DIDN’T WORK

- Predictions on how long it should take to do certain tasks were underestimated, meaning that more hours were required in a sprint during which which many external factors (i.e. assignments and examinations in other modules) added hugely to our overall workload
- The extension of the sprint into the midterm break, while leading to the completion of a couple of features, was a failure overall, mostly due to circumstances outside our control, but also partly due to a lack of clarity of purpose and less communication between team members as each of us had returned home from UCD for the midterm break.

HIGH-LEVEL PLAN FOR NEXT SPRINT

The plan for the next sprint at the moment consists mostly of adding features that we failed to complete during the extension of this sprint:

- Data Analytics graphs
- Display DarkSky weather icons
- Clean up display of weather data
- Modal Design (giving an onclick function to each marker on the map
- User-flow diagram after all intended functionality is added to the application
- Decide on extra features to be added with guidance from Karl
- Extra functionality for geo-location: be able to give the user directions to a station

REVIEW
SPRINT 3: DAY 10 OF 10
03/04/2020

WHAT DID WE GET DONE
- Completed work on geolocation/directions to nearest station to user
- Directions to any station on dropdown menu
- Graphs depicting both daily and hourly station data (i.e. how many bikes)
- Styling of graphs
- DarkSky weather icons
- Design of onclick map markers
- Cleaned up display of weather data
- Research into Data Analytics section of project
- User-flow diagram for app with full functionality

WHAT DID WE NOT GET DONE
- Data Analytics work: no model or interface completed
- Finalisation of text colours and text layout for onclick modal
- The providing of an option to allow the user to display either the daily or hourly data for a specific station from the dropdown menu

DID ANYTHING GO WRONG?
We ran into difficulty with our data analytics work: the research on what we were supposed to do was completed, but we were unable to implement our model and interface in the permitted time for this sprint. This work will be pushed forward into Sprint 4.

BURNDOWN/BACKLOG ANALYSIS
Behind on Burndown on first day due to first meeting and sprint planning taking place on Day 2 during practical time.
We did not complete all planned tasks for this sprint. This was an ambitious sprint as we had dedicated 40 hours worth of tasks to be completed, but we only completed 75% of this work, with about 10 hours' worth of work having to be pushed back to the final sprint. There are a number of reasons for this failure to keep schedule:
- a lack of overall project organisaton during the second week of the sprint
- it took longer than envisaged to carry out many of the tasks that we did complete (it took us 37 hours to do 30 hours' worth of work, as seen in the backlog chart)
- the data analytics work in this sprint was unsuccessful.

RETROSPECTIVE
SPRINT 3: DAY 10 OF 10
03/04/2020

WHAT WORKED

- Organisation for first week of the sprint: Despite having delegated more hours of work than usual we were well ahead of schedule by the middle of the sprint (end of day 5).As with previous sprints, wetried to do as much as possible early in the sprint so that we would be under less pressure during the final days of the sprint.
- We managed to get a lot of work done in this sprint as we had allocated more hours and completed more tasks in Sprint 3 than we did in previous sprints.

WHAT DIDN'T WORK

- As with all of the previous sprints, predictions on how long it should take to do certain tasks were underestimated: we did 37 hours of work when 40 hours were allocated, but only completed 30 hours worth of tasks, accounting for just 75% of the allocated workload.
- Communication during second half of the sprint: the lack of communication at the end of the spring caused by external factors meant that we would start off Sprint 4 on the back foot.
- Overall organisation of tasks with Trello: the tasks were set up appropriately, but there was little or no activity on the trello board in week 2.

HIGH-LEVEL PLAN FOR NEXT SPRINT

The plan for the next sprint includes some of the data analytics work we failed to complete during this sprint, as well as some finalisations of the overall project:

- Data Analytics model
- Data Analytics interface
- Clean up overall presentation of UI
- Get Flask app running on EC2.
- If we have time: set up SSL certificate on EC2 so app user can share their geo-location. If not, hardcode a city centre location to show functionality of geolocation feature.

REVIEW
SPRINT 4: DAY 10 OF 10
17/04/2020

WHAT DID WE GET DONE
- Finished the data analytics model
- Implemented the model in the front-end
- Cleaned up user interface to an acceptable degree
- Implemented bike lane overlay and Dark Mode on the google map
- Deployed the application to EC2
- Added comments to code
- Cleaned up code
- Improved map marker loading speed by adding extra column in tables
- Improved chart loading speed by indexing tables
- Removed problems occurring as result of deployment of app to EC2

WHAT DID WE NOT GET DONE
- Implementation of SSL certificate for geo-location feature; hardcoded user location implemented instead
- Cookies feature: we began researching it, but hadn't started implementing it

DID ANYTHING GO WRONG?
Problems with application after deployment to EC2: all of these ironed out before deadline (more details in project report, most of which was written before this sprint review/retrospective

RETROSPECTIVE
SPRINT 4: DAY 10 OF 10
17/04/2020

WHAT WORKED
- Work ethic: we were more focused on the end goal in this sprint than in any of the previous sprints.
- The decision to begin working on the sprint on day 1 rather than day 2: we were in the zone in terms of project work before the first meeting even took place

WHAT DIDN'T WORK
- Time management: we spent far more time on the tasks for this sprint that originally intended