



COMP30670 Software Engineering

OnYourBike

Students:

Alan Treanor, Student Number 11288396

Sheena Davitt, Student Number 97133809.

Thomas Anderson, Student Number 17202065

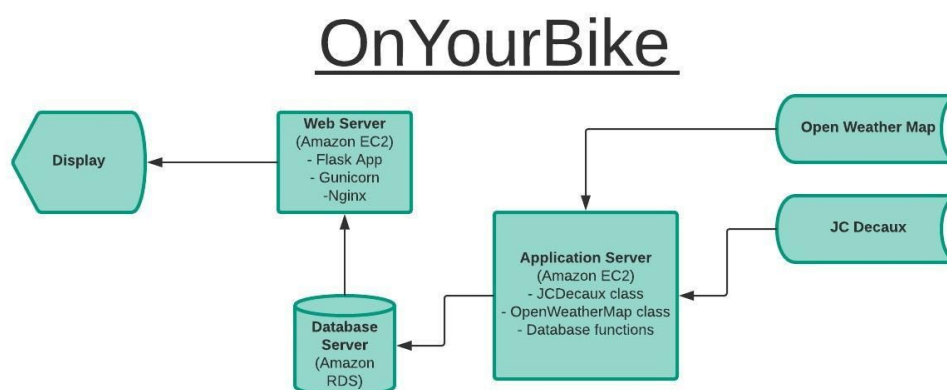
### Sprint 3:

# Project Management Report

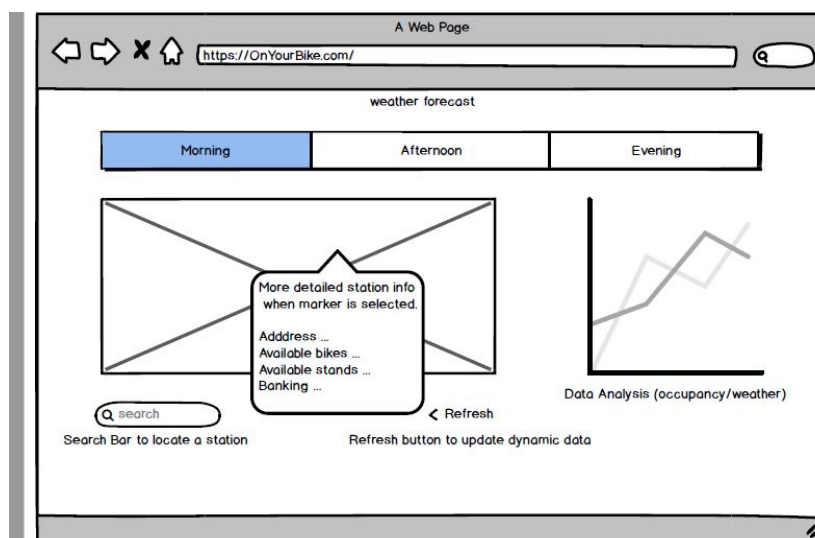
## 1. Design, planning notes and materials

This is the design template that we created for the entire project displaying how all of the elements fit together. We drew this up at the early stages in order to help us to visualise what is a very large project with many moving parts.

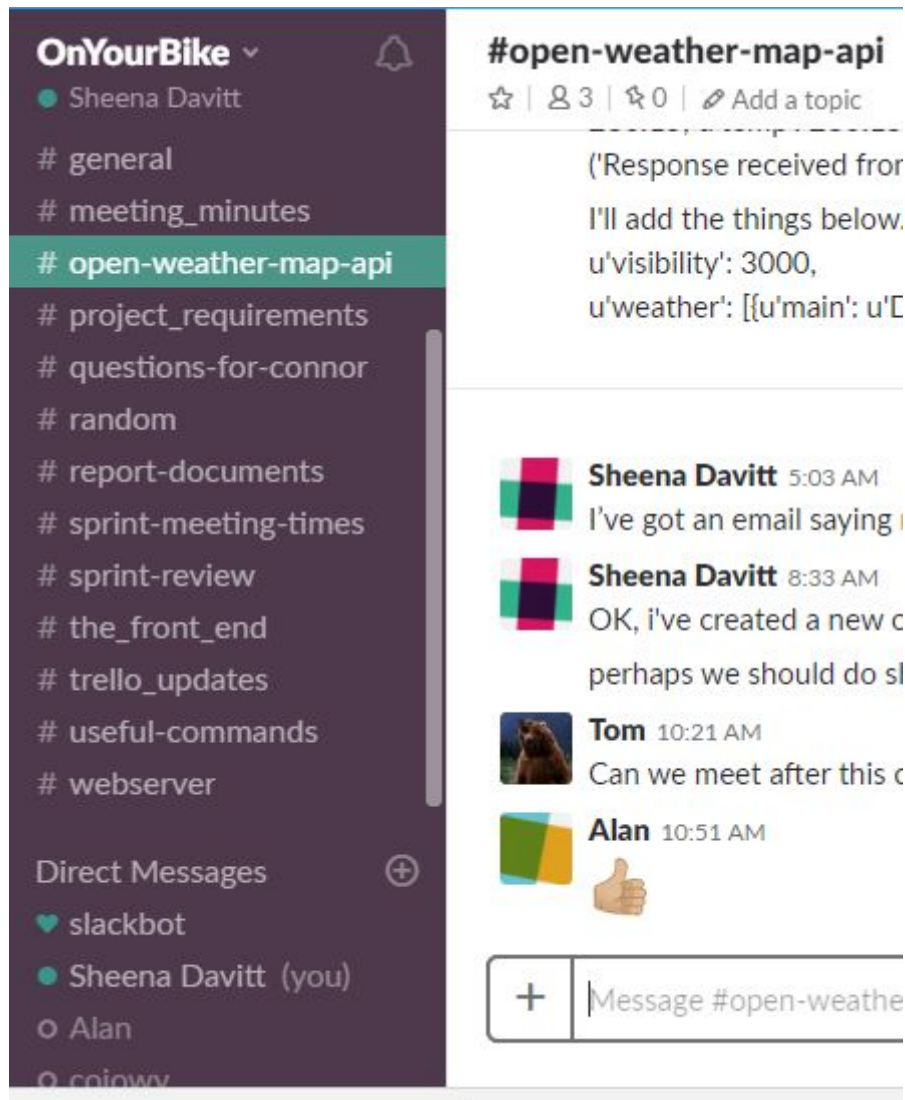
In the template, below, data is gathered by api requests from OpenWeatherMap and from JCDecaux, then processed in the Application Server, inserts the data into the appropriate tables in the RDS database. The Web server then fetches the most recent data from the RDS tables using SQL requests and, jsonifies the results and then passes them to the front end, where they are rendered as web pages with javascript features.



Below is how we originally envisioned the main web page, with weather data, a map, a google search bar and charts. As you can see from section five, our final web page was not exactly as we first planned it, but this design, created in Balsamiq, was a useful starting point on which we could build.

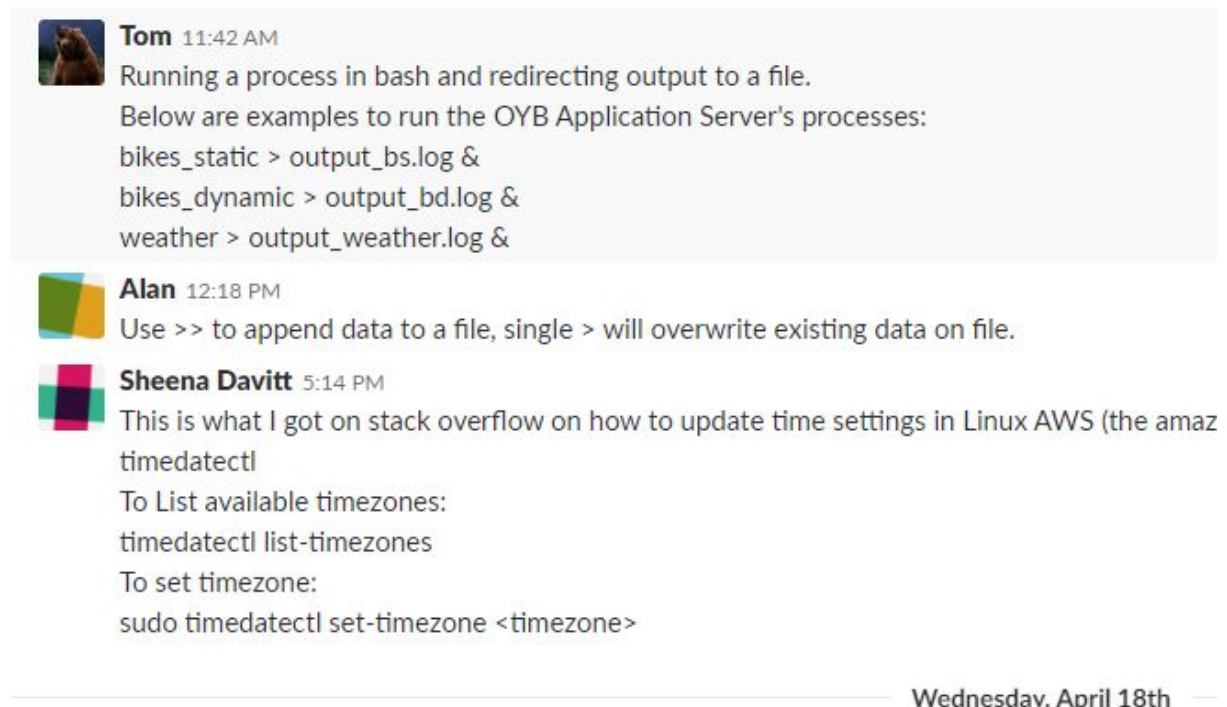


We also used Slack to share materials and resources (eg images, files, api keys etc). Below is a screenshot showing how we created different themed channels so that we could have a way of keeping all these resources in one place, stored by subject so we could find them at a later time. We also used it to raise issues when we were not in a position to discuss matters face to face and to arrange our scrum meetings.



Probably the most helpful channel was the 'useful commands', where we posted bash commands we used for a range of issues, such as installing packages or manipulating the EC2 instances, or SQL statements for changing tables. We were able to access these commands over and over in the course of the four sprints. It was a useful resource in compiling the readme.txt file on the final day.

See below for an example of our posts in the 'useful commands' channel.



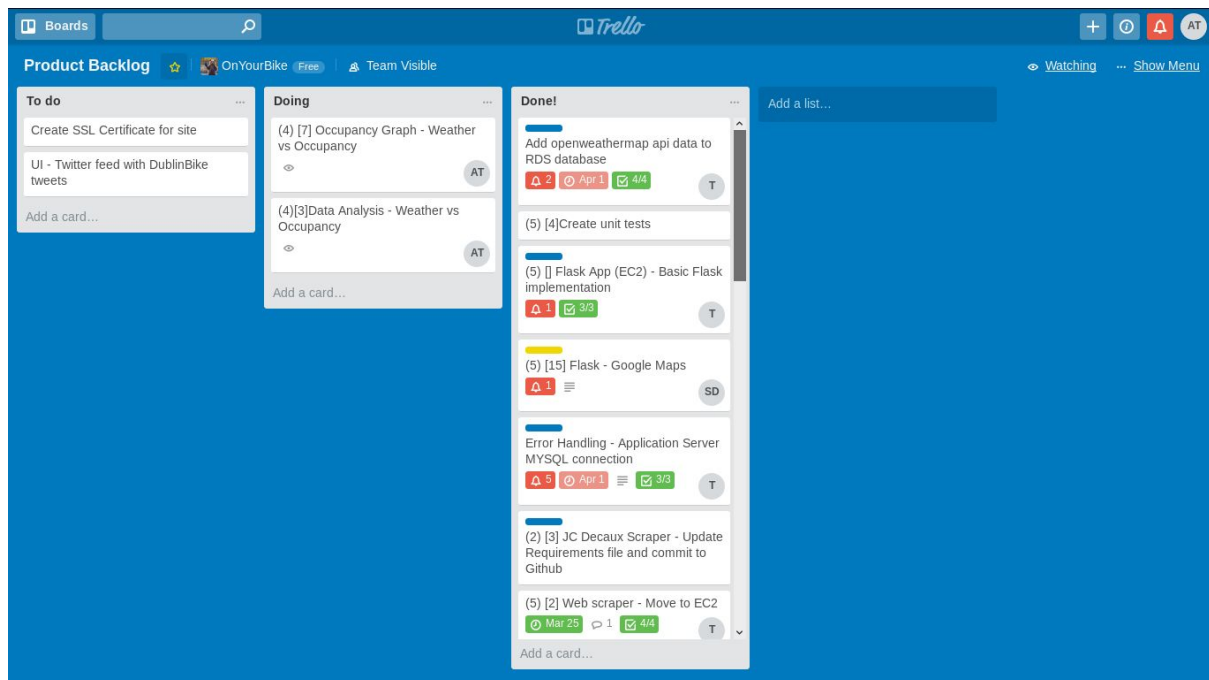
## 2. Scrum project management

The final product backlog can be seen below. We managed to complete the majority of our features with the data analysis and occupancy graph being the only 2 features that were not deployed. The other 2 features of create SSL certificate and Twitter feed were additional features we would have added given we had time leftover at the end of the project.

We followed the scrum project management throughout, holding regular scrum meetings to keep abreast of what stage each feature was at, who is doing which feature. Reviewing each sprint at the end of the week also gave us ideas as to how to improve our approach for the following week.

We found the regular scrum meetings very effective in terms of making sure that no work was ever duplicated - and also, if someone encountered a problem, the other team could offer assistance or advice.

## Product Backlog:



As mentioned above, the team used Slack for communication during the project, this was a useful application as it allowed us to break our conversations down into threads relating to different areas of the project. If we needed to review something from a previous conversation we could simply go to the related Slack thread and scroll through the conversation to find the information we required. The alternative to this would have been a single channel messaging service such as Whatsapp or Facebook Messenger, both are good services but a single channel would have quickly become busy and confusing to find relevant information. The screenshots below give an example of some of the channels we used.

## Web Server Channel:

OnYourBike

Alan

# flask-app

# general

# meeting\_minutes

# open-weather-map-api

# project\_requirements

# questions-for-connor

# random

# report-documents

# sprint-meeting-times

# sprint-review

# the\_front\_end

# trello\_updates

# useful-commands

# webserver

Direct Messages

slackbot

Alan (you)

cojowy

Sheena Davitt

Tom

Tom, Sheena Davitt

+ Invite People

Apps

#webserver

3 | 0 | Add a topic

Wednesday, March 28th

Tom 8:24 AM

<https://www.digitalocean.com/community/tutorials/how-to-set-up-password-authentication-with-nginx-on-ubuntu-14-04> (edited)

Thursday, March 29th

Alan 8:49 AM

Web server ssh access: [ec2-34-223-225-36.us-west-2.compute.amazonaws.com](https://ec2-34-223-225-36.us-west-2.compute.amazonaws.com)

Alan 8:50 AM

added this Plain Text snippet: [OnYourBikeDB.pem](#)

```
1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEpAIBAAKCAQEAwJZ4xh19FbK/ChZvqXXVJpzbFvB5zuVlJk3XHU3301gQ0P0vpNLInMmXQ1e
3 ZP19axTg0pvtQ80vho4huq+I1k55SzCN54s+19MqcIZ50VF4RGECo0vQ5Bqg0THTRHy0BTG4qS0
4 6DpLhXJPGH1rhWaxQ0WwZ1LVCKoPL8TN8rbLBcPSEhcy5BvoJmm+CyGcx80TNMhqNKEGTFxap/4
5 u1cdyyIHaztV69xZvc3HHzVcVBWx31z1QTkmAG39ZVg62+2E5KLCrKfzYEVSnFv+MxyY5t1Q03UW
6 -----
```

Alan 10:13 AM

To start Nginx: [ `sudo service nginx start` ]

The endpoint address used to ssh to this account can be used to access the server from web.

Friday, March 30th

Alan 10:43 AM

SERVER INSTALLATION INSTRUCTIONS

`sudo vi /etc/apt/sources.list`

Add the following lines at bottom of file:

`deb http://nginx.org/packages/mainline/ubuntu yaketty nginx`

+ Message #webserver

## Open Weather Map API Channel:

OnYourBike

Alan

# flask-app

# general

# meeting\_minutes

# open-weather-map-api

# project\_requirements

# questions-for-connor

# random

# report-documents

# sprint-meeting-times

# sprint-review

# the\_front\_end

# trello\_updates

# useful-commands

# webserver

Direct Messages

slackbot

Alan (you)

cojowy

Sheena Davitt

Tom

Tom, Sheena Davitt

+ Invite People

Apps


#open-weather-map-api

3 | 0 | Add a topic

Thursday, April 12th

Alan 10:18 AM

uploaded and commented on this image: [Screenshot from 2018-04-12 10-16-09.png](#)



<https://weatherwidget.io/>

Alan 10:18 AM

Above is the weather widget we can use if allowed.

Sheena Davitt 10:54 AM

MY openweather API key is: &APPID=66040549d2cc38abfa2a0be1019ad3b5

Tom 11:39 AM

Thanks. Here is the response we are getting for current weather:

```
{
  "clouds": {
    "all": 90
  },
  "name": "Dublin",
  "visibility": 3000,
  "sys": {
    "country": "IE",
    "sunrise": 1523560848,
    "message": 0.0022,
    "type": 1,
    "id": 5237,
    "sunrise": 1523511107
  },
  "weather": [
    {
      "main": "Drizzle",
      "id": 300,
      "icon": "09d",
      "description": "light intensity drizzle"
    },
    {
      "main": "Mist",
      "id": 701,
      "icon": "50d",
      "description": "mist"
    }
  ],
  "coord": {
    "lat": 53.35,
    "lon": -6.26
  },
  "base": "stations",
  "dt": 1523527200,
  "main": {
    "pressure": 1006,
    "temp_min": 280.15,
    "temp_max": 280.15,
    "temp": 280.15,
    "humidity": 100
  },
  "id": 2964574,
  "wind": {
    "speed": 4.1,
    "deg": 80
  },
  "cod": 200
}
```

(Response received from Open Weather Map.; 200)

+ Message #open-weather-map-api

## Meeting Minutes Channel:

### 3. Meeting Notes for each sprint

#### Sprint 1

**Monday:** We discussed the project requirements and what documentation it would involve. We decided that we needed to produce a plan, and that we would have to use Trello boards in order to list all of the features needed in the project and use Slack to communicate and store resources (commands, files, screenshots etc). We also decided we needed to create a GitHub repository for our code, which we did. We also set up a Slack group and Trello Group. We also created a product backlog and resolved to set up burndown charts. To the Trello board for sprint 1 we added the following features:

- Python code to write to/read from MySQL database
- Prepare a python package (Object Oriented) for a Bike\_Scraper that will fetch data from JCDecaux
- Move the web scraper to EC2
- Create an RDS Instance on EC2
- Create tables on the RDS
- Parse the data from the web scraper

**Tuesday:** We had no major issues. At this stage we were mostly learning how to implement the features chosen for this sprint.

**Wednesday:** Again, no major issues arose and we continued working with our features.



**Thursday:** We had no major issues at this stage. The database code was taking much longer than expected, so this was a minor issue.

**Friday:** No major issues. The delay with the database code was eventually fixed so the following week we would be able to write to the database.

**Project Retrospective:** Alan posted some helpful reflections on Slack about the experience of the first sprint, including the suggestion that we should put comments on all code and try to make it as readable for others as possible, and to seek help from Conor or a demonstrator as early as possible if any serious technical problem emerged.

## Sprint 2

**Monday:** we discussed what we would like to achieve during this sprint and how we felt the previous sprint went. The following features were moved from the product backlog to the sprint 2 board on Trello:

- add Open Weather Map API data to RDS database
- method to retrieve data from database for flask application
- front-end Google maps
- install web server Nginx on EC2 instance
- update requirements file on Github
- handle errors with MySQL connection to application server

**Tuesday:** we discussed how we were proceeding, no issues came up and we were happy to continue with our tasks.

**Wednesday:** the sprint was on plan with no issues arising

**Thursday:** the sprint was on plan with no issues arising. Sheena mentioned that she had been having trouble with getting the markers to display in the map, but that after discussing this with demonstrators and the support centre had eventually got the markers displaying correctly.

**Friday:** the sprint finished up on time and we signed off on the features we had selected from the backlog

**Project Review:** we had two meetings with the project demonstrator during practical sessions and he was happy with the progress we were making.

**Project Retrospective:** we planned to spend more time on other projects in week 3 as we had two assignments due that week. Otherwise there were no changes to be made to the process.



## Sprint 3

**Monday:** We decided that for this sprint we needed to do the following, (each of which was then added to the Trello boards):

- create a diagram to visualise the index.html
- Add a refresh button that will update dynamic data
- Add a google maps search bar to the UI
- Set up a css file
- Get flask and DB query code to talk to each other
- separate the HTML and JS files
- Create a table in the MYSQL database that is populated with the latest data from the JCD\_Dynamic\_Data & OWM
- Start Weather data analysis to produce projections on occupancy
- Update scraper to refresh static data once per day
- edit the light table to include all variables we require at startup

At our scrum meeting, no major issues were reported and we resolved to start implementing the features listed above.

**Tuesday:** One issue emerged - the fact that the RDS database on Alan's EC2 was facing storage issues and his account was temporarily closed. We discussed taking a snapshot of the RDS and migrating it to another instance.

**Wednesday:** We eventually decided to resolve the storage issue by removing some extra elements from Alan's EC2, and by moving the server to Sheena's instance. The RDS was fully functional again. Sheena mentioned that there were also some ongoing delays in getting flask and the database code to talk to each other. We resolved to seek the advice of a demonstrator or the support centre to see if they could offer any new ideas.

**Thursday:** No serious issues. The issue of the database/flask communication was resolved.

**Friday:** No major issues arose. Most of the features we started on were finished by the end of the sprint.

**Project Review:** we met with the project demonstrator during practical sessions and he said he was pleased with our progress.

**Project Retrospective:** We felt that, although much is still to be done, the scrum process was going well and there was no major change needed to the general approach to implementing our features.

## Sprint 4

**Monday:** We decided that for this sprint we needed to do the following, (each of which was then added to the Trello boards):

- Display Current Weather Information

- change link pointers to make most recent version live
- Further design improvements to the front end
- Data Analysis - Weather vs Occupancy
- Unit testing
- Initiation of scraper methods
- write README with installation & file location
- Getting OYB server to run on EC2

At the scrum meeting no major difficulties emerged and we decided to start implementing the features straight away.

**Tuesday:** No major issues at this stage.

**Wednesday:** At our scrum meeting, we discussed how, although the front end is now working (displaying weather info/bike occupancy info on markers etc) on Sheena's laptop, there are major issues with getting it to run on EC2. We resolved to spend time trying out all the various import statements or versions of mysqldb/sqlalchemy (the EC2 error messages were suggesting it was an issue with mysqldb or sql alchemy).

**Thursday:** We realised that there was an issue with displaying charts on the main web page. We decided to create a branch of our GitHub (the master would have the 'working' version), and other branches could be used to try out Google charts without 'breaking' the existing javascript.

**Friday:** We were still having issues with getting the web server to run on EC2 (specifically, it was displaying the web page correctly, but the markers and updated weather info - which require a connection to the database - were not displaying, despite many hours of trying to import every possible related package on the EC2. We resolved to go to the support centre to see if they could help.

### **Final sprint and overall Project Retrospective:**

The most significant thing we learnt in the final week - and in the project as a whole - is that it is easy to underestimate how difficult it is to get the separate parts of such a project to fit together. Coding individual features was one thing - but getting flask to talk to the database, for instance, or getting the working application server to run properly on an EC2 (with a different OS) was a major challenge. So it was interesting to realise that coding takes up only a small fraction of the time involved, while co-ordination of all the moving parts and management of the project is the crucial skill.

We also learned about the importance of different data structures - when to use lists and dictionaries for storing results from search queries. Also, we learnt a lot about different data types and how important it is that they be consistent across database tables, in sql queries and format strings (this is something that we spent a considerable amount of time fixing in response to error messages - but we now know that if we came across this issue again, we would make sure they would be consistent from the very start).

In the final sprint we also learnt that Google charts is an extensive and complicated feature - we did not have time to fully implement it in our project, we just have a basic chart that appears when you click on the 'Graph' button on a station infowindow, but at least we now know more about the feature overall and would be able to implement it in a future project.

In terms of the data analysis, our weather data and bike occupancy data didn't overlap so we were unfortunately unable to implement this feature.

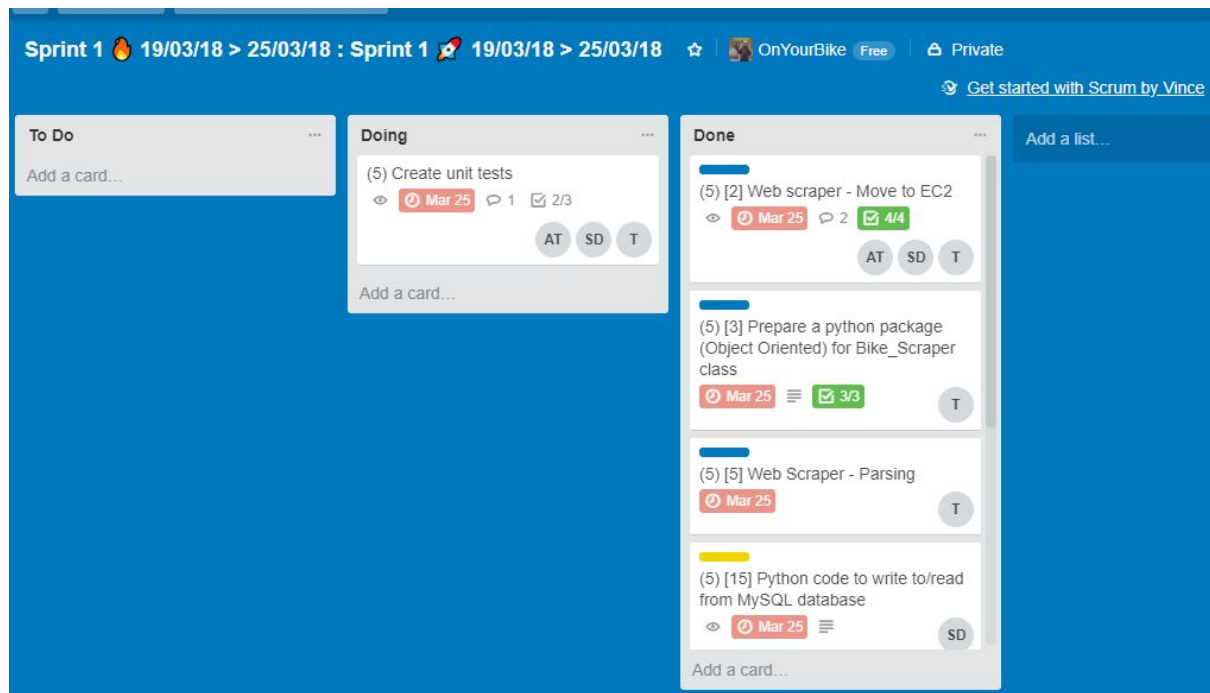
We also learned the importance of branches in GitHub in storing a 'working' version of the application, which can then be altered in new branches without damaging the original (important when using a language like javascript).

We also learned how scrum was a very useful technique for us - especially in a college project, where we are all learning from scratch. We were able to continually learn from each other, to help each other out as soon as any issue arose, and to co-ordinate our efforts so that we could spread the workload fairly and without anyone duplicating another team-member's efforts. We found it particularly useful in the first week, when, during the study break, we met every day and worked together during the day - this got us off to a strong start. Using this technique in a full time project where you have no other assignments must be a very effective way of managing workload and workflow.

#### 4. Feature selection/product backlog for each sprint

Our feature selection moved from the back-end to the front-end and we selected the features that had other features dependent on them being completed to work. This helped prevent any bottlenecks in the management of the project by one individual waiting on another to finish their feature before they could proceed.

##### Sprint 1:



## Sprint 2:

The screenshot shows a Scrum board for 'OnYourBike : Sprint 2' with a date range of 26/03/18 to 01/04/18. The board is divided into three columns: 'To Do', 'Done', and 'Done'. The background features a scenic view of snow-capped mountains and evergreen trees.

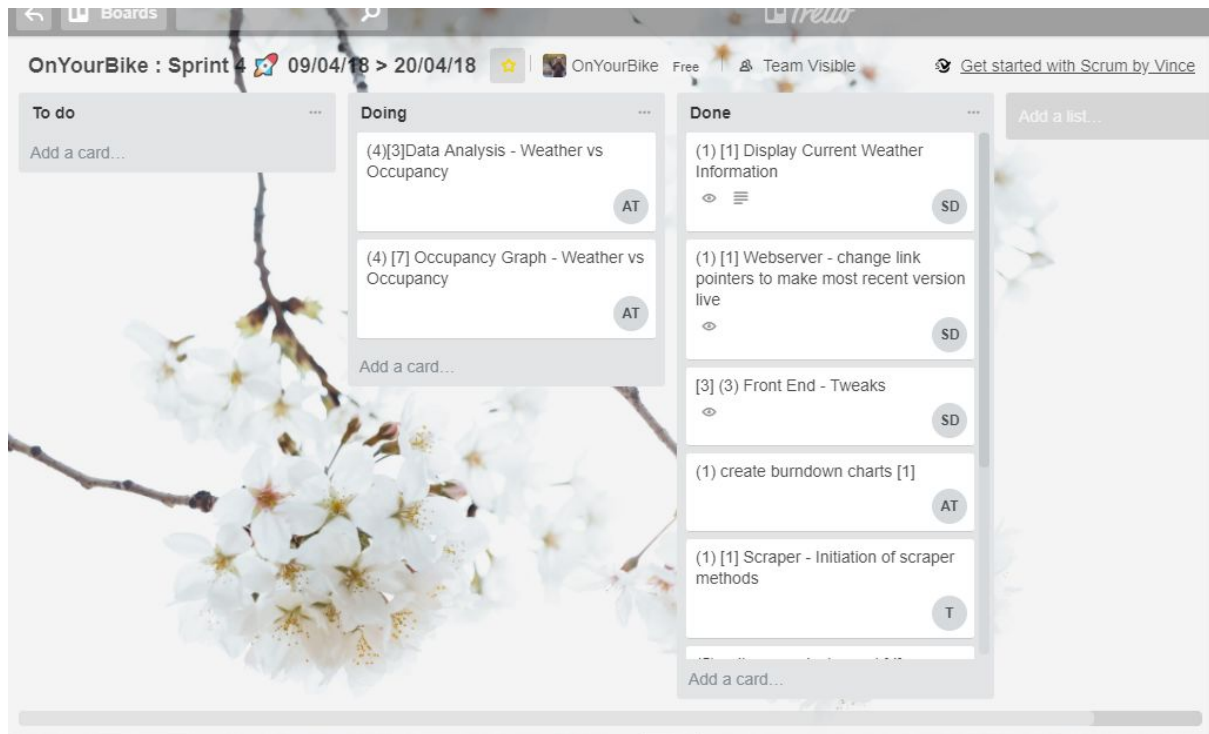
- To Do:** Contains one card with the text 'Add a card...'.
- Done:** Contains one card with the text 'Add a card...'.
- Done:** Contains four cards:
  - Card 1: '(2) [3] JC Decaux Scraper - Update Requirements file and commit to Github' with a progress bar and a 'T' status icon.
  - Card 2: '(5) [5] Add openweathermap api data to RDS database' with a progress bar and a 'T' status icon.
  - Card 3: '(5) [4] Methods to retrieve data from database for flask application on web server' with a progress bar, a comment icon showing '1', and an 'AT' status icon.
  - Card 4: '(5) [15] Flask - Google Maps' with a progress bar, an eye icon, and an 'SD' status icon.

## Sprint 3:

The screenshot shows a Scrum board for 'OnYourBike : Sprint 3' with a date range of 02/04/18 to 08/04/18. The board is divided into four columns: 'To Do', 'Doing', 'Done', and 'Done'. The background features a warm, orange-toned landscape with rolling hills.

- To Do:** Contains four cards:
  - 'Weather data analysis to produce projections on occupancy'
  - 'JD Decaux - Update scraper to refresh static data once per day'
  - 'UI - Twitter feed with DublinBike tweets'
  - 'Create SSL Certificate for site'
- Doing:** Contains two cards:
  - 'UI - graph for occupancy information when station is selected'
  - 'UI - weather information'
- Done:** Contains one card with the text 'Add a card...'.
- Done:** Contains five cards:
  - Card 1: '(1) [1] UI - create a diagram to visualise the index.html' with a progress bar and an 'AT' status icon.
  - Card 2: '(1) [2] UI - refresh button to update dynamic data' with a progress bar, an eye icon, and an 'SD' status icon.
  - Card 3: '(1) [2] UI - google maps search bar' with a progress bar, an eye icon, and an 'SD' status icon.
  - Card 4: '(1)[1] set up css file' with a progress bar, an eye icon, and an 'SD' status icon.
  - Card 5: '(1) [1] Database - edit lite table to include all variables we require at start up' with a progress bar.

## Sprint 4:

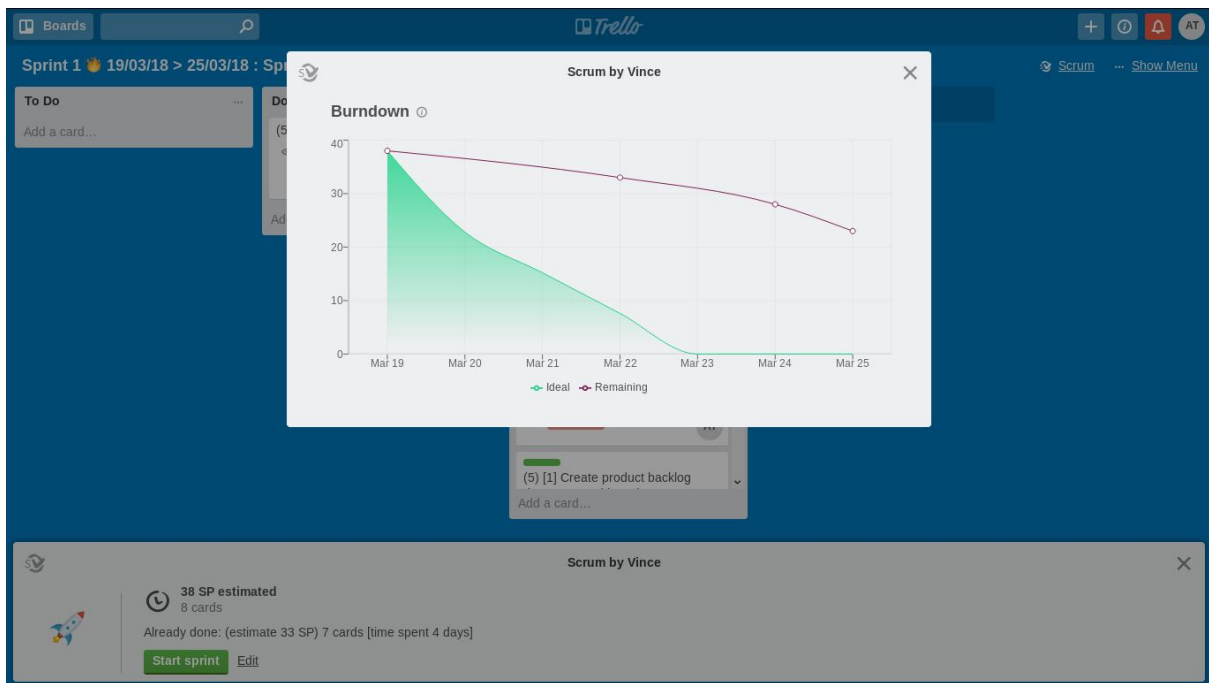


## 5. Burndown Charts

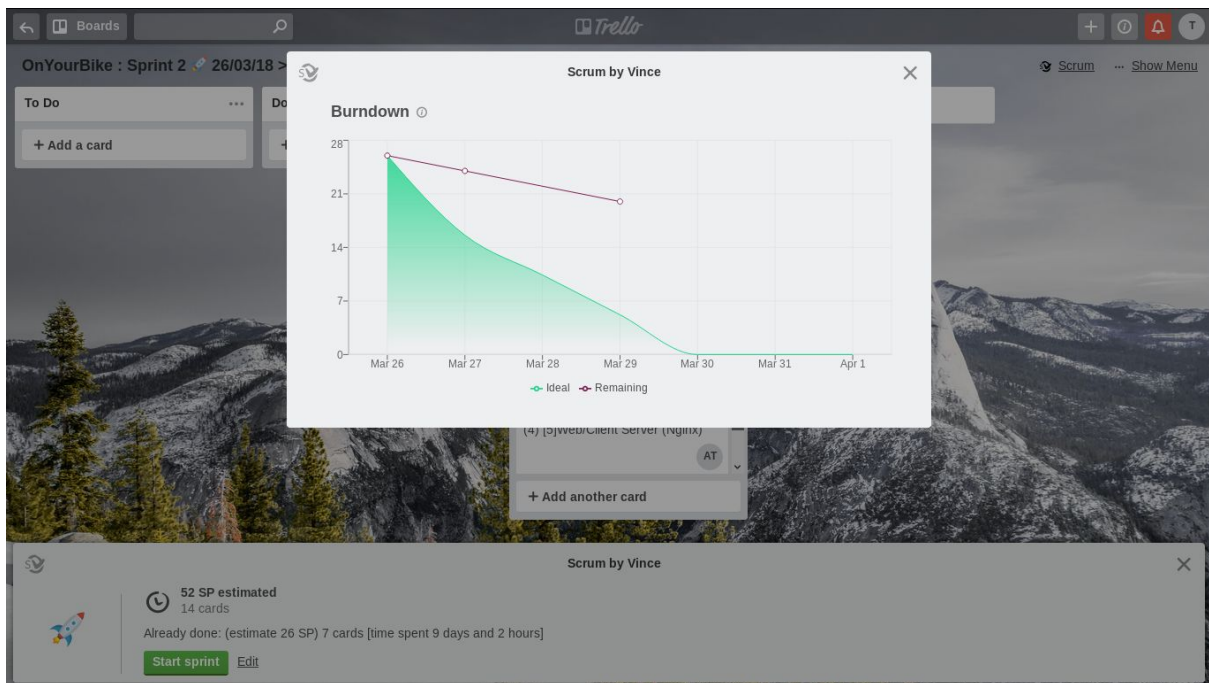
### Overall Burndown Chart:



## Sprint 1:

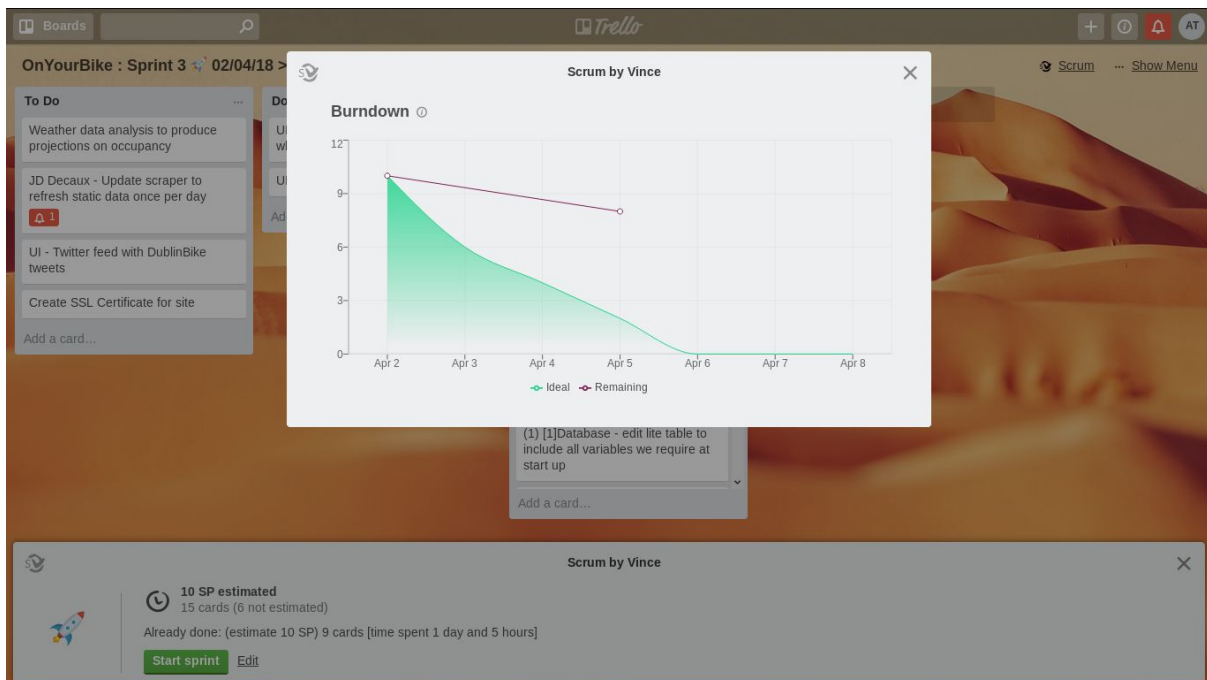


## Sprint 2:

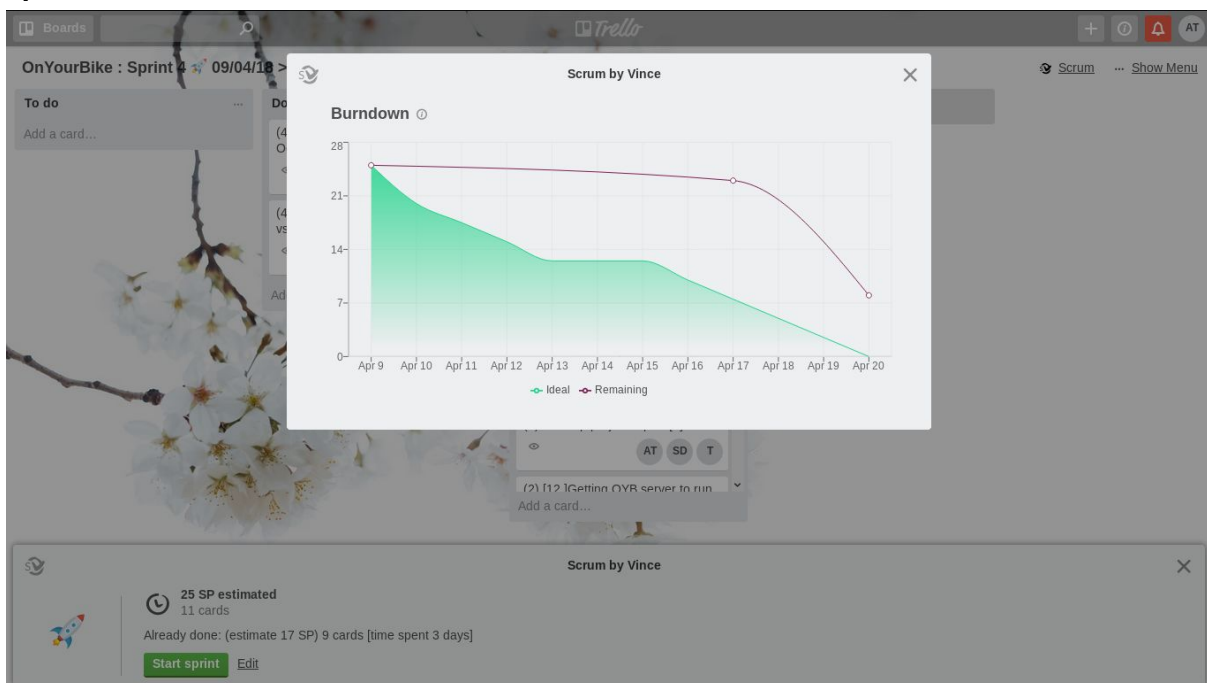




## Sprint 3:



## Sprint 4:





## 6. Managing deliverables, prototypes

	URL / IP / End Point	Note
Github Repository	<a href="https://github.com/atreanor/OnYourBike">https://github.com/atreanor/OnYourBike</a>	See readme.md
Application Server	172.31.38.249	External IP
Web Server	54.194.68.226	External IP
RDS Server	onyourbikemysql.cquggrydnjcx.eu-west-1.rds.amazonaws.com	EndPoint

The URL for our GitHub repository, called OnYourBike, is: <https://github.com/atreanor/OnYourBike>

The following three screenshots show the functioning RDS on Alan's EC2 - on which we have three tables (the screenshots show them as viewed through Workbench).

The screenshot displays the AWS Data Workbench interface. On the left, a sidebar shows the database structure for 'onyourbikemysql', including tables 'JCD\_dynamic\_data' and 'JCD\_flask'. The main pane shows a query result grid for the 'JCD\_flask' table. The query is 'SELECT \* FROM onyourbikemysql.JCD\_flask;'. The result grid shows 23 rows of data. The columns are: number, name, contract\_name, status, bike\_stands, available\_bike\_stands, available\_bikes, last\_update, address, and OYB\_timesta. The data represents bike rental information for various locations in Dublin.

number	name	contract_name	status	bike_stands	available_bike_stands	available_bikes	last_update	address	OYB_timesta
1	CLARENDON ROW	Dublin	OPEN	31	4	27	2018-04-20 10:47:29	Clarendon Row	2018-04-20 0
2	BLESSINGTON STREET	Dublin	OPEN	20	19	1	2018-04-20 10:47:04	Blessington Street	2018-04-20 0
3	BOLTON STREET	Dublin	OPEN	20	17	3	2018-04-20 10:48:36	Bolton Street	2018-04-20 0
4	GREEK STREET	Dublin	OPEN	20	15	5	2018-04-20 10:53:03	Greek Street	2018-04-20 0
5	CHARLEMONT PLACE	Dublin	OPEN	40	40	0	2018-04-20 10:45:08	Charlemont Street	2018-04-20 0
6	CHRISTCHURCH PLACE	Dublin	OPEN	20	20	0	2018-04-20 10:52:22	Christchurch Place	2018-04-20 0
7	HIGH STREET	Dublin	OPEN	29	29	0	2018-04-20 10:49:11	High Street	2018-04-20 0
8	CUSTOM HOUSE QUAY	Dublin	OPEN	30	1	29	2018-04-20 10:52:41	Custom House Quay	2018-04-20 0
9	EXCHEQUER STREET	Dublin	OPEN	24	2	22	2018-04-20 10:47:24	Exchequer Street	2018-04-20 0
10	DAME STREET	Dublin	OPEN	16	5	11	2018-04-20 10:48:27	Dame Street	2018-04-20 0
11	EARLSFORT TERRACE	Dublin	OPEN	30	16	14	2018-04-20 10:47:50	Earlsfort Terrace	2018-04-20 0
12	ECCLIS STREET	Dublin	OPEN	20	15	5	2018-04-20 10:46:19	Ecclis Street	2018-04-20 0
13	FITZWILLIAM SQUARE...	Dublin	OPEN	30	7	23	2018-04-20 10:44:54	Fitzwilliam Square ...	2018-04-20 0
14	FOYNES STREET UPPER	Dublin	OPEN	30	12	18	2018-04-20 10:52:59	Foynes Street Upper	2018-04-20 0
15	HARDWICKE STREET	Dublin	OPEN	16	16	0	2018-04-20 10:53:49	Hardwicke Street	2018-04-20 0
16	GEORGES QUAY	Dublin	OPEN	20	17	3	2018-04-20 10:49:16	Georges Quay	2018-04-20 0
17	GOLDEN LANE	Dublin	OPEN	20	18	2	2018-04-20 10:52:13	Golden Lane	2018-04-20 0
18	GRANTHAM STREET	Dublin	OPEN	30	30	0	2018-04-20 10:50:26	Grantham Street	2018-04-20 0
19	HERBERT PLACE	Dublin	OPEN	30	0	30	2018-04-20 10:47:32	Herbert Place	2018-04-20 0
21	LEINSTER STREET SO...	Dublin	OPEN	30	0	30	2018-04-20 10:49:18	Leinster Street South	2018-04-20 0
22	TOWNSEND STREET	Dublin	OPEN	20	1	18	2018-04-20 10:44:53	Townsend Street	2018-04-20 0
23	CUSTOM HOUSE	Dublin	OPEN	30	15	14	2018-04-20 10:54:13	Custom House	2018-04-20 0

**Above:** FLASK TABLE, which holds all the up-to-date JCDecaux bike data

The screenshot shows a database client interface with a query window containing the SQL statement: `SELECT * FROM onyourbikemysql.JCD_dynamic_data;`. The result grid displays 23 rows of data with the following columns: `id`, `number`, `status`, `bike_stands`, `available_bike_stands`, `available_bikes`, `last_update`, and `OYB_timestamp`.

id	number	status	bike_stands	available_bike_stands	available_bikes	last_update	OYB_timestamp
1	42	OPEN	30	23	7	2018-04-18 17:27:51	2018-04-18 16:32:26
2	30	OPEN	20	2	18	2018-04-18 17:24:31	2018-04-18 16:32:26
3	54	OPEN	33	33	0	2018-04-18 17:24:40	2018-04-18 16:32:26
4	56	OPEN	40	40	0	2018-04-18 17:31:16	2018-04-18 16:32:26
5	6	OPEN	20	19	1	2018-04-18 17:28:43	2018-04-18 16:32:27
6	18	OPEN	30	28	2	2018-04-18 17:31:31	2018-04-18 16:32:27
7	32	OPEN	30	0	30	2018-04-18 17:31:19	2018-04-18 16:32:27
8	52	OPEN	32	32	0	2018-04-18 17:29:36	2018-04-18 16:32:27
9	48	OPEN	40	9	31	2018-04-18 17:31:33	2018-04-18 16:32:27
10	13	OPEN	30	6	24	2018-04-18 17:31:26	2018-04-18 16:32:27
11	43	OPEN	30	18	12	2018-04-18 17:31:18	2018-04-18 16:32:28
12	81	OPEN	40	35	5	2018-04-18 17:29:08	2018-04-18 16:32:28
13	31	OPEN	20	1	19	2018-04-18 17:28:37	2018-04-18 16:32:28
14	98	OPEN	30	14	16	2018-04-18 17:30:01	2018-04-18 16:32:28
15	14	OPEN	30	23	7	2018-04-18 17:30:59	2018-04-18 16:32:28
16	1	OPEN	31	12	19	2018-04-18 17:31:02	2018-04-18 16:32:29
17	23	OPEN	30	5	24	2018-04-18 17:31:28	2018-04-18 16:32:29
18	68	OPEN	40	39	1	2018-04-18 17:31:23	2018-04-18 16:32:29
19	74	OPEN	30	14	16	2018-04-18 17:31:22	2018-04-18 16:32:29
20	87	OPEN	38	35	3	2018-04-18 17:26:10	2018-04-18 16:32:29
21	84	OPEN	30	17	13	2018-04-18 17:30:54	2018-04-18 16:32:29
22	90	CLOSED	40	0	0	2018-04-18 17:25:51	2018-04-18 16:32:30
23	11	OPEN	30	30	0	2018-04-18 17:29:42	2018-04-18 16:32:30

**Above:** Dynamic table, which stores dynamic (changing) data from JCDecaux.

The screenshot shows a database client interface with a query window containing the SQL statement: `SELECT * FROM OpenWeatherMap.OwM_current;`. The result grid displays 20 rows of data with the following columns: `clouds`, `name`, `visibility`, `w_d_main`, `w_d_id`, `w_d_icon`, `w_description`, `coord_lat`, `coord_long`, `owm_dt`, `id`, `humidity`, `pressure`, `temp`, and `temp_min`.

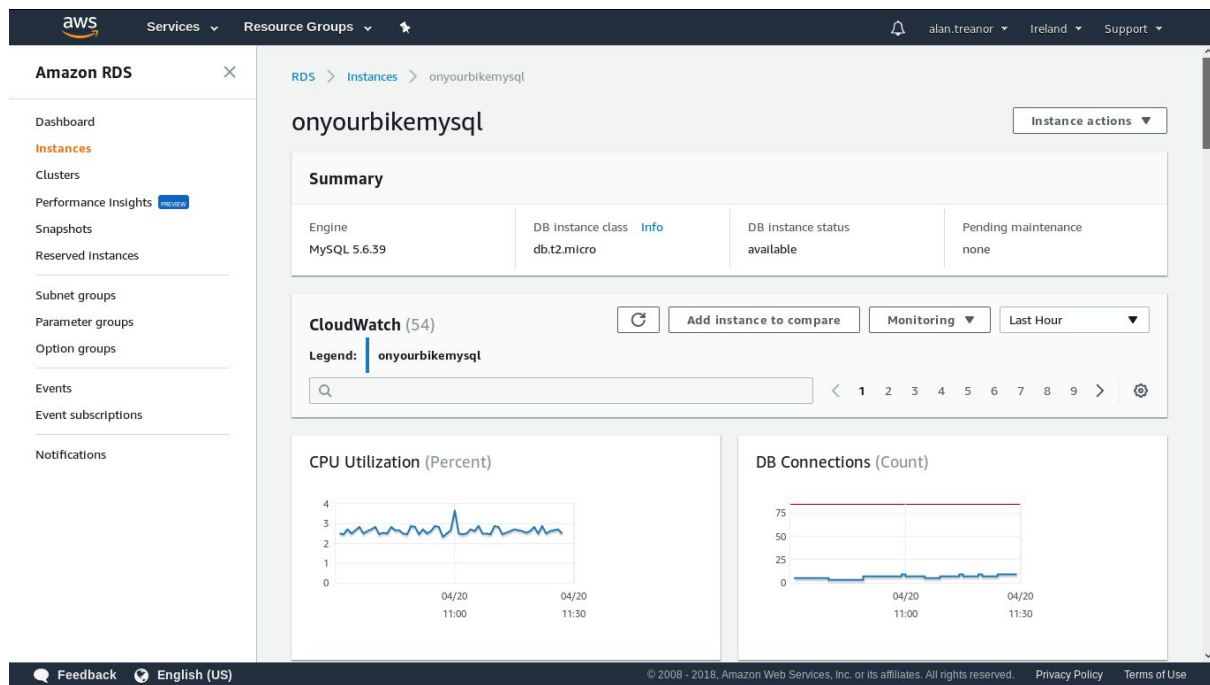
clouds	name	visibility	w_d_main	w_d_id	w_d_icon	w_description	coord_lat	coord_long	owm_dt	id	humidity	pressure	temp	temp_min
75	Dublin	10000	Clouds	803	04d	broken clouds	53.35	-6.26	2018-04-18 17:00:00	2964574	76	1018	17	14
75	Dublin	10000	Clouds	803	04d	broken clouds	53.35	-6.26	2018-04-18 17:00:00	2964574	76	1018	17	14
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 18:00:00	2964574	72	1017	17	15
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 18:30:00	2964574	59	1017	18	17
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 18:30:00	2964574	59	1017	18	17
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 19:30:00	2964574	59	1017	17	17
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 20:00:00	2964574	63	1017	16	16
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 20:00:00	2964574	63	1017	16	16
20	Dublin	10000	Clouds	801	02d	few clouds	53.35	-6.26	2018-04-18 20:30:00	2964574	59	1017	16	16
40	Dublin	10000	Clouds	802	03n	scattered clouds	53.35	-6.26	2018-04-18 21:30:00	2964574	76	1018	13	13
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-18 22:00:00	2964574	71	1017	13	13
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-18 22:00:00	2964574	71	1017	13	13
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-18 23:00:00	2964574	71	1018	13	13
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-18 23:30:00	2964574	71	1018	13	13
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 00:00:00	2964574	76	1018	12	12
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 00:00:00	2964574	81	1018	12	12
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 01:00:00	2964574	82	1019	12	12
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 01:30:00	2964574	76	1019	12	12
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 02:00:00	2964574	81	1018	11	11
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 02:30:00	2964574	87	1019	11	11
75	Dublin	10000	Clouds	803	04n	broken clouds	53.35	-6.26	2018-04-19 03:00:00	2964574	81	1019	12	12

**Above:** the open weather map table stored on our RDS (as seen through Workbench).

```
(softeng) [ec2-user@ip-172-31-38-249 OYB_logs]$ ps
  PID TTY          TIME CMD
 3843 pts/2        00:00:00 bash
16163 pts/2        00:00:02 bikes_dynamic
16168 pts/2        00:00:29 bikes_flask
16173 pts/2        00:00:00 weather
24670 pts/2        00:00:00 ps
```

**Above:** Application Server Processes running on Tom's EC2

**Below:** our RDS as viewed through the AWS web interface.





**Below:** a very early version of the map with markers and a clear button (which has since been dropped). The finished map is in the following image.

Dublin maps with multiple markers and clear button



**Below:** Screenshot of final working version of our web page with markers displaying individual station info taken from RDS. Also shows up-to-date weather information from RDS OWM weather table. NB this screenshot is from Sheena's laptop, not the EC2 instance.

# OnYourBike with DublinBikes

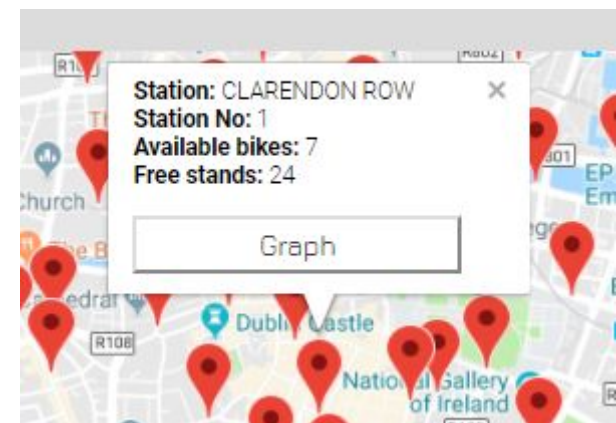
The weather in Dublin:

Clouds

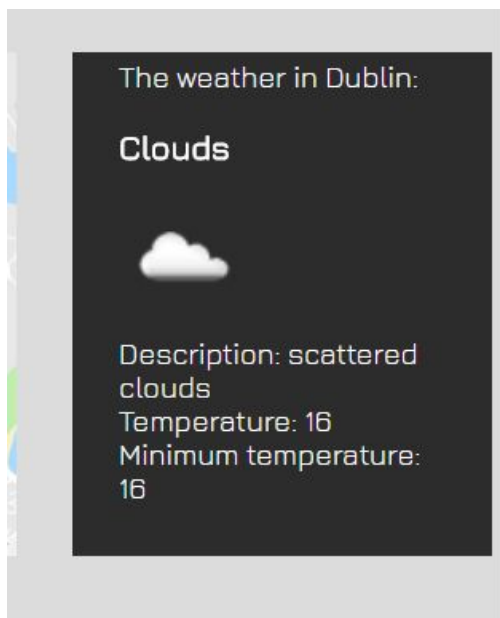
Description: scattered clouds  
Temperature: 15  
Minimum temperature: 15

Type in a streetname or area below to see it on the map

Search for a street

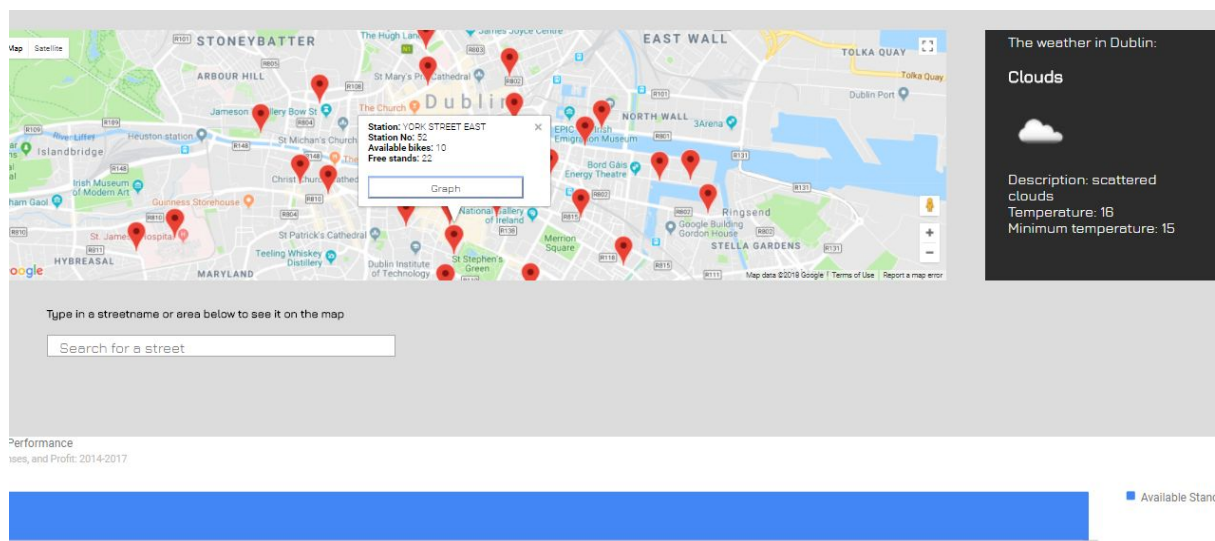


**Left:** close up of a marker displaying individual bike station data on station occupancy. The information has been fetched from our database

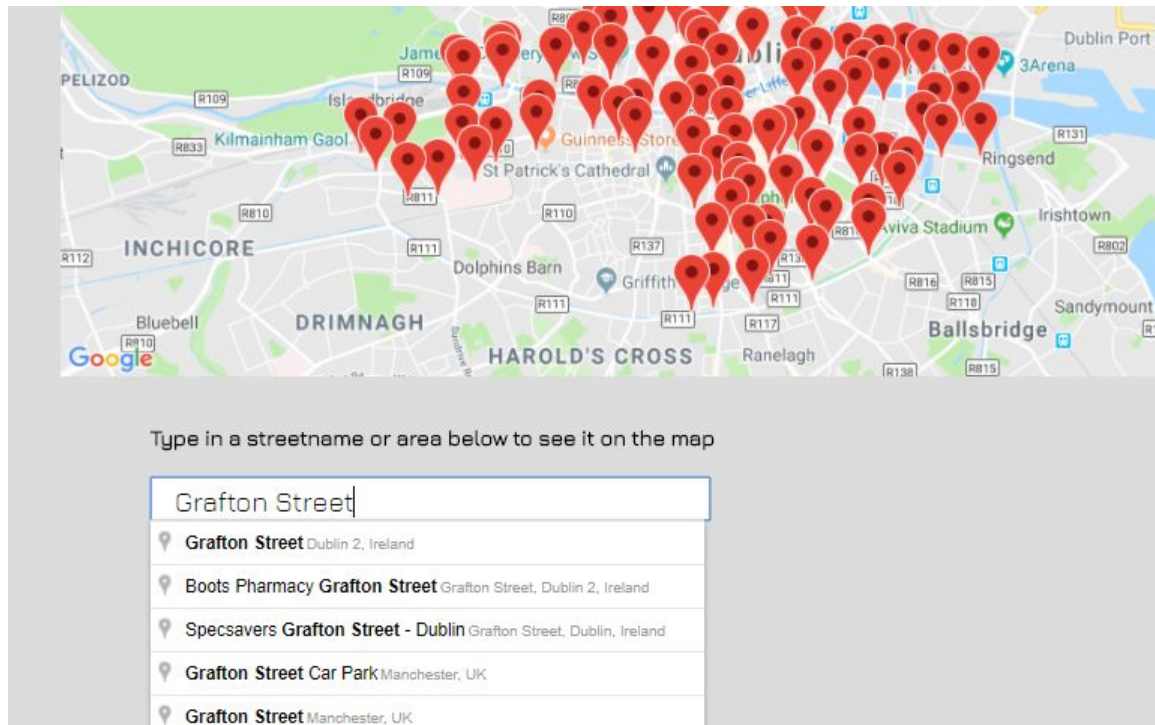


**Left:** close up of the live weather display on our main web page. This information has been fetched from the OWM weather table on our RDS hosted on an EC2.

**Below:** the image shows how pressing the 'graph' button on the infowindow relating to a particular bike station brings up a simple bar chart (at the bottom of the web page).



We also added a handy feature that lets users search for a street or area (see below) so that they can see where they are (or will be) and how close to a bike station they would be. When they enter a street name, they can see a marker (different to the bike station markers) so they can relate that to the nearest bike station.



**Left:** this shows the result of when a user searches for 'Grafton Street' - a small marker appears so that they can see where Grafton Street is and where the nearest markers for Dublin Bikes are.

### Final list of features delivered by our project:

- A python package that contains schedulers that scrape data, parse data and execute SQL insert statements
- Execution of python package using entrypoints
- Object oriented programming for the application server (two classes)
- Obtained (get request) and parsed data from JC Decaux
- Obtained (get request) and parsed data from OpenWeatherMap
- Deployment of Nginx on our web server (EC2 instance)
- Deployment of Gunicorn to start the Flask app
- Database code that connects to our RDS and inserts data into tables
- Web page with Google map that displays a marker for every DublinBikes station

- Infowindow that appears when each marker is clicked that displays up to date information on available bikes/free stands, the station's name and number
- Up-to-date weather information displayed on index.html, that shows the most recent entry in our weather database.
- Search bar allowing user to type in street name so that they can see where they are and if there are any nearby bike stations
- Simple google chart
- RDS with tables running on an EC2 instance
- Web scraper running on EC2 instance
- Basic map page running on EC2 instance - see web page at 54.194.68.226 to view this. The connection between the server on this instance and the RDS instance meant that this version - just on the EC2 - did not display markers/weather info. (However, these displayed perfectly on the local host).

#### List of features we did not have time to fully implement

- Advanced google charts with longer-range information (although all the work in the background was completed we simply did not have time to play with the Javascript any longer to make this feature work)
- More extensive unit tests
- Data analytics
- SSL certificate - we realise the importance of this and looked into getting one for our page, but unfortunately did not have time to implement it
- Twitter feed with live DublinBikes information on web page.