Dublin Bikes Group Project Report

Software Engineering COMP30670

Team Members:

Anurakta Behera [Student Id: 16203521]

Fangxue Mei [Student Id: 15210672]

Daniele Strafile [Student Id: 16203437]

Group Project Report

Software Engineering COMP30670

Introduction:

Dublin bikes is a public bike rental scheme which is operated in Dublin since 2009. Dublin is the 17th city to implement such a scheme. As of April 2013, the network consists of 102 stations and 1500 bikes. There is a whopping estimate of 70,000 subscribers in Dublin only.

Overview:

The aim of the project is to develop a web application to display occupancy and weather information for Dublin Bikes. This is done by scraping data from an API over a period of time. Once enough data is generated, it is to be stored in a database and a web page is to be created which shows current and historical station information for 'Dublin bikes' on each bike station in Dublin. This visualization is done using the data scraped from the API.

Tools Used:

1. SQLite
2. MySql
3. Flask
4. EC2
5. RDS
6. Javascript
7. Python
8. Github

Name: Dublin Bikes Project

Team: Anurakta Behera (Ann),

Fanxue Mei (April),

Daniele Strafile (Dan)

Percentage of Group Submission:

Daniele Strafile (Dan): 40%

Fangxue Mei (April): 30%

Anurakta Behera (Ann): 30%

Skills Practiced: Teamwork, web development with Flask, Sprint and scrum meetings, Javascript, project management, EC2 Amazon, RDS, Sqlite, Mysql.

Learning Done:

We learnt a lot in this project.

- Time management being the most crucial lesson of all.
- We started with SQLite and moved on to MySql as we were not clear about the requirement before.
- We also learnt that before writing any piece of coded, having a clear idea about what is required is very imperative.
- Explaining your code to someone is a arduous task as well. There are various streams of logic that can be applied on the same piece of code.
- Learnt Flask, its application and uses.

Our Project Phases:

Our entire project was executed in 3 phases. We kept our demonstrator Sourabh Dixit in loop though out the phases. Our main challenge through the phases was to allocate more time to this project.

Sprint No.: 1

Scrum Master: Anurakta Behera (Ann):

Date: 20th Mar to 31st Mar.

Phase I : We started our preliminary meetings with discussions on how to go about with the project? The very first question we asked each other, is how do we strategize the project? Do we divide tasks according to each team members forte, or do we start at scratch, from the very beginning and divide tasks to each other as we trickle down the project timeline. I am so glad we chose the latter approach. This helped each team member to exercise his/her skills and learn more at the same time. We started with designating tasks this week and creating victory pointers throughout the timeline. This was something done to keep the team motivated and to keep us involved through the project.

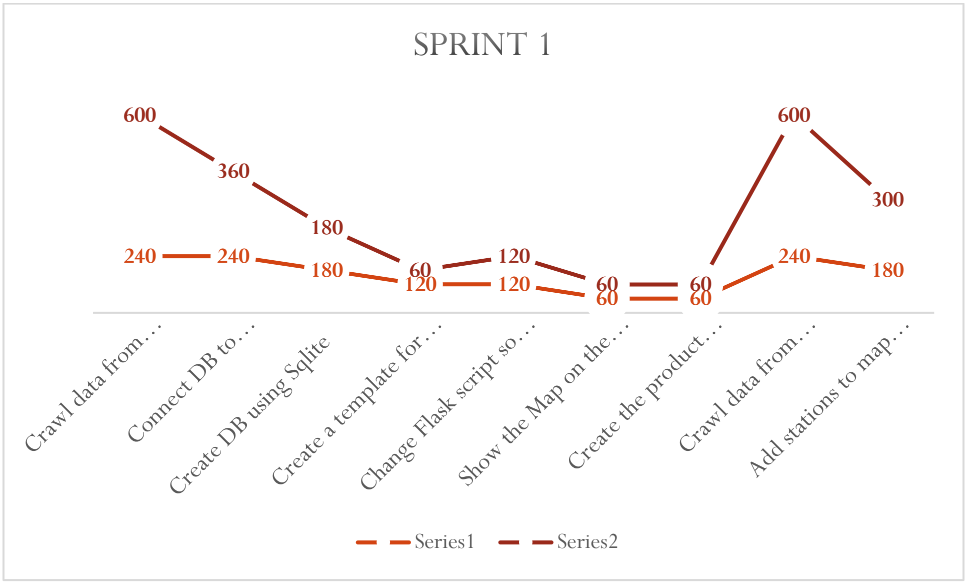
When we decided what approach will suit us best, we started discussing about timings and daily meetings. Then we moved on to working on 3 things simultaneously ; the website, learning a new tool Flask and on how to get our data to be stored on Flask. But before starting on the project, we created a repository on Github, an account on Trello and shared our progress and weekly challenges with each other. This was the start of Sprint 1, led by Anurakta Behera (Ann).

I, Ann created a rough sketch on paper as to how the website would look and what elements would go where. We collaborated ideas which were stemmed from the original Dublin bikes website and whether or not we use the same format as they have, or create our own. We reached on a conclusion to use the current header and logo of the website. The website was created by Ann. She also wrote a code for scraping the data from the API to be stored in a database. This led to duplicity of work. As the scrum master I thought this was required at that point in time, as I was not able to understand and execute a piece of code written by the team members.

Daniele took up the task of learning Flask and teaching it to the other team mates as well . This was really stimulating and it resulted in Daniele learning it and starting the implementation of Flask through python. He had a working prototype of flask application that connects to an empty database and opens up on the web that can request data to be entered to the database manually.

April took up the task of creating the databases on SQLite. Sqlite was very easy to understand and implement as well. In due course of time, this was changed to Mysql as suggested by the demonstrator. Now updating this database took up some time and demanded effort. She wrote a script that allowed her to scrape data from the API.

A burndown chart of Week 1, Sprint 1.



Challenges Faced in Sprint1:

Ann had created a scraper that was working, however it couldn't be automated.

The data was in XML,was not getting JSONified.

Duplicating the work led to better understanding but a lot of time was consumed here.

We were initially working on SQLite, migrating it to RDS and then to MySQL. This migrate took up a lot of time.

Sprint No.: 2

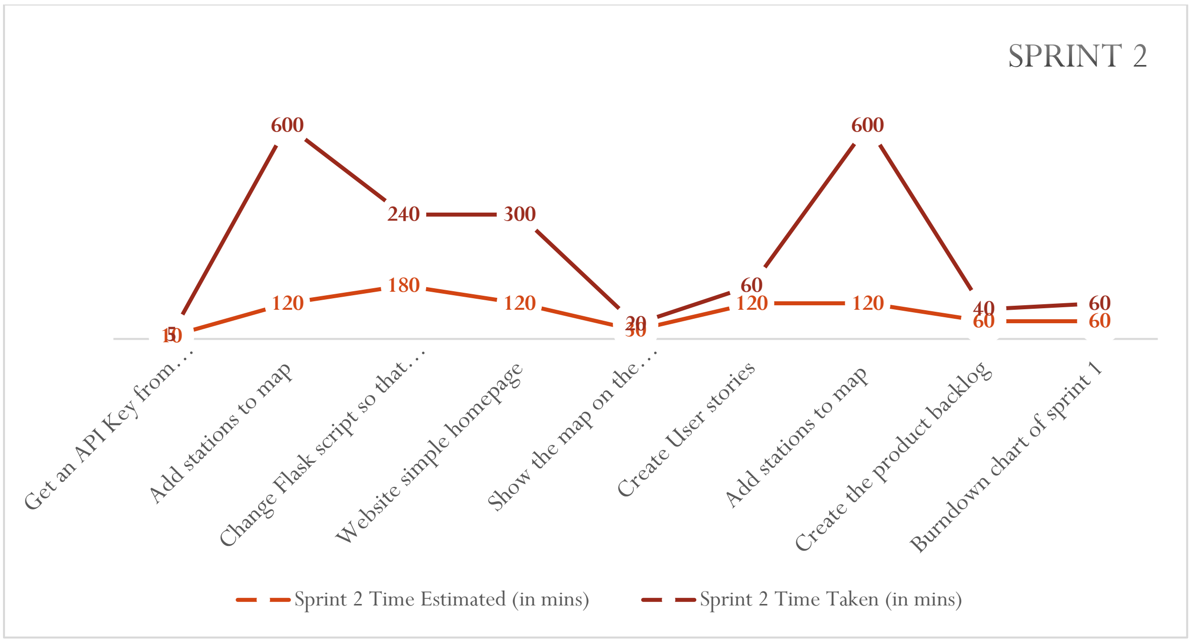
Scrum Master: Fangxue Mei (April):

Date: 3rd Apr to 7th Apr.

Phase II: We kickstarted this week with a detailed meeting with our demonstrator explaining him our ideas and our vision o]for the project. We were also bickering about the enormity of the project and how this would help us in out work environment as well. The last Sprint turned out to be a success as we reached our victory pointers. This week we focused primarily on 2 victory pointers; To train flask to extract data from the database and Display the stations markers on the website's Map.`

Daniele altered the flask script so that it gets data from RDS instead of SQLite. After displaying the map on the website, all three of us as a team worked on getting the station markers up and running. This task alone consumed 10 hours of our time. The biggest challenge this week was JavaScript. We had to use JS to retrieve the station's longitude and latitude to illustrate the station markers on the Google Map. This took our longest time and demanded a lot of effort from everyone in the team.

A burndown chart of Week 3, Sprint 2



Sprint No.: 3

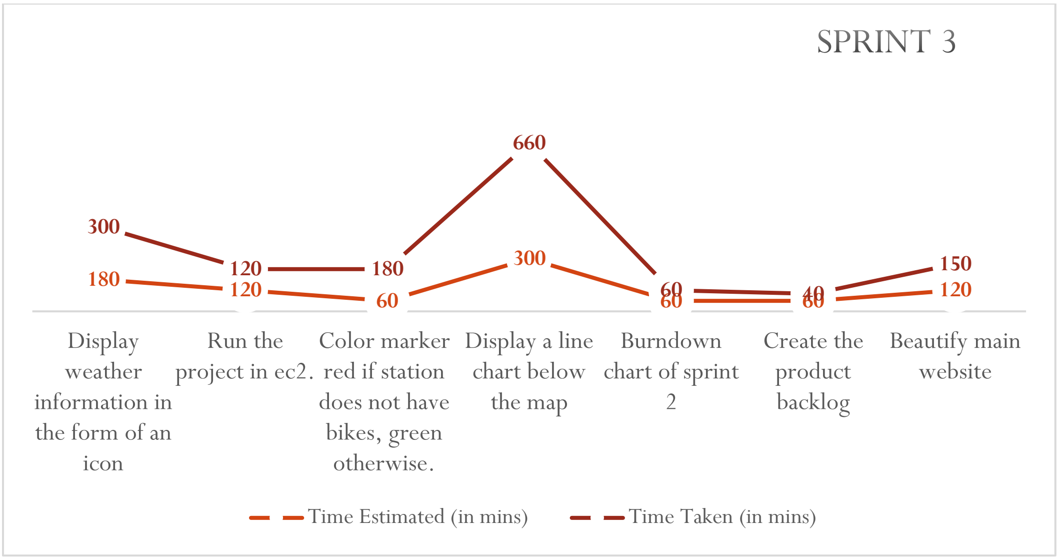
Scrum Master: Daniele Strafile (Dan):

Date: 9th Apr to 23rd Apr.

Phase III: On Monday we had our sprint meeting with demonstrator. Firstly, we had a discussion about the progress so far with the whole project and our individual contributions to it. The product backlog had unfinished task from the previous sprint, namely the burndown charts and the scraper that had to be run on Amazon EC2. Secondly, we discussed about the new features we wanted to add to our project. We came up with a simple layout of the website, which included a google map, station markers with basic on-click information along with some statistical information to be displayed below the map itself.

Ann also took care of the data analysis from the vast depth of data.

A burndown chart of Week 4, Sprint 3



Backlog

Backlog Item	Responsible	Started on	Finished on	Status	Estimate Time(in mins)	Actual Time (in mins)
Sprint 1	Anurakta (Ann)	20th Mar	31st Mar	Complete	1400	1970
Create a burn down chart.	Ann			Moved to next Sprint		
Website simple homepage	Ann			Moved to next Sprint		

Change Flask script so that it gets data from RDS instead of Sqlite.	Dan			Moved to next Sprint		
Create DB using Sqlite	April	21st Mar	31st Mar	Compl ete	180	180
Migrate Database from sqlite to rds.	April	31st Mar	31st Mar	Compl ete	180	240
Apply free RDS Services.	April	21st Mar	21st Mar	Compl ete	20	20
Checked that we couldn't use AWS RDS, as it needs to charge fee. We will use sqlite3 to create DB and manage our DB.	April	21st Mar	21st Mar	Compl ete	180	180
Get the dynamic data from Dublin bike API and inserted data in Data Base	April	22nd Mar	22nd Mar	Compl ete	60	
Migrate sqlite to RDS	April	31st Mar	31st Mar	Compl ete	60	60
Created the DB with sqlite and input stations data in it and still need to get the dynamic data and find the way to insert it to DB	April			Compl ete	60	120
Flask	Dan	21st Mar	21st Mar	Compl ete	120	150
Crawl data from dublin bikes. Scraper is working.	Dan, Ann, April	21st Mar	31st Mar	Compl ete	240	600
Add stations to map (longitude and latitude from RDS).	Dan, Ann, April	21st Mar	31st Mar	Compl ete	180	300
Input stations data in it and still need to get the	April	1st April	1st April	Compl ete	120	120

dynamic data and find the way to insert it to DB						
Total					1400	1970
Backlog Item	**Responsible**	**Started on**	**Finished on**	**Status**	**Estimate Time(in mins)**	**Actual Time (in mins)**
Sprint 2	Fangxue(April)	3rd April	7th April	Complete		
Create a burn down chart.	Ann	3rd April	3rd April	Complete	60	40
Create a backlog Chart	Ann	3rd April	3rd April	Complete	60	60
Run the api_read python program on EC2 instance	April			Moved to next Sprint		
Website simple homepage	Ann	7th April	7th April	Complete	120	300
Change Flask script so that it gets data from RDS instead of Sqlite.	Dan	5th April	5th April	Complete	120	240
Add stations to map						
Get an API Key from Google Map	Dan	9th April	9th April	Complete	5	5
Create User stories	Ann			Complete	120	120
Get an API Key from Google Map	Dan			Complete	10	5

Add stations to map	Dan			Compl ete	120	600
Change Flask script so that it gets data from RDS instead of Sqlite.	Dan			Compl ete	180	240
Website simple homepage	Ann			Compl ete	120	300
Show the map on the website.	Ann			Compl ete	30	20
Create User stories	Ann			Compl ete	120	60
Add stations to map	Dan			Compl ete	120	600
Total					1185	2590
Backlog Item	**Responsi ble**	**Start ed on**	**Finish ed on**	**Status**	**Estimate Time(in mins)**	**Actu al Time (in mins)**
Sprint 3	Daniele (Dan)	9th April	23rd April	Compl ete		
Display weather information in the form of an icon from openweather map when you click on a marker.	Dan	20th April	20th April	Compl ete	180+JJ45:J59	300
Run the project in ec2.	April				120	120
Color marker red if station does not have bikes, green otherwise.	Dan	21st April	21st April	Compl ete	60	180

Display a line chart below the map	Ann	10h April	15th April	Compl ete	300	660
Beautify main website	April	23rd April	23rd April	Compl ete	120	150
Create a burn down chart.	Ann	23rd April	23rd April	Compl ete	60	40
Create a backlog Chart	Ann	23rd April	23rd April	Compl ete	60	60
Craete Line charts	Ann	10th April	20th April	Compl ete	180	300
Bullet point at bottom left corner of empty box which says "last updated at"	April	21st April	21st April	Compl ete	120	60
Add the code to let the program to run every 5 mins	April	18th April	21st April	Compl ete	60	120
Copy the environment to EC2	April	18th April	21st April	Compl ete	60	120
No hup to run the program on background	April	18th April	21st April	Compl ete	60	120
Make the program continue running after logout	April	18th April	21st April	Compl ete	60	120
Onclick of the marker, display a line chart below the map with available bikes occupancy.	Dan, Ann, April	10th	19th April#	Compl ete	240	660
n flask, get the data of availability for every station and show it on the website	April	20th April		Compl ete	120	180
Total					1620	3190

Technical:

The entire project is split into 2 large sections:

1. The Database named (db_related)
    a. The data base that is being pulled from the scraper is being stored at DS Store.
    b. This data gets tabulated in dbconnect. On this file the data gets differentiated to timestamps, station information, etc.
    c. Rds_connect is the connector between the database and MySql.

2. The Flask Architecture (flaskr)
    a. The website template is stored here on the static folder. Moreover all the static items: images, png files, js, css, all the items that do not change is stored here.
    b. Template folder holds the html file.
    c. Flaskr is the glue between both the segments.

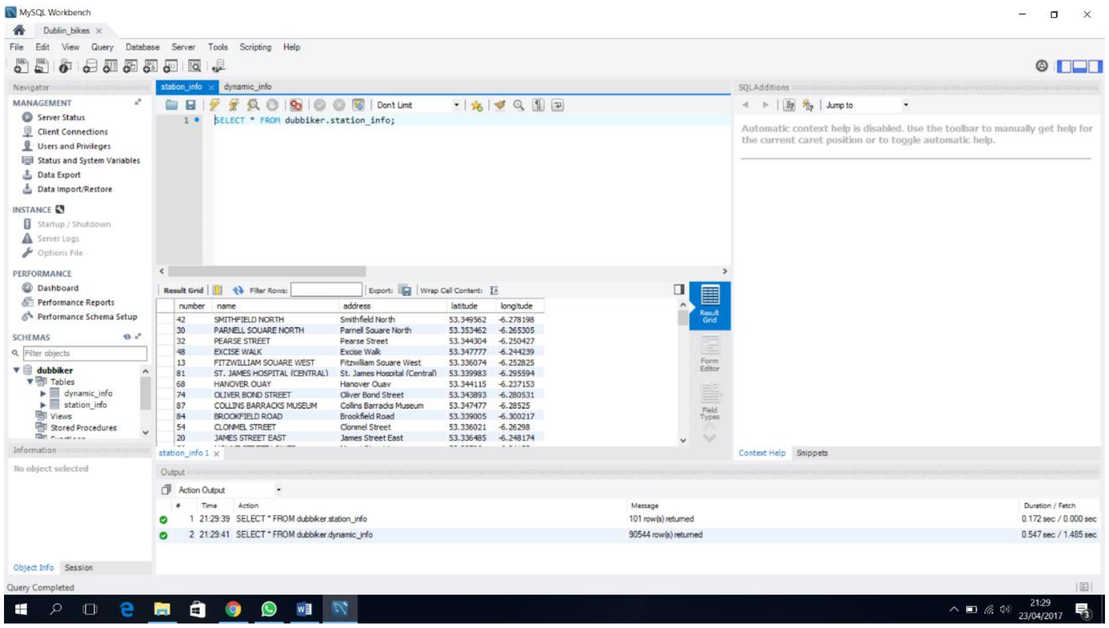
Link to the EC2 instance: http://ec2-52-26-65-232.us-west-2.compute.amazonaws.com/

Github Link: https://github.com/ucd2017comp30670/Dublin_bikes_project

Trello Link: https://trello.com/b/yfbpam0Z/sprint-1

Screenshots of data

Station_info table



```
SELECT * FROM dubbiker.station_info;
```

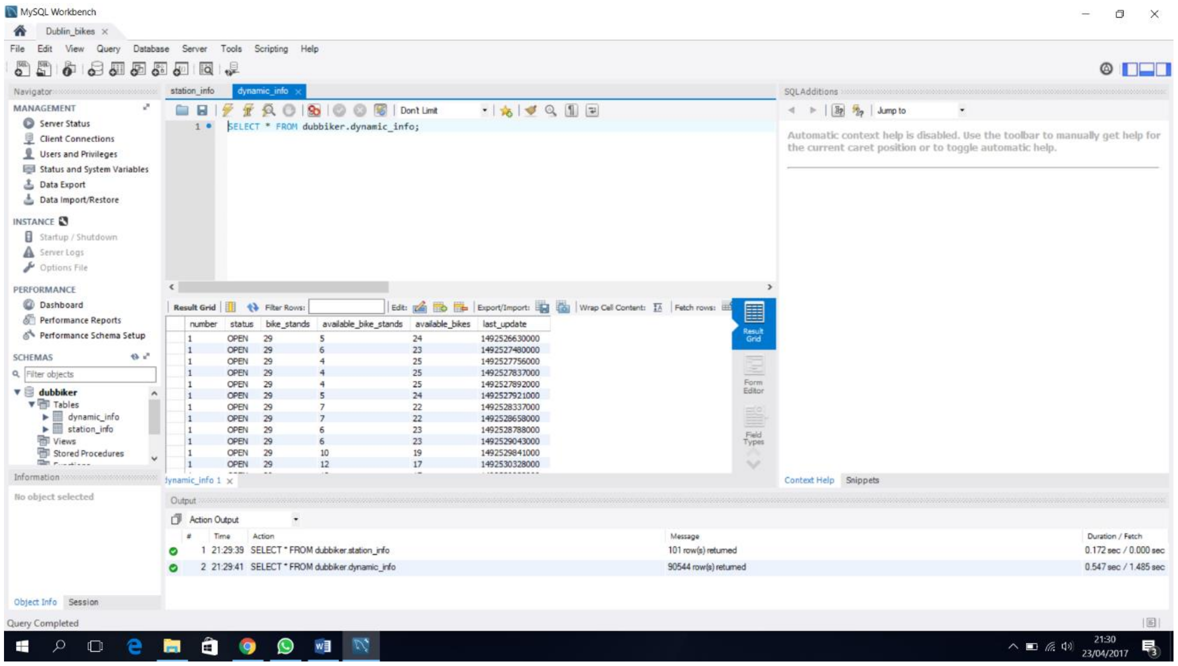
number	name	address	latitude	longitude
42	SMITHFIELD NORTH	Smithfield North	53.349562	-6.278198
30	PARNELL SQUARE NORTH	Parnell Square North	53.353462	-6.265305
32	PEARSE STREET	Pearse Street	53.344304	-6.250427
48	EXCISE WALK	Excise Walk	53.347777	-6.244239
13	FITZWILLIAM SQUARE WEST	Fitzwilliam Square West	53.336074	-6.252825
81	ST. JAMES HOSPITAL (CENTRAL)	St. James Hospital (Central)	53.339983	-6.295594
68	HANOVER QUAY	Hanover Quay	53.344115	-6.237153
74	OLIVER BOND STREET	Oliver Bond Street	53.343893	-6.280531
87	COLLINS BARRACKS MUSEUM	Collins Barracks Museum	53.347477	-6.28525
84	BROOKFIELD ROAD	Brookfield Road	53.339005	-6.300217
54	CLONMEL STREET	Clonmel Street	53.336021	-6.26298
20	JAMES STREET EAST	James Street East	53.336485	-6.248174

```
#  Time      Action                                Message                  Duration / Fetch
1  21:29:39  SELECT * FROM dubbiker.station_info   101 row(s) returned      0.172 sec / 0.000 sec
2  21:29:41  SELECT * FROM dubbiker.dynamic_info   90544 row(s) returned    0.547 sec / 1.485 sec
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Query Completed

Dynamic_info table



```
SELECT * FROM dubbiker.dynamic_info;
```

number	status	bike_stands	available_bike_stands	available_bikes	last_update
1	OPEN	29	5	24	1492526630000
1	OPEN	29	6	23	1492527480000
1	OPEN	29	4	25	1492527756000
1	OPEN	29	4	25	1492527837000
1	OPEN	29	4	25	1492527892000
1	OPEN	29	5	24	1492527921000
1	OPEN	29	7	22	1492528337000
1	OPEN	29	7	22	1492528658000
1	OPEN	29	6	23	1492528788000
1	OPEN	29	6	23	1492529043000
1	OPEN	29	10	19	1492529841000
1	OPEN	29	12	17	1492530328000

```
#  Time      Action                                Message                  Duration / Fetch
1  21:29:39  SELECT * FROM dubbiker.station_info   101 row(s) returned      0.172 sec / 0.000 sec
2  21:29:41  SELECT * FROM dubbiker.dynamic_info   90544 row(s) returned    0.547 sec / 1.485 sec
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Query Completed