Software Engineering Dublin Bikes Project

Orla Gartland 17204139
Fatima Mohamed 17205708
Robert De Bhal 12751005

COMP30670: Software Engineering

UCD School of Computer Science

20-04-2018

Contents

Foreword

All members of this group contributed equally to this project.

Product Backlog:

Tasks	Expected completion
Create the following: • RDS MySQL instance • Trello account • GitHub repository for this product • Project template using PyScaffold and push to GitHub • Amazon ec2 instance (to perform the scraping of DublinBikes Data for at least a week)	Sprint 1
Write a script to scrape dymamic data from the JCDecaux Dublin bikes database every 5 minutes. • Access the data using the API • Save this data to Amazon RDS	Sprint 1
Write a script to obtain weather information from the OpenWeatherMaps API	Sprint 1
Make 3 tables on the RDS instance: • dynamic (to store fields of the Dublin Bikes data which changes such as available bikes) • static (to store the fields of the Dublin Bikes data which does not change such as address) • weather (to store data from the OpenWeatherMaps API)	Sprint1
Create a basic Flask App and add this Flask app to the project template	Sprint 1
Display a basic Google Map on this Flask App	Sprint 1
Learn how to display circles of varying size on the Google Map	Sprint 1
Do a basic mock up of what the web app should look like	Sprint 1
Perform basic analytics on the trends observed far using Google Data Studio and/or pandas	Sprint 1
Work on the front end – i.e. use CSS to style the web app	Sprint 3
Query the database using php to display pointers on the map which indicate the position of a station	Sprint 2
Query the database using php to access the number of available bikes and then size the circles on google maps accordingly	Sprint 2
Display number of bikes available and number of stands available on-click of a station in a Google Maps Info Window	Sprint 2

Work on efficeincy – i.e. reduce the waiting time for an Info Window to display bike availability and stand availability	Sprint 2
Analyse the availability of bikes with respect to: • Time of the day • Weather ↘ Rain ↘ Temperature ↘ Clouds	Sprint 2
Make charts of availability by rain for each station	Sprint 2
Integrate availability charts with the web page	Sprint 2
Display pie charts of current availability on click of a station	Sprint 2
Perform unit testing on the code	Sprint 3
Make a model to predict availability of bikes for every station for every day for every hour using pickle	Sprint 3
Make dropdowns to select a day and time to predict availability	Sprint 3

* Note: Tasks in yellow were not in the initial version of the product backlog

SPRINT 1 – SCRUM MASTER: FATIMA

Note:
In the notes below of the daily stand up meetings:

- ‘yesterday’ refers to what tasks that person completed yesterday for this project
- ‘today’ refers to what tasks that person or the group as a whole intends on doing today/ before the next daily stand up
- ‘anything to overcome’ refers to any challenges/difficulties a person may encounter with a particular task and if they need help with it

Also, for the first sprint we performed most tasks as a group during the day because we were on midterm break and dedicated our evenings towards other subjects. For this reason most of the tasks are labelled as ‘Teamwork’.

Sprint 1: Planning meeting notes

- We decided upon who would be scrum master for each sprint
- We made a flow chart of the overall system to get a high level understanding of what we needed to do and how different components were connected (see Appendix 2)
- We made a product backlog and assigned tasks to each sprint
- We assigned each of user story and task a complexity number based on the Fibonacci numbers (1,2,3,5,8,13) - i.e. a task assigned as 2 is twice as complex as a task assigned as 1.
- We assigned tasks to each individual for this sprint as well as the tasks expected completion dates
- We set out our medium term high level goals which were as follows:

- Sprint 1:
    - ➘ Collect Dublin bikes and weather data.
    - ➘ Display circles of varying sizes (proportional to number of bikes available) on a basic Flask app
- Sprint 2:
    - ➘ Analyse the data (possibly using pandas)
    - ➘ Make average occupancy graphs
- Sprint 3:
    - ➘ Focus on the front end and styling the web page
    - ➘ Clean up code, ensure it works on the instance etc.

Sprint 1: Day 1 Stand Up (Monday 12/3/18)

Robert, Orla, Fatima

- Yesterday:
    - n/a on Day 1 of sprint
- Today:
    - Create GitHub account and repository for this project (Robert)
    - Create Amazon RDS instance (Fatima)
    - Learn how to connect to Amazon RDS via MySQL workbench (Fatima)
    - Create project template using PyScaffold and push to GitHub (Orla)
    - Create Amazon ec2 instance, install Anaconda on this, send everyone the key for instance (Orla)
    - Create Trello account, invite members (Orla)
- Anything to overcome:
    - First time using this Amazon RDS

Sprint1: Day 2 Stand Up (Tuesday 13/3/18)

Robert, Orla, Fatima

- Yesterday: Completed the assigned individual tasks

-Today:
- Orla and Robert do not have MySQL workbench on their laptops so this must be installed today
- Use python to create a scraper script to query JC Decaux API
- Store this data locally in an appropriate format (i.e. a pandas data frame)
- Connect the web scraper to RDS (using pymysql) and store this data
- Perform basic queries of RDS data

- Anything to overcome:
    - Our limited knowledge of RESTful APIs making it difficult to use JC Decaux API
    - No prior experience with using Amazon RDS
    - How to convert the JSON data obtained from the API into a relational database

Sprint1: Day 3 Stand Up (Wednesday 14/3/18)

Teamwork (entire group)

- Yesterday:
    - Connected the scraper to Amazon RDS and started storing data
    - Realised we need to store data in a format that will be most convenient for analysis
    - Decided our database would require the following:

- ➘ static table – for station information that does not change (such as address)
- ➘ dynamic information - for station information that will change (such as bike availability)
- ➘ weather information – to hold the weather which we intend on scraping from Open Weather Maps API
- ➘ foreign keys to link the above 3 tables
- ➘ stored procedures – such as: find the time of the most recent insertion into the dynamic table (to ensure our scraper is still running)

- Today:
  - Make a static, dynamic and weather table in our database
  - Write a console script to run the scraper on ec2 (currently we have only practiced using the scraper locally on our machines)
  - Change any security settings on the ec2 instance if required
  - By the end of the day have the scraper for Dublin Bikes data fully functioning on ec2 instance

- Anything to overcome:
  - Figure out appropriate data types to store variables in (e.g. may need to change date into a more human readable format)
  - Figure out which foreign keys should be used to link the tables
  - Need to revise how to create stored procedures in MySQL workbench
  - Need to look over previous assignments to learn how to create console scripts

Sprint1: Day 4 Stand Up (Thursday 15/3/18)

<u>Robert, Orla, Fatima</u>

- Yesterday:
  - Achieved the tasks we had set out to do as a group – made 3 tables with appropriate foreign keys
  - When we left last night the scraper was running and updating our database
  - This morning before the stand up, I noticed that the scraper was no longer running on the instance (I used the command ps) ; we looked into this at the stand up (i.e. the time of the last entry to the database) and quickly realised that this happened because I logged out of my account last night and did not run the process with ‘nohup’

- Today:
  - Add try/catch block to catch certain exceptions such as Integrity errors when inserting into the database
  - Look into seeing if we can make e-mails be sent to us if an exception occurs that is not in the try/catch block so that we will be notified immediately if the scraping script stopped working because of an error.
  - Obtain a key from Open Weather Maps API and do basic queries
  - Determine which fields to store in RDS (i.e. which fields will be most useful to our application)
  - Convert the data from Open Weather Maps API from JSON to Pandas Data frame to SQL which can be stored on RDS

- Anything to overcome:
  - Have never sent e-mails via python before but are aware that such functionality exists (is not particularly important if we cannot achieve this (low priority task))
  - Getting the data into a format that can be inserted into our database can be tricky and time consuming

Sprint1 Day 5 Stand Up (Friday 16/3/18)
n/a: Did not meet up today (Midterm break)

Sprint1 Day 6 Stand Up (Monday 19/3/18)
<u>Robert, Orla, Fatima</u>
- Yesterday (i.e. Day 4)
    - We got the scraper back up and running on ec2 using the nohup command before the name of the process
    - We got the e-mail functionality working – i.e. an email is sent if an error occurs in the scraper that was not covered in the catch block (However this was only tested on our local machines, not on the instance; perhaps there are security issues in sending emails?)
    - Started saving weather data down and added a foreign key of last updated weather to dynamic table for ease of analysis later in this sprint
- Today:
    - Create a basic Flask App (Robert)
    - Look into how to visualise data using Google Data Studio and make basic plots with our data using this tool (Orla)
    - Create a basic web page that can connect to Google Maps API and display a map of Dublin. (Use Brackets for now to display the webpage – not Flask as we do not have this up and running yet) (Fatima)
    - Add additional extras to this page such as footers etc. (Fatima)

-Anything to overcome:
- Have never used Google Data Studio before
- Need to revise the directory structure and code needed for a Flask App from Assignment 2

Sprint1 Day 7 Stand Up (Tuesday 20/3/18)
<u>Robert, Orla, Fatima</u>
- Yesterday:
    - Created a basic Flask app but we will still use Brackets currently for testing (Robert)
    - Worked on scrum master notes such as how to make a Burn Down Chart (Orla)
    - Tried to use Google Data Studio but found it quite difficult and unintuitive to use – also found it difficult to find tutorials/help online for this tool (Orla)
    - Got a map of Dublin to display on the browser (Fatima)
- Today:
    - Ensure scraper is still running
    - Try get circles to appear on Google maps which are at the locations of the bike stations
    - Try get circles of varying sizes and colours to appear
- Anything to overcome:
    - Have never displayed a circle on a google map before; only displayed a hardcoded marker in Web Development module before
    - Have to read the stations location from RDS database and store this in a form that can be read by JavaScript (May have to convert SQL to pandas dataframe to JSON to JSON of the correct format)

- May have to use pandas to analyse the data in future instead - also will ask Jake if this was what he recommended or was it called something else
- Fatima noticed that the stylesheet is not working, also having issues getting footer to stay at bottom of page

Sprint1 Day 8 Stand Up (Wednesday 21/3/18)

n/a: Did not meet up today (Midterm break)

Sprint1 Day 9 Stand Up (Thursday 22/3/18)

Robert, Fatima, Orla

- Yesterday:

- Got blue circles to display by reading from a JSON object with three hard coded locations
- Had difficulty getting the locations of the stations into the correct JSON format (Currently it is in a JSON format but not one which is required by the Google Maps API

- Today:

- Ensure scraper is still running
- Get JSON into the format needed by Google Maps API
- Goal again for today is to have a circle on Google Maps everywhere there is a Dublin bikes station
- Attend Sprint Retrospective meeting which will be held later in the day

- Anything to overcome:

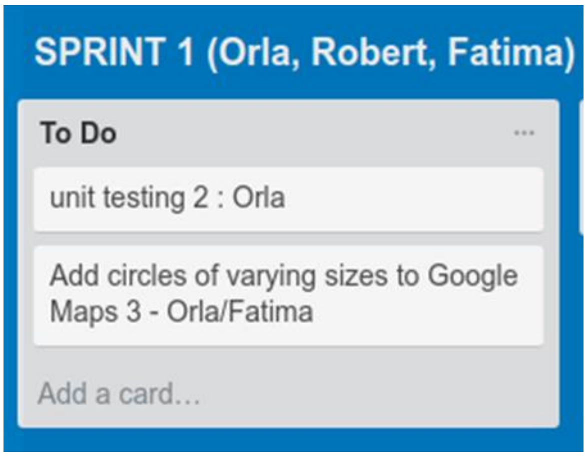
No - Should get circles to appear today after progress that was made yesterday

Sprint1 Day 10 Stand Up (Friday 23/3/18)

n/a: Did not meet up today (Midterm break)

Sprint 1: Retrospective Meeting Notes (Thursday 22/3/18)

- We looked at our Trello board to see which tasks were still in progress and not completed (See screenshot below)
- We added tasks which were not completed to the next sprint
- We reflected upon the progress made and acknowledged that we had reached our medium term goals that we set out to achieve for this sprint.
- We analysed our burndown chart (See comments below)
- We reflected that we should take more detailed and informative minutes of our daily stand ups. The minutes were originally written using pen and paper in the morning and typed up that evening. In future we will type these directly in to save time and avoid duplication of work.


SPRINT 1 (Orla, Robert, Fatima)

To Do

unit testing 2 : Orla

Add circles of varying sizes to Google Maps 3 - Orla/Fatima

Add a card...

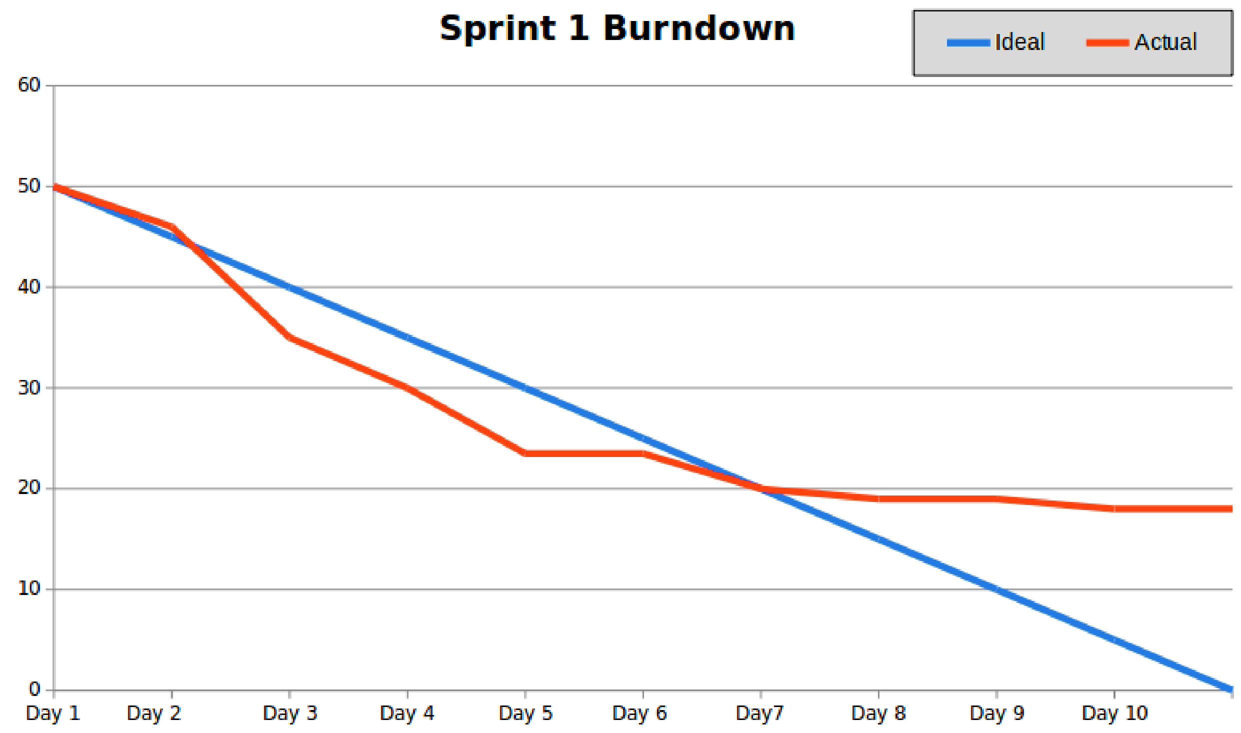
Sprint 1: Review Meeting Notes (Tuesday 27/3/18: during practical)

- We met with the product owner (Jake) to show him the progress made over the past 2 weeks
- Jake also helped us determine what should be next on our priority list.
- We learned from this meeting that we do not have to use php to query the RDS data base to display information for each station
- We learned we should use Google Charts instead of Google Data studio to display our graphs on our website
- Jake recommended us to look at the nohup.out file to see why our scraper went down in the 1st place
- We looked at this file in the practical and realised there was a TypeError caused by trying to convert an object of type none into a string in date format. This means that either the Dublin bikes scraper or weather scraper must have returned none for the date

Sprint 1 Burndown Chart

Comments on burndown chart:

- We got more work than we had anticipated to get done for the first 6 days.
- From Day 7 onwards we did not get as much work as we had hoped to because we did not meet up in order to focus on our other subjects
- As a result we were left with uncompleted tasks at the end of this sprint. (Hence the reason the chart ends with 18 hours work remaining)



SPRINT2 SCRUM MASTER: ORLA

Sprint 2: Planning meeting notes

- We set up a Trello board for this sprint.
- We analysed the product backlog and moved tasks that were not completed such as testing to the 'to do' list of this Trello board.
- We assigned tasks to people as well as expected completion dates.

Sprint2 Day 1 Stand Up (Monday 26/3/18)

<u>Robert, Fatima, Orla</u>

- Yesterday:

- Robert checked the RDS database last night and realised that the dynamic table in the database was last updated on Tuesday 20 March meaning that we have collected data for 5 consecutive days (from 15th March – 20th March) but have not collected any data for the past 4 days

- Today:

- Get the scraper back up and running on the instance
- Determine what caused the scraper to stop working in the first place to ensure this does not happen again
- From now on check the database more regularly (ideally every day before we leave college) to ensure that if it does stop working again, the issue will be fixed as soon as it arises and start collecting data again straight away

- Anything to overcome:

- Determine what caused the scraper to stop working and why we did not get an e-mail to notify us of the error (perhaps there are security settings on the instance with sending outbound traffic?) We need to look into this further.

Sprint2 Day 2 Stand Up (Tuesday 27/3/18)

<u>Robert, Fatima, Orla</u>

- Yesterday:
  - The group worked together to get the scraper process back running on the instance and to start storing data into the dynamic table in RDS.
  - We did not have any issues with this but did not change the code as we still did not yet know what caused the scarper to stop running
- Today:
  - Determine what caused API scraper to stop working and catch any errors.
- Anything to overcome:
  - Need to add a try/catch block to ensure that the scraper does not stop working in cases like this in future
  - Need to test that the scraper will still work with this piece of code included

Sprint2 Day 3 Stand Up(Wednesday 28/3/18)

<u>Robert, Fatima, Orla</u>

- Yesterday:
  - We worked on adding an appropriate try catch block (e.g. try insert this formatted date – except type error – if this happens insert 1/1/1970)
  - We implemented this, re-installed the package and tried to get scraper working on ec2 but this was not working
  - We decided to drop the column from RDS that converts the Unix time stamp to a date
  - We worked on the front end and now have the name of the station appearing on click of one of the circles but our scraper was still not functioning
  - An issue is that 'on-click' , these boxes will not disappear (perhaps we need an if/else statement)
  - Robert realised last night that 4 new Dublin bikes stations were added which was the cause of our scraper not working because of the foreign key constraint of 'station number' imposed on the dynamic table – The scraper was trying to insert numbers into the dynamic table which were not found in the primary key of the static table
- Today:
  - Back end: Figure out a way to update the static table if a new station gets added and implement this. (Robert and Fatima)
  - Front end: Figure out how to have the circles displaying on a map in a way that is not hard coded; currently the circles where the stations are located are being displayed by reading from a JSON object that has 104 entries in the JavaScript file, which is not good practice. (Orla)
- Anything to overcome:
  - Back end: Need to revise foreign key/referential integrity constraints from Databases module.
  - Front end: Need to look up pandas documentation to convert to JSON. Also need to revise XMLHttp requests because to display the markers/circles in a way that is not hard coded we will need to read JSON from a file

Sprint2 Day 4 Stand Up (Thursday 29/3/18)

<u>Robert, Fatima, Orla</u>

- Yesterday:

- weather.html file was updated so that it reads the latitude/longitude and name of station from a json file as opposed to it being hard coded in. This is still not ideal as it does not allow stations to be dynamically added (Orla)
- updateStaticTable.py file was made and imported into API_scraper.py file so that the database tables can handle new stations being added. (Robert and Fatima)
- Robert got the pointers on the map to display by reading a json object in flask. In other words there is now a Flaks route which holds the static JSON data such as station number, latitude and longitude for each station.

- Today:

- Front end: Get Info Windows to close when you click another station (currently they stay open and display no information)
- Since Good Friday and Easter Monday are coming up and we may not be meeting up until Tuesday, we set up 3 different routes on the flask app (localhost:5000/<persons_name>) and 3 different weather files. This will enable us to have 3 different development environments thereby preventing merge errors over the weekend.

- Anything to overcome:

- No- there should be no issues setting up different routes on flask as we have done this in the past.

Sprint2 Day 5 Stand Up(Friday 30/3/18)

n/a: Did not meet up today (Good Friday)

Sprint2 Day 6 Stand Up(Monday 2/3/18)

n/a: Did not meet up today (Easter Monday)

Sprint2 Day 7 Stand Up (Tuesday 3/4/18)

Robert, Fatima, Orla

- Yesterday:

- Currently the project is set up as a package i.e. it can be installed using ‘pip install git+<url>’ instead of git clone <url>. The only issue with this is that the static and templates files are not getting copied into the site packages folder in Anaconda. Over the weekend all members of the group looked into whether it was a good idea to have a Flask app set up as a package. The general consensus was that it was not recommended
- Robert looked into MANUSCRIPT.in which can be added into package structure which should overcome the issue of templates and static not being installed.

- Today:

- From now on don’t install it as a package. Use git clone when we need to run it on the instance.
- Get website to display static data (i.e. station number and name) and dynamic data (i.e. current availability) on-click of a circle/marker

- Anything to overcome:

- Displaying dynamic information will be more difficult than static information

Sprint2 Day 8 Stand Up (Wednesday 4/4/18)

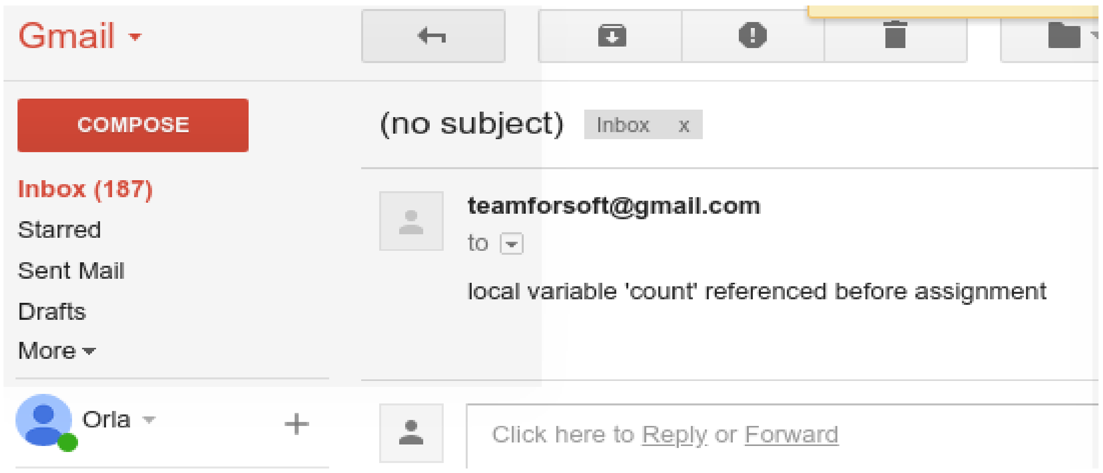
<u>Robert, Fatima, Orla</u>

- Yesterday:
  - We tried to get the circles to display dynamic data for the stations but did not achieve this. We had issues with the correct ordering of functions in JavaScript.
  - We did manage to get a webpage which displays the dynamic information for each station in json format (as provided in lectures) (i.e. 'localhost:500/availability/6' will display most recent bike availability for station number 6)
- Today:
  - We realised that our database was last updated at 6am this morning so we plan on getting the scraper back running today and looking at the nohup.out file to determine the error that occurred.
- Anything to overcome:
  - Need to get scraper back running.

Sprint2 Day 9 Stand Up (Thursday 5/4/18)

<u>Robert, Fatima, Orla</u>

- Yesterday:
  - We read the nohup.out file to find out the scraper stopped working because of a MaxRetry error from the Open weather maps API. We thought we were only querying the database every 15 mins but because of a bug in our code we were actually querying data every 5 mins. We are getting data from Dublin bikes and open weather maps every 10 minutes.
  - Also deleted the code that sends us an e-mail because we got an e-mail to say that there was a syntax error when we were testing the adjusted code. We maintained that it is best for all errors to go to the 1 file.
  - We also added a logger file where our exceptions that we are catching will get written to
- Today:
  - Get dynamic information such as current availability to display on-click of a station. (We did achieve this yet because we were derailed when the scarper stopped working, but we did make an attempt at it and pushed it to GitHub)
- Anything to overcome:
  - No



(no subject)

teamforsoft@gmail.com

local variable 'count' referenced before assignment

Sprint2 Day 10 Stand Up (Friday 6/4/18)

Robert, Orla

- Yesterday:
  - We got dynamic data in particular bike availability and number of stands to display for each station. The only issue is that a call is made to the database everytime the user clicks on a station meaning it takes 4 seconds for the data for a particular station to display on click.
- Today:
  - Check that the web application (which now includes dynamic information) will still work on the ec2 instance.
  - Start graphing the availability data by rain/no_rain for a given station using pandas
- Anything to overcome:
  - Need to look over the 'df.resample' method that was recommended to us in lectures
  - Need to look over Data Analytics practical on generating bar charts that are grouped by a categorical feature.

Sprint 2: Retrospective Meeting Notes (Monday 9/4/18)

- The group acknowledged that we need to get into the habit of using Trello more. We decided that we should have this open during every stand up meeting in future as opposed to relying on what tasks still need to be done from memory.
- We analysed our burn down chart.
- We acknowledged that we may have performed too many tasks as a group during this sprint and may need to split up more for the next sprint if we wish to implement all features stated in the specification.

Sprint 2: Review Meeting Notes (Tuesday 10/4/18: during practical)

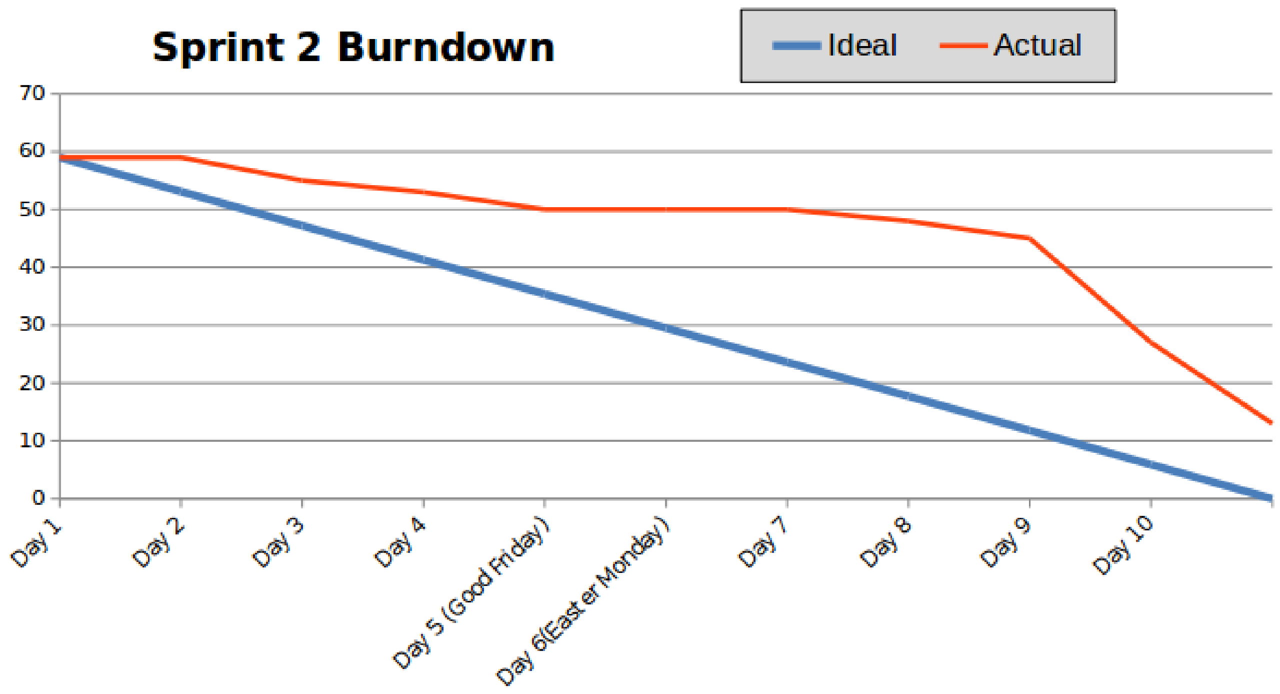
- We had a meeting with our product owner, Jake, who recommended that we implement a location feature – i.e. get location/enter location and find the nearest station to you
- We added this to the product backlog but still intend on focusing on getting the availability charts to display
- The group acknowledged that we did not achieve our medium term goals of having availability charts for every station displaying on the web page
- We looked at our Trello board for the previous sprint and noted tasks that did not get completed which were:
  - ➘ Integrate the availability charts for each station with the web page.
  - ➘ Display real time weather information on the web app
  - ➘ Testing
  - ➘ Comment our code
  - ➘ Make our keys hidden

Sprint 2 Burndown Chart

Comments on burndown chart:

- We realised in the middle of this sprint that we were not meeting the deadlines we had set ourselves for our tasks. We did not realise how complex some of these tasks were.
- Also the 2 bank holidays meant that we had 2 less stand up meetings so we had less time to co-ordinate as a group.

- We did not complete all of our tasks, so some of these must be brought forward to the next sprint. (This is why the orange line does not reach 0 in the graph below)



SPRINT3 SCRUM MASTER: ROBERT

Sprint 3: Planning meeting notes (Monday 9/4/18)

- We made a Trello board for the new sprint
- Since we did not reach our medium term goal for the previous sprint, we re-prioritised our tasks.
- We set out a goal of having the entire website done by next Monday (Monday 16th April) so that we could focus on other aspects of the project (such as personal reflection/documentation etc.) for the remainder of the week.
- We decided to all focus on getting the essential parts of the website completed (such as integrating the graphs withthe web page) before we go about minor tasks (such as hiding our keys, commenting code etc.)

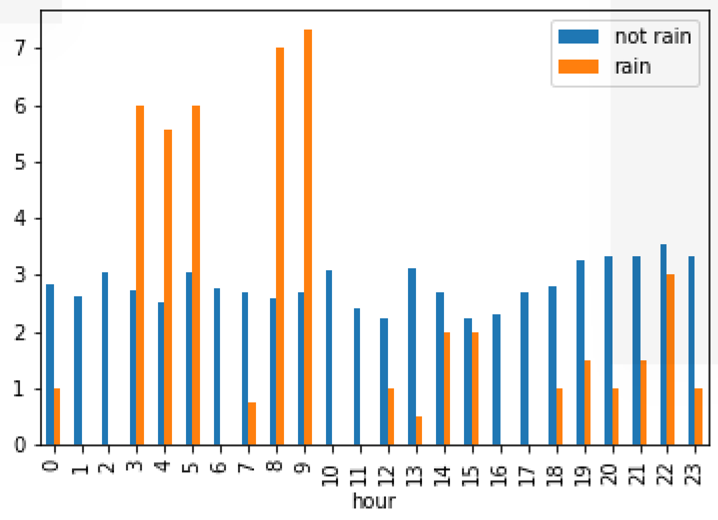
Sprint3 Day 1 Stand Up (Monday 9/4/18)

<u>Robert, Orla, Fatima</u>

- Yesterday:

- We displayed the web page on the instance by changing the URL from localhost to the URL of the instance.
- We used pandas to manipulate the data to extract the necessary data such as bike ‘availability by station number by time of day by weather’.
- We used pandas to plot a graph when you hardcode the station number into it. See graph below.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff5a7012a90>
```



- Today:
  - We realised that the scraper stopped working at 6.30 am on Sunday morning so we aim to get this back working by the end of the day and determine what caused the error.
- Anything to overcome:
  - Do not know what caused the scraper to stop working again

Sprint3 Day 2 Stand Up (Tuesday 10/4/18)

Robert, Orla, Fatima
- Yesterday:
  - We analysed the nohup.out file
  - Determined that 1 of the stations (station 90) is not updating its information according to the Dublin Bikes website.
  - The API_scraper file only updates the dynamic table when all stations are updated (to avoid duplicates). However because station 90 was not updating, our entire table was not storing new information. We edited our API_scraper file to account for this in future.
- Today:
  - Do a basic sketch of what the website should look like
  - Work on the front end i.e. include a picture of Dublin bikes logo, add a navbar with links to Facebook etc.
  - Get a basic Google Chart to display on click of a station (Does not have to be a meaningful chart as of yet)
- Anything to overcome:
  - None of us have ever used Google Charts before.

Sprint3 Day 3 Stand Up (Wednesday 11/4/18)

Robert, Orla, Fatima
- Yesterday:

- We managed to display a basic pie chart of available bikes and available stands on click of a given station

- Today:
  - Brainstorm how we should go about extracting and parsing the data we need to plot an availability by rain chart for each station using Google Charts.
  - Determine how the graphs we generated using pandas will help for plotting the information with Google Charts

- Anything to overcome:
  - Accessing the required data and getting it into the format required by Google Charts may prove difficult.

Sprint3 Day 4 Stand Up (Thursday 12/4/18)

<u>Robert, Orla, Fatima</u>

- Yesterday:
  - No progress made yesterday as we were all pre-occupied with other modules.

- Today:
  - Get rid of a bug we have in the code; Currently the info window for the first station stays open (i.e. it will not disappear on-clcik of another station, but all other info windows are functioning as expected)

- Anything to overcome:
  - No. It should not be too difficult to fix this issue.

Sprint3 Day 5 Stand Up (Friday 12/4/18)

<u>Robert, Orla, Fatima</u>

- Yesterday:
  - We fixed the issue of the info window staying open for the first station that was clicked.
  - We then fixed another problem we encountered whereby the pie chart would not display on the first station clicked but it would display there-after.

- Today:
  - Put data such as average availability by hour and average availability by day into a local URL using Flask and then plot data from this URL using Google Charts and have it display on click of a station

- Anything to overcome:
  - No- We now have a better understanding of how to go about this because of recent lectures.

Sprint3 Day 6 Stand Up (Monday 16/4/18)

<u>Robert, Orla, Fatima</u>

- Yesterday:
  - We now have bar charts of availability by rain vs no rain (for ach hour) on click of a station (as well as the pie charts we had made before). However we currently only have hourly availability trends chart (NOT daily availability trend chart).

- Today:
  - Backend: Consider what features to include in the model and look into how to use pickle to store objects in Python
  - Frontend: Make drop downs on the web page to select time and day for a prediction

- Frontend: Display real time weather information (icon and temperature) as well as current time on the web app

- Anything to overcome:

- We have never used pickle before

Sprint3 Day 7 Stand Up (Tuesday 16/4/18)

<u>Robert, Orla, Fatima</u>

- Yesterday:

- Robert practiced using pickle to make objects in python

- Today:

- Have a sprint 3 review meeting with Jake during the practical (See below)
- Use pickle to create a model for each station for each hour for each day.
- Create directories do not exist (104 directories for the stations, each with 7 folders for the days, each with 24 folders for the hours, each of these containing a model.pkl file)
- Realised the scraper stopped working – Need to get it back running today and determine the cause of the problem.

- Anything to overcome:

- Unfamiliar with using pickle

ASIDE: Sprint3 Review Meeting (Tuesday 16/4/18)

- We acknowledged that we are behind for this sprint and are at risk of not meeting all the requirements of the specification.
- We looked at our Trello board to see which tasks were of a low priority and decided to only implement these later in the week if we had time even if they were not complex tasks (such as re-entering the map a better position that displays more stations.
- Our product owner informed us that we should try to have a chart that varies depending on the day (currently our bar chart will be the same for weekends and weekdays)
- We acknowledged that such a feature would be more beneficial and informative to the user, but is of a lower priority right now. We will most likely not have time to implement such a feature.
- We set ourselves the following targets for the rest of the week:
  - ↘ By Tuesday night – get a predictive model for each station for each day for each hour
  - ↘ By Wednesday night – have the predictive model integrated with the website
  - ↘ By Thursday night – have all unit tests, documentation and burndown charts completed
  - ↘ By Friday evening – have GitHub repository tidied up and have personal reflections completed.
- Here is our Trello board for this sprint:

Sprint3 Day 8 Stand Up (Wednesday 17/4/18)

- Yesterday:

- Robert and Fatima created models using pickle
- Orla wrote unit tests to check the connection to Open Weather Maps API and JC Decaux API

- Today:

- Get a legend to display on the Google Map indicating what the size of the circles represent (Fatima)
- Remove old CSS file that is not being used. (Fatima)
- Create a JavaScript function to recognize when the user clicks on the drop downs (Robert)
- Parse the data to get weather forecast of wind speed and temperature for a specific day and use this in the predictive model (Orla)

- Anything to overcome:

- Although we have used the Google Maps API before, we have never displayed a Legend on a Google Map before.

Sprint3 Day 9 Stand Up (Thursday 18/4/18)

- Yesterday:

- We did not yet display a Legend. We realized that this will take longer than we had anticipated
- We wrote code that will round the hour picked to the nearest 3 hour interval. This was necessary because Open Weather Maps API forecast only gives data for every 3 hours but our drop down allows the user to get a prediction for every hour of the day. (In hindsight we could have overcome this by only providing 3 hour intervals in the drop down, although this would not have been as user friendly.)

- Today:

- Push yesterday's code to GitHub that will get the nearest hour (Orla)
- On click of a time and day from the drop down menu, display the prediction on the website. (Begin by getting the prediction to display in an alert-box.) (Robert)
- Change the titles and name of the axes of the Google Charts to something meaningful (Fatima)
- Push the sprint notes to GitHub (Orla)

- Anything to overcome:

- Integrating the prediction with website will be the most difficult task. We are not yet completely certain of how to implement this.

Sprint3 Day 10 Stand Up (Friday 19/4/18)

- Yesterday:

- Robert managed to fully integrate prediction with website.
- Fatima edited the titles of the charts and worked on repositioning them.
- Orla made a burn down chart for this sprint and removed unnecessary files.

- Today:

- Fix the legend on the Google Chart. (Currently half the legend is not showing – You have to hover over it to display full name) (Robert)
- Make a README and review documentation. (Fatima)
- Make final changes to scrum notes (Orla)

- Anything to overcome:

- No

Sprint 3 Retrospective (Thursday 18/4/18)

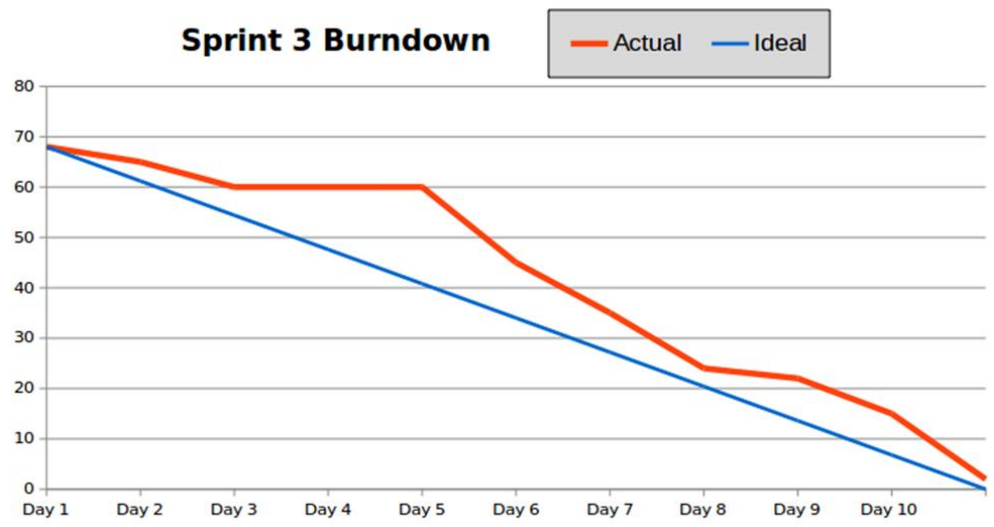
- We reflected on the entire process and acknowledged that we worked co-ordinated very well as a group.

- We acknowledged that the ‘divide and conquer’ approach we used over the past few days was a more efficient use of our time than trio programming (which we used mainly in Sprint 2). The ‘divide and conquer’ approach enabled us to get more tasks completed.
- We also analysed our burndown chart for this sprint (See notes below).

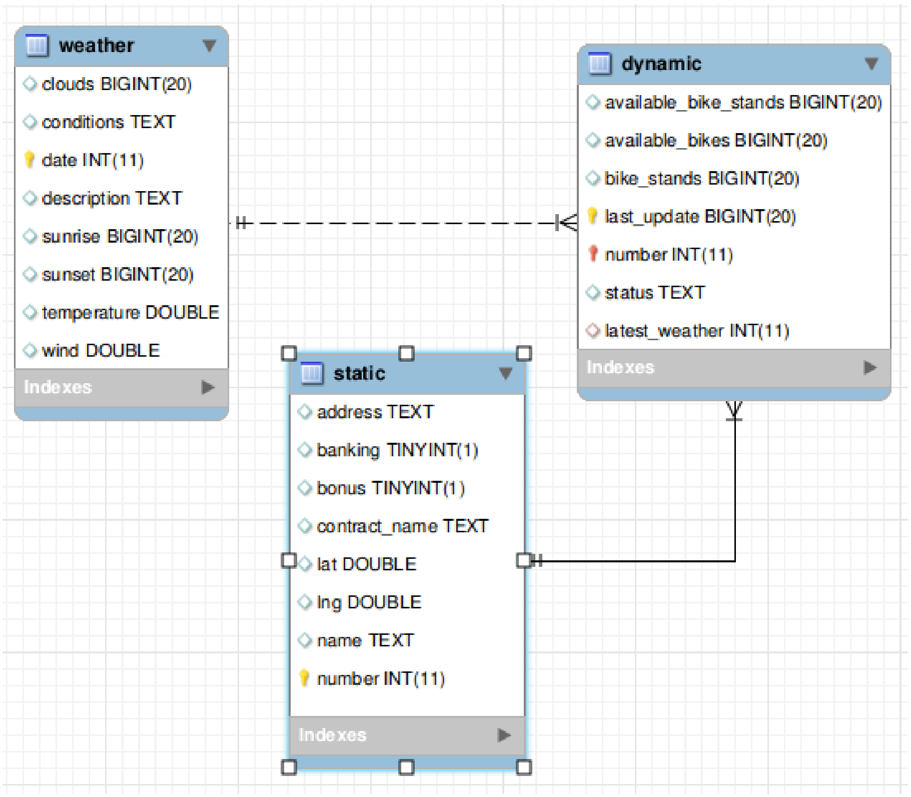
Sprint 3 Burndown Chart

Comments on burndown charts:

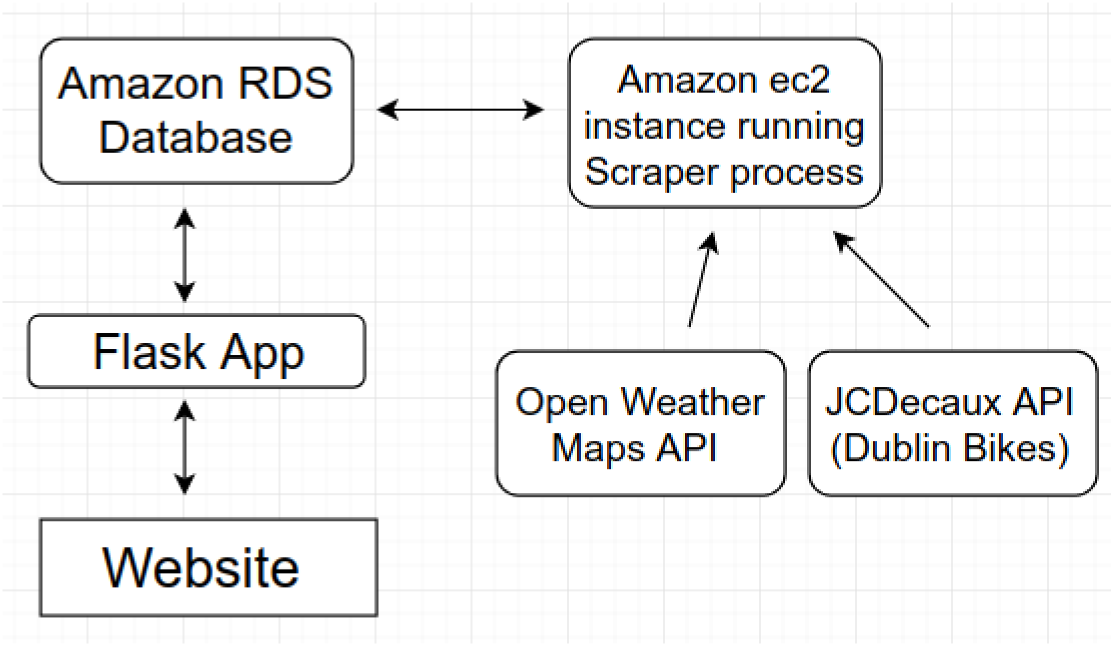
- During this sprint we were continuously not managing to complete our tasks by the estimated dates. The burndown chart reflects this.
- In the middle of this sprint we were worried we would not get all features implemented due to the complexity of certain tasks. For example the chart shows little progress in terms of tasks completed on days 3-5.



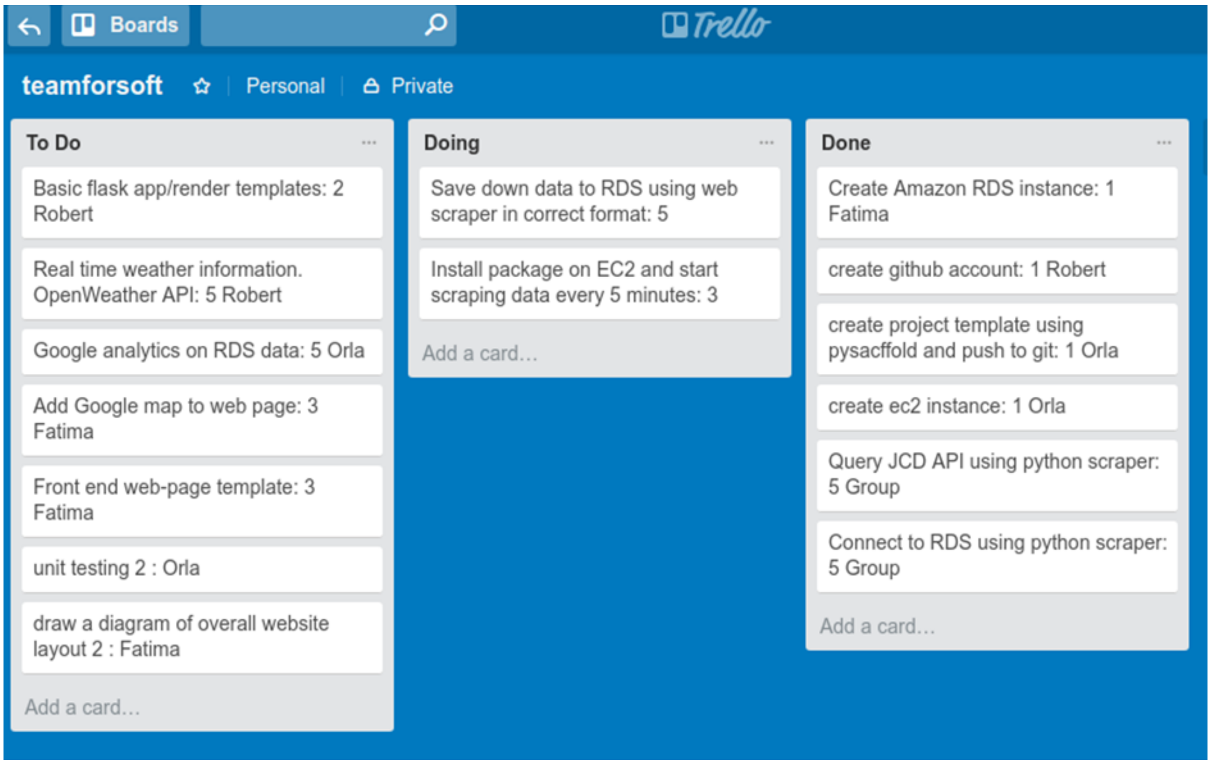
Appendix 1: Structure of our database



Appendix 2: Flow Chart of the overall system


Amazon RDS Database
Amazon ec2 instance running Scraper process
Flask App
Open Weather Maps API
JCDecaux API (Dublin Bikes)
Website

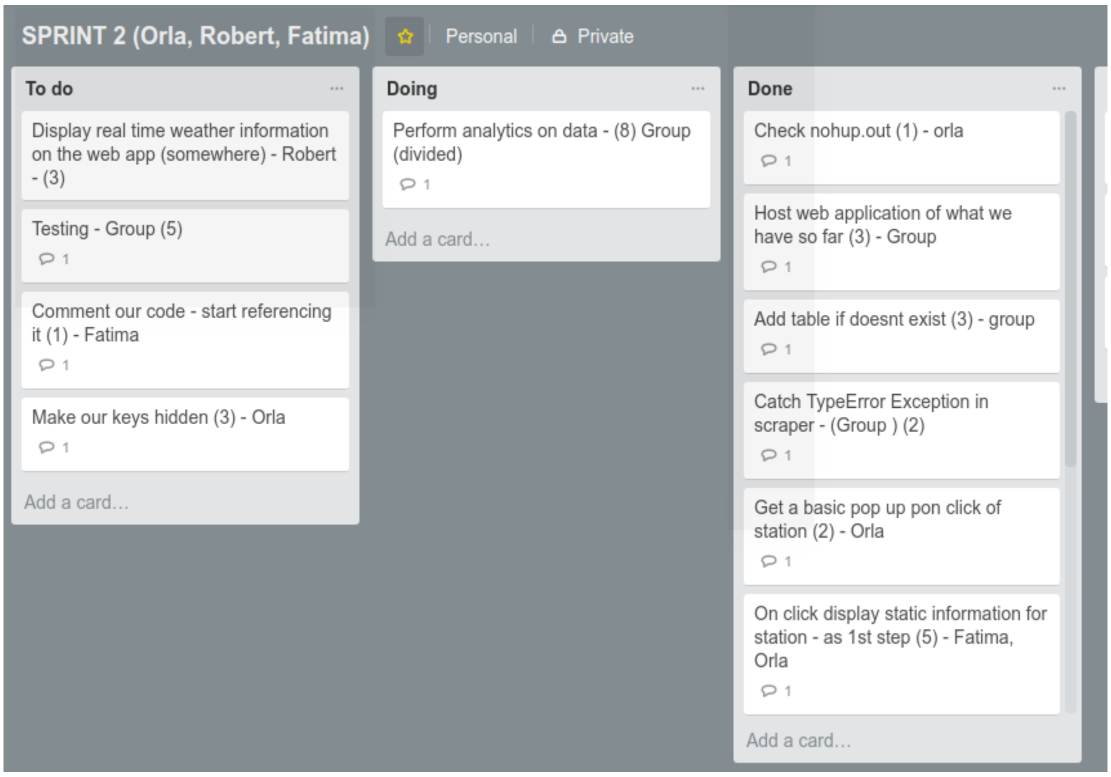
Appendix 3: Trello Boards for each sprint:



To Do
- Basic flask app/render templates: 2 Robert
- Real time weather information. OpenWeather API: 5 Robert
- Google analytics on RDS data: 5 Orla
- Add Google map to web page: 3 Fatima
- Front end web-page template: 3 Fatima
- unit testing 2 : Orla
- draw a diagram of overall website layout 2 : Fatima
- Add a card...

Doing
- Save down data to RDS using web scraper in correct format: 5
- Install package on EC2 and start scraping data every 5 minutes: 3
- Add a card...

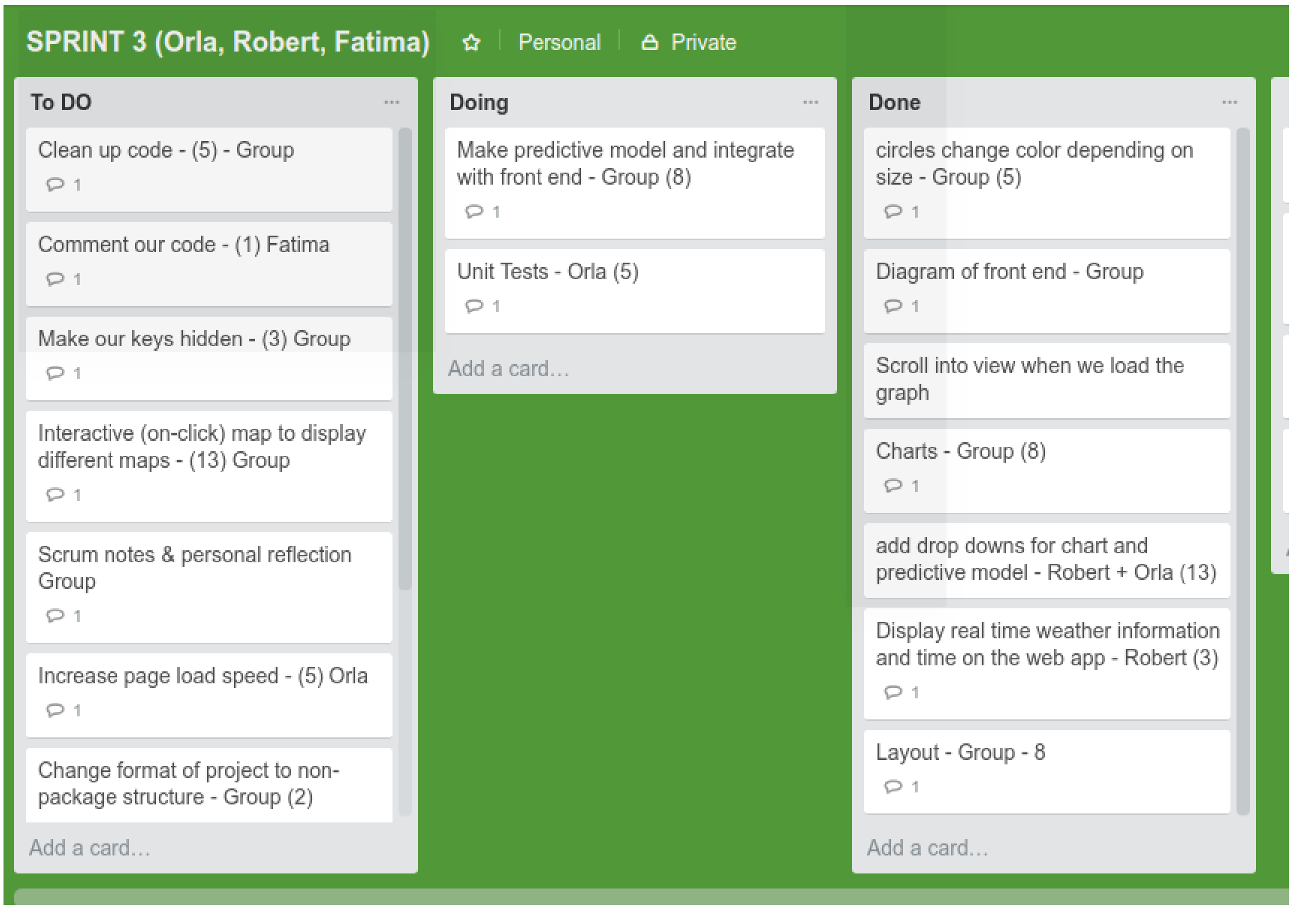
Done
- Create Amazon RDS instance: 1 Fatima
- create github account: 1 Robert
- create project template using pysacffold and push to git: 1 Orla
- create ec2 instance: 1 Orla
- Query JCD API using python scraper: 5 Group
- Connect to RDS using python scraper: 5 Group
- Add a card...



To do
- Display real time weather information on the web app (somewhere) - Robert - (3)
- Testing - Group (5) 1
- Comment our code - start referencing it (1) - Fatima 1
- Make our keys hidden (3) - Orla 1
- Add a card...

Doing
- Perform analytics on data - (8) Group (divided) 1
- Add a card...

Done
- Check nohup.out (1) - orla 1
- Host web application of what we have so far (3) - Group 1
- Add table if doesnt exist (3) - group 1
- Catch TypeError Exception in scraper - (Group) (2) 1
- Get a basic pop up pon click of station (2) - Orla 1
- On click display static information for station - as 1st step (5) - Fatima, Orla 1
- Add a card...



SPRINT 3 (Orla, Robert, Fatima)

Personal | Private

To DO

- Clean up code - (5) - Group 💬 1
- Comment our code - (1) Fatima 💬 1
- Make our keys hidden - (3) Group 💬 1
- Interactive (on-click) map to display different maps - (13) Group 💬 1
- Scrum notes & personal reflection Group 💬 1
- Increase page load speed - (5) Orla 💬 1
- Change format of project to non-package structure - Group (2)

Add a card…

Doing

- Make predictive model and integrate with front end - Group (8) 💬 1
- Unit Tests - Orla (5) 💬 1

Add a card…

Done

- circles change color depending on size - Group (5) 💬 1
- Diagram of front end - Group 💬 1
- Scroll into view when we load the graph
- Charts - Group (8) 💬 1
- add drop downs for chart and predictive model - Robert + Orla (13)
- Display real time weather information and time on the web app - Robert (3) 💬 1
- Layout - Group - 8 💬 1

Add a card…

Sprint 1 Backlog Monday 12 March – 26 March 2018

Backlog Item	Story Points	Responsible	Status
Story #1 - Get set up with necessary infrastructure	**4**		
Create RDS MySQL instance and connect to it via MySQL workbench	1	Fatima	Completed (Day1)
Create Trello account, add group members	0.5	Orla	Completed (Day1)
Create GitHub repository for this product	0.5	Robert	Completed (Day1)
Project template using PyScaffold and push to GitHub	1	Orla	Completed (Day1)
Create Amazon ec2 instance, install Anaconda on it	1	Orla	Completed (Day1)
Story #2 - Scrape Dublin Bikes Data	**28**		
Create a Python script to query the JCD API	5	Group	Completed (Day2)
Connect to RDS using python scraper	5	Group	Completed (Day2)
Store the data in correct format (i.e. in a dataframe) using web scraper	5	Group	Completed (Day2)
Create a table in RDS to store dynamic data and start saving down the data	3	Group	Completed (Day3)
Create a table in RDS to store static data such as address of stations	2	Group	Completed (Day3)
Write a console script to run this package	1	Orla	Completed (Day3)
Get the package installed on EC2 and start scraping data every 5 minutes	2	Robert	Completed (Day3)
User Story #3 – Scrape Weather Data	**4**		
Get a key for the OpenWeather Maps API	1	Robert	Completed (Day4)
Practice using this API by obtaining real time weather information	1	Robert	Completed (Day4)
Create a table to store the most recent weather data	1	Group	Completed (Day4)
Link weather table to dynamic Dublin bikes table (for ease of analysis later)	1	Group	Completed (Day4)
User Story #4 – Create Basic Web app	**7**		
Create Basic Flask App	2	Robert	Completed (Day 6)
Add Google map to web page	2	Fatima	Completed (Day6)
Do a mock up of final website layout	2	Group	NOT COMPLETED
Add basic extras to the web page such as footers etc.	1	Fatima	Completed (Day6)
User Story #5 – Display Position of Stations on Map	**7**		
Add circles of varying sizes to Google Maps	3	Group	Completed (Day 7)
Try to read the positions of the stations from Amazon RDS	2	Orla	NOT COMPLETED
Store the locations of the stations in JSON format	1	Orla	Completed (Day9)
Display the stations by reading this JSON object	1	Robert	Completed (Day9)
Actual Hours Remaining			

Sprint 2 Backlog Monday 26 March – 9 April 2018

Backlog Item	Story Points	Responsible	Status
Story #1: Basic Information on click of a station	**14**		Completion Date
Get a basic info window displaying on click of each station	3	Orla	Day 8 – 4 April
Display static information (e.g.name) of a station in info window	3	Fatima	Day 9 – 5 April
Display dynamic information (availability) of a station in info window	5	Group	Day 9 – 5 April
Make Info Windows disappear on click of another station	3	Group	Day4 – 29 March
Story #2: Reduce wait time of our application	**10**		
Write station locations to JSON file as temporary fix to increase speed	2	Orla	Day 3 – 28 March
Edit scraper to make new table of static data in a local .db fiile	5	Group	Day 9 – 5 April
Edit weather.html so that it displays the markers by reading local db file	3	Group	Day 9 – 5 April
Story #3 : Availability chart by rain for each station	**13**		
Look into overall availability trends using df.resample	3	Group	Day 10 – 6 April
Parse the data into availability during rain & no rain (2 extra columns)	5	Group	Day 10 – 6 April
Plot this data on a bar plot using pandas	5	Group	Day 10 – 6 April
Story #4 : Fix issues with scraper when they arise	**7**		
Scraper down at start of sprint 2 – determine cause of the problem	2	Group	Day 2 – 27 March
Add appropriate try/except blocks to catch these errors in future	2	Group	Day 2 – 27 March
Write exceptions from scraper to a file & the time they occurred	2	Group	Day 9 – 5 April
Tidy up scraper file (add comments and have less code in 'main')	1	Orla	NOT COMPLETED
Story #5 : Miscellaneous	**15**		
Make our keys hidden	2	Fatima	NOT COMPLETED
Display real time weather information on the web app	2	Robert	NOT COMPLETED
Ensure the web app can run on the instance	1	Orla	Day 10 – 6 April
Make our code more readable (use jquery instead of XMLHttp)	3	Group	NOT COMPLETED
Reconsider the current package structure of our project	2	Group	Completed (Day 7)
Do Python Unit Tests of Code we have so far	5	Orla	NOT COMPLETED
Total			

Sprint 3 Backlog Monday 9 April – 20 April 2018

Backlog Item	Story Points	Responsible	Completion Date
Story #1: Display pie chart of availability (Google Charts)	**8**		
Display any basic Google chart on click of a station	3	Group	Day 1 – 9 April
Display a pie chart with data relevant to our application	5	Group	Day 2 –10 April
Story #2: Display availability by rain chart (Google Charts)	**15**		
Parse the data to obtain availability (by rain/no rain) and time	5	Group	Day 5 – 13 April
Make a Flask URL and store this data in a URL for each station	2	Group	Day 5 – 13 April
Display a basic bar chart on click of a station with hard coded values (for now!)	3	Group	Day 5 – 13 April
Populate the bar chart by reading from the Flask URL for that station	5	Group	Day 5 – 13 April
Story #3 : Make a model to predict bike availability	**13**		
Consider what features to include in the model and learn about pickle	2	Robert	Day 6 – 16 April
Use pickle to create a model for each station for each hour for each day	8	Group	Day 7 – 17 April
Test the models with data for 1 station to ensure reasonable results	3	Group	Day 7 – 17 April
Story #4 : Integrate model with website	**20**		
Make drop downs on the web page to select time and day	5	Orla	Day 6 – 16 April
(OPTIONAL) Make a submit button on the main html page (to get prediction)	2	Fatima	NOT COMPLETED
Make a Flask URL in views.py to hold the model for that station	5	Group	Day 9- 19 April
Edit the main html file to access these predictions on click of submit	8	Group	Day 10- 20 April
Story #5: Other	**12**		
Display real time weather information and time on the web app	3	Orla	Day 6 – 16 April
Do Python Unit Tests of Code we have so far	5	Orla	Day 10- 20 April
Fix issues with scraper if they arise	2	Group	Day 8 – 18 April
Tidy up website, make legend for map etc.	2	Fatima	Day 9- 20 April
Total			