Project Rothar

COMP30670- Software Engineering

Dr. Aonghus Lawlor

Conor Devoy - 15201361

David O Keeffe - 15203741

Andrew Poole - 12252073

Sprint 1

Team	Project Rothar
Names	Conor Devoy, David O Keeffe, Andrew Poole
Present (Initials)	CD, DOK, AP
Student Number	15201361, 15203741, 12252073
Date	Monday 29th February – Friday 11th March
Sprint No.	1
Sprint Length	2 Weeks

Sprint Backlog

Task	Owner	Estimate	Completed
Breakdown of Project- Tasks clearly defined	Collab	1 Day	Yes
Division of Tasks into individual Sprints- 3 sprints envisioned with 3 distinct stages of development to the project.	Collab	1 Day	Yes
Tasks for first sprint assigned to group members & date for next meeting set	Collab	1 Day	Yes
Establish Group repository	CD	1 Day	Yes
Database setup #Move to Sprint 2 Andy	CD	3 Days	Yes
Server Configuration & Implementation	AP/CD	2 weeks	Yes
Write code to scrape API & run for 1 week	AP	1 Week	Yes
Documentation & Sprint Management	DOK	2 weeks	Yes

Breakdown of Project

Our first meeting primarily concerned itself with accustoming ourselves with the specifications for the project so we could best approximate the possible Sprint cycles and overarching objectives. We discussed the linear path from start to end by creating small achievable tasks for each objective and were thus able to best divide up the work for the first Sprint according to the individual strengths and weaknesses of the group members in order to get the project off the ground.

User Requirements

A large portion of the meeting was spent analysing the end user requirements in order to have a better understanding of how we needed to format and store the data we would query. Some of the items raised were:

- Current and Historical Station Information
    - Our plan is to request station information every 5 minutes for a week.
    - From the data collected, we plan for the user to be able to get current information i.e. when the user clicks a station marker, we will get the current day and time and look up information from the same day and time in the data we retrieved over a week.
    - We also plan to have a feature where the user is able to click a day of the week and a chart will display a time series of the available bikes for that station on that day.
    - This requires a best approximation of the closest timestamp to the queried time as they are unlikely to exactly match our data as it is only in 5 minute intervals.
- Marker design
    - Each station will display a circle around the station. Depending on the number of bikes available, the circle's size and colour will vary.
- Directions and Geolocation

- We plan on having a feature where the user can click a button and the map will display a route to the nearest station to the user with bikes available.
- Another feature of the application will be for the user to plan their bike journey. Assuming the user has a bike, they will be able to enter a location and the map will display directions to the nearest station with available bike spaces to that location.

Division of tasks

Facing a 2 week break before we would have the chance to meet again we thought it best to just get the initial stages of the project underway in the first sprint. This would allow us to reconvene after the break and begin the next Sprint where we envision the majority of the collaboration taking place.

Setting up the Server

A member of another group had finalised their server setup using the Amazon Web Services and Andy was keen to come to grips with that infrastructure so he took on the task of setting up the server.

Conor availed of the server space offered by Aonghus to set up another one as a backup just in case the learning curve for AWS was too steep and also as an exercise in curiosity.

Using the server space provided by the lecturer, we used SSH to remotely access the server. Screen was then used keep the API Scrapper code running on the server, without a dedicated connection from the local machine. The code was then let run for an entire week. This data was collected in both CSV format and in a database. Conor checked on the process twice to make sure that it was still running over the week, but screen allowed us to "fire and forget" the code until the week was over.

Database Configuration / API Scraper

The plan for the script to run on the server is to have it request data every 5 minutes and to have this requested data appended to a SQLite database. To do this we expect it will require creating a database and a database table for the data to be added to. We will also need to get an API key so that we can make requests to the API.

Sprint 1

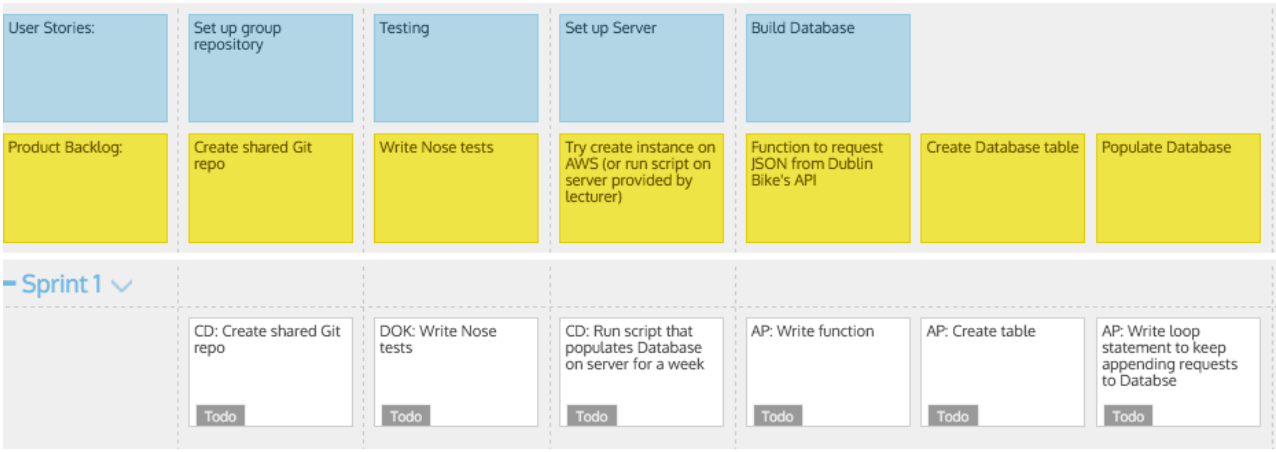
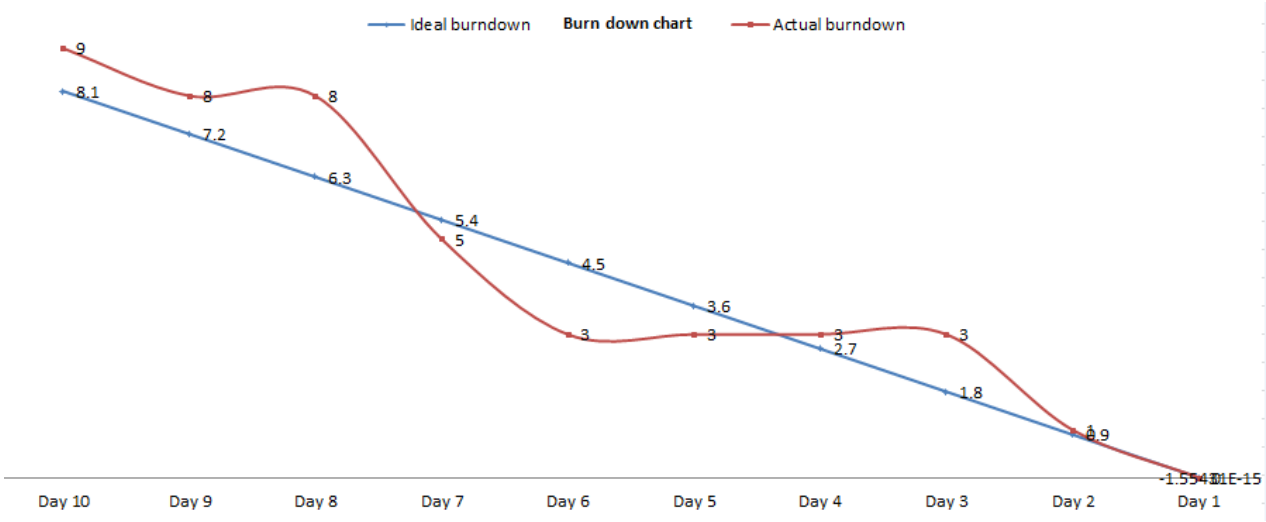


Fig. Sprint 1 Backlog

Burn Down Chart

Sprint 1: Burn down chart

Project	Story	Task	Start	Day 10	Day 9	Day 8	Day 7	Day 6	Day 5	Day 4	Day 3	Day 2	Day 1
ProjectRothar	Repo	Create shared Repo	2	2	1	0	0	0	0	0	0	0	0
ProjectRothar	Data	Write Function	2	2	2	3	0	0	0	0	0	0	0
ProjectRothar	Data	Create Table	1	1	1	1	1	0	0	0	0	0	0
ProjectRothar	Data	Loop to append requests for entire week	1	1	1	1	1	0	0	0	0	0	0
ProjectRothar	Server Setup	Run Scrapper	3	3	3	3	3	3	3	3	3	1	0
Ideal - Remaining efforts in uninterrupted working hours			9	8.1	7.2	6.3	5.4	4.5	3.6	2.7	1.8	0.9	-0
Actual - Remaining efforts in uninterrupted working hours			9	9	8	8	5	3	3	3	3	1	0



Review & Retrospective Summary

Problem:

Andy was unable to properly attempt the creation of an instance on the Amazon EC2 web service because of an SSH problem. He attempted to create this instance on the UCD campus which meant using the EDUROAM connection. He was prevented from doing so due to his laptop experiencing connectivity issues.

Resolution:

We had foreseen potential problems (although not connectivity problems) with setting up the AWS server because no one in the group had used the web service before.

We pre-empted this problem by having Conor set up the server using space provided by Aonghus on his machine as a back-up as Andy was keen to get to grips with the AWS infrastructure.

Lesson:

Detailed planning saves time from potential disruptions. Had we not planned for Conor to set up a server as a backup to our original plan we may have ended Sprint 1 without any data. By having the failsafe option it allowed a member of our team to be ambitious and have the freedom to fail. In this way, a possible failure to meet his goals didn't disrupt the progress of the team as a whole.

Sprint 2

Team	Project Rothar
Names	Conor Devoy, David O Keeffe, Andrew Poole
Present (Initials)	CD, DOK, AP
Student Number	15201361, 15203741, 12252073
Date	Monday 21th March – Friday 1st of April
Sprint No.	2
Sprint length	2 week (1st week during Easter Break)

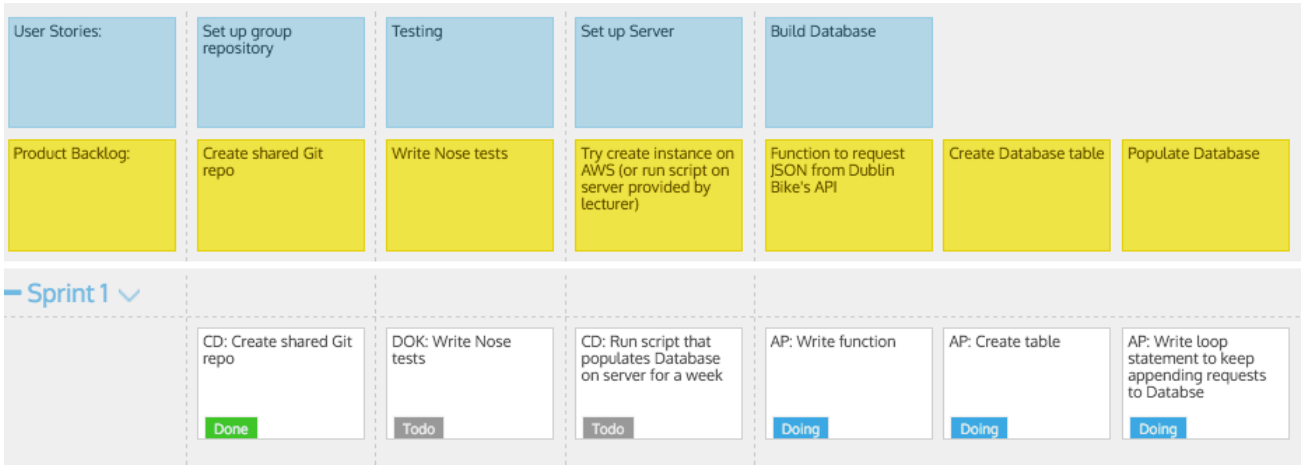
Sprint Backlog

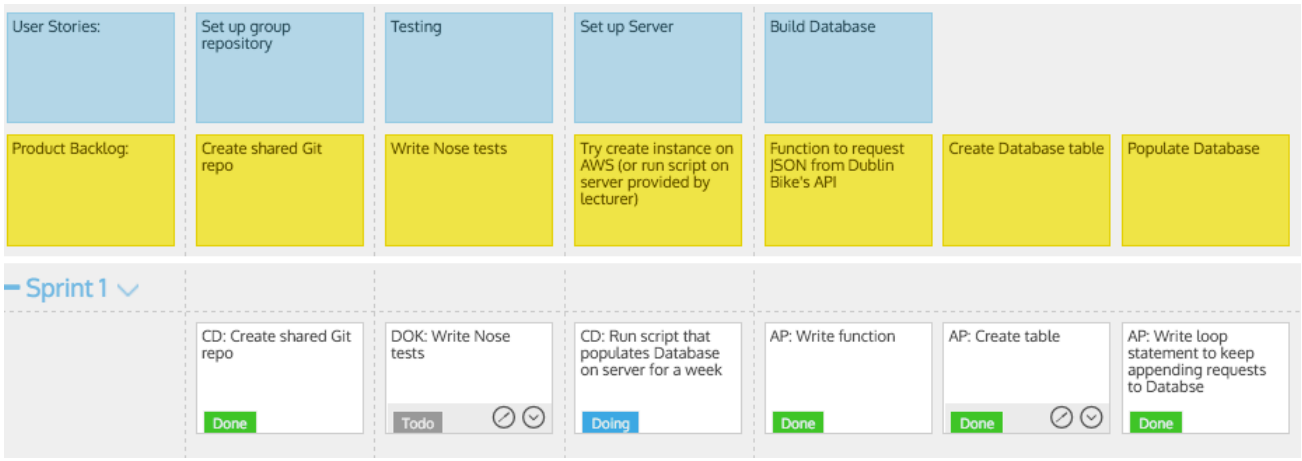
Task	Owner	Estimate	Completed
Sprint 1 review meeting	Collab	1 Day	Yes
Sprint 2 set-up meeting	Collab	1 Day	Yes
HTML/CSS	DOK	3 days	Yes
JavaScript - Initialise map	DOK	5 days	Yes
Render HTML & run on server	Conor	5 days	Yes
Dynamic map markers representing station info	AP	2 week	Yes
Documentation	Conor	2 week	Yes

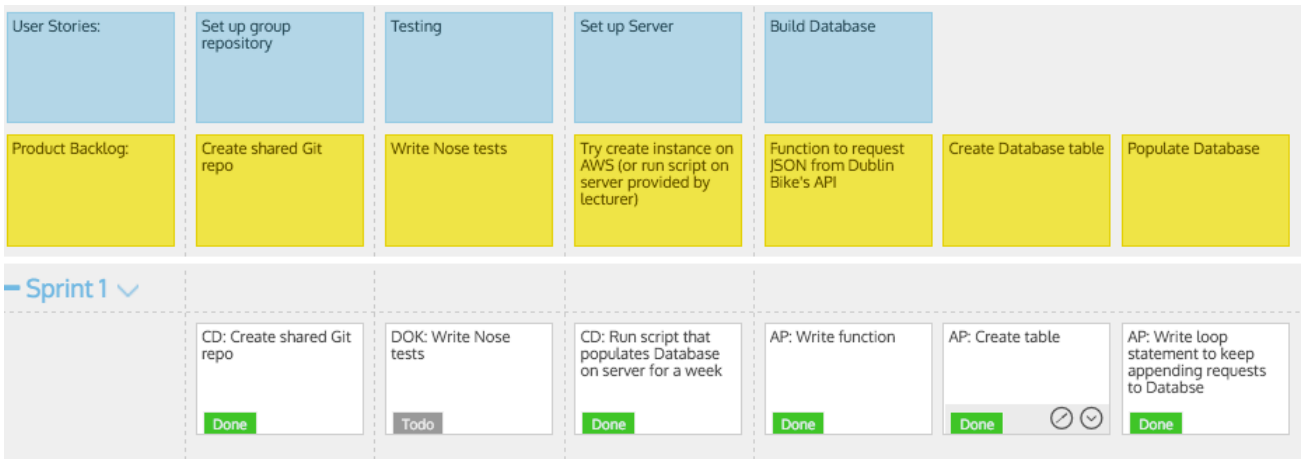
Sprint 1-Review

Apart from not being able to set up the AWS EC2 instance to run our script, Sprint 1 went well with no major issues. The database table was successfully populated with data from the API. After a week of working as a team, we realised how important it is to communicate as a team. We all agreed that each of us needed to work on this aspect because the project is meant to be a collaborative effort.

Sprint 1 Backlog Progress







Sprint 2 Set-up

With data to work with we had to break down the next set of tasks and divide them up for the following week. This meeting focused more in depth on what general themes we wanted for our HTML page, what design we imagined our map markers taking and the responsive nature of those markers. We all gave our input on the look we envisioned for different aspects of the application and drew our ideas on a white board. This allowed us to finalise a general layout we were happy with. We decided that we should try to stay faithful to the design but that ultimately this would be subject to the discretion of the person coding.

We decided that writing the HTML page should be an iterative process based on the cumulative ideas of the person writing the code and the other 2 members applying functionality to it.

The most important thing in terms of getting this Sprint off the ground was having a very basic HTML structure and map initialisation very early in the Sprint so the other members could then work on that to carry out their responsibilities and feedback layout and styling suggestions.

HTML/CSS/JS: Map Initialisation

David took on the role of initialiasing the basic outline of the webpage. It was very important to get the basic foundation done quickly so other group members could start to build on that and so this was given a deadline of 3 days at the latest.

The goal was to have a HTML layout with some general styling and to have our a Google map created and displayed on the page.

Render HTML & Run on Server

It was decided that Conor and Andy would be responsible for the majority of the work involved in setting up the web application framework. However, we felt that we needed to as a group to do some research on what web application framework to use. In this sprint, it is Conor's task to set up the initial script which will allow us to create a simple web application. Our goal is to have our application running on the local server and to eventually have it handle any database queries (to be completed in Sprint 3).

Map Markers, Circles and Station Information

Initially, we had thought we would use historical data from our database to populate our map with station markers but we have decided to alter this plan because it would be more beneficial to make an API request as the page is being loaded and using this current information to place the station markers around the map. It is Conor and Andrew's task to populate the Map with station markers, circles (which display different colours and sizes depending on the percentage of available bikes) and information windows (which will display the current station information from the API request made on page load).

Sprint 2 Story

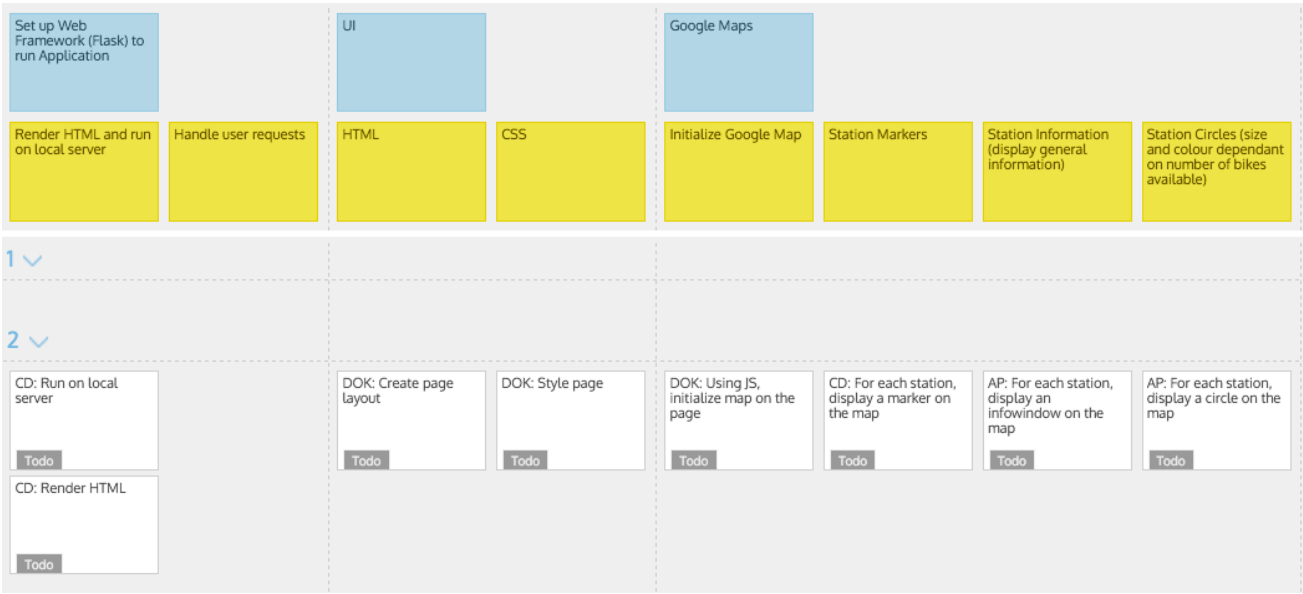
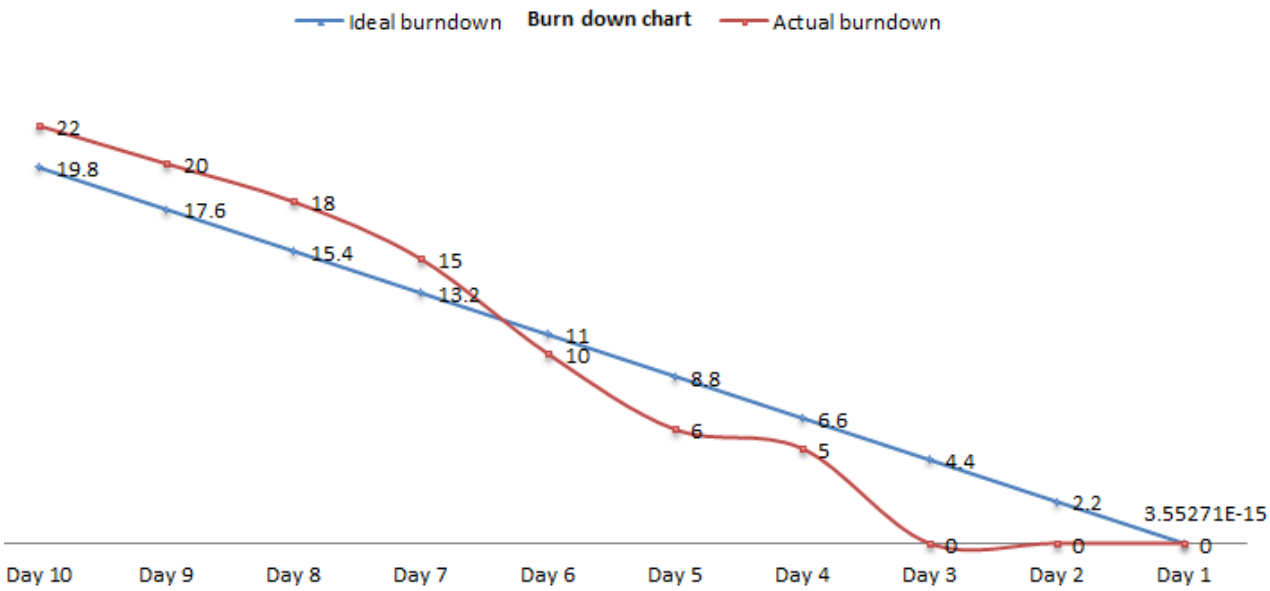


Fig. Sprint 2 Backlog

Burn Down Chart

Sprint 2: Burn down chart

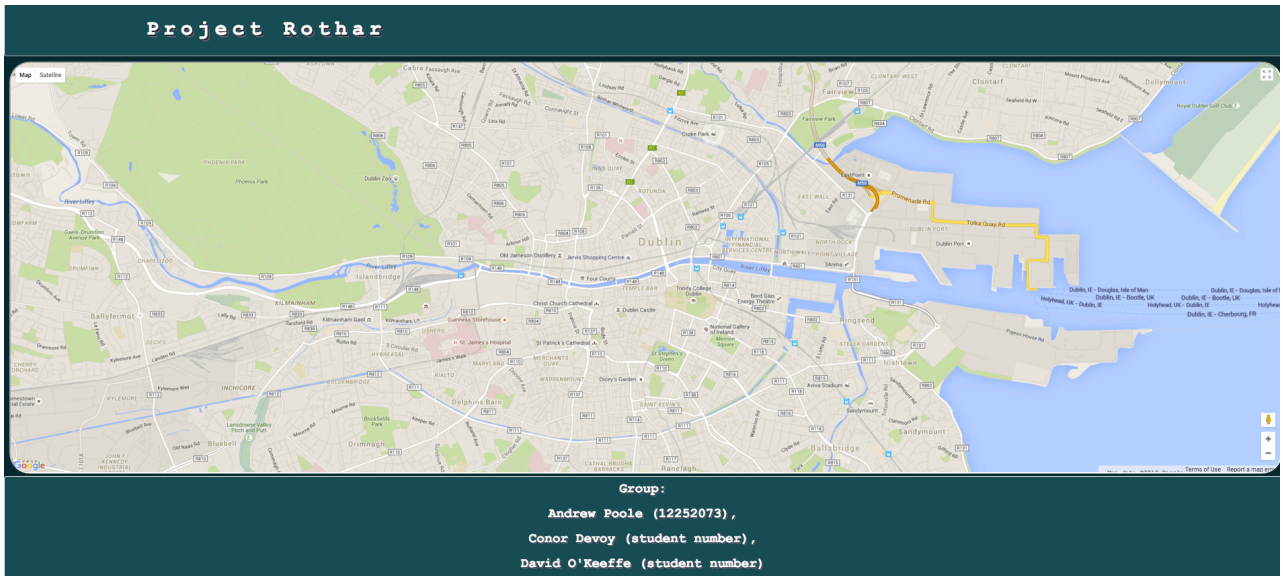
Project	Story	Task	Start	Day 10	Day 9	Day 8	Day 7	Day 6	Day 5	Day 4	Day 3	Day 2	Day 1
ProjectRothar	Flask setup	Render HTML	3	3	1	0	0	0	0	0	0	0	0
ProjectRothar	Flask setup	Run on Local Server	3	3	3	2	0	0	0	0	0	0	0
ProjectRothar	UI	Create Page Layout	3	3	3	3	2	0	0	0	0	0	0
ProjectRothar	UI	Style Page	3	3	3	3	3	0	0	0	0	0	0
ProjectRothar	Google Maps	Initialize Map	2	2	2	2	2	2	0	0	0	0	0
ProjectRothar	Google Maps	Station Marker	2	2	2	2	2	2	0	0	0	0	0
ProjectRothar	Google Maps	Station Info Window	3	3	3	3	3	3	3	2	0	0	0
ProjectRothar	Google Maps	Station Circle (Bike Availability)	3	3	3	3	3	3	3	3	0	0	0
Ideal - Remaining efforts in uninterrupted working hours			22	19.8	17.6	15.4	13.2	11	8.8	6.6	4.4	2.2	0
Actual - Remaining efforts in uninterrupted working hours			22	22	20	18	15	10	6	5	0	0	0



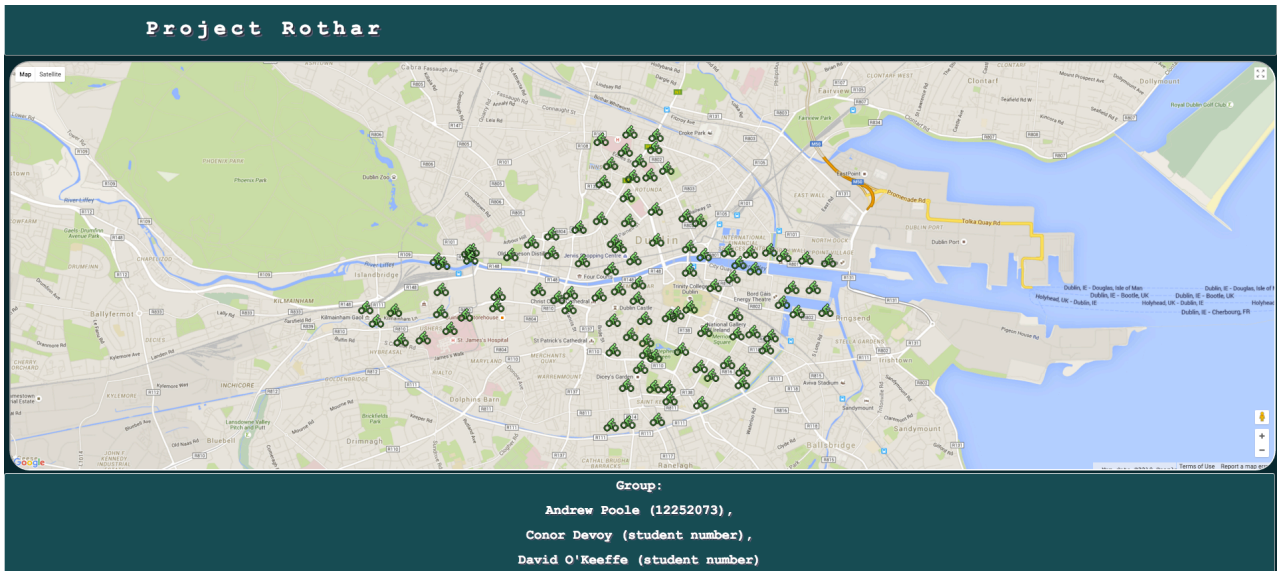
Review & Retrospective Summary

Application Progress

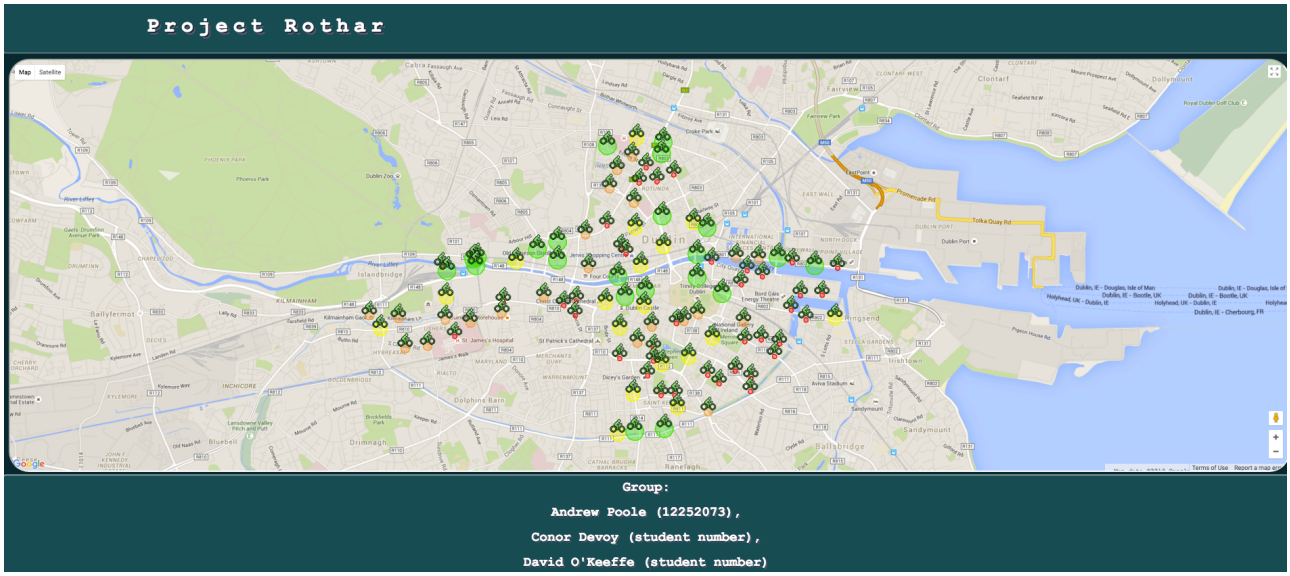
1. Initialized Google Map on page



2. Placed markers for stations on page

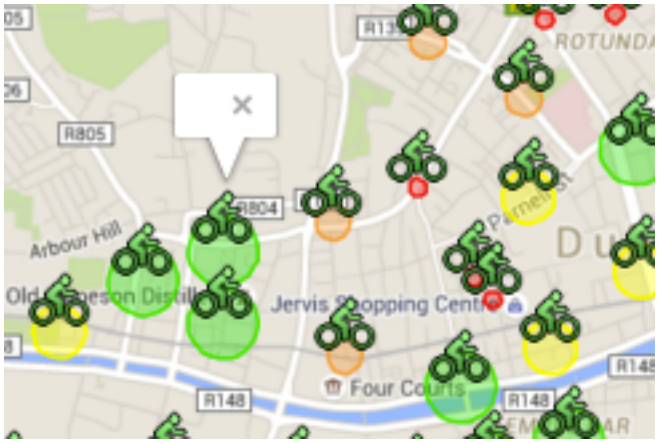


3. Displaying circles for each station. Colours and size give a visual indication of the percentage of available bikes



4. Infowindow display for each station.

Information content is already contained within infowindow. In the next sprint we will need to style the infowindows so that the information is visible.



Problem:

At this point we had not figured out how to have Flask query our database. We initially planned to have the database return each stations latitude and longitude on page load.

Resolution:

This issue was resolved by performing an API request on page load. This request returned each station's most recent information.

Lesson:

We felt that due to our lack of experience, we were unable to perform a task we had set ourselves. In this circumstance it actually worked to our advantage because we felt the API request offered current information about each station. Lack of foresight and detailed planning about what is involved in each task to be completed led to this problem. Being more meticulous about the tasks we set ourselves will allow us to know what is more feasable when we set out our sprint backlog.

Sprint 3

Team	Project Rothar
Names	Conor Devoy, David O Keeffe, Andrew Poole
Present (Initials)	CD, DOK, AP
Student Number	15201361, 15203741, 12252073
Date	Monday 4th April – Friday 15th April
Sprint No.	3

Sprint Backlog

Task	Owner	Estimate	Completed
Sprint 2 Review Meeting	Collab	1 day	Yes
Sprint 3 Set up Meeting	Collab	1 day	Yes
Handling Database Requests	AP	2 weeks	Yes
Style Station Info windows	AP	2 weeks	Yes
Time-Series chart for stations	CD	2 weeks	Yes
Geolocation of user	DOK	2 weeks	Yes
Display route on map	CD	2 weeks	Yes
Find nearest station to destination	AP	2 weeks	Yes
Documentation	DOK	2 weeks	Yes

Sprint 2 Review

In sprint 2, there weren't any major issues. Members completed their respective tasks in a timely fashion. The plan set out for the order of the sprint backlog to be completed was made in such a way that each member would get a chance to code.

We decided to use the Flask web framework to support the development of our Dublin bikes web application. Conor was able to get the web page to be rendered on the local host using Flask. This involved creating an application object, rendering our HTML and running the application object on the server with Flask's run method.

Each member of the group was already familiar with HTML, CSS and Javascript, and the Google maps API so the learning curve wasn't too steep in setting up the web page. David had already created the initial template to be rendered by the time Conor got Flask set up so we could test the Flask rendering of the template straight away.

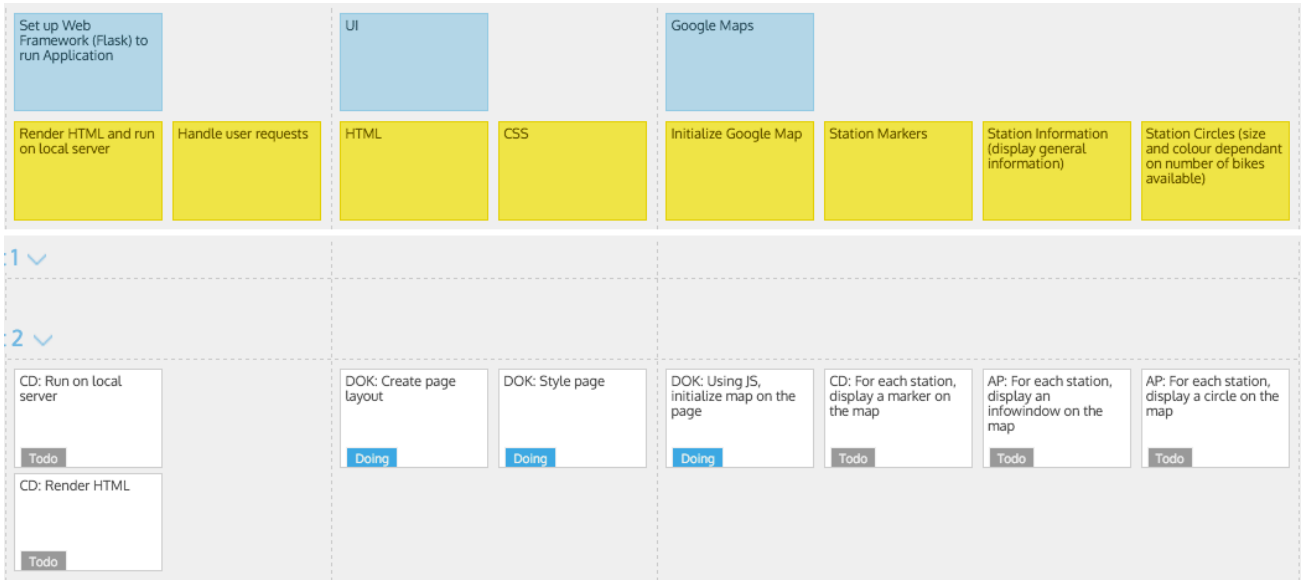
David set the Google map to centre on Dublin city centre at a height that allowed for all the Dublin bike markers to be visible on page load. Conor and Andrew worked on how each station would be displayed on the map. After looking into using the database to display the markers, i.e. query database for each stations latitude and longitude, we realised it would be more useful to have the application perform an API request on page load. Doing it this way gave us each station's latitude and longitude, and also each station's most recent information.

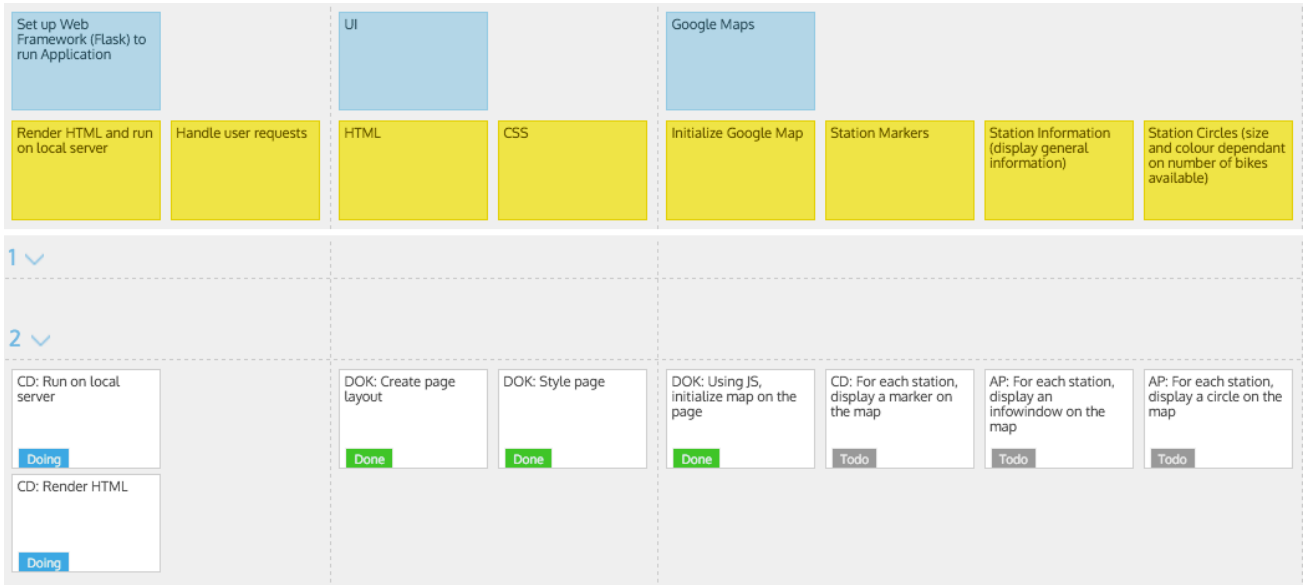
Each station's location marker is displayed on the map using the following image  .
Each station's respective circle's size and colour will depend on the percentage of available bikes at that station. This gives the user a quick reference to look at to see where bikes are available.

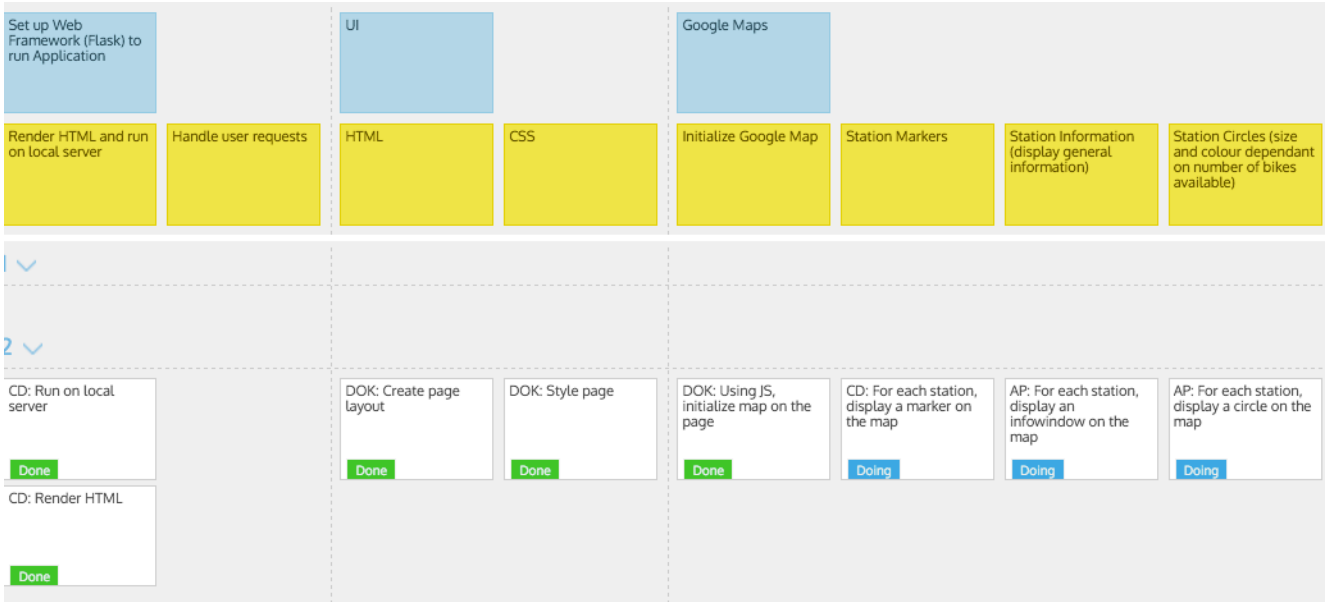
For each station, an infowindow will be displayed with the most recent information if the user clicks the icon but is a task for our next Sprint. We imagine the information for each info window will be:

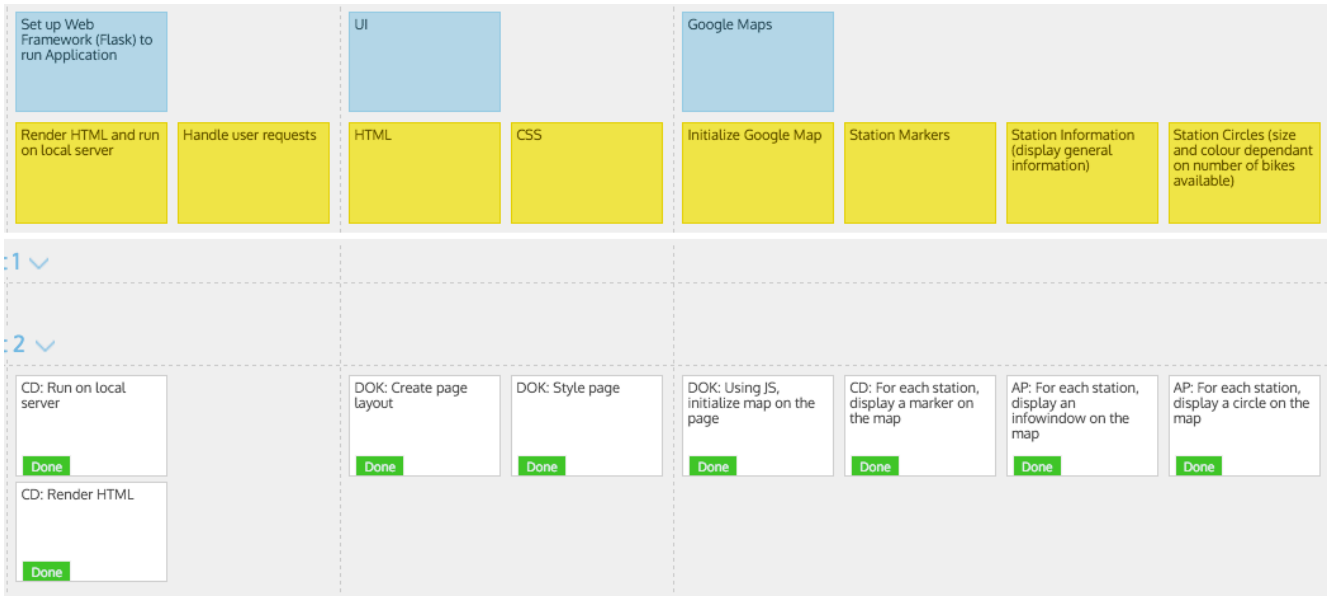
1. Station status i.e. open or closed
2. Available bikes
3. Available spaces
4. Card machine availability
5. Time of last update

Sprint 2 Backlog Progress









Sprint 3 Set up

The main objective for this sprint is to have our application framework handle database queries for station and day specific information in JSON format. From this data, we will display a time series chart for the day selected in the infowindow. This chart is the basic functionality we want our info window to have and will show the number of available bikes historically available at different times of the day.

We have allocated tasks that allow us to aim for extra user features. The first will allow the user to click a button that locates the nearest station with available bikes. Once this station is located, the map will display a route from the user to the station.

Assuming the user has gotten a bike, the second feature allows the user to enter a destination location. When this location is submitted, a route will display on the map which locates the nearest station to the destination with spaces currently available.
If we have time after all features have been implemented, we would like to be able to deploy the application. However, we realize this goal may not be achieved because our group isn't familiar with what would be involved in completing this task.

Handling Database Requests

After our shortcomings last week with getting Flask to handle querying the database, we all made an effort in gaining a better understanding of the web framework. For this sprint, it was Andy's task to get the application to query the database for the relevant station and day. Conor would use this information to display a time series chart.

Time series charts for stations

The plan is to have the days of the week displayed in each station's infowindow. When a day is clicked, a chart for that station will display its historical information for that day i.e. how many bikes are available at different times throughout the day. The database will be queried each time a day is clicked.

Geolocation/ Route Display/ Closest destination

These tasks in the Sprint were the responsibility of individual group members but the tasks were carried out in conjunction with one another as they are so closely related.
Two features:

1. Find nearest station with available bikes to user
2. Find nearest station with spaces available to the destination entered

David handled the first task to get geolocations of the user and the destination entered. This location will be used for checking which station is closest and also to render a route to that

station. The second task will be to locate the nearest station with available bikes / spaces to the user destination. The last task is to display a route from the user's location to the station found.

Sprint 3 Story Map

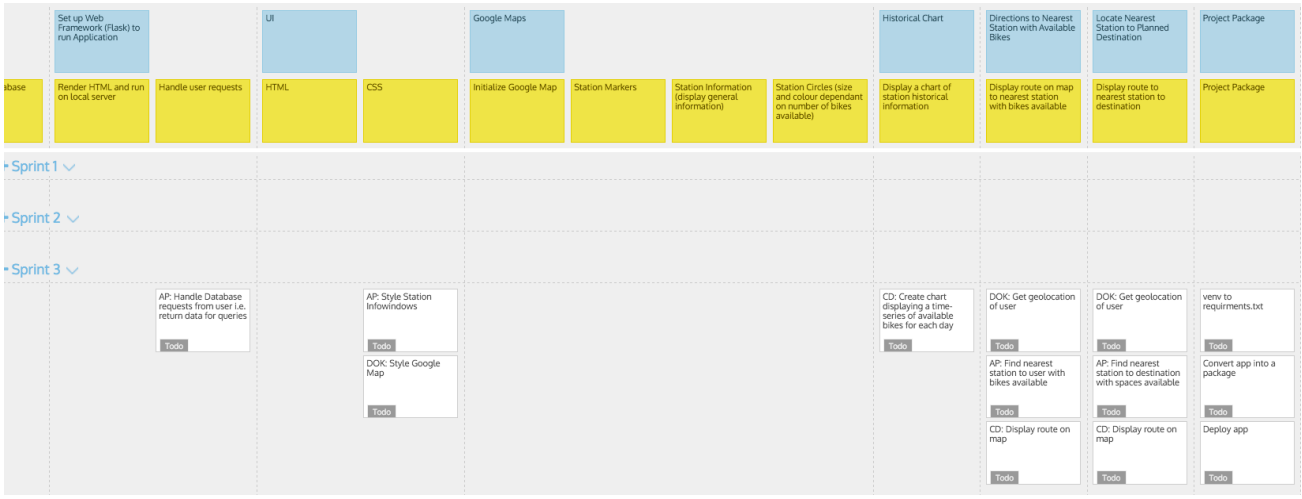
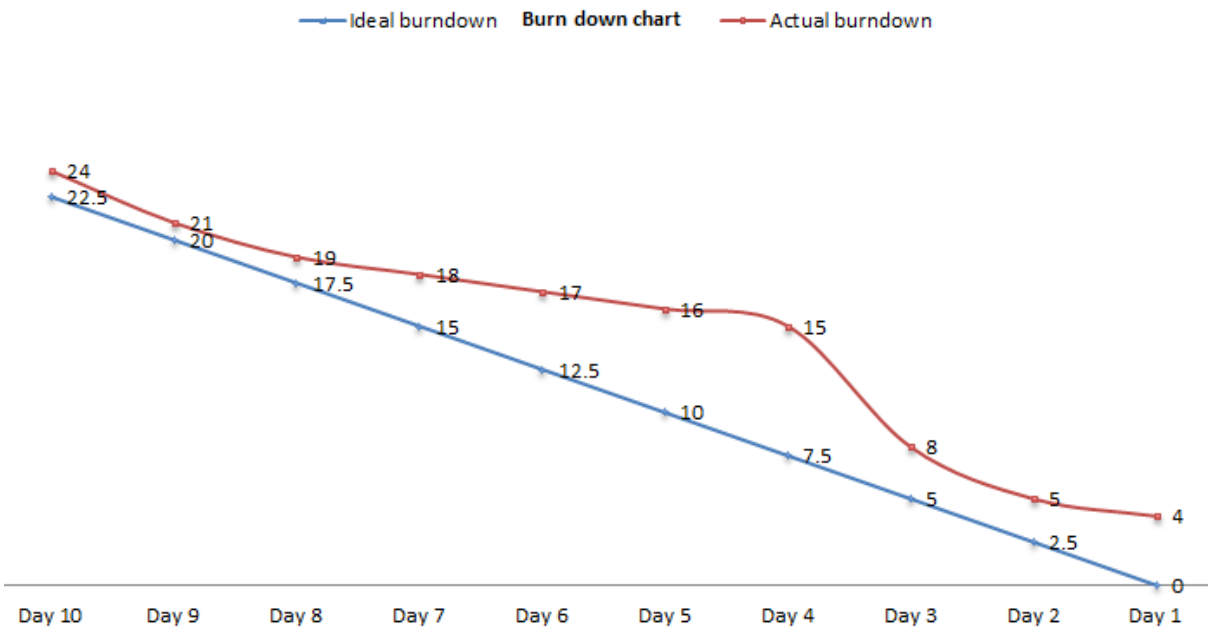


Fig. Sprint 3 Backlog

Burn Down Chart

Sprint 3: Burn down chart

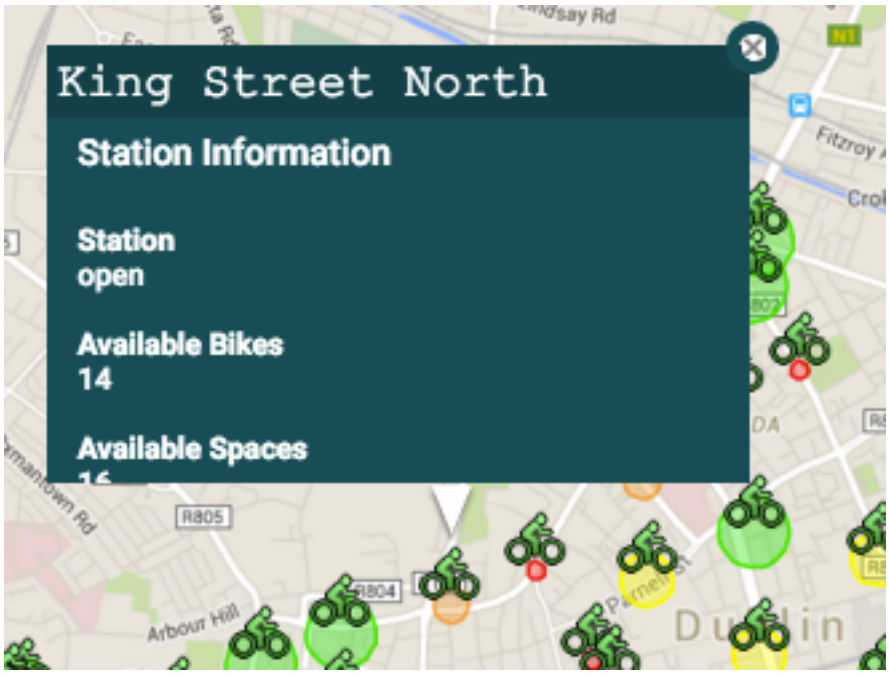
Project	Story	Task	Start	Day 10	Day 9	Day 8	Day 7	Day 6	Day 5	Day 4	Day 3	Day 2	Day 1
ProjectRothar	Testing	JS error handling	1	0	0	0	0	0	0	0	0	0	0
ProjectRothar	Flask App	Handle DB requests	3	4	3	2	0	0	0	0	0	0	0
ProjectRothar	UI	Style Station Info Window	3	2	0	0	0	0	0	0	0	0	0
ProjectRothar	UI	Style Google Map	1	1	1	0	0	0	0	0	0	0	0
ProjectRothar	Historical Chart	Create Chart	3	3	3	3	4	3	2	0	0	0	0
ProjectRothar	Geolocation and Routing	Get location of user	2	2	2	2	2	2	2	2	0	0	0
ProjectRothar	Geolocation and Routing	Find nearest station	2	2	2	2	2	2	2	3	0	0	0
ProjectRothar	Geolocation and Routing	Display route	2	2	2	2	2	2	2	2	0	0	0
ProjectRothar	Locate nearest station to planned Dest.	Location of user	1	1	1	1	1	1	1	1	1	0	0
ProjectRothar	Locate nearest station to planned Dest.	Find nearest station	1	1	1	1	1	1	1	1	1	0	0
ProjectRothar	Locate nearest station to planned Dest.	Display route	1	1	1	1	1	1	1	1	1	0	0
ProjectRothar	Project Package	venv to req.txt	1	1	1	1	1	1	1	1	1	1	0
ProjectRothar	Project Package	App to package	4	4	4	4	4	4	4	4	4	4	4
Ideal - Remaining efforts in uninterrupted working hours			**25**	**22.5**	**20**	**17.5**	**15**	**12.5**	**10**	**7.5**	**5**	**2.5**	**0**
Actual - Remaining efforts in uninterrupted working hours			**25**	**24**	**21**	**19**	**18**	**17**	**16**	**15**	**8**	**5**	**4**



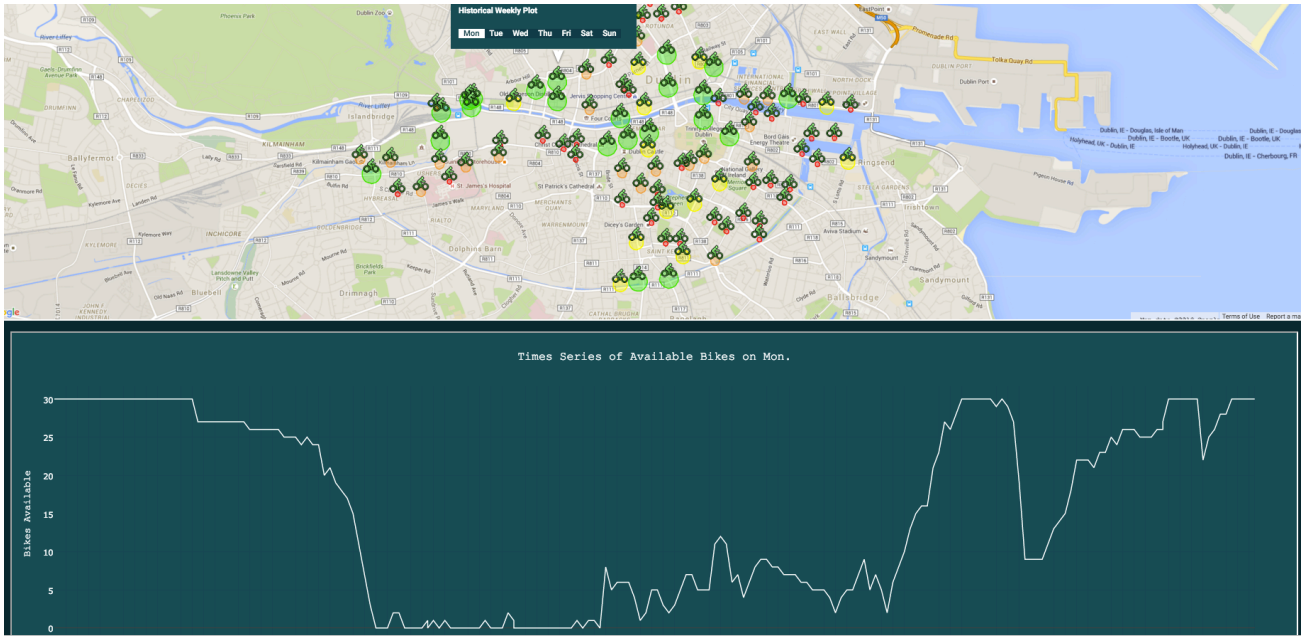
Review & Retrospective Summary

Application Progress

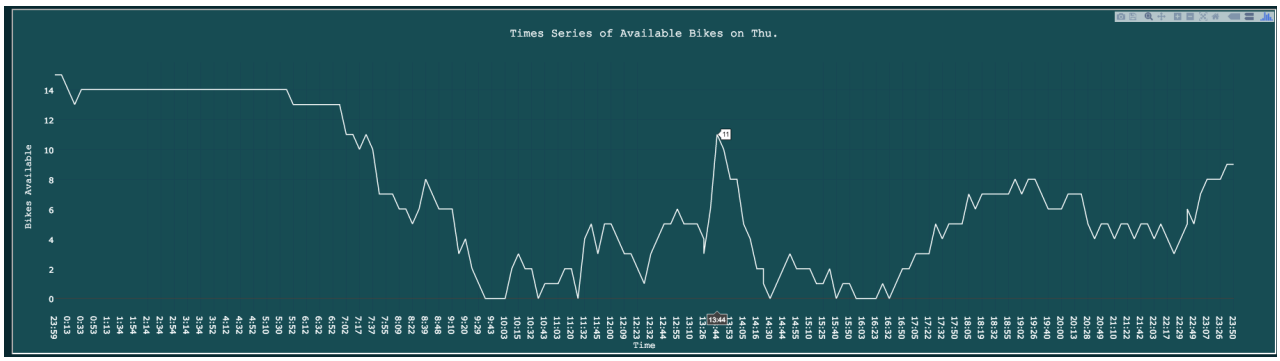
1. Infowindow styling



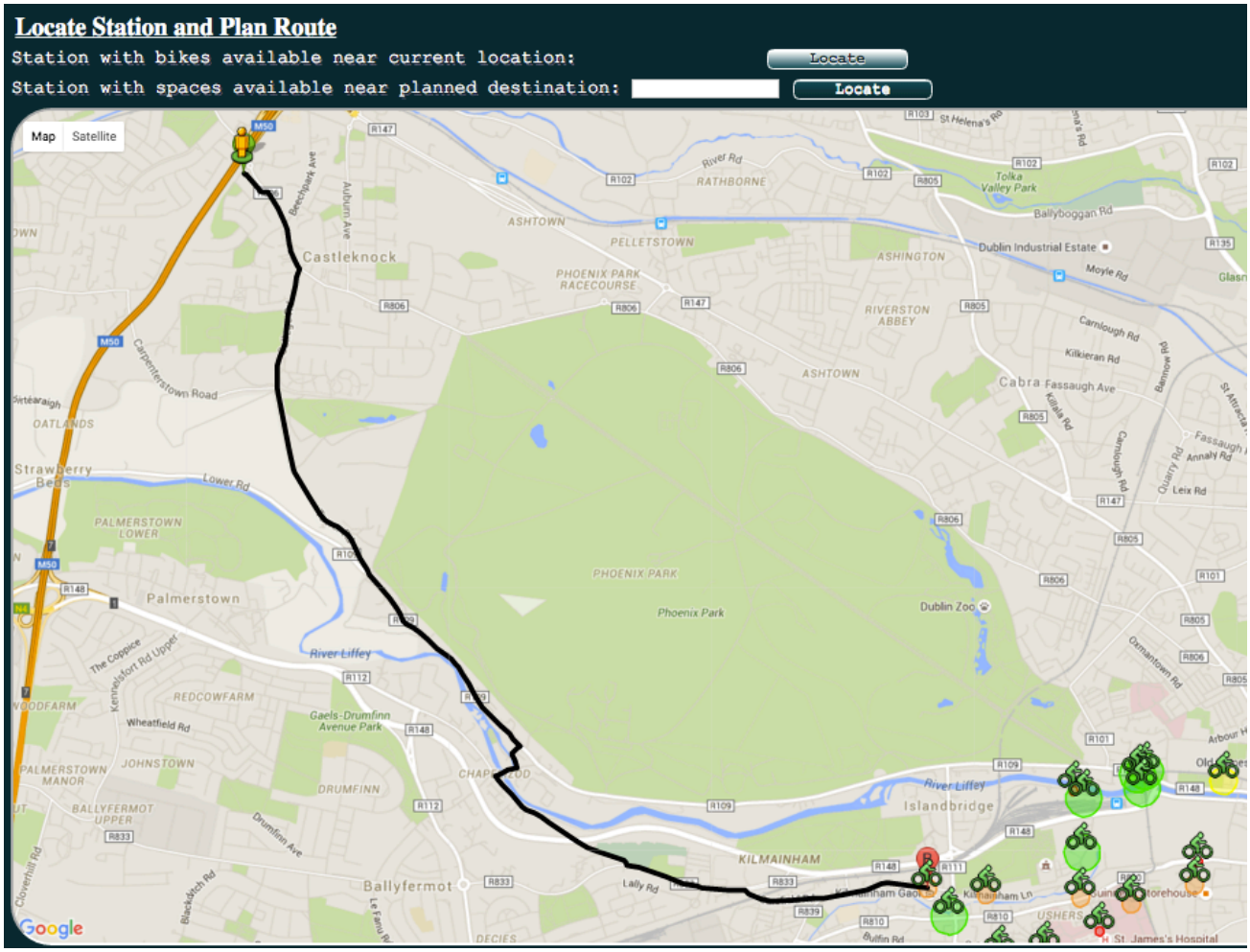
2. Displaying historical chart when a day of the week is chosen for a station



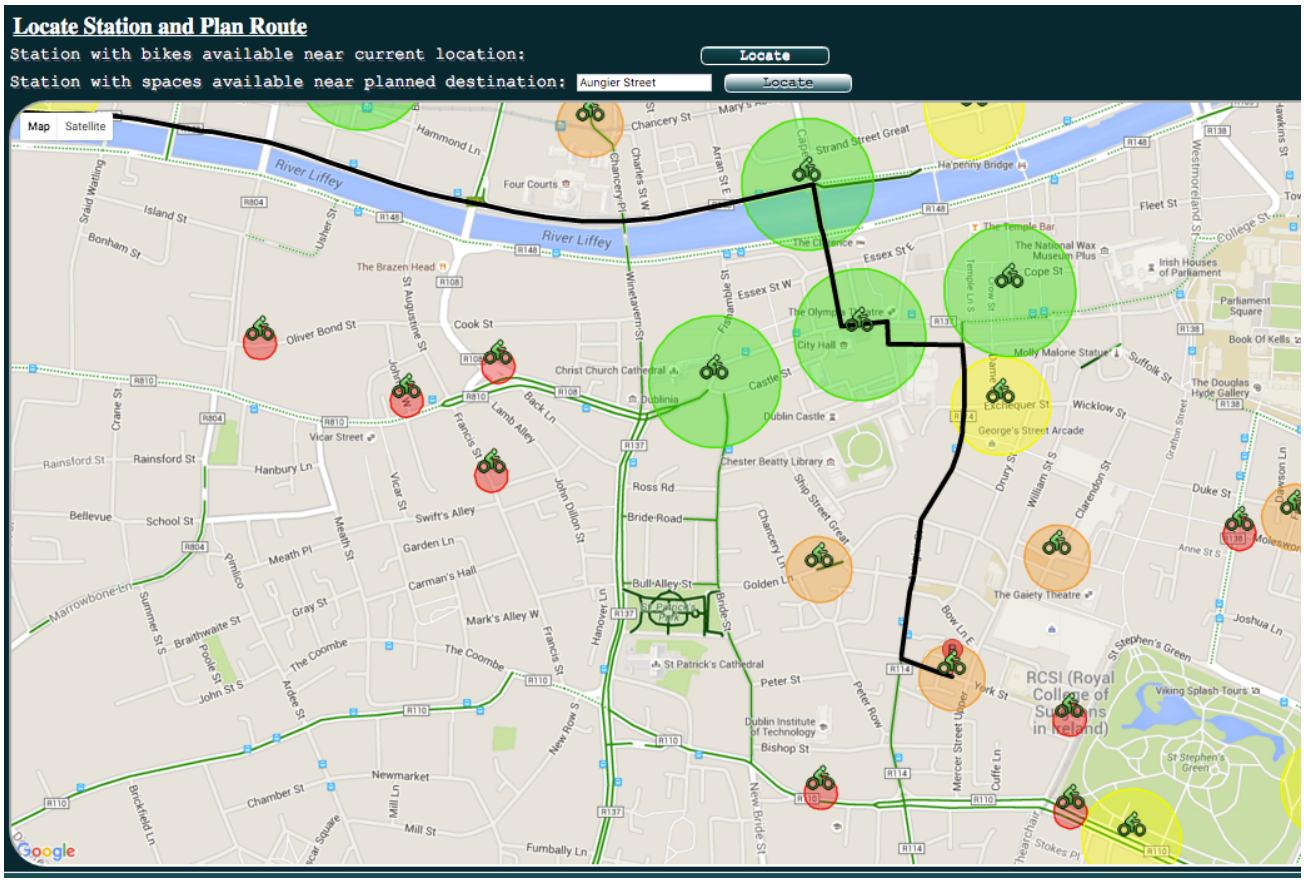
3. Historical chart displays the number of bikes available when hovering mouse over time series.



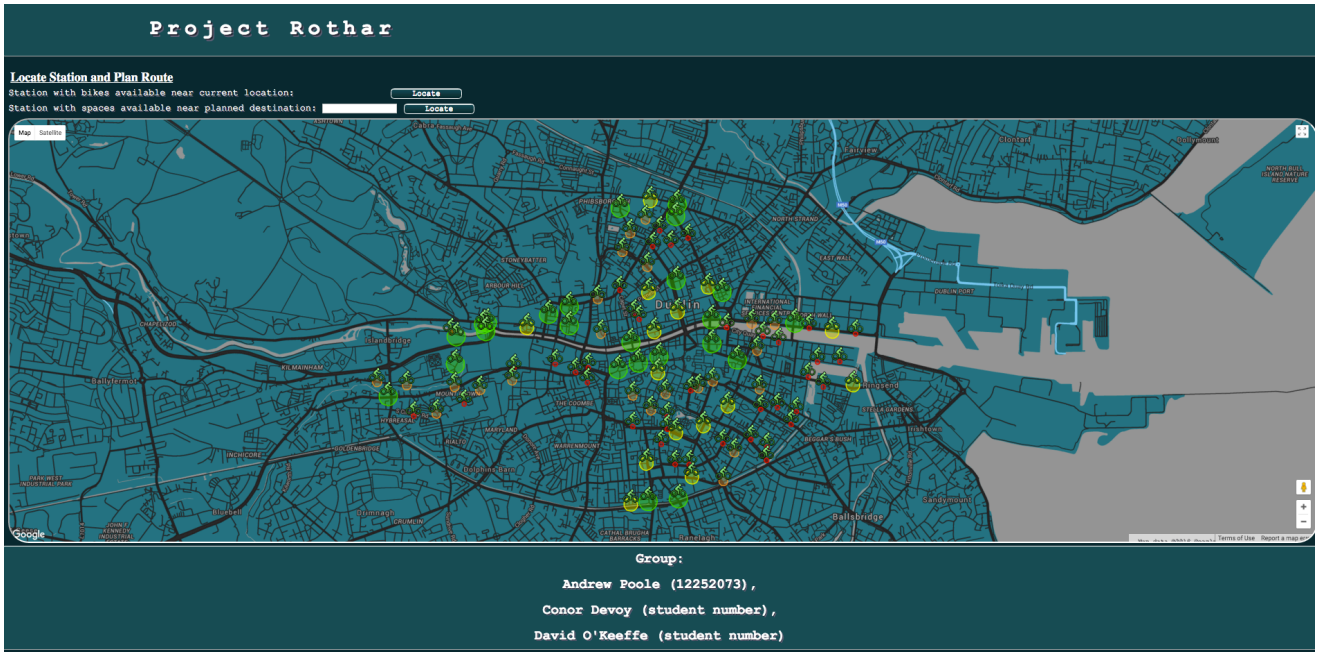
4. Locating nearest station with available bikes to the user and displaying route to station



5. Locating nearest station with available spaces to the destination entered and displaying route to station



6. Google map styling



Project Rothar

Station with spaces available near planned destination: Capel Street Locate

Problem

An issue encountered was having each subsequent chart being appended to the content div. We wanted only a single chart to be displayed at a time i.e. when the user selected a second chart to display, the initial chart would be removed and the second would then be displayed.

Resolution:
This is issue was resolved by doing some further research into how plotly.js worked. Instead of having the chart being immediately appended to our content div, the div holding the chart would be deleted and then a new chart div would be appended to the content div.

Lesson:
This was a similar issue to the last sprint i.e. a lack of experience with tools we were unfamiliar with. However, this time the issue was resolved in the current sprint.

Sprint 3 Review

Andy set up accessing the database via Flask in the following way. First, he wrote a function in the application script that would create a connection to the database. After connecting to the database, a query executes using variables passed to it from the url request handler. The station id and the day of the week is passed to the query from the request handler.

Request Handler:
localhost:5000/historical_plot/ <int:station_number>/<plot_day>

Depending on the station and day selected, the above url will return the queried information from the database. If the user clicks Monday in the station's infowindow, the station id and the day of the week values are used as parameters in the graphChart function. In this function, a jQuery getJSON request is called using these parameters to build a URL to request from.

Once the requested JSON is retrieved, it is iterated over. For each iteration, the time and the available bikes are pushed two separate arrays. These arrays are used to build time series chart's x and y axis.

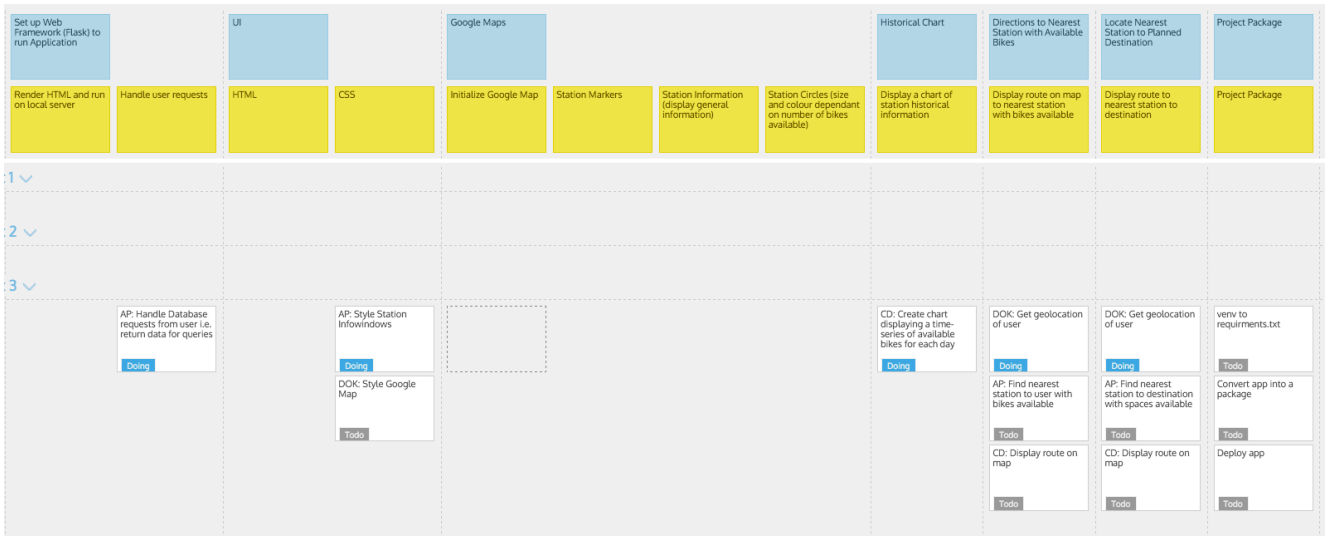
For the chart, we used plotly.js. Conor found that it was a relatively easy process to create the chart.

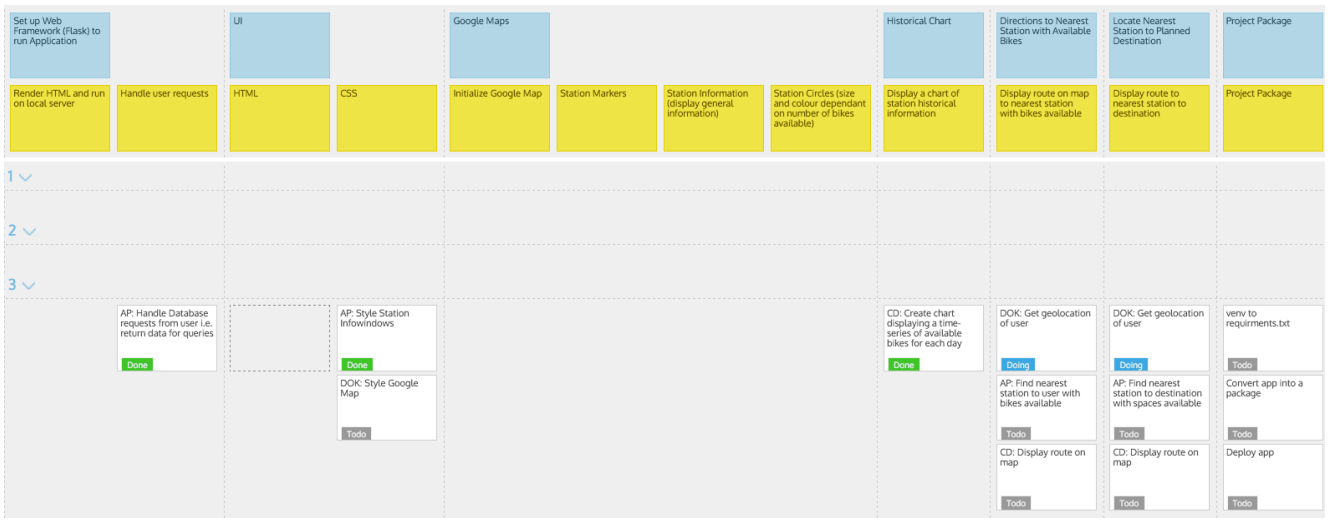
For the two features where we would find the nearest station to the user / destination entered with available bikes / spaces, we created a function to handle both features because both features had many tasks in common. For example both features required the following inputs:

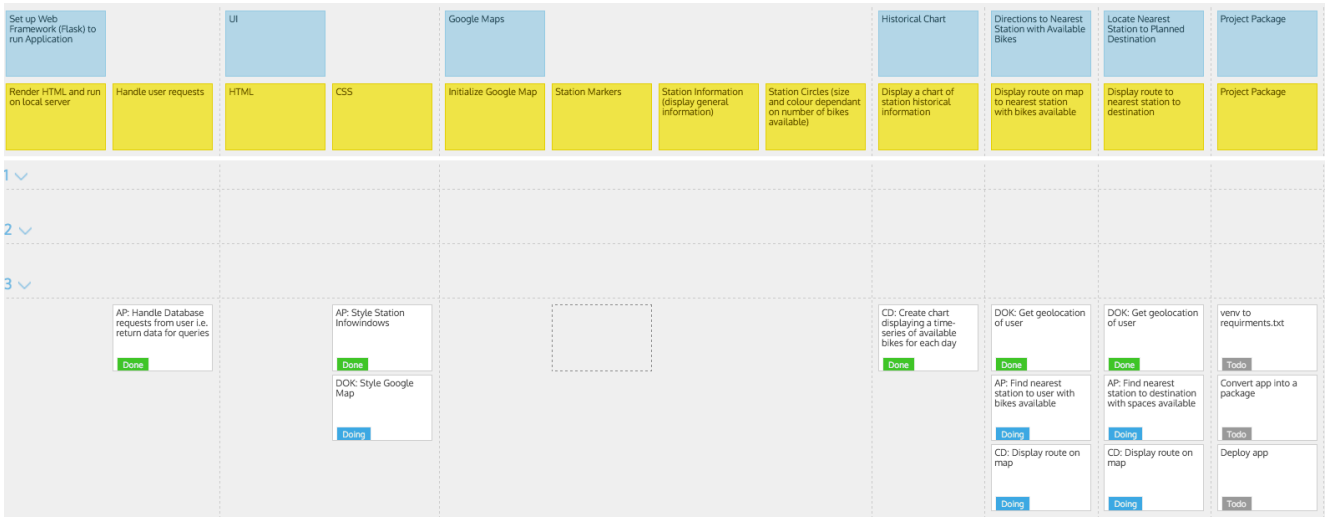
1. An array of stations to iterate through. Both of the following arrays are populated on page load when the JSON is retrieved from the API request:
    a. Available bikes array
    b. Available spaces array
2. User location – latitude and longitude. This is used to render a route from the user's location to the station selected.
3. Location to use when finding the closest station:
    a. For the feature that finds the closest station with available bikes, the user location will be used.
    b. For the feature that finds the closest station with available spaces, the inputed destination's latitude and longitude is found and used.
4. The travel method. When finding the closest station with avalaible bikes we made the assumption that the user is walking and is looking for a bike. When finding a station with avalaible spaces, we made the assumption the user currently has a bicycle and is looking to return it to the nearest station to their destination with available spaces.
5. The last parameter is the map instance being used to render the route onto.

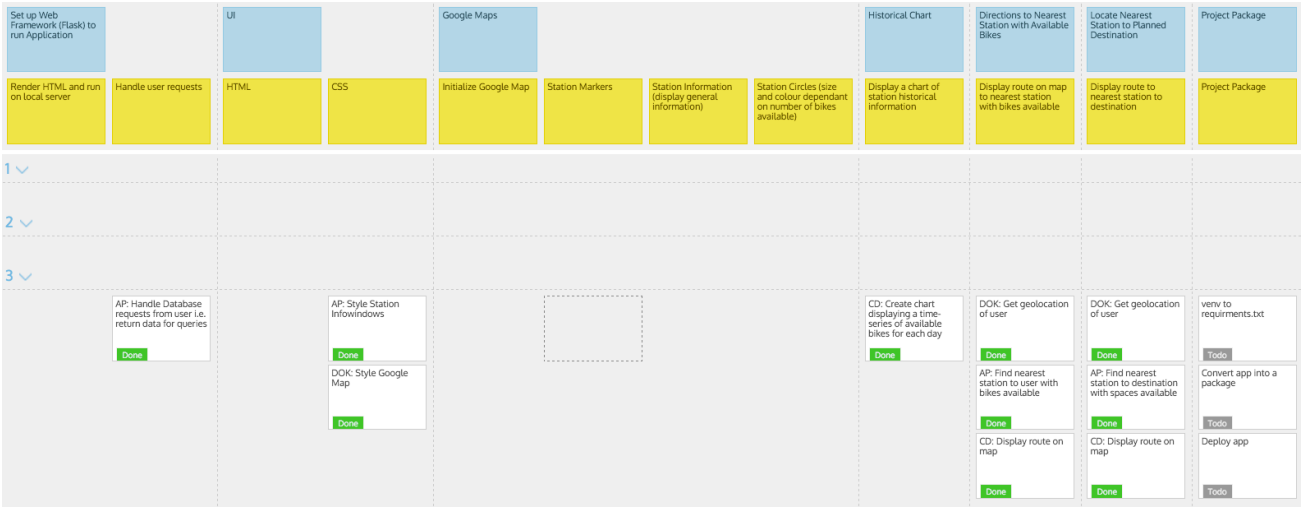
Other tasks completed during this sprint were styling of infowindows and the map. These can be seen above in the application progress images.

Sprint 3 Backlog Progress








Set up Web Framework (Flask) to run Application
UI
Google Maps
Historical Chart
Directions to Nearest Station with Available Bikes
Locate Nearest Station to Planned Destination
Project Package
Render HTML and run on local server
Handle user requests
HTML
CSS
Initialize Google Map
Station Markers
Station Information (display general information)
Station Circles (size and colour dependant on number of bikes available)
Display a chart of station historical information
Display route on map to nearest station with bikes available
Display route to nearest station to destination
Project Package
1
2
3
AP: Handle Database requests from user i.e. return data for queries
Done
AP: Style Station Infowindows
Done
DOK: Style Google Map
Done
CD: Create chart displaying a time-series of available bikes for each day
Done
DOK: Get geolocation of user
Done
AP: Find nearest station to user with bikes available
Done
CD: Display route on map
Done
DOK: Get geolocation of user
Done
AP: Find nearest station to destination with spaces available
Done
CD: Display route on map
Done
venv to requirments.txt
Todo
Convert app into a package
Todo
Deploy app
Todo

Meetings Minutes

Meeting Minutes – Sprint 1 Set-up

Location: In Practical

Date: Monday 29th February

Time: 13:00-14:00

Attendance

Conor Devoy

David O Keeffe

Andrew Poole

Agenda Items

1. Breakdown individual project requirements
2. Sketch preliminary schedule for completion of project
3. Assign tasks to be completed for Sprint 1
4. Assign tasks to group members
5. Schedule next meeting

Action Items

1. Establish group repository
2. Establish Database
3. Scraper code
4. Server set-up
5. Schedule meeting with TA to discuss this meeting
6. Next group meeting after Easter break

Other Notes

1. Set up meeting with TA to proof our first meeting.

Meeting Minutes – Sprint 1 Review

Location: UCD Physics Building

Date: Friday 11th March

Time: 11:30 - 11:45

Attendance

Conor Devoy, David O Keeffe, Andrew Poole

Agenda Items

1. Present to rest of group on Sprint 1 work done
2. Individual Problems?
3. Constructive Criticism?
4. Schedule Sprint 2 set-up meeting

Action Items

1. Reconvene for Sprint 2 set-up meeting, in second week of Easter Break.

Meeting Minutes – Sprint 2 Setup

Location: UCD Physics Building

Date: Monday 21th March

Time: 13:00 – 14:00

Attendance

Conor Devoy

David O Keeffe

Andrew Poole

Agenda Items

1. Compare work done to product timeline
2. Breakdown and analyse potential pitfalls for each task
3. Assign tasks
4. Schedule next meeting

Action Items

1. Set up initial webpage
    a. HTML/CSS
2. Initialise map on webpage
    a. . Javascript
3. Render HTML
4. Dynamic Map Markers
5. Next meeting set for end of next week dependent on tasks completed.

Other Notes

Meeting Minutes – Sprint 2 Review

Location: In Practical

Date: Friday 1th April

Time: 11:30 - 11:45

Attendance

Conor Devoy

David O Keeffe

Andrew Poole

Agenda Items

1. Present to rest of group on Sprint 2 work done
2. Individual Problems?
3. Constructive Criticism?
4. Schedule Sprint 3 set-up meeting

Action Items

1. Sprint 2 was a very collaborative effort so no presentation needed.
2. Individual Problems
    a. Conor talked about difficulty in rendering the HTML through flask. Instead of trying to pass in the queried information from the database on a user click it is now done through the Javascript script dynamically instead.
3. Constructive Criticism
    a. Group reminder that we needed to stick to given deadlines as much as was possible to avoid slowing down overall progress.
4. Reconvene for Sprint 3 set-up meeting.

Other Notes

Meeting Minutes – Sprint 3 Setup

Location: UCD Physics Building

Date: Monday 4th April

Time: 1:00 – 12:30

Attendance

Conor Devoy

David O Keeffe

Andrew Poole

Agenda Items

1. Compare work done to product timeline
2. Decide on essential tasks vs optional task for final Sprint
3. Breakdown and analyse potential pitfalls for each essential task
4. Assign tasks
5. Schedule next meeting

Action Items

1. Style station Info windows
2. Time Series chart for stations
3. Geolocation of user
4. Display route on map
5. Nearest Station to destination
6. Next meeting set for end of next week dependent on tasks completed.

Other Notes

Meeting Minutes – Sprint 3 Review

Location: UCD Physics Building

Date: Friday 15th April

Time: 11:30 – 12:00

Attendance

Conor Devoy

David O Keeffe

Andrew Poole

Agenda Items

Action Items

1. Pats on the back all round for a job well done

Other Notes