Dublin Bikes Report

Group 7

15th of April 2022

Address of web app: http://34.252.137.171:5000/

Gus Boothman, Bryan Agar, Michael Scallon

Table of Contents

Introduction and Overview

This report provides a detailed review of our team's development of a DublinBikes web application. The architecture, design, process, retrospective, and future work are all explored. The contributions of each member can also be found at the end of the report.

The project specification was to create a fully interactive website that displayed real-time data that helped the end-user find information on a Dublin bikes location, discover Dublin bike locations and display useful information such as predictive availability to the user. The ultimate aim of this project was to create a web app for Dublin Bikes. In this report objectives, unique features, design, architecture, analytics, software engineering process, retrospective and future work are all explored.

This project consisted of various components and technologies, these include backend and frontend development, UI & UX and other elements which will be discussed later in the report.

Objectives

Some of the key requirements for the success of this application included features such as

- Interactive map displaying real-time available bikes data across Dublin
- Ability to click on the location of a bike station and reveal information at the station
- Current availability of bikes at a given location
- Current availability of bikes stands at given location
- Current weather at the location
- Station address
- Key information about station location(Banking available etc)
- Display the closest station to the user
- Display the user's current location

- Use industry standards for project management(Scrum and sprints)

The final product that is displayed to the end-user is a fully interactive Dublin bikes app that allows a user to find information about a certain location and scan through the entire Dublin bikes system with ease.

Target Users

Dublin has a relatively poor public transport system when compared to other major European cities. This coupled with the threat of climate change and pandemics, cycling around the city has never been more encouraged. Dublin Bikes is a bike renting scheme around since 2009 to facilitate commuters in the city centre. Thus, our target audience is those that use Dublin Bikes. As Dublin is a capital city this audience captures a diverse set of people, from college students to workers, and tourists. All of which would be using the app daily.

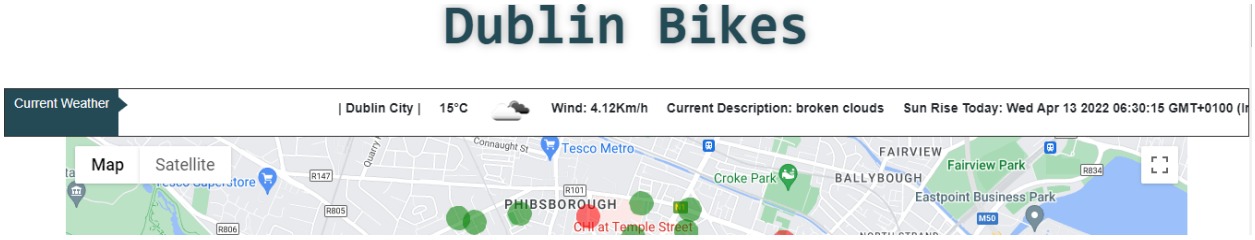
The Structure of Our App

When a user opens the web application, they are first greeted by the splash page while the content of the application loads. They are then on the main page where the interactive map is with all the features. Below the map, there is a section where the user can explore station information and statistics in more detail. At the bottom of the page is a journey planning section whereby the user can choose a day, a time and a station and it will predict the number of available bikes at that particular station. The user can then choose the station where they will be returning the bike, the number of predicted free bike stands will be predicted in this instance. There is a separate about page where the user can find out more about the application.

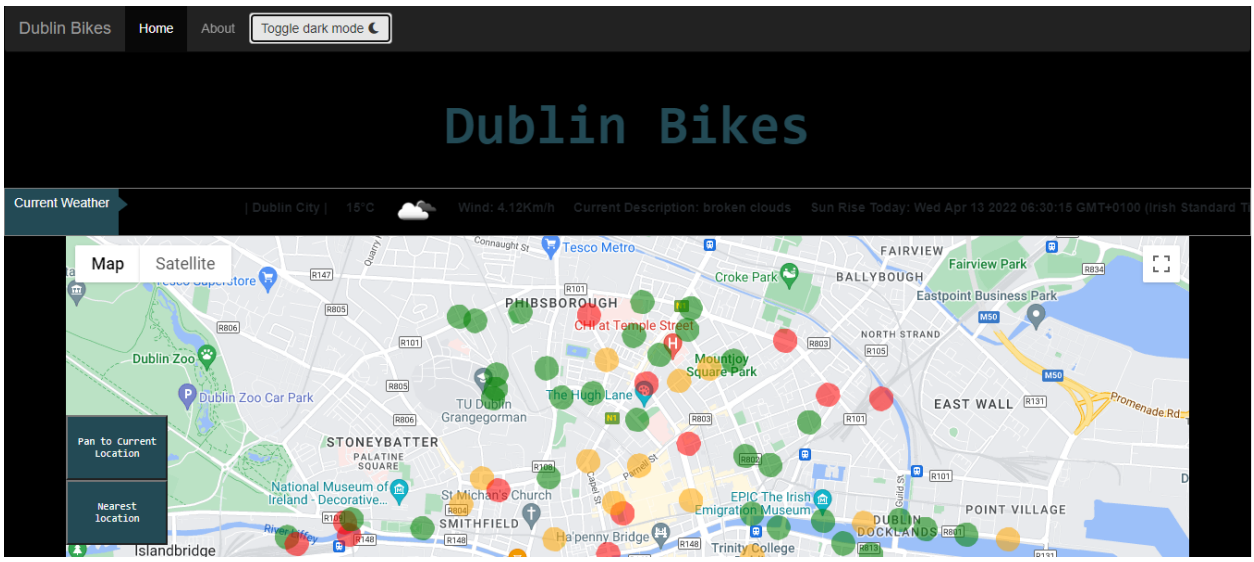
Unique Selling Points of Our Application

Our App has a number of unique features not found on the current Dublin Bikes app in operation:

- **Weather Data-** Weather is obviously a hugely important factor for a cyclist to consider. Our up to date news style ticker graphic displays all the important information a cyclist would consider before making a journey.

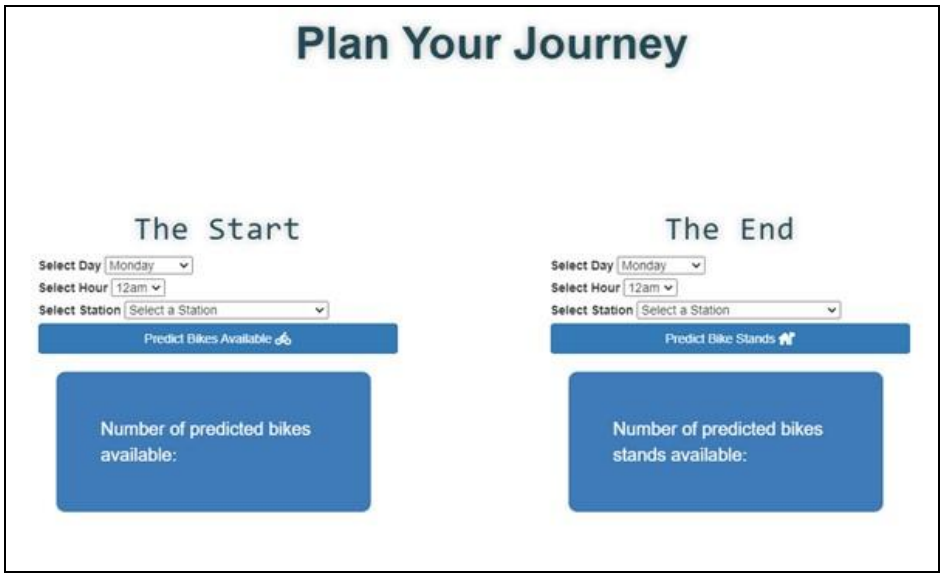


- **Dark Mode**- This is for cyclists tha may be using the app at night. Toggling dark mode may reduce eye strain and make it easier to use.

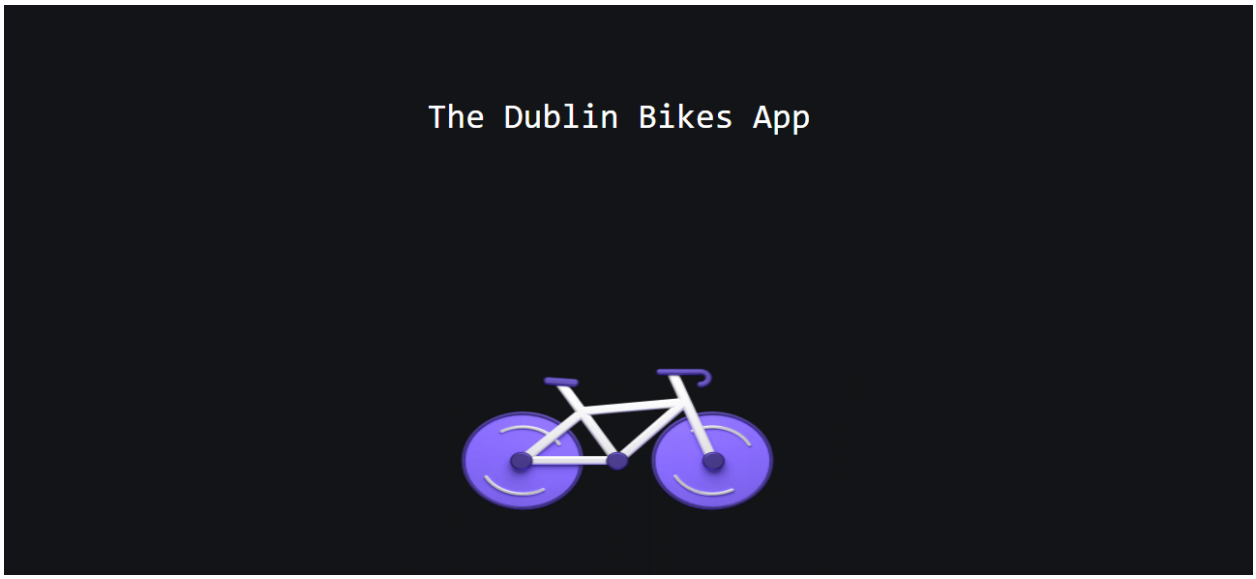


- **Nearest Station-** as you can see with the buttons shown in the demonstration of dark mode below. The user can view their current location and then select the button to find the nearest station to them.

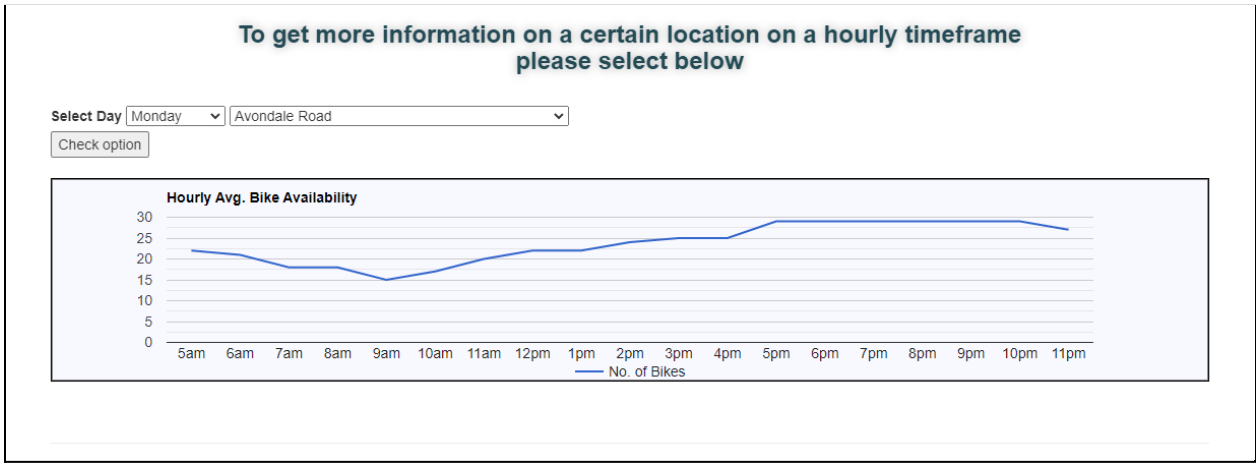
- **Plan Your Journey Section-** Users can select a time and station they will start their journey at and get a prediction of how many bikes are available. They can also select when they will finish their journey and get a prediction of free bike stands. This will be particularly useful for those users that might have an appointment at a certain time and want to plan ahead. See the image below of the section.



- **Fresh User Experience Design and User Interface Design** - Our aim was to create an application that was sleek and intuitive to use. Commuters are often in a rush. They do not want to have to deal with a slow and cluttered app. For this, a minimal number of pages were implemented to keep navigation simple. Coloured Markers to indicate occupancy levels were implemented so the user can instantly see where the most bikes are without having to click anything. This is particularly useful if you are with a group and want to quickly see where your best chance of getting several bikes quickly is located. Furthermore, the app is responsive to different screen sizes. A Splash page is displayed while the content is loading, this increases the user experience as the user knows that the app is loading and not simply non-functioning. See the image below of the splash screen.



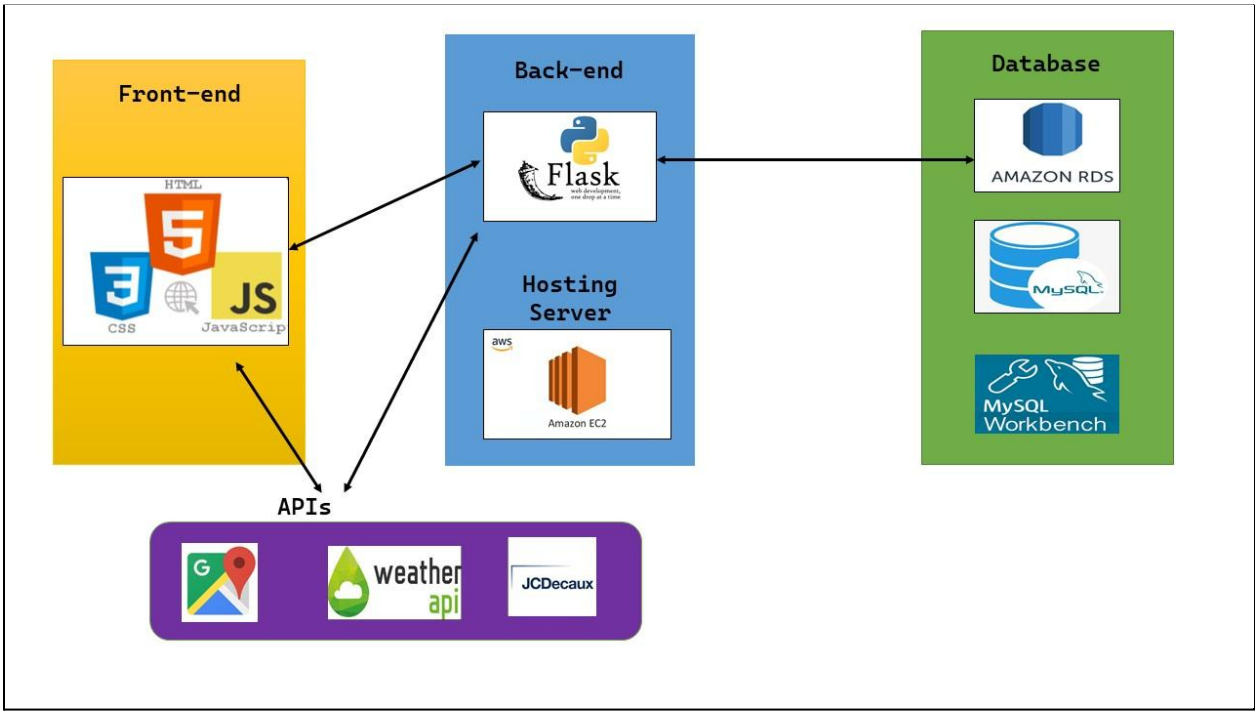
- **Daily Usage Statistics-** The user can view statistics on average daily bike availability and hourly bike availability per station. This will be useful for cyclists to determine how dependable a certain station is to them.



The Architecture of the Application

The architecture of the app refers to both the boundaries and the communication between different parts of the app. As well as the responsibility of each part of the app. It ensures that the application works correctly and efficiently.

It was very important as a group that we had the architecture completely clear from the beginning of development. As any sort of discrepancy could lead to issues in all facets of the project but in particular time waste. See the image below for an illustration of the architecture of our app.

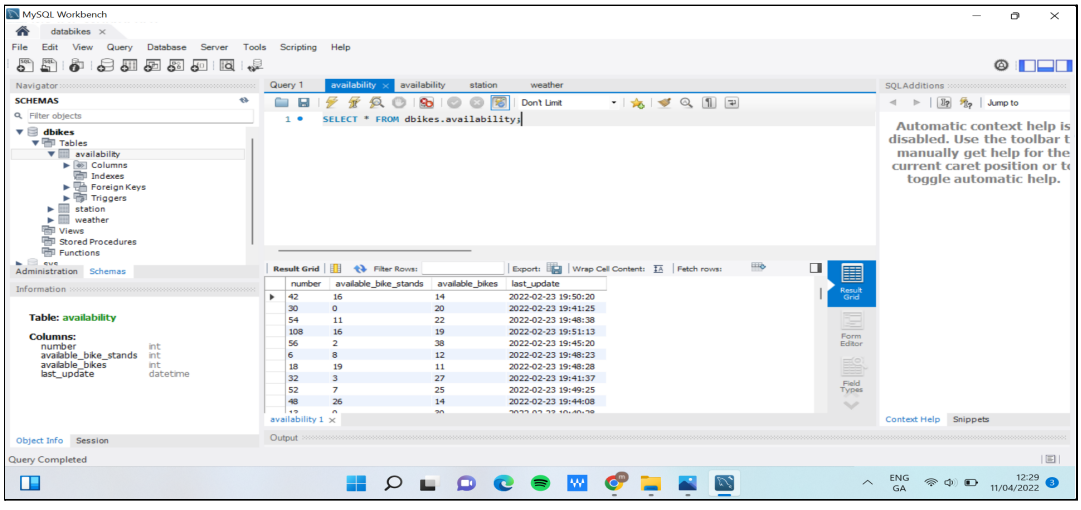


The Database

A vital element of the architecture of the app is the database. Two elements of Amazon Web Servers (AWS) were implemented in order to create a functional database. An Ec2 was used to create instances or virtual machines to run code and an RDS was used to store all of the data collected.

The benefit of EC2 instance was that it allowed for any code that was running to be active at all times and meant that the running processes were not reliant on any physical machine which would have been impractical to have running at all times. As we were conscious of the cost we opted for all free tier elements on Ec2 and made sure to carefully monitor the Aws cost explorer to view when our free credits were close to expiring.

We developed two python files to scrape information from JCDeacux's API and Open Weather's API and create the tables in our database. Aws Rds was used to store this information meaning all members of the group could access the information at any time. Again in a similar fashion to the Ec2 instances, an Rds was also monitored on the cost explorer to ensure no major expenses were accrued. We had two python files that would connect to the API's and populate our database. The first of our database creation and web scraper files were titled scraper.py. This file served several functions. Firstly it connected to a JCDecaux API which was our source of the information for the physical bikes. The bike data from JCDecaux was updated every 5 minutes. The weather data from Open Weather was updated every 30 minutes. MySQL Workbench was used to visualise our data and write SQL commands, see *figure below.*



The Back-End and Front End

Flask is the framework we used to handle the backend. SQLAlchemy was used to connect to the database and collect the data. Specific SQL commands were written to only get the necessary data from the database. This data can then get passed to the frontend in order to be displayed. In order to get the data to the front end, the Fetch API in Javascript was used. Once the data was fetched we could then manipulate it and display it where necessary. Where the user was submitting a form in order to make a prediction AJAX was utilised. This was necessary as AJAX allows us to pass data from the user to the backend into our machine learning model and send back the result without reloading the entire page. In order to create the machine learning models, a Jupyter notebook was used. Python modules Pandas and Numpy were used to clean and combine data. Scikit-learn was used to make the necessary machine learning algorithms. Pickle was used to convert our model into a file that we can open on the Flask server and take input and produce predictions.

The front-end of the website was crucial to the success of the project as discussed within the design section below. While vanilla Javascript was used to make the web application dynamic and connect to the Flask server. HTML, CSS and Bootstrap were utilised to position, display, and style the data. Bootstrap was particularly useful in making our app responsive to different screen sizes.

Functionality and Issues

As a whole the app works well and is intuitive to use. However, there are issues:

- The map was quite slow to load. This was due to originally having to fetch all the data from the database which was ever-growing. It was made faster by changing to a table with less data as it was only displaying the most recent information it did not need historical data.
- The google charts are also quite slow to load. As mentioned in the bugs, it was not an issue we had the chance to fix. A more optimised SQL query would probably help the issue.
- There was an issue with the location features when the web app was hosted. The Geolocation API requires HTTPS as opposed to just HTTP. In order to achieve this, we

would have had to apply for an SSL certificate in order to authenticate our web app and enable an encrypted connection. Unfortunately, we did not have enough time to apply for SSL certification. As a solution, we decided to hardcode the longitude and latitude in order to show how the feature works.

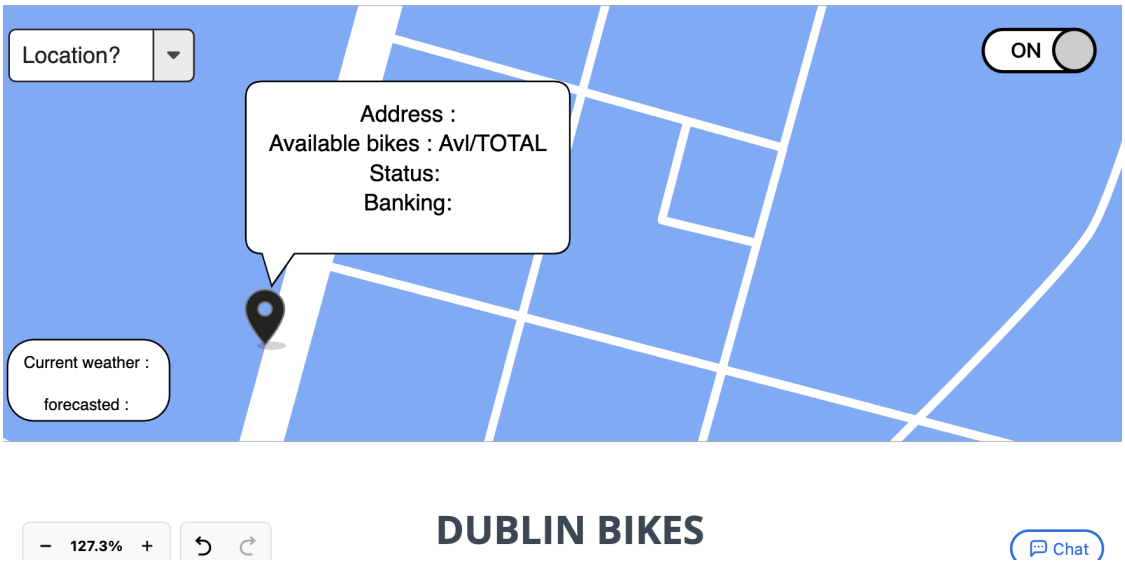
Design

The overall design of the application was a key area that we decided to focus on for this project. We felt that a well-planned design with industry UX and UI tools used would create a much better experience for the end-user, which in return would increase the probability of the user returning to the site whenever they needed Dublin bikes data again.

A simple but uniform layout was decided from the offset as we felt that adding distracting and unnecessary design futures such as side columns and offsetting plugins would take away from the application on the viewer's side. We wanted the app to be as practical as possible for its diverse set of users. For example, the map is what we believe users will use the most. Thus, we displayed it front and centre with minimal clutter. All other extra features were placed away from the map.

Before any wireframe, line of code or UX diagram was created, design research was carried out on various other data platforms to see what elements we wanted to recreate, how they displayed their data and finally how the use of colour was used throughout the website.

Some interesting and notable finds from our research included websites such as https://www.storytellingwithdata.com/ and https://infogram.com/blog/map-examples-from-the-web/ were the main inspiration that we tried to glean.



User interaction

As the goal of this project was aimed at the general public and presumes no technical ability, it was vital that the flow of the application and the user's actions within the app could be guided clearly without having any technical issues for the user. We designed the application in such a way that it was ‘foolproof’ and that everything the user wanted to do was in a different section so they wouldn't get confused or overwhelmed by the data/info being displayed.

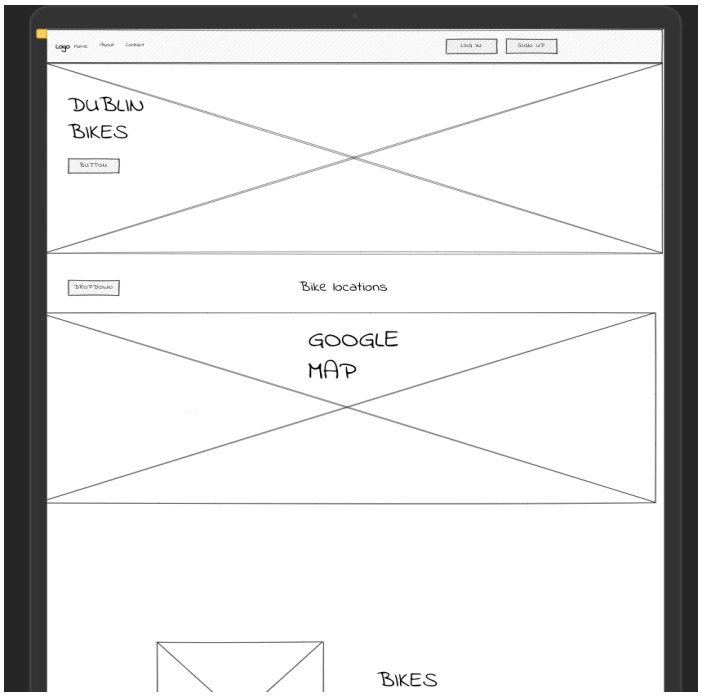
The user flow of the user can be seen further down in the report and displays what happens with each user interaction with the website

Wireframing

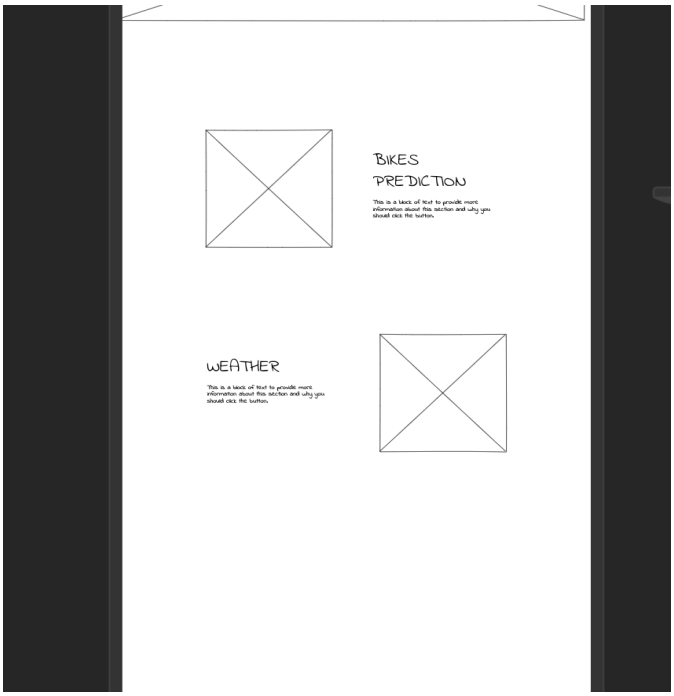
Having a clear and complete idea of what we wanted the website to actually look like with the naked eye was something we thought about right from the beginning of the entire process. We knew that we wanted something simple to operate for the user, have detailed and clear stats and finally something that would clearly show the user the bike's station in question.

Wireframing was the best tool that allowed us to get a solid picture of what we wanted to create. The website ww.app.uizard.io was used to create the wireframes. Screenshots below show the final agreed-upon wireframe and our starting point for coding the UI for the website.

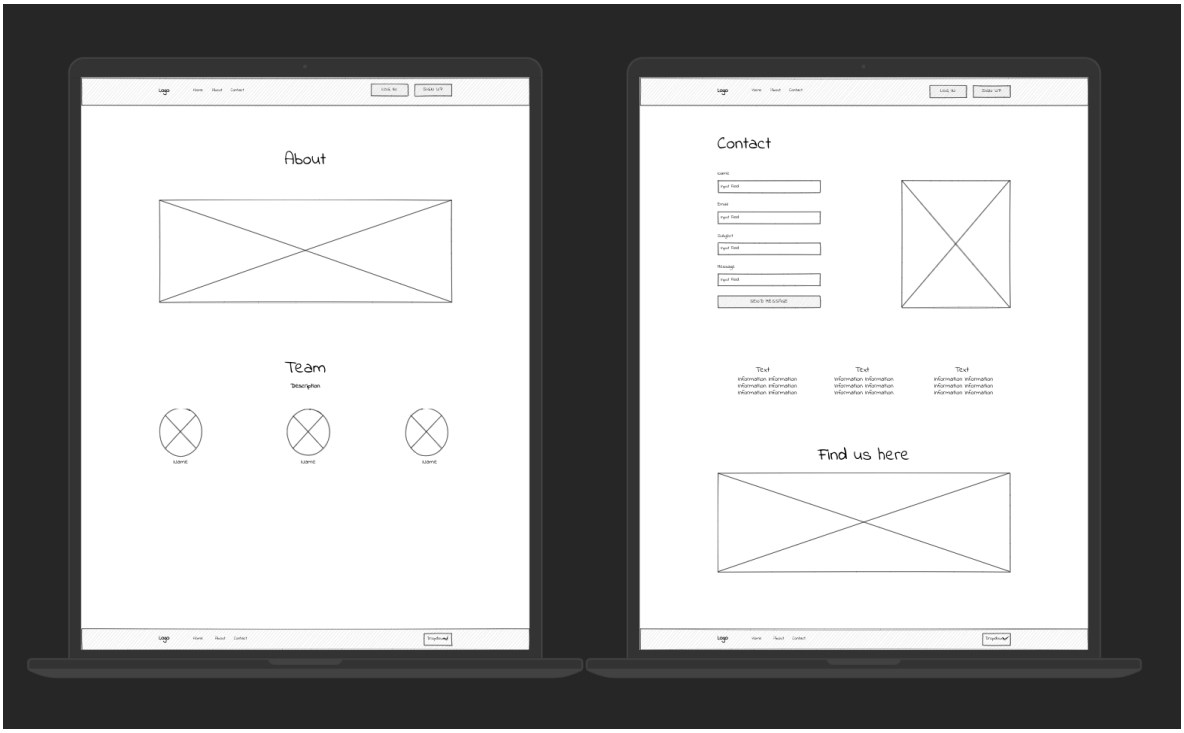
Homepage wireframe



Continuation of Homepage



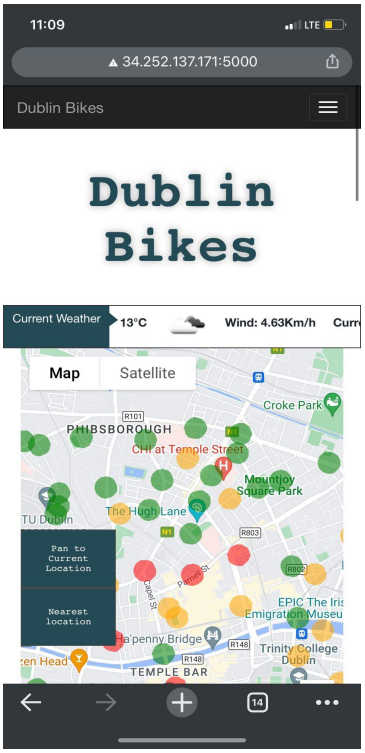
About Page Wireframe

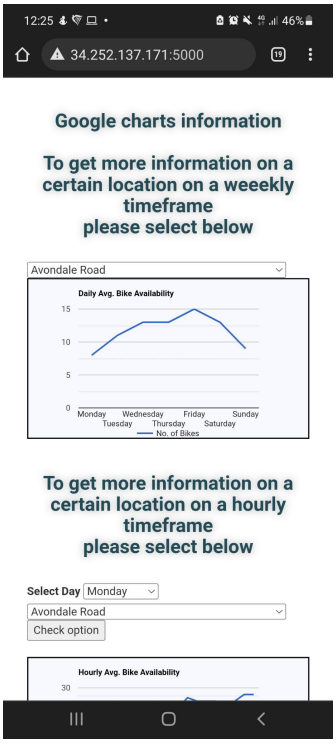


Bootstrap

In order to have a more uniform design and structured website, the bootstrap library was implemented to achieve this aim. Another benefit of using bootstrap within the web app is it easily allows the website to be responsive, mobile and different screen size friendly. This attribute of bootstrap is a core reason why many websites are built with bootstrap.

The main use-case of bootstrap within our application was for the structuring of the content. Bootstrap uses dynamic columns to allow for easier placement of data within the website. Furthermore, Bootstrap was very beneficial for making the app responsive, see images of our app on an iPhone and a Samsung phone below.





Design process/planning

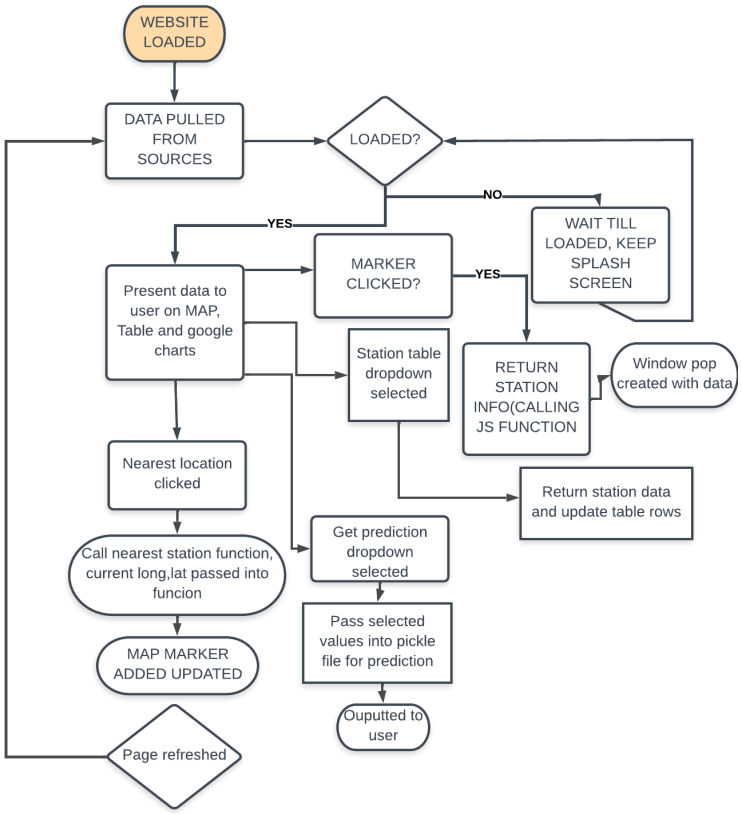
We approached the design building of the website in various stages throughout the project just like the sprint model. We first decided to design the core elements of the website, these included features such as a basic map, flash interaction and other core features, once the core features were implemented we moved on to more styling and designing additional features such as the information tables for example. We did not want to be too restrictive in our design process, we let the design develop as the project developed. While also keeping the core idea of simplicity to keep us on track.

Handling the design in different stages allowed us not to get too far ahead of the process and also gave us reassurance that we were making solid progress throughout the project as the design of the website felt uniform in each stage/sprint.

User Flow of Our App

Below shows the various ways a user can interact with the website and how the interaction affects/updates the page and data being presented.

Having this user flow allowed us to test for any issues that the user may encounter, whether it was an invalid link or an invalid http request. It also helped us as a team understand features that another team member may have implemented and how we could approach the design of each feature.



Data Analytics

In this section the way in which we incorporated data analytics and machine learning into our web application. The aim was to allow the user to predict the number of available bikes and the number of available bike stands at some point in the future. This premise as discussed previously was implemented as a 'Journey Planner' section.

```
In [12]: df_avail
```

Out[12]:

	number	available_bike_stands	available_bikes	last_update	day	hour	just_date
0	42	16	14	2022-02-23 19:50:20	2	19	2022-02-23
1	30	0	20	2022-02-23 19:41:25	2	19	2022-02-23
2	54	11	22	2022-02-23 19:48:38	2	19	2022-02-23
3	108	16	19	2022-02-23 19:51:13	2	19	2022-02-23
4	56	2	38	2022-02-23 19:45:20	2	19	2022-02-23
...	...	...	...	...	...	...	...
1199637	39	9	11	2022-04-10 15:49:38	6	15	2022-04-10
1199638	83	19	21	2022-04-10 15:55:01	6	15	2022-04-10
1199639	92	37	3	2022-04-10 15:47:09	6	15	2022-04-10
1199640	21	27	2	2022-04-10 15:47:57	6	15	2022-04-10

The data below shows the data we used for machine learning. The first step was to evaluate the data and clean the data using Jupyter notebook as well as modules Pandas and Numpy. For example, the features were changed to the appropriate type, see *figure data types.* There was also a test station that was included in the data set, that was removed, *see figure removing station.*

```
df_avail.dtypes
```

```
number                          int64
available_bike_stands           int64
available_bikes                 int64
last_update            datetime64[ns]
day                             int64
hour                            int64
just_date                      object
dtype: object
```

Removing illogical data from dataset

```
df_avail.loc[df_avail['number']==507]
```

	number	available_bike_stands	available_bikes	last_update	day	hour	just_date
1032891	507	0	1	2022-04-05 09:56:28	1	9	2022-04-05
1033002	507	0	1	2022-04-05 10:06:34	1	10	2022-04-05
1033113	507	0	1	2022-04-05 10:06:34	1	10	2022-04-05
1033224	507	0	1	2022-04-05 10:16:40	1	10	2022-04-05
1033335	507	0	1	2022-04-05 10:16:40	1	10	2022-04-05
...	...	...	...	...	...	...	...

Machine Learning Models

As all group members had very little prior experience with machine learning and data analytics. Thus, the aim was to keep things as simple as possible. Firstly, all the necessary data was read into a jupyter notebook. A linear regression model was implemented at first. The hour, the day and the station number were the features that were used to train the model to predict the target features of available bikes and available bike stands. The data set was split as 70% for the training set and 30% for the testing data set. The model was then trained. The metrics were as follows;

R squared = 0.07740, Mean squared error = 105.455, Mean Absolute Error = 8.340

As is shown the MSE and MAE results are much larger than 0 and the R squared value is considerably less than 1. These results showed the linear regression model performed relatively poorly. This is due to a number of reasons. It seems as though one of the main reasons is that we are dealing with a data set that is non-linear. As there is no linear relationship between the target feature and the training features.

We decided next to try a Random Forest Model. The reason is that this type of model is better at dealing with non-linear data. The data was split in the same proportions as the linear regression. The evaluation metrics were as follows for predicting bike availability:

r2 score = 0.654, MSE = 39.50764780518432, MAE = 4.862

The evaluation metrics were as follows for predicting bike stands:

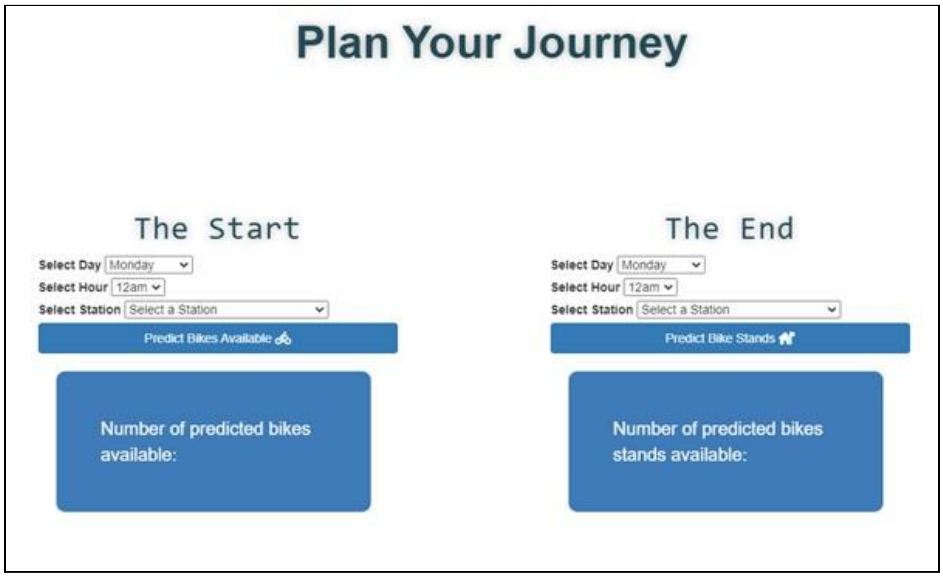
r2 score = 0.5421, MSE = 37.529, MAE = 4.784

All metrics performed considerably better in comparison to the linear regression model. Thus our models can be summarised as follows:

Available Bikes = randomForest(hour, day, bike station)

Available Bike Stands = randomForest(hour, day, bike station)

Integration of the Models with the App



As mentioned, we designed a 'Journey Planner' section for the predictions. See the *figure above.* The user can select their starting station, the time will be looking for a bike, and the day. They will be given a predicted number of available bikes based on the inputted information. 'The End' section is the same except the user selects the information based on where they will finish their journey and the prediction will be the number of available bike stands. The flow of data to make the get the outputted prediction is as follows:

- The user selects their information through the dropdowns (hour, day, station) all of which are required. The stations that are displayed are configured to correspond to their station number. The user then clicks the predict button.
- The data is sent through a POST request to the backend (flask) where the numbers are converted to a NumPy array and then passed into the necessary model for prediction. The output result is rounded to the nearest whole integer and is returned in JSON format to be displayed on the front end.
- Ajax was used to handle the requests. This allowed the web app to make asynchronous requests, meaning we did not have to reload the entire app in order to send information to

the backend. This was a much better user experience as it was much faster for the user to get their prediction.

Issues with Machine Learning

This is probably the part of the project we struggled with most as a group. We found it difficult to develop models that included multiple features. Originally, we wanted to include the weather data that we had collected as well as the forecast API that we had set up. Due to issues with combining data frames, the forecast API configuration and time pressures we decided it was best to focus on a more simplified model. Furthermore, seasonality would already be worked into the occupancy data anyway (i.e if it is raining more during a month this will be reflected in the occupancy levels).

Another issue we had was the size of the pickle files. This caused issues when trying to push them to Github as there the files exceeded the allowed file size. Our workaround was to zip to the pickle files and then push them to Github that way.

Process

One of the key learning objectives of this project was to develop the app using the Scrum methodology. This methodology provides a framework for developing and delivering projects in a team environment.

Scrum Process and Applications Used

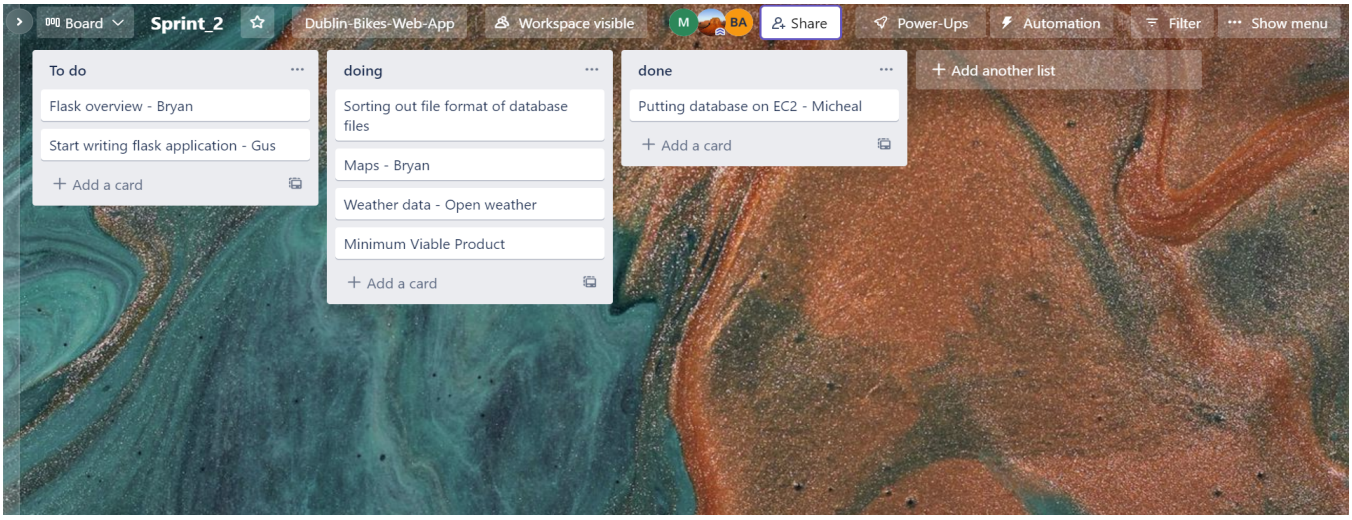
During the course of our project scrum and agile worked in tandem to provide an optimum workflow and allow for a better team atmosphere in an effort to provide the best possible application. Scrum refers to the structure of the workflow, while agile can be seen as a mindset where each member strives for continuous improvement.

There are five main principles to scrum. The first of which is commitment. We were each dependent on each other's work and a delay in one facet of the project may have led to other delays to other members' work as they may have been dependent on this for their tasks. One element we found that we struck well was balancing our expectations and work given to each member so that an achievable workload was decided upon for a given time period.

Courage, openness and respect were another three of the principles that we felt that we became much more confident with as time progressed. By this, we are speaking of the ability to question other decisions, provide constructive feedback and speak up when we felt that changes needed to be made to better the team. While originally we were cautious of both coming across as bossy or rude and of also hurting the feelings of other team members, we worked hard to create an open and honest work environment where anything that was said and received came from a place of wanting to improve rather than a place of criticism.

The final principle of scrum we felt that we became much more aware of was that of focus. As we progressed we became better and better at sticking to what was most important, while spending less time on smaller time-consuming aspects that were not necessary to our success. We became better at picking certain goals for our sprints and working on firstly getting a minimum working version of a feature before later refining them once we were confident we had the time to do so. These five principles we found very helpful in not just producing our product on time but also producing an optimal work atmosphere.

Our scrum process was based on four sprints. Three of these were two weeks in length and one was four weeks to include the midterm college break. The aim of each sprint was to allow for a certain number of tasks to be completed in the short term to aid the final product in the longer term. For each sprint, a member of the group was assigned as scrum master, and we took turns to participate. As the scrum master one was tasked with the additional job of organising when each stand up meeting would occur, deciding in collaboration with the other members the tasks which were hoped to be completed by the end of the sprint, to keep minutes of each stand-up and construct a sprint review and sprint backlog, while still maintaining their standard workload on the application. This job was important in ensuring that the structure of the scrum was maintained, while also ensuring that we remained at a desirable pace towards our final target.

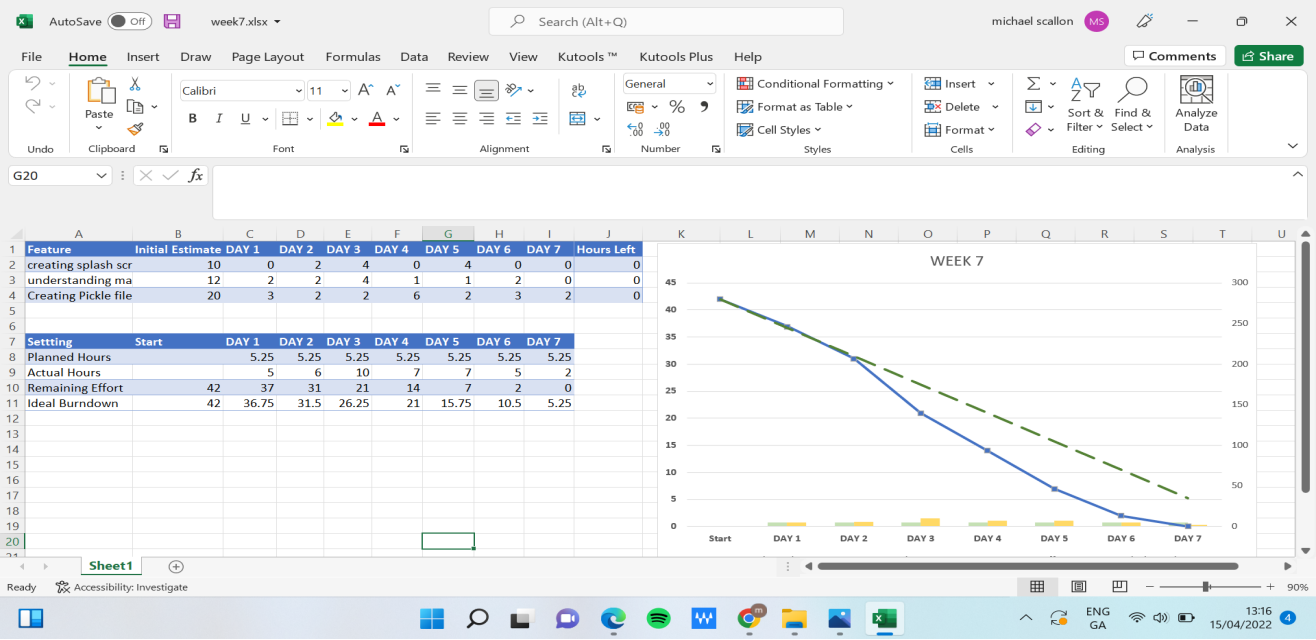


For the scrum process meetings with our product owner, we had weekly meetings at which we were required to be present. We met every Thursday morning with our product owner Daniel who would check up on our progress and tell us what he envisioned for us to have completed by the following week. We found him to be a pleasure to work with as alongside his reviews and expectations, he was incredibly helpful in the cases where we had questions regarding the project. What we appreciated the most however was his comforting attitude towards us as he always made us feel that our hard work was being appreciated and that we were on track to complete our project to a good standard while also being able to give feedback constructively

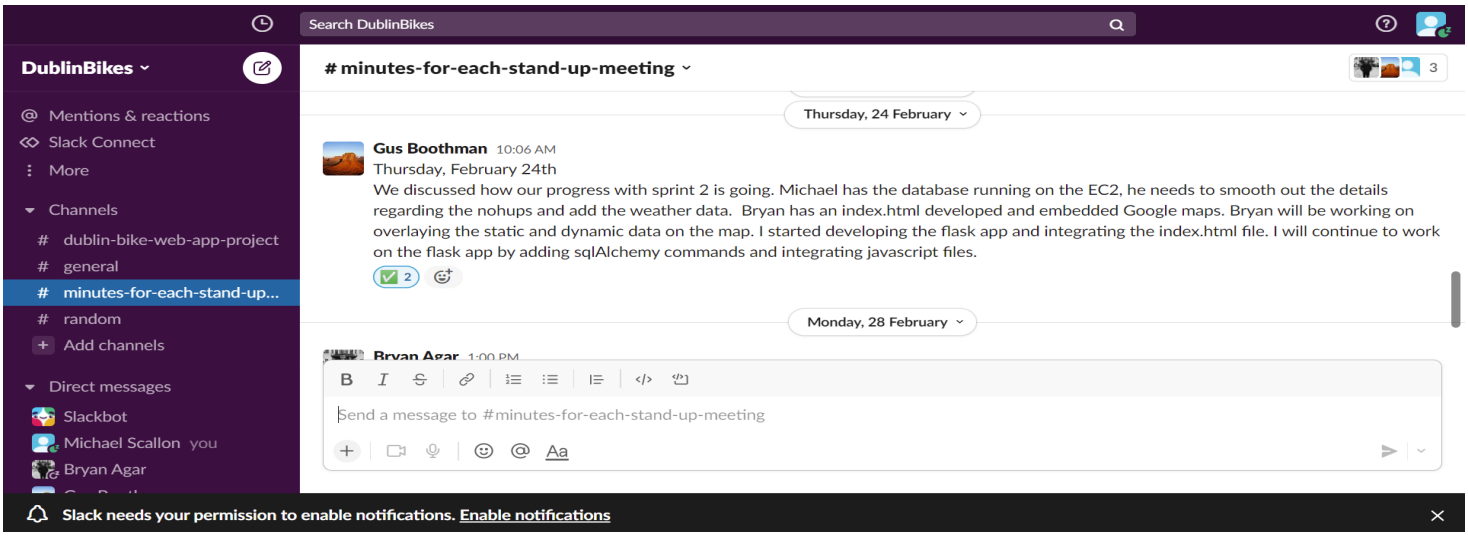
Stand up meetings were also a key and regular occurrence during our product development. We originally were not sure how many stand-ups to have per week. We originally thought we would do a stand-up every day. However, we quickly found out that 3 or 4 times a week gave us enough time to discuss developments and was flexible enough to fit in with other commitments. These were incredibly beneficial for a number of reasons. They allowed us to check in with each other to see what progress had been made to allow for changes to be made regarding expectations, should a member have been either ahead of schedule or behind due to unforeseen struggles with a certain task. They also meant that we were much more able to help each other through struggles, while also allowing for us to bounce ideas off one another as our envisaging of our final product evolved over time. They were also again very comforting in the sense that each member was reassured that while they were putting a great effort into the project, we could see that this effort was replicated by the other group members. We kept a minutes record of what had been spoken of to ensure we were also always moving forward in the correct direction.

We also created documentation in the form of sprint backlogs before each sprint and sprint reviews after each sprint. The backlog was a report on what we expected to achieve by the end of the sprint. This allowed us to remain focused on what we needed to get done by the end of the sprint and allowed us to plan out the sprint in advance. The sprint review was then a record of how the sprint went. It detailed what went well and what could have been done better in the next sprint as unforeseen events, delays and completing tasks early was a frequent event. The sprint review allowed for better planning of the following sprint backlog as it was a record of what had been done as opposed to what was planned to be done. Both documents were very helpful for planning future tasks.

Another tool that was helpful with our planning and general workflow was the use of Gantt and burndown charts. These enabled us to more accurately gauge expectations for each week while also allowing each of us to have a better expectation of how much work we would have to allocate each week in order to hit our targets. The burndown charts allowed us to set goals for the week and then each day log how much work we had completed, with the goal of reaching zero by the end of the week. These burndown charts were created using excel and each group member had access to these and were able to log in their daily time spent on the project. The Gantt chart allowed us to visualise and to set future blocks of work to be completed.



We also implemented several different apps and resources during the course of this project. These included whats app, Trello, Zoom, GitHub, google docs and slack, with each a different but necessary purpose. Slack was used to document information of moderate formality such as stand up minutes which we felt were too formal for whats app but didn't need to be on google docs.

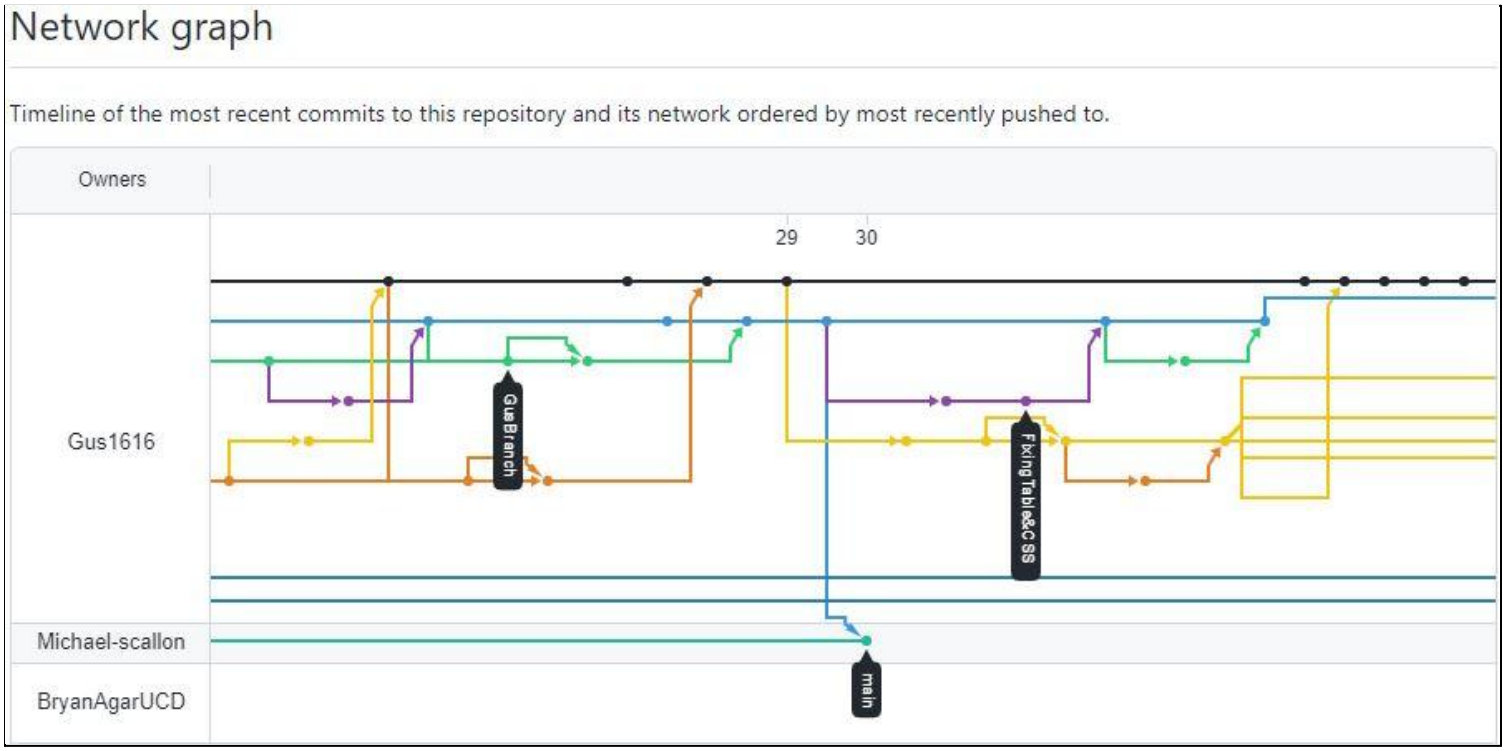


The least formal communication platform used was whats app. This was used to communicate the times for each stand up as well as more informal questions we may have had, while google

docs were used for more important documentation which we may have needed to refer to multiple times during the project such as wireframes or other important planning documents. Zoom was used for occasions where in-person standups were impractical, while Trello was used to make sure everyone knew which tasks were currently being completed to avoid the potential for two members to be working on the same task at the same time.

GitHub WorkFlow

GitHub was arguably the most important resource used. It was to allow for code to be shared online so that we could each work individually, while still being able to merge all of our work together. While at the beginning we struggled to come to terms with the git workflow. There were many issues such as repeated merge conflicts and accidental overwriting of key files. However, we became much more proficient at merging code, fixing conflicts and keeping our repository up to date as time progressed. See a snippet of our network of branches below.



Meeting Logs

Full meeting logs can be found detailing the contents of each meeting. Slack was used to log these meetings and can be found with the below link. - https://join.slack.com/t/dublinbikesworkspace/shared_invite/zt-17ch80g0i-Da20pnyK88eh9Dx9KukQ3w**.** Below is a summary table of meetings time , date and attendance. There are also sample meetings in the included appendices.

Date	Time	Attendance
08/02/2022	9.00	Full
09/02/2022	10.30	Full
10/02/2022	18.00	Full
11/02/2022	9.00	Full
14/02/2022	12.00	Full
16/02/2022	9.00	Full
17/02/2022	9.30	Full
21/02/2022	11.00	Full
22/02/2022	10.00	Full
24/02/2022	10.00	Full
28/02/2022	13.00	Full
01/03/2022	9.00	Full
03/03/2022	10.00	Full
11/03/2022	17.00	Full
14/03/2022	11.00	Full
17/03/2022	10.00	Full
20/03/2022	18.00	Full
24/03/2022	14.00	Full
26/03/3022	14.00	Full
30/03/2022	19.00	Full
01/04/2022	10.30	Full
03/04/2022	9.30	Full
05/04/2022	10.00	Full

Retrospective

What Worked Well

We think one of our greatest strengths was that we were all at a similar level in terms of programming ability. At the start, we thought that this could potentially put us under pressure as we were aware of other groups being more experienced. However, it turned out to be the reason why this project was so enjoyable. Since we were all at the same level we all worked on the different areas together (front end, back end, database). This meant we were able to learn from each other at a steady pace. There were no issues whereby one member was much further ahead and other members had to keep up. Conversely, there was no member much weaker causing the other team members to slow down to allow them to catch up. There was a very good balance in the team. Furthermore, as per the Scrum guidelines, smaller groups are more efficient. This was apparent as our communication was largely effortless and we were well organised.

What Was Problematic

As a team, we are very proud of what we produced. However, as is highlighted in the 'Future Work' section there are a number of areas that we would like to improve upon. The reason we could not have added some of these additions was mainly due to time pressure. While we managed our time well, for the most part. It would have been hugely beneficial to have a little bit more time. More specifically, we felt slightly inexperienced with the machine learning component. What we produced worked well, in particular, from a UI perspective. However, we feel as though our model could have been more advanced.

We also could have been a bit stricter without our coding style with regards to things like aspects like variables naming conventions and code sections. We understand that clean and clear code is paramount to software engineering and that lax coding guidelines can destroy a company. However, in particular with regards to our javascript file we could have been stricter with our layout of the code.

Bugs

- The map content such as the markers is a bit slow to load and loads after the splash page.
- Google charts content is slow to load which leads to slight page movement.
- In Dark mode, it might be slightly challenging to see text like the weather data. It depends on your screen quality. This would have been a quick fix by adjusting the colour slightly.

Future Work

There are a number of features given the time would be valuable additions:

- Obtain SSL certification for HTTPS to allow for user location tracking.
- Improving the machine learning model- Adding more features to the machine learning model. Due to time constraints, we were not able to incorporate weather data from a weather forecast API. Possibly exploring other models like Neural Networks,
- User Login system- Users could sign up and login and view their history of use and other elements like favourite features.
- Directions to bike station- the app could provide exact directions to their nearest bike station both in text form and visually on the map.
- App speed up - the loading of the map is a bit slow. A mechanism to speed up loading would increase user satisfaction. The google chart loading time is also quite slow. If we had more time a cleaner SQL query would probably have fixed that issue.
- Display statistics regarding the total number of bike rides or length of bike rides for a user. Such statistics may give the user a sense of accomplishment and thus increase usage.
- The nearest station with availability - currently our nearest station feature only displays the nearest station to the user regardless of availability.
- An option to translate the app to a different language. We suspect that many tourists would be interested in using the app. Having the option to translate the app into their language would greatly increase user satisfaction and usability in general.
- We would have liked to do unit testing of our app to test its robustness.

Conclusion

This report outlined our team's development of a Dublin Bikes web application. The design of the app, the objectives, the architecture, data analytics, the Scrum process, bugs, and future work were all discussed.

All three of us in this group are still very new to software engineering and more broadly computer science. We all have quite different backgrounds. From the completion of this project, we have learned a huge amount. Our practical skill levels were very similar prior to the completion of this project. This allowed us all to work on all aspects of the project synergistically, teaching each other as we learned.

Furthermore, we all have a much clearer understanding of the role of a software engineer. To us, software engineering is about integrating complex systems together to form a single product. A good software engineer is not just proficient at coding. They are also an effective communicator, they have the ability to understand and explain business requirements, they have a willingness to learn, work in a team and they have good time management skills.

Of course, this project was not without its struggles, from accidentally overwriting and corrupting important files on GitHub and frustration with Ajax.. However, it was a very enjoyable project and we learned a huge amount.

Contributions:

We have decided as a group to split the credit evenly amongst the team members:

- Gus = 33.33%
- Michael = 33.33%
- Bryan = 33.33%

Appendix

Sprint Reviews

Sprint 1

What went well?

- We have a clear understanding of what is expected for this web app. The architecture of the app was clearly defined.
- The team communicated well and received clear feedback from the product owner. The standups ensured all team members were on the same page.
- The AWS database was set up and the connection was made successfully.
- Python scripts were written that scrape the necessary data from the JCDecaux API.

What needs to be addressed for the next sprint?

- We need to clearly define the road map of the project
- Our git work-flow needs to be improved. Each member will have their own branch on which they will work from, while the development branch will only have code that has been completely debugged and is fully functional.
- We need to separate tasks more clearly to increase productivity. As opposed to all working on a specific task at the same time.
- We need to be disciplined with standups and ensure our work is being logged appropriately on slack.
- We need to get a better understanding of how a flask app works and start writing code.

Sprint 2

What went well?

- We gained a much better understanding of flask and were able to set up a file to run our application.
- We were able to manipulate the data from our database and now have a better understanding and have more ideas of where this information can be used , creating many ideas as to new features that we could implement.
- We had a basic shell of our application created which was working in a minimalist manner, however this allowed us to work on more detailed ideas as we had a better platform to test these features.
- We became much better at formalising our stand ups and our documentation of these meetings was also done in a much more professional manner.

- We became better at working on individual tasks , but also worked well at combining these together.

What needs to be addressed for the next sprint?

- While our git flow still needs some work we did improve on this aspect from the last sprint as we become more familiar with the core concepts of git.
- Our productivity , while having also improved from sprint 1 could still be improved. We need to work on balancing our time better so that the project moves forward in a more consistent manner as we worked in bursts within the sprint at times. However this is not a major concern and is moving in the right direction.
- The use of some of the applications to manage our work flow could be better . I (Michael) feel for example that I could be much better at using trello and can see that it would be very useful if I used it in a more efficient manner.

Sprint 3

What went well?

- We have a basic machine learning model for both predicted availability and for predicted available bike stands. These were both implemented as a way to create a start and end point for the journey of a user.
- The machine learning model predictions were given a user input that allowed the user to select a start time, day and station while doing similar for the end station using a mixture of javascript(ajax) and flask.
- We continued to work well refining the design of our web app and worked well together in designing an application layout that we as a group are becoming more and more proud of.
- We started to think of extra features that would bring our app to life such as a splash screen . Elements such as these, while not crucial elements of the app, add to the experience of using it and we found that these kinds of features are more achievable as we begin to move towards a more finished product.
- Our time management and use of both trello and github got much better. There is now a certain level of confidence working with github which was missing in earlier sprints and now have a much better ability to collaborate in a more fluid fashion. We seemed to streamline our rate of work more and we were able to continuously move the project along better than in the last sprint also.

What needs to be addressed for the next sprint?

- Our machine learning model is not dealing with weather data. This is something that we are spending a lot of time at , but unfortunately are struggling. While at the minute we have a model that deals with just bike data , it would be nice to have the weather data also available should time allow**.**
- There were some small bugs that would have ideally have been fixed during this sprint but were pushed to the final sprint like a small issue with the dark mode feature. While not a major issue we still need to work on our ability to get features 100 percent finished in a given time frame. We hope to fix these issues in the final sprint as we must reach a hard deadline.

Sprint 4

What went well?

- We hit our deadline on the creation of our app. While there were some elements we would have still liked to have completed we were still very happy with the final product.
- We spent a good amount of time on the report document. We feel that the documentation we had made during the course of our project was incredibly helpful during this stage and are glad we did this throughout the project.
- We were able to fix the majority of small bugs during this final sprint and worked well as a team to produce solutions for these problems.
- We were able to reflect on the project as a whole asit came to completion and realised how proud we were of our work, how much we enjoyed the process and how much we learned.

What needs to be addressed in future work outside of the assignment time?

- We never thought of what problems might arise once running the flask app on the ec2 instance. For example the geolocation aspects would not run as we were working with a http site rather than a https. In the end we did not leave enough time to fix these kinds of problems and in hindsight we should have launched our app on the instance earlier to give us time to realise these flaws and try and fix them.
- Our machine learning still does not include weather data. This is the biggest flaw we think our application still has. Unfortunately this aspect got put to one side as we wished to have a fully functioning application and there was not enough time to go back and fix this aspect, however it is something that we feel can be fixed after the project is handed in for our own piece of mind.

Sprint backlogs

Information regarding the breakdown of the following tasks and who completed each can be found in the gantt chart below.

Sprint backlog 1

-Create a virtual environment : We need to create a virtual environment to run our database/scraper code. This needs to be saved as a .yml file so it can be shared and used later on the Ec2 instance.
-Create Ec2 instance: We need to create an Ec2 instance to run our database/scraper file. This is so we can have the Rds collecting data even when our machines are closed and so it can run all the time.
-Create Rds : We need to create an AWS rds database. This is so we can all access the data and so it is not just stored on one machine in case something was to happen to it locally. It is also needed for later in the project. Getting this linked to mysql workbench is also something that needs to be done so we can view the data as you cannot view it on Rds alone.
-Creating documentation: We need to start thinking about our app , what it will look like and how it will work. We could potentially make some wireframes of ideas and we also need to nail down the architecture of how our app will run.
-Future planning: We need to look at what will need to be done for the next sprint. This can be done towards the end of this sprint so that there is a smooth transition between sprints.

Sprint backlog 2.

- Create flask app: We need to create a basic flask app so we can display our information. This will be the base of our app and will be the file that we run when we want to view our application.
- Weather scraper: We need to add a weather scraper and table similar to the one created in the last sprint about the bikes. This is so we can add weather information to the app and add weather to our machine learning later in the project.
- HTML file : We need to work on making a basic html file to be able to view our sight. The flask file is more back end and this is more of a front end file. It would also be ideal if we could embed the google maps part of the project into the page so we can start working on this part of the project.

- Flask app improvements : Once the basic flask app is running we need to start adding more details to it and learn how it works for the later parts of the project.
- Map functionality: We need to try and get all the bike locations to appear on the map. This will allow us to add more functions onto this part such as making them clickable. We need to try and get as much of this done as possible during this sprint.
- Future planning: We need to look at what will need to be done for the next sprint. This can be done towards the end of this sprint so that there is a smooth transition between sprints.

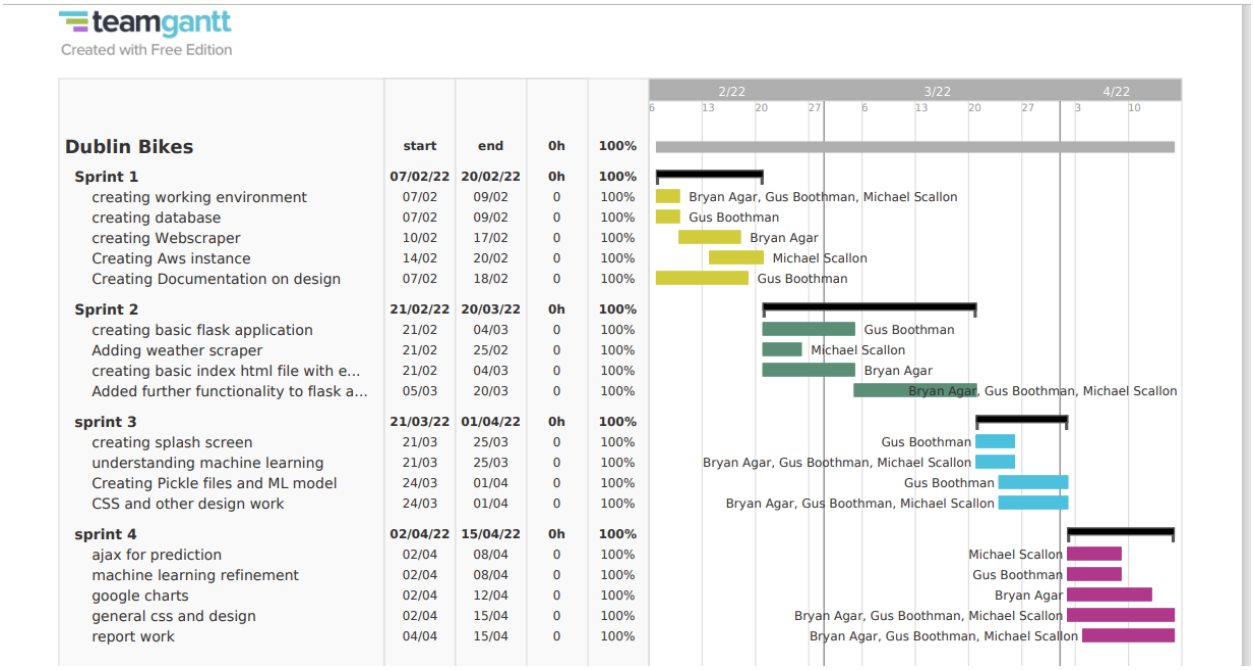
Sprint backlog 3.

-Information table : We need to make a table so that more information about the stations can be added and viewed. It would be ideal if we could get this to interact with the map also so that when the user selects a station to get more information on they can also see its location on the map.Maybe we could also add a sound effect for this too.
-Pan to location and nearest station : We need to add two buttons to our station map. One of these will show the users current location and the other will find the nearest station to the user.
-Splash Screen: The app takes a couple of seconds to load so it would be cool if we could get a splash screen working to hide this load time.
- Machine Learning : We need to start on the machine learning part of the project. We need to create pickle files based on the data in our database by training and testing it. Maybe random forest would be the best implementation but we need to test several different models and see which works best.
- Css : We need to add some more design to our application. Since there are now a few features nearly completed it would be nice to start structuring them to see how our app might take shape. It would also be very nice if it was responsive and worked well on mobile devices as well as on a computer.
- Machine learning interface: We need to work on a way to fetch and display our machine learning predictions. Maybe AJAX might be the best way to go about this so the page won't need to refresh every time a new prediction is requested.

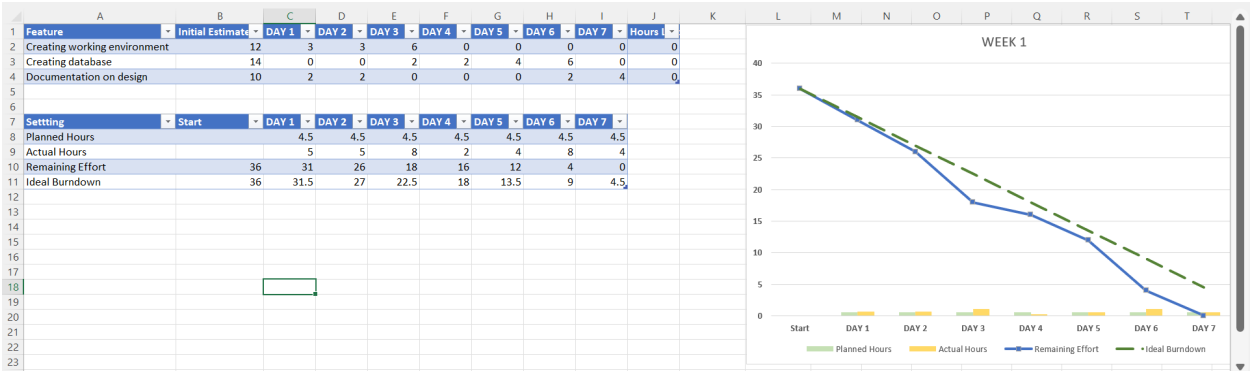
- Future planning: We need to look at what will need to be done for the next sprint. This can be done towards the end of this sprint so that there is a smooth transition between sprints.

Sprint backlog 4.

- Google charts: We need to get some google charts work done. We need to be able to allow the user to view details on average bikes each day for each station and to allow them to view the average bikes each hour of a certain day for a station.
- Report: We need to get more work done on our report. We need to start collecting all of the documents we have made so far and start adding them to a single google docs file. Splitting the report into different sections and dividing up the sections might be the best way to go about this.
- Hosting: We need to host our sight so it can be viewed on google chrome. We will need to get our flask app and pickle files on and running on another Ec2 instance for this and change some of the details so it can be viewed.
- CSS and bugs: We need to finalise our design and fix any bugs our app has. We need to test it thoroughly to see if anything doesn't work and finalise how we want our final product to look.

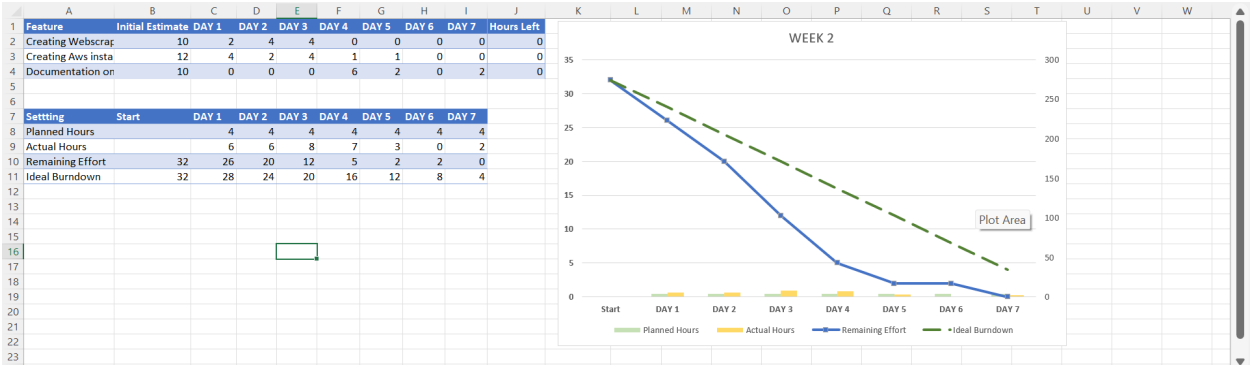


Burndown charts.



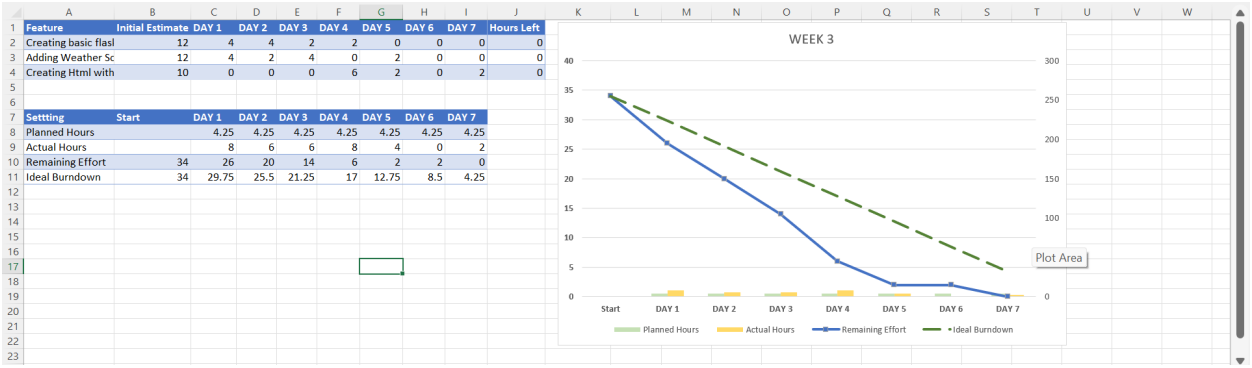
Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours
Creating working environment	12	3	3	6	0	0	0	0	0
Creating database	14	0	0	2	2	4	6	0	0
Documentation on design	10	2	2	0	0	0	2	4	0

Setting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		4.5	4.5	4.5	4.5	4.5	4.5	4.5
Actual Hours		5	5	8	2	4	8	4
Remaining Effort	36	31	26	18	16	12	4	0
Ideal Burndown	36	31.5	27	22.5	18	13.5	9	4.5



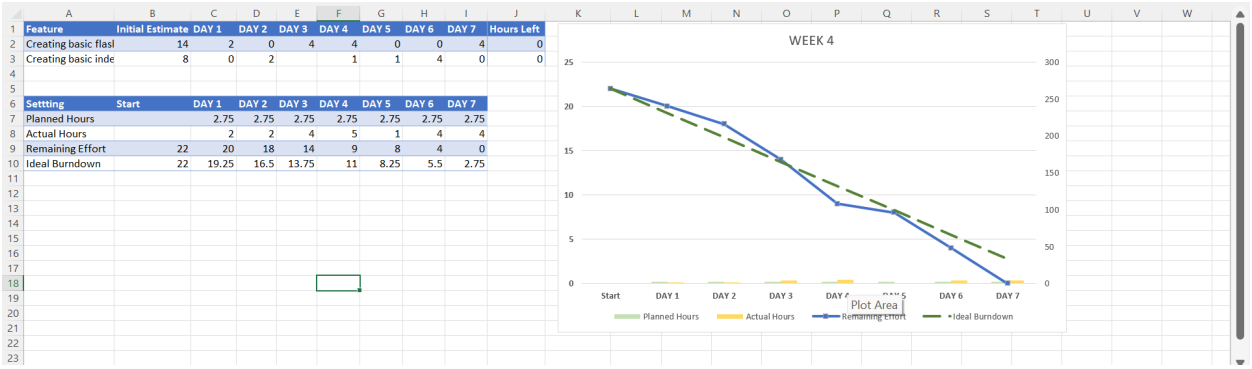
Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Creating Webscrap	10	2	4	4	0	0	0	0	0
Creating Aws insta	12	4	2	4	1	1	0	0	0
Documentation on	10	0	0	0	6	2	0	2	0

Setting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		4	4	4	4	4	4	4
Actual Hours		6	6	8	7	3	0	2
Remaining Effort	32	26	20	12	5	2	2	0
Ideal Burndown	32	28	24	20	16	12	8	4



Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Creating basic flask	12	4	4	2	2	0	0	0	0
Adding Weather Sc	12	4	2	4	0	2	0	0	0
Creating Html with	10	0	0	0	6	2	0	2	0

Setting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		4.25	4.25	4.25	4.25	4.25	4.25	4.25
Actual Hours		8	6	6	8	4	0	2
Remaining Effort	34	26	20	14	6	2	2	0
Ideal Burndown	34	29.75	25.5	21.25	17	12.75	8.5	4.25

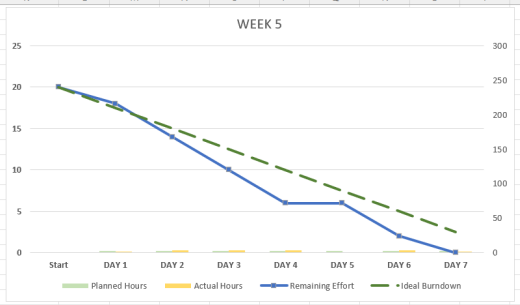


Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Creating basic flask	14	2	0	4	4	0	0	4	0
Creating basic inde	8	0	2		1	1	4	0	0

Setting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		2.75	2.75	2.75	2.75	2.75	2.75	2.75
Actual Hours		2	2	4	5	1	4	4
Remaining Effort	22	20	18	14	9	8	4	0
Ideal Burndown	22	19.25	16.5	13.75	11	8.25	5.5	2.75

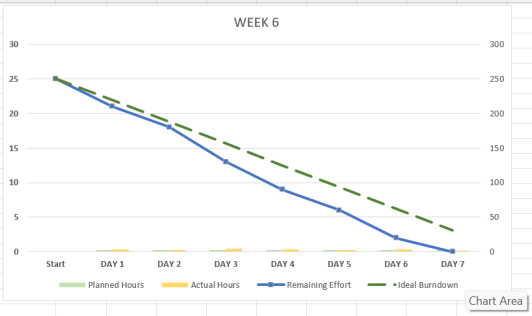
Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Added further func	20	2	4	4	4	0	4	2	0

Settting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		2.5	2.5	2.5	2.5	2.5	2.5	2.5
Actual Hours		2	4	4	4	0	4	2
Remaining Effort	20	18	14	10	6	6	2	0
Ideal Burndown	20	17.5	15	12.5	10	7.5	5	2.5



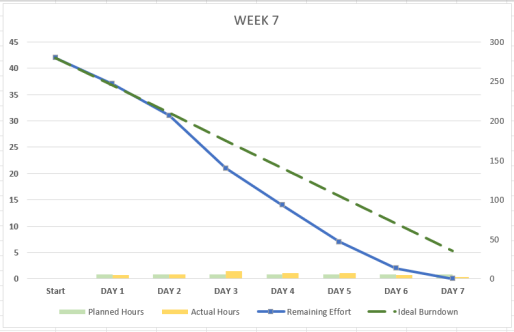
Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Added further func	25	4	3	5	4	3	4	2	0

Settting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		3.125	3.125	3.125	3.125	3.125	3.125	3.125
Actual Hours		4	3	5	4	3	4	2
Remaining Effort	25	21	18	13	9	6	2	0
Ideal Burndown	25	21.875	18.75	15.625	12.5	9.375	6.25	3.125



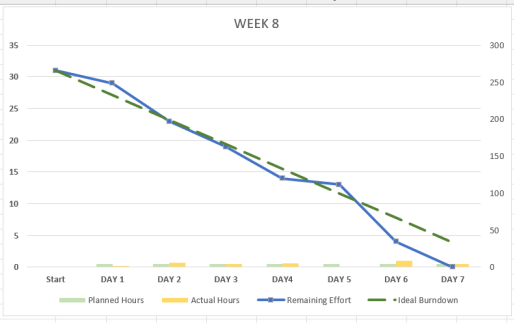
Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
creating splash scr	10	0	2	4	0	4	0	0	0
understanding mac	12	2	2	4	1	1	2	0	0
Creating Pickle file	20	3	2	2	6	2	3	2	0

Settting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		5.25	5.25	5.25	5.25	5.25	5.25	5.25
Actual Hours		5	6	10	7	7	5	2
Remaining Effort	42	37	31	21	14	7	2	0
Ideal Burndown	42	36.75	31.5	26.25	21	15.75	10.5	5.25



Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Creating Pickle file	15	2	4	0	4	0	5	0	0
CSS and other desi	16	0	2	4	1	1	4	4	0

Setting	Start	DAY 1	DAY 2	DAY 3	DAY4	DAY 5	DAY 6	DAY 7
Planned Hours		3.875	3.875	3.875	3.875	3.875	3.875	3.875
Actual Hours		2	6	4	5	1	9	4
Remaining Effort	31	29	23	19	14	13	4	0
Ideal Burndown	31	27.125	23.25	19.375	15.5	11.625	7.75	3.875

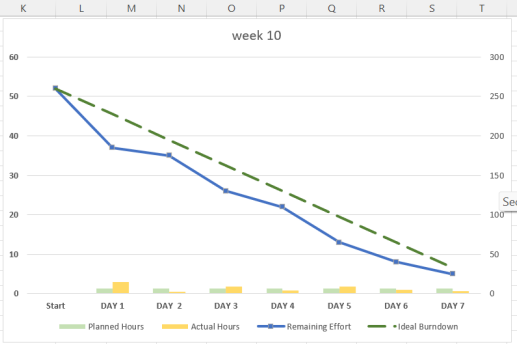


Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
Ajax	12	4	4	4	0	0	0	0	0
machine learning	12	0	2	4	1	1	4	0	0
design css	15	3	0	3	0	3	5	1	0
google charts	14	2	2	2	2	2	2	2	0
report	10	0	0	2	4	2	0	2	0

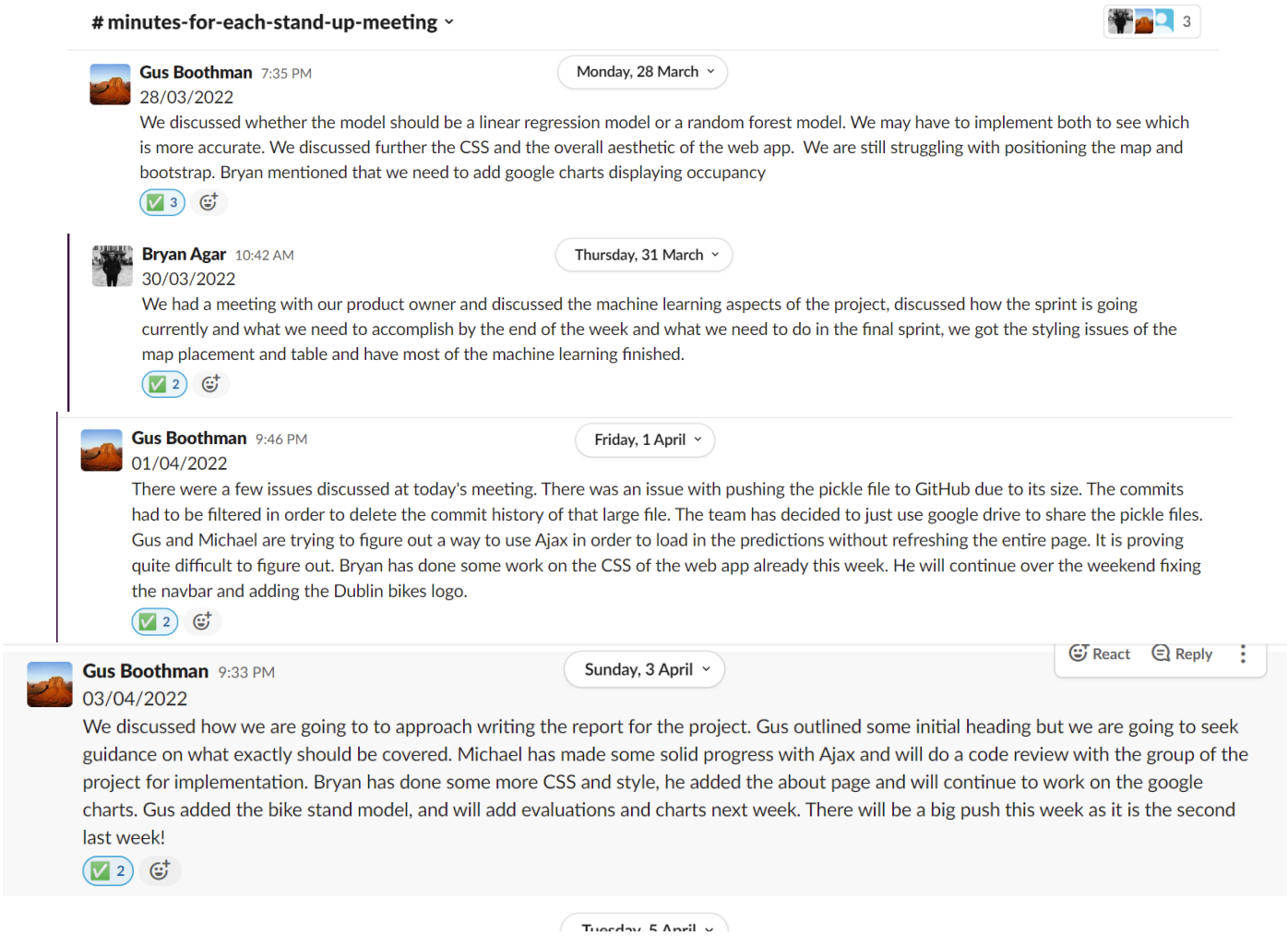
Settting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		7.875	7.875	7.875	7.875	7.875	7.875	7.875
Actual Hours		9	8	15	7	8	11	5
Remaining Effort	63	54	46	31	24	16	5	0
Ideal Burndown	63	55.125	47.25	39.375	31.5	23.625	15.75	7.875

Feature	Initial Estimate	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7	Hours Left
design css	20	3	0	3	0	3	5	1	0
google charts	12	4	0	4	0	4	0	0	0
report	20	8	2	2	4	2	0	2	0

Settting	Start	DAY 1	DAY 2	DAY 3	DAY 4	DAY 5	DAY 6	DAY 7
Planned Hours		6.5	6.5	6.5	6.5	6.5	6.5	6.5
Actual Hours		15	2	9	4	9	5	3
Remaining Effort	52	37	35	26	22	13	8	5
Ideal Burndown	52	45.5	39	32.5	26	19.5	13	6.5



Slack example stand up meeting minutes.



Trello task management.