Software Engineering

Dublin Bikes Web App

Group 24

Ezebilo Nnaemeka 22203869

Mohammed Choudhary 22203536

Mohammad Syed 17305051



Contents

[Project Overview 4](#)

[Introduction 4](#)

[Project Management 4](#)

[Sprints 4](#)

[Sprint 1 4](#)

[Sprint 2 8](#)

[Sprint 3 10](#)

[Sprint 4 11](#_t33krrv43hn0)

[**Data Analytics 14**](#_f1nqvlvnjxbh)

[**Frontend: 16**](#_39rnjllv9jum)

Details:

Github: https://github.com/nezebilo/dublin\_bikes

Group Number: 24

All team members contributed equally

# Project Overview

## Introduction

Over the last two decades, the use of Dublin city Bikes has grown significantly as more companies and multinational corporations encourage their employees to reduce CO2 emissions by opting for healthier commuting options like Dublin bikes. However, there is currently no satisfactory product available in the market to provide users with all the necessary information about Dublin bikes. Therefore, our objective is to develop a Dublin bikes web application that includes additional features and services such as predictions of available bikes and weather information. This report aims to provide an overview of the development process of this application. The product owner for this project is Farzin.

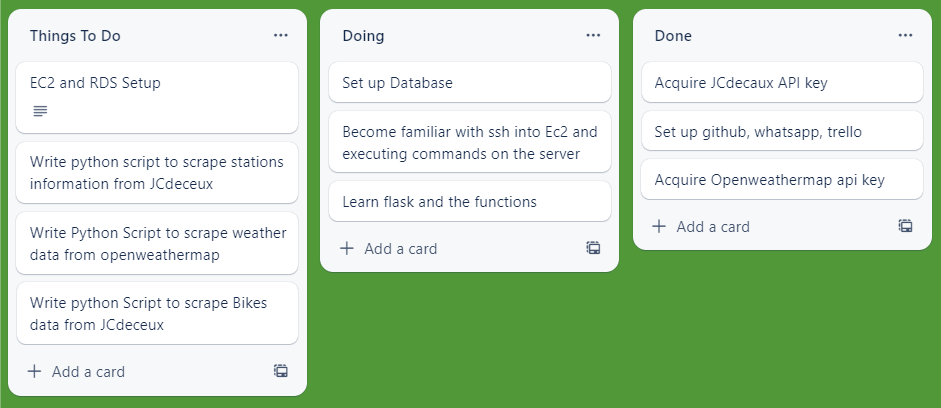
# Project Management

We have included all our team meeting logs and sprint planning throughout the project in this section. We followed the agile software development methodology of project management. The process was divided into 4 sprints with each sprint having a target which we were able to achieve. We had meetings once a week on average and on a few occasions having more as needed. Each sprint will be discussed in context of our Trello board, Backlog, challenges encountered and achieved outcomes.

## Sprints

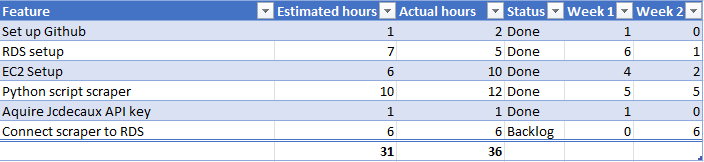
### Sprint 1

We used Sprint 1 to get familiar with the team, met with our product owner to get information on how to begin the project, set up the github, whatsapp group which we used primarily for communication, and to outline the necessary tasks.



*1Sprint 1 Trello Board*

Sprint 1 Backlog Table:



The burndown chart for the first sprint provides an approximate estimate of the time spent on the project's essential sections. The chart indicates that during the initial sprint, the team collaborated to discuss the project planning and set up the Dublin bikes application, following the agile methodology throughout the project's duration. To maintain progress, the team held regular meetings during practical sessions, including sprint planning meetings, stand-up meetings, and hurdle calls, as well as outside practical times. The backlog table shows that most of the sprint 1 requirements have been completed, but there were issues with setting up the ec2 instance. This caused a delay in working on the scraper and connecting it with the RDS database. The problem stemmed from the default security group settings in AWS, which only enabled ec2 to work on specific IP addresses. The team resolved the issue by researching the matter. After successfully creating the Python scraper and saving data to a CSV file for testing, there were further challenges in connecting it to the RDS, leading to a backlog for the following sprint.

**Process notes:**

**Meeting log 1:**

Date - 14/02/2023

Meeting minutes: 2 hours

Member attended: Mohamed, Akram, Emeka

Summary:

• The team was formed.

• A communication channel was established through WhatsApp.

• Tasks were identified, such as understanding the project description, reviewing other similar applications, and setting up GitHub and Trello.

**Meeting log 2:**

Date - 16/02/2022

Meeting minutes: 4 hours

Member attended: Mohamed, Akram, Emeka

Summary of discussion:

- Sprint planning

- Nomination of scrum master (Akram)

- Discussed the requirements of the project and the application

- GitHub has been set up

- Trello has been set up

- Google drive has been set up

- Divided the work for writing the python scraper into three (weather, static and dynamic)

between group members.

**Meeting log 3:**

Date - 21/02/2022

Meeting minutes: 3 hours

Member attended: Mohamed, Akram, Emeka

Summary of discussion:

• The progress of the scraper was discussed, and any issues were resolved.

• The work being done is a Proof of Concept (POC).

• Shorter scrum meetings were planned to take place every day.

• Hurdle calls were also scheduled to check if anyone was having difficulties with their assigned work.

**Meeting log 4:**

Date - 23/02/2022

Meeting minutes: 12 minutes

Members attended: Mohamed, Akram, Emeka

We discussed the next step after setting up the database and successfully scraping the data

We decided to add time intervals in scraping the data. This would be done by setting up Crontab on the EC2.

We also agreed to write a script that connects the scraped data from API to the AWS RDS database

**Meeting log 5:**

Date - 25/02/2022

Meeting minutes: 30 minutes

Members attended: Mohamed, Akram, Emeka

We held a phone call on Saturday

* Tested local dynamic scraper working manually
* Discussed issues with setting up the crontab on the EC2
* Discussed issues with scraper not working from EC2
* Came up with solution for continuous scraping

**Sprint 1 Backlog:**

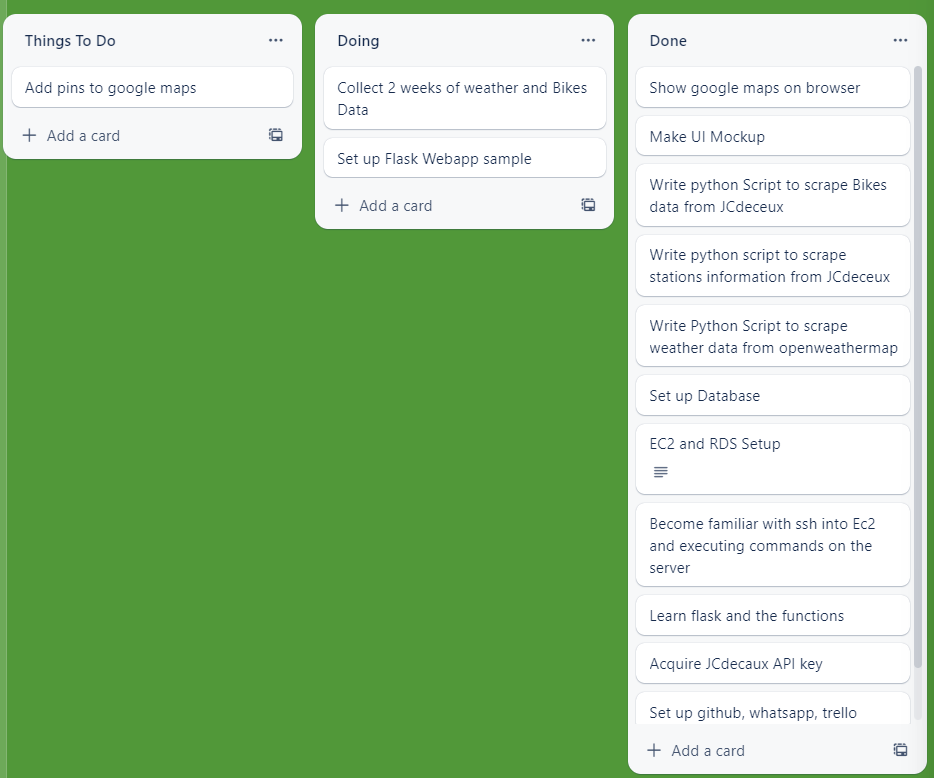
There was an issue in running the python scraper scripts off the EC2. They would just not input data to the database. The problem came from the environment not being able to install some of the dependencies. Due to the differences between windows and Linux. This delayed the next steps as we needed the scraper working to have data to use for the prediction. This was sorted out in first practical of sprint 2 and we were able to start collecting data

**Sprint 1 Reflection:**

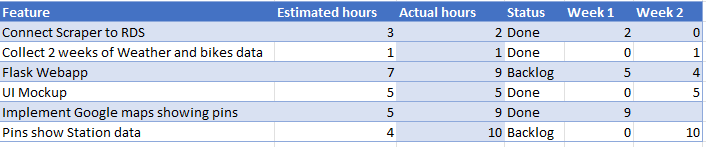
During the retrospective meeting of sprint one, the team reflected on their work during the previous sprint and discussed how they can improve moving forward. To ensure a smooth transition into the next sprint, the team agreed that if someone encounters difficulty in completing their assigned task, they can notify the team members through the WhatsApp group channel. A volunteer team member will then step in and continue the task where it was left off to finish it. This approach is expected to increase the team's productivity and help them move forward.

### Sprint 2

Sprint 2 focused on setting up the python backend for predicting the number of available bikes and the first steps of the flask web-app. We also did a Mockup of how the web-app would look. This is the Trello board for the 2nd sprint:



Sprint 2 Backlog table:



We have setup most of the backend and the initial parts of the frontend at the end of sprint 2. We haven’t finished with all the functionality of the flask and also we are able to display some pins on the map but not all the station pins yet.

**Process notes:**

**Meeting log 6**

Date - 28/02/2022

Meeting minutes: 45 minutes

Members attended: Mohamed, Akram, Emeka

Notes:

* The team had a meeting with the product owner to discuss sprint 2. During the meeting, they reviewed the backlogs from sprint 1 and discussed how to approach them.
* The team agreed that all members will focus on completing the backlog tasks from sprint 1 before moving on to sprint 2.

**Meeting log 7**

Meeting minutes: 45 minutes

Member attended: Mohamed, Akram, Emeka

Date – 2/03/2023

Notes:

* The backlog from sprint 1 (connect scraper to RDS database) has been resolved by installing dependencies manually instead of using environment.yml file
* The team also decided on the next steps for setting up the Flask application and acquiring the Google Maps API

**Meeting log 8**

Meeting minutes: 30 minutes

Member attended: Mohamed, Akram, Emeka

Date – 7/03/2023

* We set up the flask app and tested it working
* We discussed how to connect this with the google maps api
* Discussed about building the model for predicting the number of bikes at a future time and divided the remaining work between ourselves

**Meeting log 9**

Meeting minutes: 30 minutes

Member attended: Mohamed, Akram, Emeka

Date – 9/03/2023

* Discuss remaining backlog for sprint 2 and how to resolve them
* Evaluate our performance compared to our projected targets
* Next sprint will begin with finalizing the prediction model

**Sprint 2 retrospective**

The team was able to apply the lessons learned from sprint 1 to improve their efficiency in sprint 2. By working together and helping each other to solve problems, the team was able to effectively tackle any issues encountered by individual team members.

### Sprint 3

During Sprint 3, the team focused on developing data analytics capabilities to predict the number of bikes that would be available at each Dublin bike stand on a future date. This required the team to collect and process historical bike usage data, which was then analyzed using various machine learning algorithms. The resulting prediction model was then integrated into the Flask application, enabling users to view predicted bike availability at each station over the next seven days. This feature added significant value to the application by providing users with valuable insights into bike availability, which can help them plan their journeys more effectively.

**Meeting log 10:**

Date: 26/04/2023

Time: 1:30

Duration: 60 Mins

Attendees: Mohammad S, Mohammad A, Imeka

Agenda:

1. Discuss Flask application implementation
2. Review HTML and CSS design progress
3. Provide updates on data analytics integration
4. Identify issues and potential solutions
5. Plan documentation, meeting schedule, and burndown chart

Minutes:

1. Course of action:
   * Finalize Flask application structure and routing
   * Integrate data analytics code into the Flask application
   * Complete HTML and CSS design for the web application
2. Justifications and Design decisions:
   * Use Flask due to its lightweight nature and flexibility
   * Utilize Bootstrap for responsive web design and faster development
   * Integrate data analytics predictions using Pickle serialisation
3. Issues:
   * Challenges in integrating data analytics code into Flask
   * Ensuring seamless user experience across different devices
   * Balancing workload among team members
4. Documentation:
   * Document Flask application structure, routing, and data analytics integration
   * Record design decisions, justifications, and issues encountered
   * Update burndown chart and analysis
5. Meetings and Standups:
   * Schedule weekly meetings to discuss progress, address issues, and set goals
   * Conduct daily standups to track individual progress and identify blockers

### Sprint 4

Meeting log 11:

Date: 26/04/2023

Time: 1:30

Duration: 60 Mins

Attendees: Mohammad S, Mohammad A, Imeka

Agenda:

Review Flask application progress

Discuss HTML and CSS design improvements

Address issues in data analytics integration

Update on documentation and burndown chart

Plan next steps and future meetings

Minutes:

Course of action:

Finalise Flask application and data analytics integration

Address issues and bugs in HTML and CSS design

Complete documentation and update burndown chart

Justifications and Design decisions:

Use modular Flask application structure for better organisation and scalability

Incorporate user feedback in HTML and CSS design improvements

Optimise data analytics integration to ensure fast and accurate predictions

Issues:

Minor bugs in Flask routing and data analytics integration

Some inconsistencies in HTML and CSS design on different devices

Time constraints for documentation and burndown chart updates

Documentation:

Complete documentation on Flask application, HTML and CSS design, and data analytics integration

Record final design decisions, justifications, and issues resolved

Update burndown chart and provide final analysis

Meetings and Standups:

Plan final project review meeting to ensure all tasks are completed

Continue daily standups to track progress and address any last-minute issues

**Retrospective/Future Work**

**Retrospective:**

Reviewing the project, the team successfully developed a Dublin Bikes web application that integrated data analytics for predicting bike availability.

The team effectively utilized Flask for the application's backend, Bootstrap for responsive web design, and data analytics techniques like Random Forest Regression for prediction.

Despite some challenges in integrating data analytics code into Flask and ensuring seamless user experience across different devices, the team managed to address these issues and deliver a functional application.

Regular meetings, standups, and burndown chart analysis played a crucial role in keeping the project on track and resolving issues promptly.

**Future Work:**

Enhance prediction accuracy: Further refine the data analytics model and consider incorporating more features to improve the accuracy of bike availability predictions.

User Interface improvements: Continuously gather user feedback and make necessary improvements to the application's UI and UX, ensuring a more user-friendly experience.

Mobile application: Develop a mobile application for the Dublin Bikes service, enabling users to access the platform from their smartphones easily.

Real-time data updates: Integrate real-time data updates from Dublin Bikes and weather APIs to provide users with the most up-to-date information on bike availability and weather conditions.

Scalability: Optimize the application's backend architecture and infrastructure to support a growing number of users and expand the service to other cities or bike-sharing platforms.

Additional features: Implement new features such as bike route suggestions, user profiles, or gamification elements to enhance user engagement and promote the use of the Dublin Bikes service.

Collaboration with local authorities: Establish partnerships with local authorities to provide better insights into bike-sharing usage patterns, which can inform urban planning and infrastructure development efforts.

## 

# Data Analytics

1. Introduction

In this section, we discuss the data analytics process implemented for the Dublin Bikes web application project, focusing on the exploration, analysis, and prediction of bike availability at various stations. We will cover the machine learning models tested, the chosen model, the approach for weather prediction, and a detailed explanation of the code used in the project.

1. Machine Learning Models and Selection

2.1. Model Testing

To predict bike availability, we experimented with several machine learning models, including Linear Regression, Logistic Regression, K-Nearest Neighbors (KNN), and Random Forest Regression. We aimed to select the model that provided the best accuracy score.

2.2. Random Forest Regression

We ultimately chose the Random Forest Regression model, as it yielded an accuracy score of over 60% for all days of the week. The Random Forest model is a classification algorithm consisting of multiple decision trees, which are capable of fitting complex datasets and performing both classification and regression tasks. The model's main idea is to search for the optimal variable-value pair in the training set to split the data and generate the "best" two child subsets.

1. Predicting Bikes Availability and Weather Data

3.1. Weather Data API

For accurate future weather forecasts, we utilized an API to display the predictions. However, we also performed some weather data predictions using our collected data to assist in estimating future bike availability at each station.

3.2. Bike Availability Prediction Model

We predicted bike availability based on the collected weather data using the Random Forest Regression model. The model took into account features such as weather description, temperature, humidity, wind speed, and other relevant information.

1. Data Analytics Code Explanation

4.1. Importing Dependencies and Database Connection

We began by importing the necessary dependencies such as Pandas, NumPy, and various modules from Scikit-learn. We then established a connection to our database using SQLAlchemy, providing the appropriate credentials and connection string.

4.2. Data Preprocessing

We queried the weather and bike data from our database and performed data preprocessing steps like changing data types, selecting important features, and sorting the data by the 'last\_update' column. We then merged the weather and bike data using the 'last\_update' column as the key.

4.3. Feature Engineering

After merging the datasets, we performed feature engineering to extract additional information such as the month, day of the week, hour, and minute from the 'last\_update' column. We also encoded categorical variables, such as 'weather\_description', using Scikit-learn's LabelEncoder.

4.4. Model Training and Evaluation

We split the data into training and testing sets, with a 70-30 ratio. We trained the Random Forest Regression model on the training set and evaluated it on the testing set. The model's performance was assessed using metrics such as Mean Absolute Error, Mean Squared Error, Root Mean Squared Error, and R-squared score.

4.5. Model Serialization

After training the model, we serialized it using Python's Pickle module to save it as a '.pkl' file. This allowed us to use the trained model in our Flask application without having to retrain it every time.

1. Conclusion

In this section, we have detailed the data analytics process for predicting bike availability at Dublin Bikes stations using machine learning models. We have also provided a comprehensive explanation of the code used for data preprocessing, feature engineering, and model training and evaluation. The Random Forest Regression model, with its accuracy score of over 60%, has proven effective in predicting bike availability, contributing to the overall usefulness of the Dublin Bikes web application.

# Frontend:

The code provides the functionality for a web application that displays Dublin bikes stations occupancy levels and weather information. The outline of the main features are as follows.

Navigation bar

-A navigation bar that allows users to go to different sections of the page including the map, weather , news , safety tips and info about the company.

Map

-A map displaying Dublin bike stations' locations, with markers for each station, and an information window that shows additional information about each station which includes the name, number, address, card acceptance, and the total number of bike stands. Users can also click a button to view further details, including the current bike availability at the respective station.

Weather widget

-A weather widget that fetches and displays the current weather conditions in Dublin, including the description, temperature, and an icon which represents the weather conditions. It uses the openweather API to collect current weather information

Prediction

-A bike demand prediction form that allows users to select a station and input a date and time. The application fetches weather data and sends it along with the station and date/time information to the server to get a predicted bike demand. The prediction is then displayed on the page.

-Furthermore it includes Sections displaying recent news, cycling safety tips, and contact information for the company.

Javascript libraries like Jquery and underscore.js which allowed us to simplify the event handling for the front end.

The code is then stylised using CSS and the CSS file sets the font-family, font colors, and also the background colors for the body and various headings. It also styles the navigation menu , map section, weather widget, and different content sections with appropriate features such as the padding, margin, and background colors. Furthermore , the form elements, such as labels, input fields, and submit button, are styled for better usability and overall appearance.

The front end calls various APIs such as openweather and Jcdecaux using the respective API keys in order to fill the application with the information that users will need. It also reads static information from a dublinNEW.json file to display information that is not regularly changed on the website for simplicity.