Bike Sharing In Dublin
Whisper of Wheels
Group 3 - Report

for

Comp 30830

Alexandru Butuc (33%)- Backend, Frontend, Management
Dharnesh Vasudev IR (33%)– Backend, Machine Learning, Management
Diksha Kumar (33%)- Frontend, Report, Management

Introduction Video URL:

https://drive.google.com/drive/folders/1m2N3jmcNxjkiiDxeTEJOlLi9MduOoTFR?usp=sharing

Github URL:

https://github.com/AlexButuc/Software-Engineering.git

1. Overview:

This document outlines the requirements and the process for developing an online web application called “Whisper of Wheels,” which is designed to track bikes in Dublin. This project is part of the Software Engineering model (COMP 30830) at University College Dublin (UCD) and was completed by three group members under the supervision of our product lead, Akila Wickramasekara.

1.1 Project Objective:

The primary objective is to create a user-friendly and informative web application that displays real-time occupancy and weather data for bike stations around Dublin. This application aims to enhance user experience by providing meaningful insights through the analysis of historical trends and by utilizing machine learning to forecast bike availability. In addition to providing access to shared bike accessible occupancy information, the project seeks to:

- Improve public access to shared bike availability across Dublin.
- Offer predictive insights into future bike usage based on changing weather conditions.
- Encourage smarter travel decisions and promote sustainable urban mobility.

By the end of the project, we plan to deploy a responsive and interactive web application hoisted on Amazon’s Elastic Compute Cloud (AWS EC2). This will be developed in Python using Flask, integrating several technologies including:

- Real-time Application Programming Interface (API) data consumption from JCDecaux and OpenWeather for weather information.
- Cloud infrastructure for deployment and accessibility.
- Machine learning model trained on historical data to predict occupancy.
- JSON based data handling and trend visualization utilizing CSV sources instead of the previously planned AWS Relational Database service (RDS) usage.

All development will aim to adhere to clean modular code, organized within a GitHub repository to ensure maintainability, readability, and ease of reuse.

1.2 Target Users:

This web application is tailored for two main audiences:

1. Dublin citizens and commuters who use bicycling as their main mode of transportation and need real-time bike availability and weather information to streamline their travel.
2. Tourists in Dublin that are looking for an easy way to locate bike-sharing stations to navigate the city on two wheels.

1.3 Main Features with Screenshots:

1) Live Bike Station Map:

An Interactive map that displays all of Dublin Bike stations with real-time availability status.

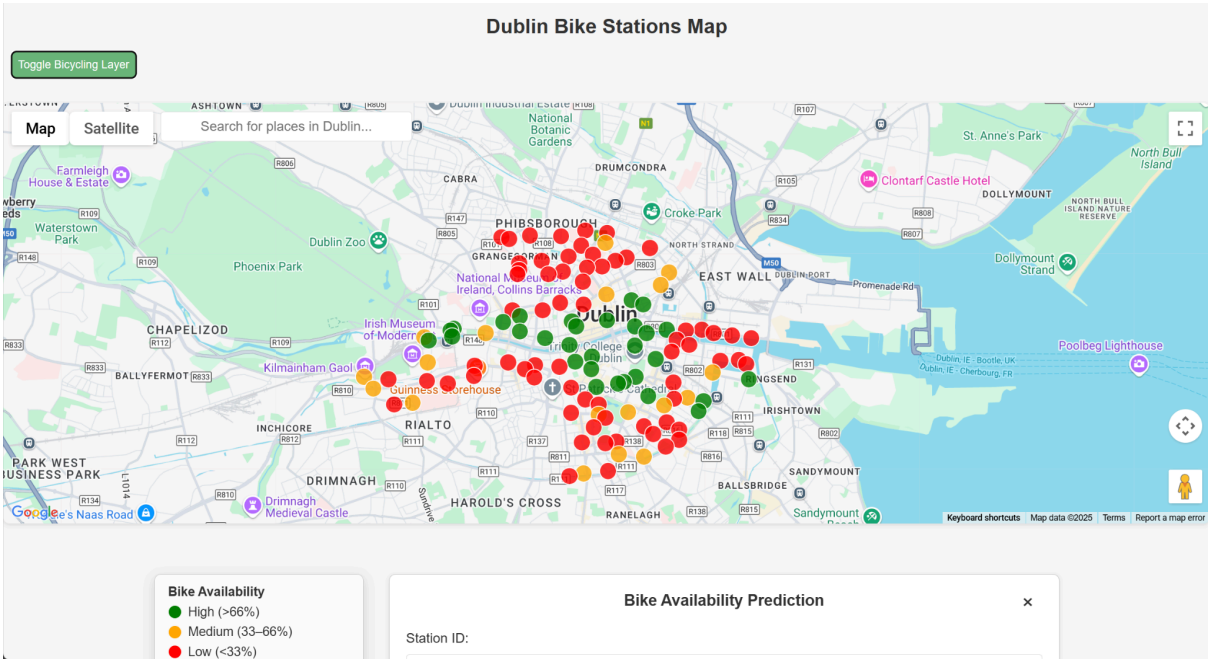


Figure 1.1 The Interactive map of Bike Stations on the website

2) Weather Information:

A display of the current weather information and conditions that will influence bike usage allowing the user to make informed decisions.

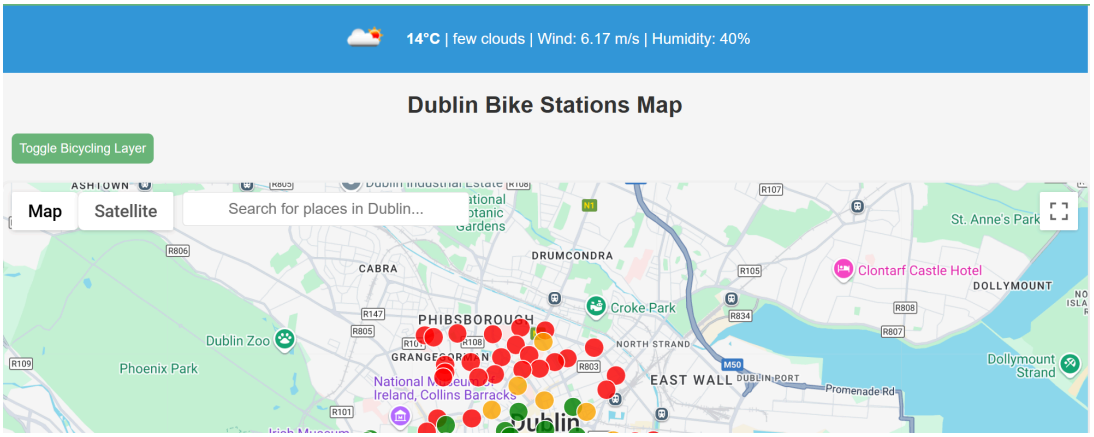
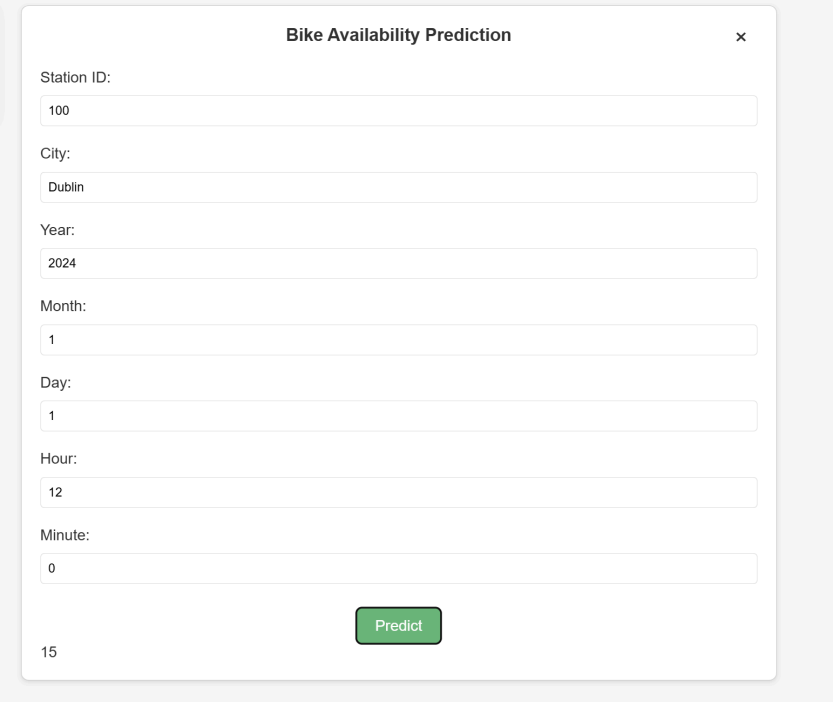


Figure 1.2 The live weather on the website

3) Occupancy Forecasting:

Machine learning powered predictions of future bike availability using historical data and weather patterns. Users can either manually enter in the station ID number or click on the marker they are interested in and it will automatically fill the number in the form. Once they enter all the information and hit predict, it will give them a prediction.



4) Station Information:

Users can interact with the map directly and click on a station marker that will allow them to view more detailed information such as the name of address, station id number, status, available bikes, and more details about the weather at that location.

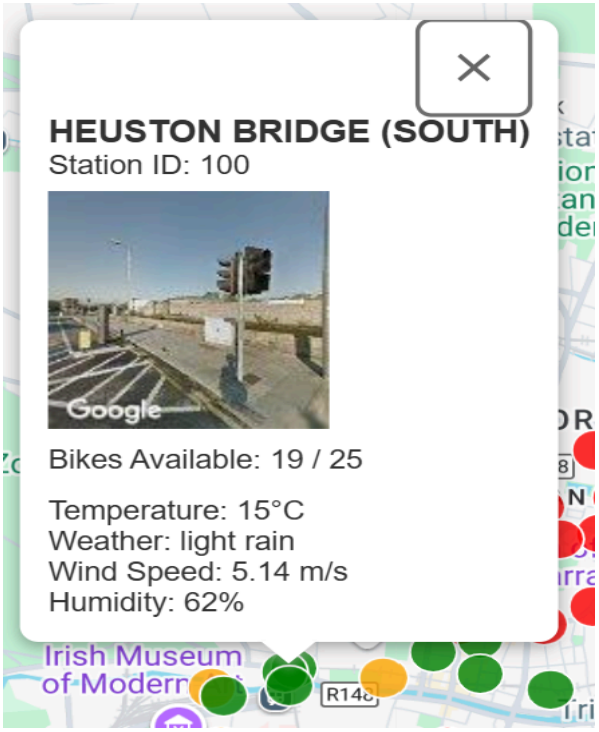


Figure 1.3 Result of Station Marker Interaction

5) API Integration and data Handling:

Real-time data that is fetched using JSON containing structured bike station data (converted from a CSV). This format replaces the suspended AWS RDS.

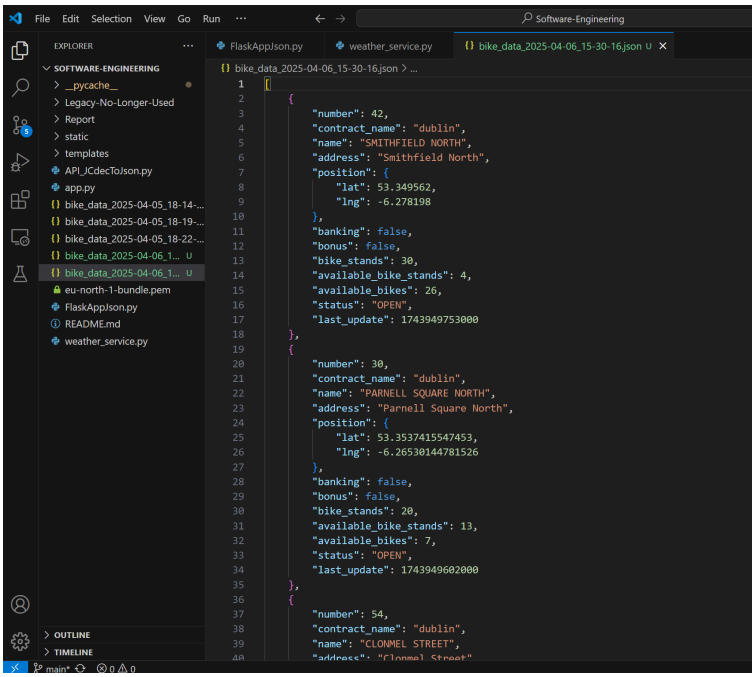


Figure 1.4 Code to Fetch Bike Data from JSON

6) Subscription and Purchase Options:

With this feature users can purchase options suited to their needs where they can choose up to three options. Users can click on the buy button and enter in their details to buy the travel pass.

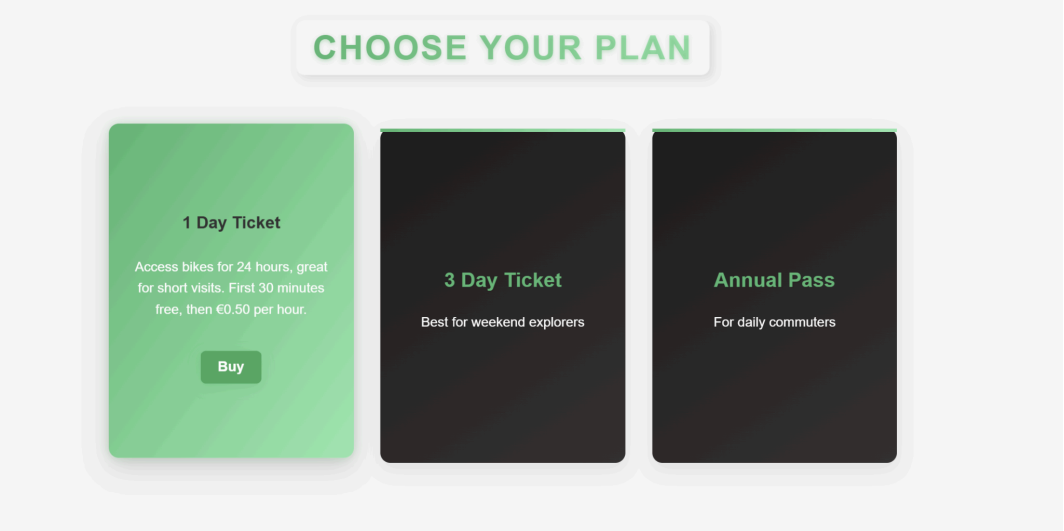


Figure 1.5 Subscription Options

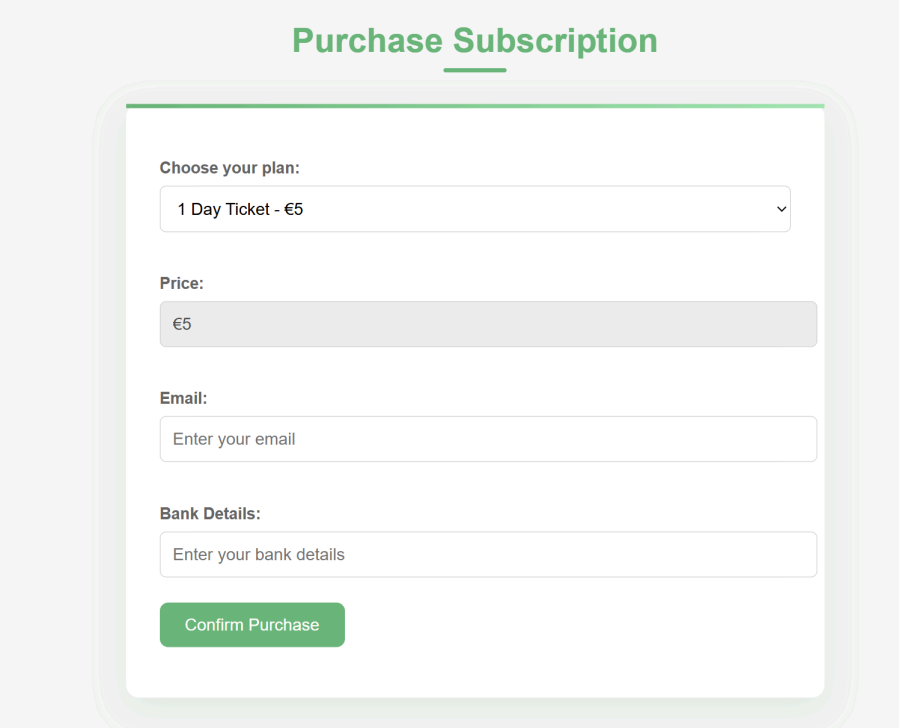


Figure 1.6 Purchase Page

2. Requirements:

The following section details the key user stories and the corresponding acceptance criteria for our web application. These user stories are crafted from the perspective of users engaging with the system and aim to capture the essential functionalities and features we implemented. Each story is accompanied by acceptance criteria, which outline the specific conditions that

must be fulfilled for each feature to be considered complete and operational. These user stories have been instrumental in the development of our web application, which is reflected in the various mockups presented below. To access the complete collection of our mockups, please refer to the GitHub link located on the title page within the "Additional Material" folder, specifically in the subfolder titled "Mockups."

2.1. Initial and Final Mockups:

2.1.1. Initial Mockup:

Our initial mockup was a minimal wireframe that focused on the basic structure of the app. It featured Google Maps as the central element, with clickable pins representing bike stops, and tabs for navigation across different pages. Although the design was minimalist, it effectively helped us outline the basic layout and user flow, ensuring that elements like interactivity and navigation were defined from the very beginning.

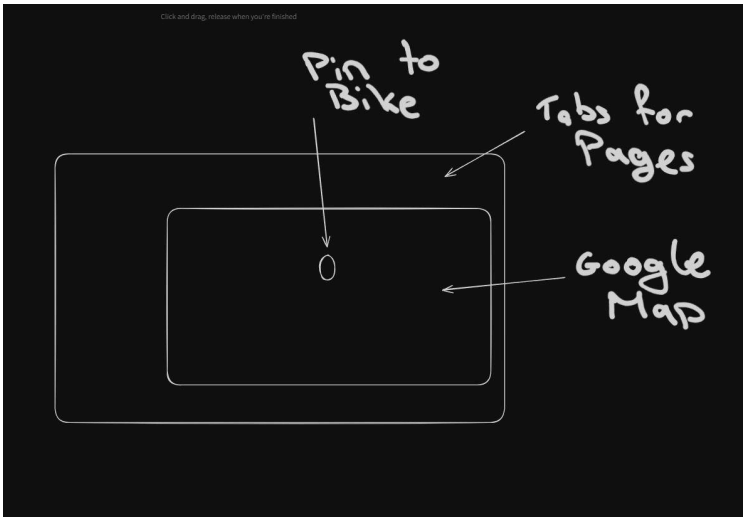


Figure 2.1 Initial Mockup

2.1.2. Final Mockup:

Our final mockup features a more polished and visually appealing design, complete with clear navigation tabs, branding elements, and engaging visuals. We incorporated real content areas and a structured layout that includes images, along with a footer section for contact information and social media links. The layout is designed to enhance usability and improve the overall user experience while meeting the functional requirements outlined in our user stories.

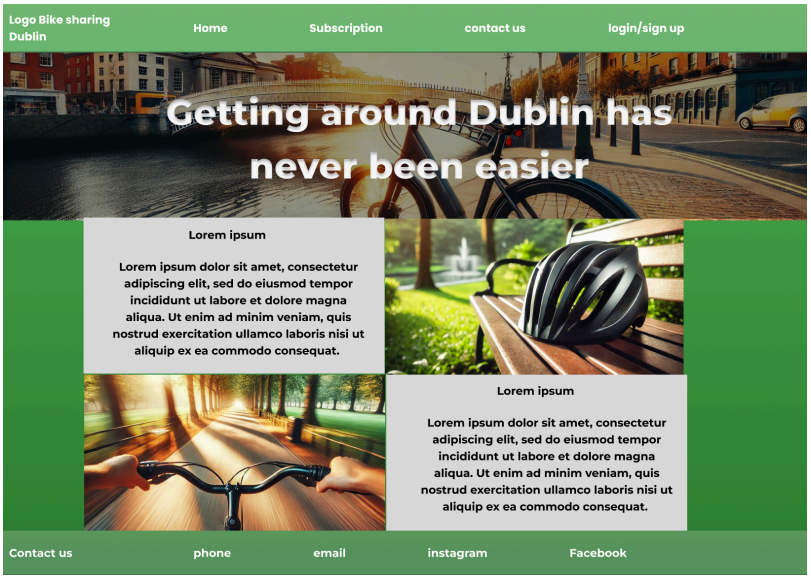


Figure 2.2 Final Mockup

2.2. User Stories & Acceptance Criteria:

1) **View all Bike Stations on a Map**:

   As a commuter or tourist, l want to see all the bike stations for Whisper of Wheels on a map so that I can easily find nearby stations.

   Acceptance Criteria:

   A dedicated maps section displays an interactive map of Dublin. All stations are marked with pins or icons. Clicking on a station shows its name and the bike availability and the station ID number.

2) **Check Available Bikes:**

   As a cyclist, I want to see how many bikes are available at each station so that I can plan my trip.

   Acceptance Criteria:

   Each station shows the number of available bikes while refreshing the data regularly (approximately every five minutes). The availability is indicated with color coding (e.g. green = high number of bikes available, orange = half the bikes are available, red = low number of bikes available) for quick understanding.

3) **See Live Weather Forecast:**

   As a user I would like to see the current weather and forecast so that I can decide if it's a good day to go cycling.

   Acceptance Criteria:

   For this weather information must be present and include all the main factors that affect cycling conditions shown on the dedicated maps page for ease of access**.** The weather

data updates automatically (approximately every ten minutes via an API call), and weather icons to make the information easier to understand.

4) **Search for a Place in Dublin:**

   As a user I want to be able to search any location in Dublin so that I can check if there are bike stations nearby.

   Acceptance Criteria:

   A search bar that allows users to enter an address or place. A map on the dedicated map page that allows the users to zoom in and out.

5) **View Bike Routes on the Map:**

   As a cyclist, I want to see suggested biking paths on the map so that I can choose a safe and efficient route.

   Acceptance Criteria:

   With this it is crucial to have bike-friendly routes that overlay on the map. The bike routes are visually distinct (e.g. different color lines). The route data is pulled from a static overlay. Users can have the option to toggle bike route visibility on/off.

6) **Purchase a Bike Pass:**

   As a user I would like to choose and purchase a bike pass so that I can access the bike-sharing service for a set period of time according to my own needs.

   Acceptance Criteria:

   Users can choose the options between one day, three day, or annual pass (for regular commuters). The form will take the user's email and payment details. Once the information is entered a random confirmation code is generated and displayed. Please note that no real transactions or data storage will occur during this.

7) **View Contact information:**

   As a user I want to view the contact details and potential social media links so that I can be directed to the most efficient way when I need help or to follow for updates about the service.

   Acceptance Criteria:

   For this there should be a contact us section somewhere on the website application. For immediate contact information there should be an email and phone number. For updates on the services provided there should be links to social media pages. Please note that no actual communication functionality is included.

8) **View Predicted Bike Availability:**

As a user I want to see predicted bike availability at stations so that I can plan future trips more effectively.

Acceptance Criteria:

Occupancy predictions are shown alongside current availability at each station. Predictions are based on a trained machine learning model using historical data. The predicted number of bikes is shown for a future time (e.g. next hour).

3. Architecture and Design:

This section provides a comprehensive description of the architecture of the application for Whisper of Wheels, highlighting the essential high-level components and their interactions. The diagrams seen below showcase the various elements of the system following a description that offers additional context for each diagram. To see further sequence diagrams that further breakdown what each function does please refer to our github link and navigate to "Additional Materials" and "Diagrams."

3.1. Diagram of Overall Structure:

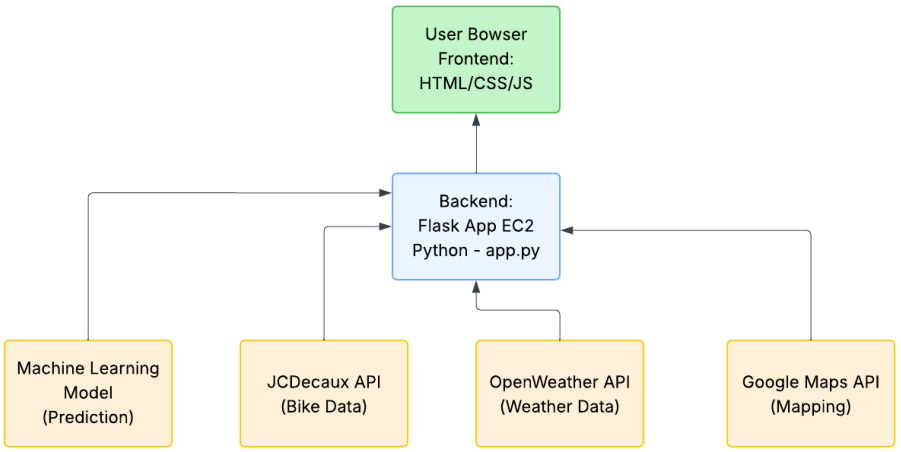


Figure 3.1 Overall Structure Diagram

3.2. Class Diagram of the Web Application and its Elements:

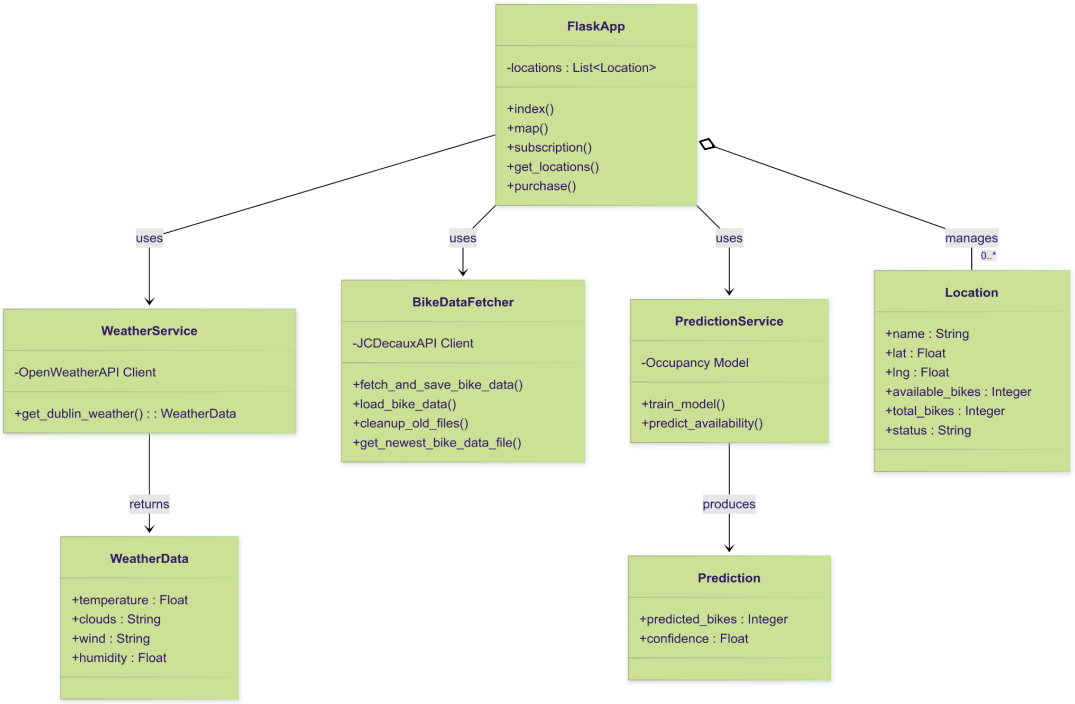


Figure 3.2 Class Diagram

3.3. Sequence Diagram of the Interactions Between Web Application Elements:

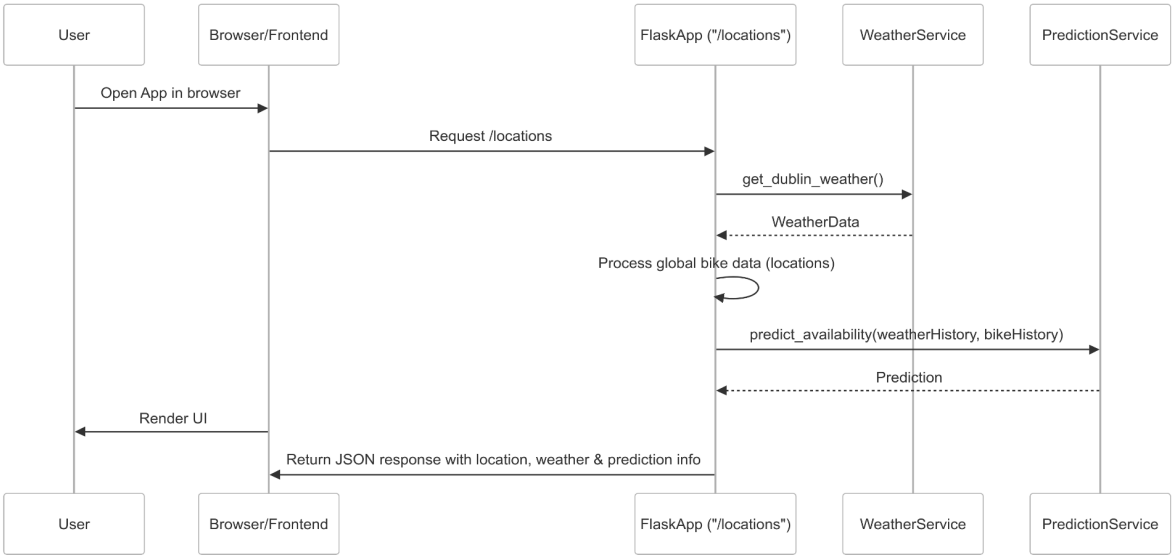


Figure 3.3 Sequence Diagram

3.4. Description of the Diagrams:

Figure 3.1 features the overall structure diagram that is useful for stakeholders to understand the overall flow of the data and the system's primary building blocks. This

illustrates the high-level architecture of our application for Whisper of Wheels and depicts the following components:

User Browser/ Frontend (HTML/CSS/JS): This is where users interact with the application. In this, requests are sent to the backend and they receive rendered pages and corresponding data in response.

Backend (FlaskApp, hosted on EC2, running using Python app.py): This is the core of the application. It processes requests, orchestrates data retrieval from external services, integrates the machine learning model, and returns appropriate responses to the frontend.

External Services: JCDecaux API provides real-time information about bike stations, such as the number of available bikes and total stands. While OpenWeather API supplies up-to-date weather information for Dublin, which is presented to user and it also used for predictions. Google Maps API renders an interactive map that displays the locations of bike stations.

Machine Learning Model: This component processes historical bike and weather data to predict future bike availability.

Connections between the backend and these components highlight how the Flask app gathers and compiles data before sending it to the user's browser.

Figure 3.2 depicts the class diagram useful for developers to understand where different pieces of functionality are located. It breaks down the logical building blocks of our application and demonstrates how they interact.

Flask App (/locations): Manages routes and maintains a global list of "Location" objects. It uses the "WeatherService" to fetch current weather data, the "BikeDataFetcher" to retrieve and manage bike data, and the "PredictionService" to forecast bike availability.

WeatherService: Communicates with the OpenWeather API and uses the method "get_dublin_weather()" that returns "WeatherData" objects containing attributes such as temperature, wind, and humidity.

BikeDataFetcher: Represents the logic for interacting with the JCDecaux API to fetch bike station data. Functions in this class handle periodic data retrieval and maintenance of JSON files.

PredictionService: This houses the machine learning model used to predict bike availability. The method in this model "predict_availability()" will produce a prediction object that shows based on the weather how many bikes will be available.

This diagram emphasizes how each entity in our system is organized and how they connect with each other.

Figure 3.3 This sequence diagram demonstrates the user's journey to obtain bike availability, weather, and predicted data through the locations endpoint:

1) **Open Application in Browser:** The user launches the application in a browser. The initial user interface (UI) (e.g., index.html) is loaded.
2) **Browser/Frontend Request:** The user via the frontend script makes a GET request to /locations.
3) **Flask App:** The Flask application receives the request and handles it and proceeds with the data assembly.
4) **WeatherService:** Flask then calls the WeatherSerice to fetch up-to-date weather information for Dublin, WeatherService returns a WeatherData Object containing temperatures, wind, and humidity.
5) **Process Global Bike Data (locations):** The Flask app retrieves the latest bike station information from its global locations variable, which is updated by a background data-fetching thread.
6) **PredicitionService:** The Flask app also calls the PredictionService. This service processes historical data for bikes and weather to forecast bike availability.
7) **Return JSON Response:** Finally, FlaskApp compiles the location data, weather information, and prediction results into a JSON response, sending it back to the browser to update the UI.

This flow demonstrates how multiple components work together to generate the necessary data for the frontend.

4. Machine Learning Model:

To improve the user experience and deliver accurate predictions of bike availability, we developed a machine learning model that estimates the number of bikes available at a station. This prediction is based on historical usage patterns, station characteristics, temporal information, and real-time weather data. This section details the data preparation process, the selected features, the target variable, the methodology for model training and performance evaluation.

4.1. Selected features, Feature Extraction/Data Cleaning Process, and Target Variables:

4.1.1. Data Cleaning and Feature Engineering:

Data cleaning is an imperative part of the preprocessing step to ensure the machine learning model could make accurate predictions. Below details the cleaning and feature engineering we used to set up our machine learning model.

- **Null Value Removal:** Rows with missing values were dropped to avoid introducing bias or errors into the model. This ensures the integrity and consistency of training the data.
- **Dropping Columns with Unique Values:** The columns with unique values aside from year and month were removed because they offer little to no predictive power. Which if left in could introduce noise and lead to overfitting.
- **Datetime Conversion:** We converted the last_reported field to datetime type, this allowed for temporal features to be used such as day of the week.
- **Weather Aggregation:** Instead of using the original min and max temperatures and humidity, the averages were computed allowing for normalization.
- **Derived Temporal Feature - day_of_week:** This feature captures human behavior trends (e.g., weekday vs. weekend commuting), which significantly impact bike demand.

4.1.2. Selected Features:

- **station_id:** This feature encodes location-based trends, different stations may have different usage patterns.
- **year:** Used for understanding seasonal and yearly patterns with the usage of bikes.
- **Day_of_week:** Depicts weekly user behavior like commuters vs weekend riders
- **num_docks_available:** Shows current or average availability which can relate to bike availability.
- **capacity:** Gets the total number of docks at the station, which normalize other metrics.
- **temperature and humidity:** Weather conditions affect behavior of riders, people may avoid biking in the cold, rain, or even hot humid days.

4.1.3. Target Variables:

Num_bikes_available: This is the output the model aims to predict, the number of bikes expected to be available at a specific station, on a specific time and date, based on certain weather conditions.

4.2. Training and Testing Process for Model Selection:

The predictive model used in our project is a Random Forest Regressor from scikit-learn; this was picked due to its robustness, ability to handle non-linear relationships, and due to its built-in feature for importance.

When training the pipeline: Data was grouped by station_id, year, month, and day to create a station-specific historical trends. The historical averages for num_docks_available and capacity were calculated to fill gaps for future predictions, when real-time data may not be available. Finally, the processed dataset was split and used to train the model, and the final model was serialised as a .pkl file for deployment.

Model Performance: The model demonstrated strong performance on the validation set with the following metrics: Mean Absolute Error (MAE): 0.06- indicating low prediction error and R^2 score: 1.00 - showing the model explained nearly all variability in the target data. This all confirms that the selected features and training process resulted in a highly effective model.

Integration and Deployment: The trained model is deployed in the Flask web application and predicts bike availability for future times using a combination of live and historical data. This integration allows the application to make real-time predictions, helping users make informed decisions about bike availability across different stations in the city.

5. Testing:

To ensure the application's reliability and functionality, multiple rounds of testing were conducted on various components of the project. The goal was to validate the system's accuracy, data flow, and integration of the user interface.

Testing Activities:

Backend Web Scraping and Data Refresh:

One team member focused on verifying that the backend's data scraping functionality worked correctly. This included checking that bike station data from the JCDecaux API refreshed at scheduled intervals, was saved properly, and that older files were automatically deleted to manage storage. Successful updates were confirmed by observing real-time logs in the terminal.

Model Prediction Accuracy:

Another team member tested the machine learning model by selecting various bike stations and comparing the predicted number of bikes available with the historical data. This ensured that the predictions aligned with known trends and validated that the model incorporated location-specific behavior effectively.

End-to-End System Validation:

Full integration testing was performed by a team member who ensured that the frontend displayed updated information and predictions as expected. This involved validating the consistency of user inputs (such as time and location) with backend predictions, checking real-time weather integration, and ensuring smooth user interface behavior.

Observed Issues and Fixes: Initial testing revealed that all stations returned the same prediction (e.g., 15 bikes), which was fixed by correcting the model preprocessing training. A later issue showed that individual stations gave the same prediction regardless of date or time inputs, indicating the model wasn't fully using temporal features. Due to time constraints it could not be solved and was noted for future improvement.

6. Process:

Our group adopted several tools and platforms to effectively organize and manage this product. These resources significantly enhanced communication and collaboration among team members, as well as task tracking and progress monitoring. These technologies improved our workflows and communication throughout the project's lifecycle.

6.1. Organisation and Management of the Project:

1) **Github:** We utilized GitHub for version control, which allowed us to share code and collaborate on the project effectively. Each team member was able to contribute to the codebase, keep track of changes, and resolve any conflicts that arose in the code.
2) **Discord:** This was our main tool for real-time communication with our professor and product owner. It allowed us to discuss our ongoing work and quickly clarify any doubts or issues.
3) **WhatsApp:** The application WhatsApp was our primary way to communicate, using text, voice and video calls. This made discussions easier, especially for sprint planning, complex problem solving, and general updates.
4) **Google Sheets:** This platform was used to track the project's progress through each sprint, ensuring transparency and accountability. We maintained a clear schedule of tasks and each Scrub Master updated the sheet during their time. It also included a table to record and calculate our burndown chart shown later in the report. Below is the link to our group's google sheets.
   https://docs.google.com/spreadsheets/d/1egNIBtgNKbbdAlXGzWMXeTOj52FBi9nzC-sRtWdMeXA/edit?usp=sharing

5) **Figma:** This website was utilized to design user interface mockups, which helped us visualize the layout and design of the project before we implemented the frontend.
6) **LucidCharts and Mermaid:** These two websites were employed to create various diagrams seen in this report such as the overall, class, and sequence diagrams. These diagrams were crucial in explaining the architecture and flow of the system, providing clarity on how the system components interact.
7) **Amazon EC2:** For deployment, we used Amazon EC2 to host our website. This cloud service offered a flexible and scalable environment for our application, ensuring it could handle the expected traffic while maintaining performance. EC2 enabled us to manage resources efficiently and deploy updates to the site as needed.

In terms of data we worked with two key datasets:

1) **Historical Weather Patterns and Bike Data(CSV data):** This dataset was instrumental in training a machine learning model designed to predict bike availability based on weather conditions. By analyzing this data, we optimized bike distribution and ensured availability across various weather scenarios.
2) **Real-time Bike Availability Data (CSV Data):** This data was automatically retrieved every five minutes, providing crucial insights into bike usage. It enabled us to maintain up-to-date information about bike availability in the system.

Additionally, with given permission from the professor, we took advantage of external resources to enhance our development process:

1) **Stack Overflow:** This platform proved invaluable for troubleshooting specific issues and finding solutions to common programming challenges. It allowed us to quickly access answers and best practices shared by the developer community.
2) **Large Language Models (LLMs):** We used this tool to clarify technical concepts, suggest coding strategies, and assist with debugging tasks. While these resources complemented our main development efforts, they significantly supported us in maintaining an efficient workflow.

By integrating these tools and resources, we effectively managed the project, collaborated seamlessly, and implemented a data-drive, machine learning-based system to allow users to use our application and make decisions about using our bikes with ease.

6.2. Sprints:

This section summarizes the progress made during each sprint of our project development cycle, highlighting implemented features, completed work, and decisions made. We'll also

review team performance through the Sprint Burndown chart, Sprint Review, and Sprint Retrospective. The main goal was to implement key features, resolve issues, and keep the project on track. Each sprint focused on tasks from the sprint backlog, yielding results aligned with project goals. At the end of each sprint, we met with our product owner, Akila, to review completed work and set the focus for the next sprint. For this project, we implemented the Scrum methodology, organizing our work into four sprints. Each group member was assigned to lead one sprint, while one member took on the responsibility of two sprints. It is important to note that our group agreed to approach this project as if it were a real job, and as such, we decided generally not to work on weekends and holidays.

6.2.1. Sprint 1 Scrum Lead Alex:

Sprint Goal:

Sprint 1 focused on establishing the foundation for the project. This included installing necessary software, defining user stories, creating mockups, setting acceptance criteria, and initiating data scraping.

Sprint Backlog & Work Completed:

- **Flask Exploration & Setup:** The team explored Flask to prepare for future web development tasks.
- **Git Setup:** A shared Git repository was established for efficient collaboration and version control.
- **EC2 and RDS Setup:** EC2 was configured for hosting, and an RDS instance was set up for database management.
- **Preliminary Data Scraping Setup:** Initial configurations were made for the data scraping feature, laying the groundwork for future scraping tasks.
- **UI Design & Mockup:** The team discussed and created a basic UI mockup to guide future design work seen below.
- **Software Installation & Setup:** All necessary development tools, including Flask and Python, were successfully installed.
- **Flask App Setup:** A basic Flask app was set up as the project's foundation.

- **Burndown Chart:** A burndown chart was created to track task completion throughout the sprint.

- **Team Bonding:** The team took time to get to know each other and build a collaborative atmosphere for future sprints

Sprint Review:

Summary of Completed Work:

In Sprint 1, the team made significant progress in setting up the project's foundation. Key tasks such as setting up the development environment, installing necessary software, creating the initial Flask app, and starting the data scraping implementation were completed. Additionally, the team defined user stories, created basic UI mockups, and set up a Git repository for collaboration. The primary goal of this sprint was to prepare for future development by laying the groundwork for the next steps.One major issue we encountered while completing our tasks was underestimating the time required for each task. This led to a slightly unfavorable burndown chart, which can be seen below in the sprint retrospective. Moving forward, we will aim to set more realistic time estimates for future sprints.

Feedback from Product Lead:

The feedback from our product lead, Akila, was highly positive. Akila appreciated the progress made, especially considering it was our first sprint. The initial setup, including the EC2 and RDS configuration, was recognized as a solid foundation for the upcoming work.

Next Steps:

Based on the feedback and completed tasks, the focus of the next sprint will be refining the data scraping functionality, improving the UI design, and continuing to build on the Flask application.

Sprint Retrospective:

Sprint 1 went smoothly, with the team adapting well to the new project and its requirements. There were no major issues that significantly hindered our progress. The positive atmosphere and open communication contributed to a seamless sprint without any significant roadblocks. Since this was our first sprint, the team was still getting familiar with the project and with each other. This led to some minor challenges in communication and coordination, but these issues were quickly addressed. Overall, the sprint was a productive experience.

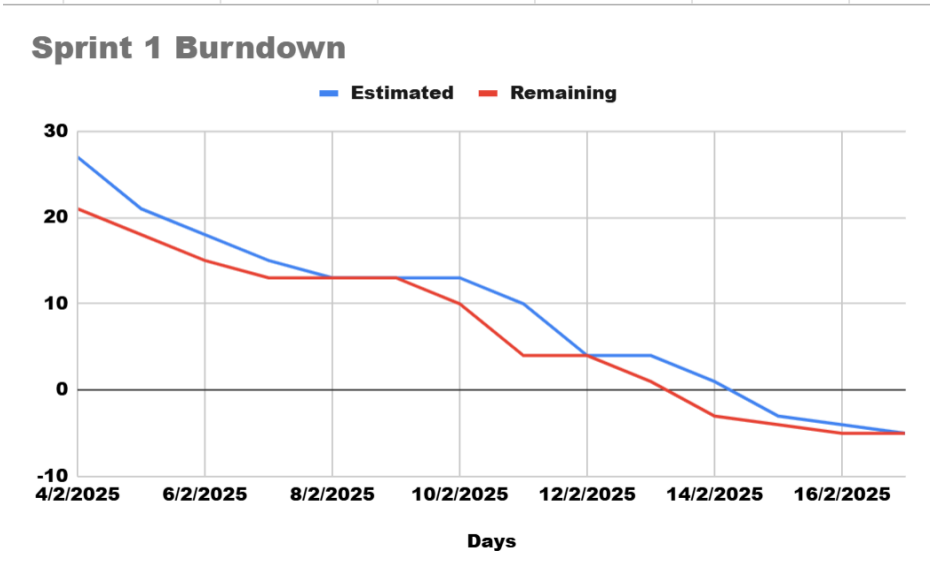


Figure 6.1 Sprint 1 Burndown Chart

6.2.2. Sprint 2 Scrum Lead Diksha:

Sprint Goal:

Sprint 2 focused on implementing data scraping, building the Flask app, and initiating preliminary front-end work. The goal was to progress significantly in terms of functionality and begin preparing for the final stages of the project.

Sprint Backlog & Work Completed:

- **Mock Website Creation:** The team created a mock website as a starting point for front-end work and to visualize the project's structure.

- **Open Weather Data Scraping:** Data scraping for weather information was successfully implemented, allowing the application to collect and utilize real-time weather data.

- **Report & README Work:** The team began drafting the final report and README file, outlining project details and functionality.

- **API Data Integration Exploration:** Research was conducted on how to combine data from different APIs, such as weather and Google Maps data, for seamless integration.

- **Google Maps Data Scraping:** The team successfully implemented data scraping for Google Maps, collecting relevant location data to integrate into the project.

Sprint Review

Summary of Completed Work:
Sprint 2 made significant strides in the development process, with successful implementations of weather and Google Maps data scraping, as well as progress on the Flask app and the front-end mock website. The team also began preparing project documentation, including the final report and README file.

Feedback from Product Lead:
The product lead provided positive feedback, noting the solid progress and the successful integration of data scraping features. The team was praised for their work on the Flask app and front-end, which set a strong foundation for the next sprint.

Challenges:
However, it was acknowledged that we once again underestimated the time required for some tasks, which slightly delayed progress.

Sprint Retrospective:

Due to the absence of a team member who was out of the country, we encountered some communication issues caused by the time difference. This resulted in delays for certain tasks and confusion for the missing member. Despite these challenges, the team did their best to collaborate effectively. We utilized Google Sheets, Discord, and WhatsApp to maintain communication and keep things on track. The team member who was away had informed the group in advance about their absence for a few weeks so the team was also aware of their time away. For future sprints, we plan to set more realistic timelines for tasks and further refine our communication tools and scheduling to ensure smoother collaboration.

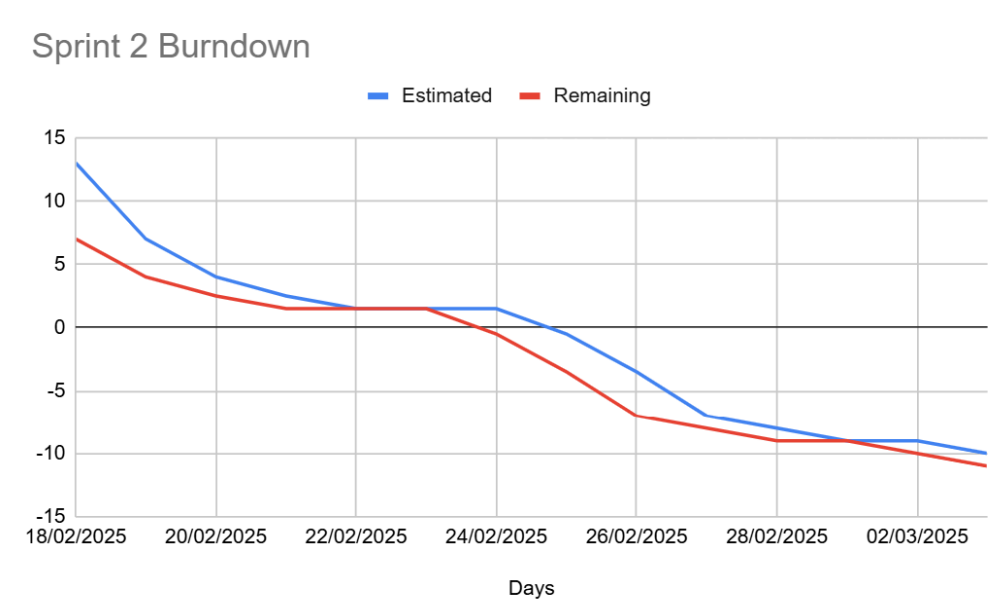


Figure 6.2 Sprint 2

6.2.3. Sprint 3 Scrum Lead Dharnesh:

Sprint 3: Implemented Features and Completed Work

Sprint Goal:
Sprint 3 focused on front-end implementation and refinement. The main goal was to start building the front-end pages and make key design decisions.

Sprint Backlog & Work Completed:

- **Main Page:** The team worked on the main page, establishing the layout and ensuring basic functionality.
- **Login/Signup (Deferred):** A decision was made to leave the login/signup feature until the end of the sprint as it was not a priority for the immediate functionality.
- **Map Page:** The map page was developed as part of the front-end, allowing users to visualize key location-based data.
- **Subscription Page (Booking & Payment):** The team began working on the subscription page, focusing on bike booking functionality as well as working on the payment section.
- **"About Me" Section on Main Page:** A small "About Me" section was added to the main page as a personal touch for users that explains what the website is for.
- **Report:** Work continued on the final report, documenting the progress and milestones achieved.

Sprint Review

Summary of Completed Work:
In Sprint 3, the team made substantial progress on the front-end, completing the main page, map page, and subscription page. The "About Me" section was added to the main page for a more personalized user experience. Significant strides were made in designing key aspects of the front-end, and the final report continued to take shape.

Feedback from Product Lead:
Akila provided positive feedback, praising the team for their continued progress. The front-end work was progressing well, with a clean layout and functional map page. However, Akila mentioned that the team could work on further refining the user interface to improve visual consistency across pages. While no major concerns were raised, the product lead emphasized the importance of completing the remaining pages in the upcoming sprint.

Challenges:
The only challenge mentioned was the fact that some tasks took longer than expected, particularly the subscription page, which required more attention to detail than initially anticipated.

Retrospective:

The sprint went smoothly, resulting in significant progress on the front end. The team collaborated effectively on the UI design, and basic functionalities were successfully established. Communication within the team was strong, which helped lay a solid foundation for the next sprint. However, we faced a major issue regarding the necessity of a login/signup page. There were conflicting opinions from various demonstrators in our lab, which created confusion about whether this feature should be included in the current sprint. After extensive numerous discussions, the team ultimately agreed to defer the implementation of the login/signup page for now. Moving forward, we will prioritize clearer decision-making regarding feature prioritization to avoid unnecessary back-and-forth discussions.

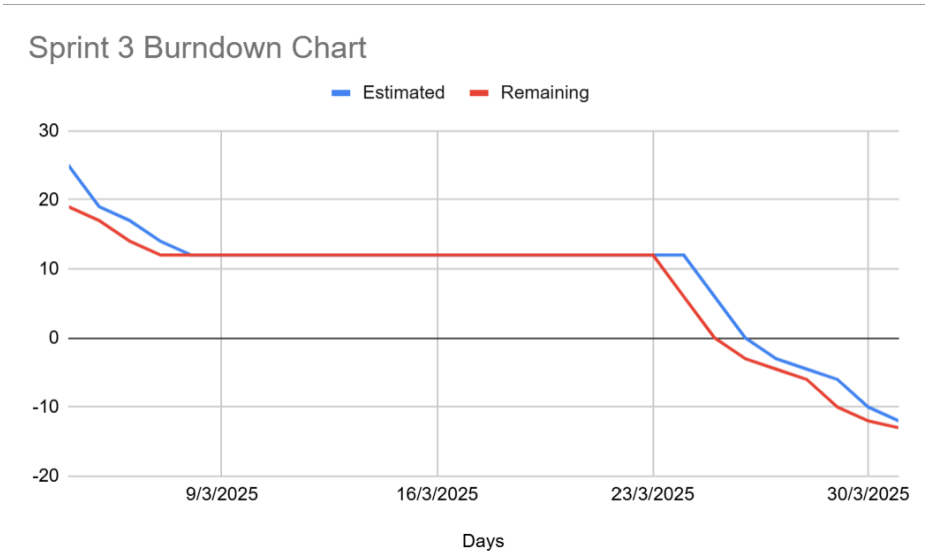


Figure 6.3 Sprint 3 Burndown Chart

6.2.4. Sprint 4 Scrum Lead Diksha:

Sprint 4: Implemented Features and Completed Work

Sprint Goal:
Sprint 4 focused on the development and testing of the machine learning (ML) model and its integration with the front-end. Additionally, final touch-ups were made to the CSS, the report was completed, and the README file was updated to reflect the final additions.

Sprint Backlog & Work Completed:

- **CSS Touch-ups:** Final adjustments were made to the CSS of previous pages, improving the overall visual appeal and user experience.

- **Report & README Work:** The final report and README file were worked on, documenting the project's objectives, implementation, and results.

- **Machine Learning Model:** The design and implementation of the ML model were completed. The model was trained and prepared for integration with the front-end.

- **ML Integration with Frontend:** The machine learning model was successfully integrated into the front-end, allowing for real-time predictions and user interaction.

- **Testing:** Comprehensive testing was carried out to ensure that the ML model and front-end integration functioned as expected, and any bugs were addressed.

- **Video for Report:** A video was created to demonstrate the project and its key functionalities, which was included in the final report.

- **Final Touch-ups:** Final refinements were made to the entire project, ensuring it was ready for submission.

Sprint Review

Summary of Completed Work:
Sprint 4 focused on integrating the machine learning (ML) model with the front-end, refining the user interface, and finalizing documentation. The team completed the key tasks, including the implementation and testing of the ML model, its integration into the front-end for real-time predictions, and final touch-ups to the UI. The team also worked on the report and README file, ensuring they were clear and informative. Additionally, a demonstration video was created to showcase the project's features.

Feedback from Product Lead:
Akila was pleased with the overall work completed during the sprint. The team successfully accomplished all assigned tasks, including the development of the machine learning model and its integration with the front-end. Akila appreciated the team's efforts in refining the user interface and completing the necessary documentation.

Challenges:
There were some challenges that impacted the sprint. One team member had previously communicated that they would be unavailable for a few weeks but contributed remotely. Additionally, one of the team members struggled with certain aspects of the ML model, causing some delays and necessitating extra coordination through phone calls to resolve one of the issues. Despite these challenges, the team adapted well, and the sprint ended with all key tasks completed.

Sprint Retrospective:

Challenges aside, Sprint 4 went quite well overall. The team's milestone for this sprint was to work on the machine learning model and integrate it with the front-end, which we accomplished. Other accomplishments included finishing touches on the report and README file and polish on the user interface. As was the case with previous sprints, this was the final sprint, and combined with the looming deadline, the team was under more pressure than usual, which created some friction. One of the members had notified the team in advance that he would be out of reach for three weeks but would do whatever he could remotely. Because of the contributions this member had already made, the sprint was not negatively impacted by his absence. On the other hand, one of the team members hit a wall with some parts of the machine learning implementation, which set the timeline back considerably. That meant a lot more calls and some rushing around to make sure that the model worked to a degree in the end. The team does need to consider how to deal with plans gone awry, especially with tasks like machine learning, foreseeing these sorts of gaps in strategy is important.

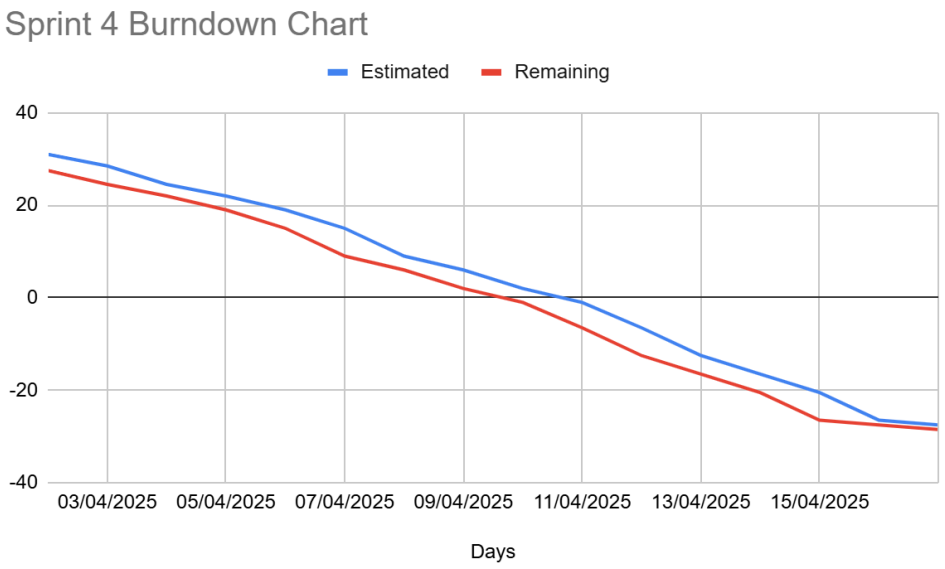


Figure 6.4 Sprint 4 Burndown Chart

7. Conclusion:

The development of Whisper of Wheels represents a software development achievement and a significant learning milestone. Throughout the planning, scrums, developments, and design phases, complex tasks were broken down into user-centered sprints, resulting in a functional web-based application. Key milestones included implementing data scraping from the JCDecaux API, integrating OpenWeather data, and building a user-friendly interactive map with real-time updates. A machine learning model, trained on historical data, was integrated into the front end, providing users with predictions for future bike availability. The entire

application was deployed using AWS EC2. The main challenges the team faced included miscalculating timelines and sporadic absences of team members. However, these issues did not hinder the team's adaptability. The integration of real-time information, along with predictive analytics, underscores the importance of data in promoting sustainable urban transportation mobility.

While we are proud of what we have achieved within the scope of this module, several future enhancements could further elevate the user experience and system capability:

- UI Refinements: Although functional, the user interface could use some visual enhancements, particularly on the maps page. Making the website more user-friendly would greatly improve the overall experience.
- Prediction Model Improvement: Incorporating more advanced models and utilizing longer-term weather data could enhance the accuracy of predictions. While also fixing the minor bugs that can be currently seen in the code when testing.
- Sign-in/ Sign-up Page: Though initially considered to be a part of the application, but later due to complexity and time constraints was discarded, adding user authentication would enable personalized experiences and features, such as tracking ride history.
- Payment System Integration: Implementing a fully functional and secure payment gateway would facilitate the purchase of bike passes, making it more practical and scalable.
- Navigation and Bike Route Suggestion: Introducing dynamic route planning that includes real-time traffic information and bike lane data would enhance the app's versatility and utility for daily commuters and tourists alike.
- Additional Features: Future versions could include features such as maintenance alerts for bike stations, user reviews or ratings, and gamification elements like rewards for frequent riders. These additions would provide a more comprehensive biking experience in Dublin.

Overall, Whisper of Wheels provides a solid foundation for a robust bike-sharing application. While it still needs to catch up with more established solutions in today's market, it serves as a valuable prototype that showcases the potential of real-time information, user-driven design, and machine learning in urban transportation. More importantly, the project has been an excellent learning experience for our team technically, creatively, and collaboratively. We have faced genuine development challenges and managed deadlines and client expectations, allowing us to learn valuable lessons that we can proudly carry into real workplace environments after completing this program.