Urban Computing Application Dublin Bikes Analytics

Tanmay Samdani
Student ID: 24348137
CS7NS4, Trinity College Dublin

November 28, 2024

Assignment 4: Urban Computing Application

Submission Date: 26 November 2024

Supervisor: Trinity College Dublin

Declaration: This assignment was performed individually.

I confirm that all work submitted is my own.

Contents

1 Introduction

This report documents the development of the **Dublin Bikes Analytics** application, which serves as a practical implementation of urban computing concepts. The project integrates real-time data and historical datasets to provide actionable insights into Dublin's bike-sharing ecosystem. The application includes interactive visualizations, advanced analytics, and clustering insights to support urban mobility planning.

2 Application Architecture

The application architecture (Figure 1) consists of:

- **Frontend:** Interactive web interface built with HTML, TailwindCSS, and JavaScript.
- **Backend:** A Flask-based RESTful API for data aggregation, processing, and analytics.
- **Data Sources:** Real-time data from the CityBikes API, complemented by historical datasets stored in Firebase Firestore.
- **Visualization Tools:** Leaflet.js for interactive maps and Chart.js for detailed analytics charts.

2.1 Application Flow Chart

The architecture flow is depicted in Figure 1, illustrating how the frontend interacts with the backend, data sources, and visualization layers.

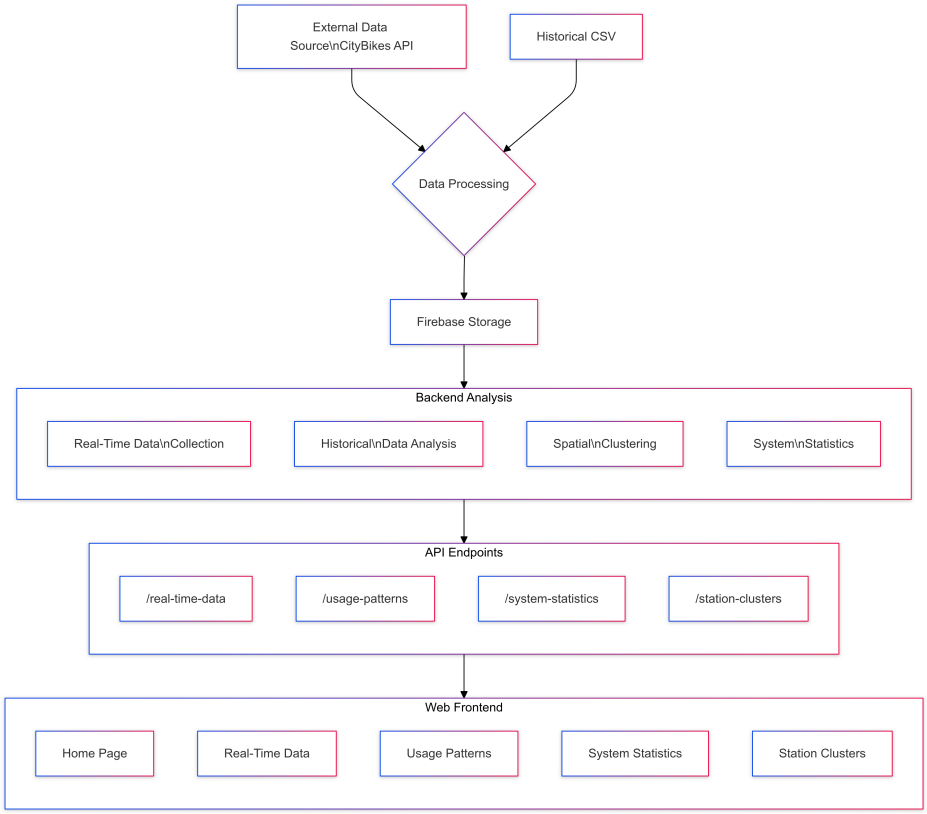

Figure 1: Application Flow Chart

3 Task 1: Data Fusion and Analysis

3.1 Motivation and Algorithms

To provide meaningful insights, the application integrates real-time sensor data with historical datasets, enabling advanced analytics. Key algorithms and techniques include:

- **Clustering:** K-Means clustering groups stations based on spatial location and utilization metrics.
- **Temporal Analysis:** Analysis of hourly and daily trends derived from historical data using pandas.
- **Data Normalization:** Min-Max scaling ensures consistent clustering results.

3.2 Data Processing Steps

The data processing pipeline includes:

1. Fetching real-time data from the CityBikes API.
2. Merging real-time data with historical data from Firebase and local CSV files.
3. Performing clustering, temporal trend analysis, and generating station-level performance metrics.

3.3 Code Implementation

The backend functionality is implemented in Flask, with key components:

- `get_real_time_data()`: Fetches live data from the CityBikes API.
- `analyze_historical_data()`: Processes historical data to generate temporal trends.
- `cluster_stations()`: Implements K-Means clustering for station grouping based on usage.
- `urban_computing_insights()`: Combines real-time and historical data for advanced metrics such as busiest and quietest stations.

4 Task 2: Data Visualization and Actuation

4.1 Visualization Goals

The application provides interactive visualizations to aid in decision-making:

- **Real-Time Data:** Displays bike availability across stations on a dynamic map.
- **Usage Patterns:** Visualizes weekly trends, peak/off-peak hours, and station performance.
- **System Statistics:** Summarizes key metrics such as total bikes, station availability, and system utilization.
- **Station Clusters:** Maps station groupings based on usage and location.

4.2 Frontend Pages

The following frontend pages were developed:

- **Homepage:** Provides an overview of the application's features (Figure 2).
- **Real-Time Data:** Displays live bike availability using a Leaflet.js map (Figure 3).
- **Usage Patterns:** Visualizes weekly trends, peak/off-peak hours, and station-level performance (Figure 4).
- **System Statistics:** Displays aggregated metrics and a detailed table of station metrics (Figure 5).
- **Station Clusters:** Visualizes clustered stations with color-coded markers (Figure 6).

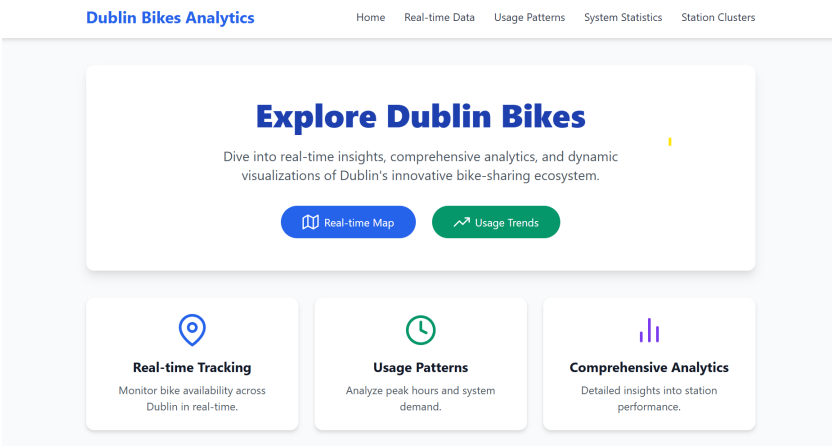

Figure 2: Homepage: Dublin Bikes Analytics

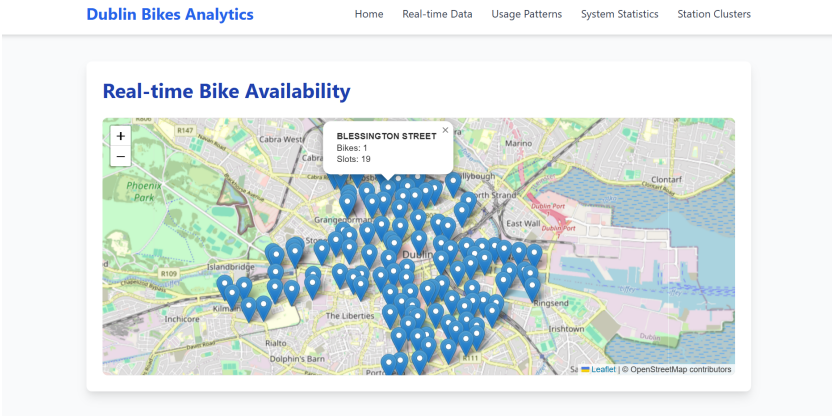

Figure 3: Real-Time Data Map

Figure 4: Usage Patterns Page: Weekly Trends and Performance

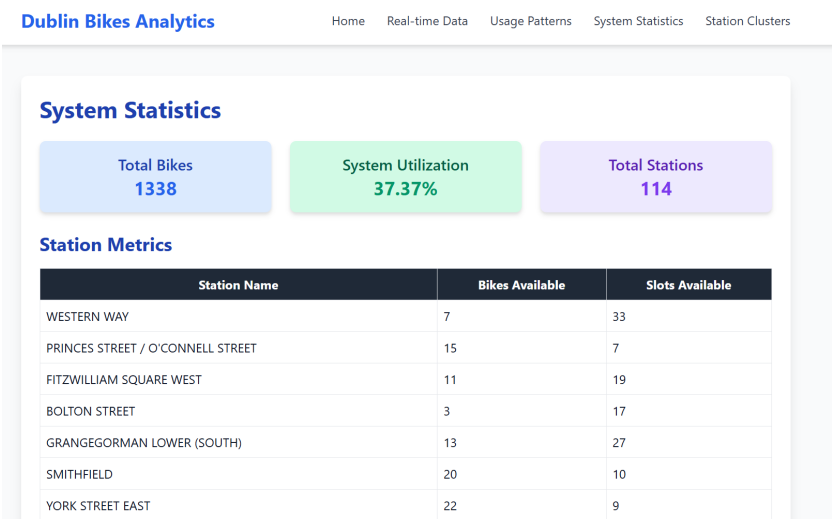

Figure 5: System Statistics Dashboard

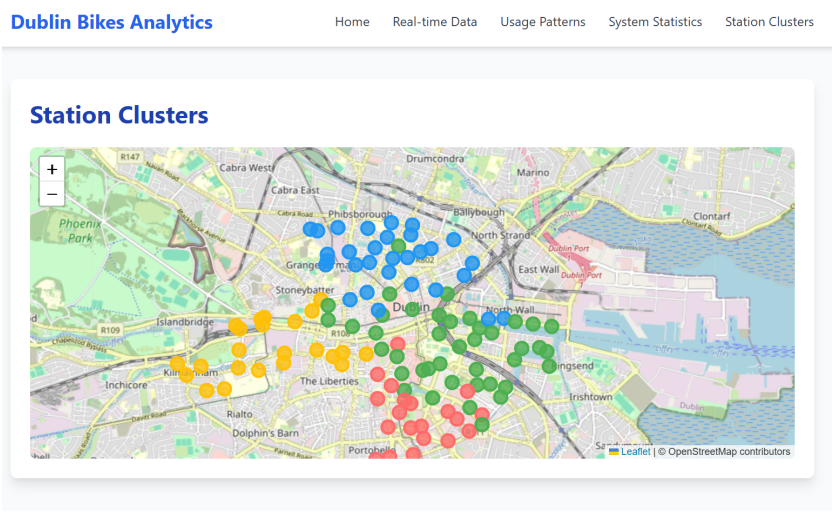

Figure 6: Station Clustering Visualization

5 Task 3: Extra Task

5.1 Motivation

To enhance the application, an advanced **Usage Patterns** feature was developed. This includes:

- Analysis of peak and off-peak usage hours.
- Weekly trends visualization using historical and real-time data.
- Identification of busiest and quietest stations to highlight performance variations.

5.2 Implementation

The `urban_computing_insights()` function powers the Usage Patterns page. Its implementation includes:

- Peak and off-peak hour identification using average hourly bike availability.
- Weekly trend visualization using Chart.js to show demand patterns.
- Station-level performance metrics, listing the busiest and quietest stations.

6 Challenges and Lessons Learned

Key challenges included:

- Ensuring real-time data consistency and synchronization with historical datasets.
- Handling missing or incomplete data during the merging process.
- Visualizing complex analytics intuitively for non-technical users.

Lessons learned:

- Modular code design simplifies debugging and future expansions.
- Effective data preprocessing is critical for ensuring analysis accuracy.
- Real-time visualizations require optimized backend processing to avoid delays.

7 Conclusion

The Dublin Bikes Analytics platform integrates real-time data, historical analysis, and advanced visualization techniques to provide actionable insights for urban mobility. Its features address key challenges in bike-sharing management, such as demand prediction, station clustering, and system utilization metrics.

Acknowledgment

This project utilized open-source datasets and tools such as Flask, Firebase, and CityBikes API. Special thanks to Trinity College Dublin for providing resources and guidance.

AI Tools Disclosure

The use made of AI tools in the preparation of this assignment is outlined below:

- ChatGPT (OpenAI): Assisted in generating sections of this report, refining text, and improving structure.