MAY 4, 2025

CITY BIKE SHARING NETWORK DASHBOARD

PYTHON-BASED INTERACTIVE WEB APPLICATION

OSAHDI YASHODHIKA
GOLD LEAF FINANCIAL GROUP INC.
2158 Armstrong Street, Sudbury, Ontario

Introduction

The City Bike Sharing Network Dashboard is a Python-based interactive web application that visualizes live data from the City Bike Network API. Built with **Streamlit**, this dashboard allows users to explore global bike-sharing networks, analyze station distributions, and generate customized PDF reports.

Overview

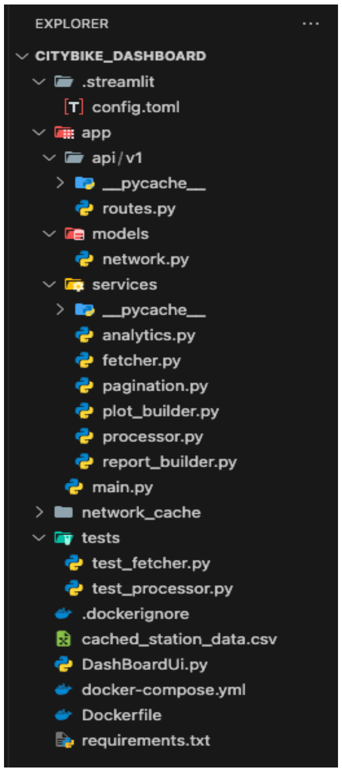
The application connects to the [CityBike API] (http://api.citybik.es/v2/networks and http://api.citybik.es/v2/networks/{network_id}) to fetch real-time data about bike-sharing networks across the world. It processes and visualizes this data to provide insights such as station counts per network, country-wise distribution, and top-performing regions.

Features

- Filter networks by name or country code
- Horizontal bar chart: Station counts per network
- Pie chart: Network distribution by country
- Summary statistics:
    - Total number of networks
    - Country with the highest number of stations
    - Network with the most stations
- PDF report generation
- Built-in error handling and feedback
- Docker support for easy deployment

Project Structure and File Descriptions

The City Bike Dashboard is structured using a modular and organized architecture. Each file is designed with a specific responsibility, making the codebase easier to maintain, extend, and debug. The application logic is cleanly separated into components for data fetching, processing, analytics, visualization, and reporting. Below is a detailed explanation of each major file and its role in the system:



```
EXPLORER ...
CITYBIKE_DASHBOARD
  .streamlit
    config.toml
  app
    api/v1
      __pycache__
      routes.py
    models
      network.py
    services
      __pycache__
      analytics.py
      fetcher.py
      pagination.py
      plot_builder.py
      processor.py
      report_builder.py
    main.py
  network_cache
  tests
    test_fetcher.py
    test_processor.py
  .dockerignore
  cached_station_data.csv
  DashBoardUi.py
  docker-compose.yml
  Dockerfile
  requirements.txt
```

Figure 1: Project Folder Structure

app/services/

- fetcher.py
  - Retrieves live bike-sharing data from the CityBike API.
  - Handles HTTP GET requests and structures the API response.
  - Includes error handling for failed requests or timeouts.
- processor.py
  - Processes raw API data into a clean, usable format.
  - Filters out incomplete entries and extracts key fields such as network name, city, country, station count, and coordinates.
  - Prepares data for visualizations and analysis.
- analytics.py
  - Computes analytical insights from the processed data.
  - Identifies key statistics such as:
    - Total number of networks
    - Country with the most stations
    - Network with the most stations
- plot_builder.py
  - Generates interactive charts using **Plotly**.
  - Builds horizontal bar charts and pie charts to visualize station counts and country-level distribution.
- report_builder.py
  - Constructs PDF reports with charts and metrics using **ReportLab** and **Matplotlib**.
  - Combines visual content and textual summaries into a downloadable file.

app/api/v1/

- routes.py
  - Placeholder for route definitions.

app/models/

- network.py
  - Optional module for defining data models.
  - Can be extended to manage structured schemas or integrate with databases.

tests/

- test_fetcher.py
  - Contains unit tests to verify API responses and error handling logic in fetcher.py.
- test_processor.py
  - Tests data cleaning, transformation, and edge case handling in processor.py.

Root Directory

- DashBoardUi.py
  - The main Streamlit UI entry point for the application.
  - Loads processed data, renders filters, visualizations, summary cards, and handles PDF generation.
- requirements.txt
  - Lists all Python libraries required to run the app.
- Dockerfile
  - Creates a container image for the app, installing dependencies and setting up the runtime environment.
- docker-compose.yml
  - Manages Docker container setup, networking, and execution using Docker Compose.
- cached_station_data.csv (in network_cache/)
  - Stores pre-fetched or backup station data for offline usage or testing scenarios.
- .streamlit/config.toml
  - Streamlit theme configuration file, used to customize layout and UI appearance.

This structured layout helps enforce separation of concerns, supports scalability, and ensures the project remains easy to understand and collaborate on, especially important in production or team environments.

Dependencies

This project relies on the following Python libraries, as specified in requirements.txt:

Library	Purpose
streamlit	Building the interactive dashboard UI
requests	Making HTTP requests to fetch data from the API
pandas	Data manipulation, aggregation, and filtering
plotly	Interactive visualizations (bar chart, pie chart)
matplotlib	Static chart rendering for PDF embedding
reportlab	Generating downloadable PDF reports
pillow	Image processing support used in PDF layouting
kaleido	Exporting Plotly charts as static images
pytest	Writing and running unit tests

Setup Instructions

This section explains how to set up and run the City Bike Network Dashboard locally or using Docker.

<u>Option 1: Run Locally (Recommended for Development)</u>

Step 1: Clone the Repository

Open terminal and run:

- git clone https://github.com/Balage-Oshadi/City-Bike-Network-Dashboard.git
- cd City-Bike-Network-Dashboard

Step 2: (Optional) Create and Activate a Virtual Environment

It is best practice to use a virtual environment to isolate project dependencies.

```
# Create a virtual environment named 'venv'
python -m venv venv

# Activate the virtual environment
# On macOS/Linux:
source venv/bin/activate

# On Windows:

venv\Scripts\activate
```

Step 3: Install Python Dependencies

Install all required packages listed in requirements.txt:

- pip install -r requirements.txt

Step 4: Run the Streamlit Application

Start the dashboard locally:

- streamlit run DashBoardUi.py

(Browser should automatically open the app at: http://localhost:8501)

Option 2: Run Using Docker (Recommended for Deployment)

If you prefer to run the application in a containerized environment, follow these steps.

Step 1: Make Sure Docker Is Installed

Install Docker Desktop from https://www.docker.com/products/docker-desktop if not already installed.

Step 2: Build and Run the Docker Container

From the root project directory, run:

- docker-compose up –build

(This will Build the Docker image, Start a container, Serve the Streamlit app on port 8501)

Step 3: Access the Dashboard

Open your browser and go to:

- http://localhost:8501

NOTE:

- To stop the app, press `Ctrl + C` in your terminal. If using Docker, stop the container with: docker-compose down.
- The app uses port `8501` by default. Ensure this port is available.

Implementation

The application follows a modular structure, where each component is responsible for a specific part of the data flow:

1. fetcher.py
   - Connects to the CityBike API and fetches raw JSON data about bike networks.
2. processor.py
   - Cleans, filters, and transforms raw data into structured DataFrames.
   - Ensures only complete, valid data is passed downstream.
3. analytics.py
   - Computes statistical summaries including:
     - Total number of networks
     - Country with the most stations
     - Network with the most stations
4. plot_builder.py
   - Builds interactive charts such as:
     - Horizontal bar chart of station counts
     - Pie chart for network distribution by country
5. report_builder.py
   - Generates a downloadable PDF containing charts and summary metrics.
6. DashBoardUi.py
   - Orchestrates the UI using Streamlit
   - Integrates filtering, chart rendering, analytics, and report generation

Exploratory Analysis (Colab)

Before developing the production-ready Streamlit dashboard, an initial exploratory data analysis (EDA) was performed in a Google Colab notebook to better understand the structure and behavior of the CityBike API.

Notebook Link:
https://colab.research.google.com/drive/1ouTfjPPlq0q2lGc0DnG8nrX3M7bednw4?usp=sharing

Key activities included:

- Fetching network-level data from the CityBike API
- Parsing and transforming nested JSON structures into normalized pandas DataFrames
- Extracting attributes like country, city, coordinates, and available slots
- Aggregating networks by country for comparative insights
- Generating visual summaries (e.g., top 10 countries by network count)

This EDA served as the foundation for designing the final schema used in the dashboard.

User Guide

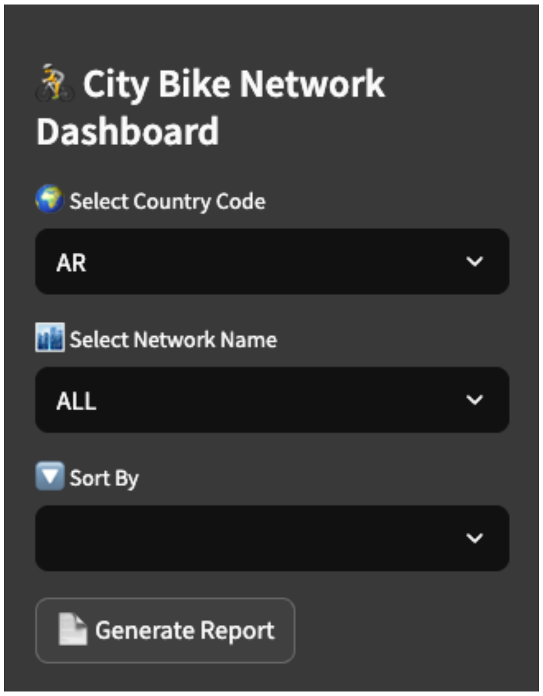
The City Bike Sharing Dashboard provides an intuitive interface for users to interact with real-time data and generate insights. Below is a guide on how to use the application effectively:

Filter Panel & Report Generation

The sidebar panel allows users to interactively filter the dashboard content and generate customized reports. Users can:

- Select a country code to filter networks by region.
- Choose a specific bike network or view all.
- Sort the data by attributes such as network name, city, or station count.
- Click the "Generate Report" button to export a PDF summary based on the selected filters.

This user-friendly panel ensures intuitive navigation and empowers users to explore the data on demand.



Select Country Code: AR

Select Network Name: ALL

Sort By:

Figure 2: Sidebar filters in the City Bike Network Dashboard (enables users to refine data and generate custom reports.)

<u>Global Summary Metrics</u>

This dashboard section provides high-level KPIs (Key Performance Indicators) to give users an immediate overview of the global bike-sharing landscape. It includes:

- **Total Networks**: The total number of bike-sharing networks currently available worldwide.
- **Top Country**: The country with the highest number of stations (e.g., Japan - JP).
- **Top Network**: The most widespread network by station count (e.g., YouBike).
- **Total Stations**: The sum of all active bike stations across the CityBike API dataset.

These summary cards offer quick insights and serve as a data snapshot on application load.

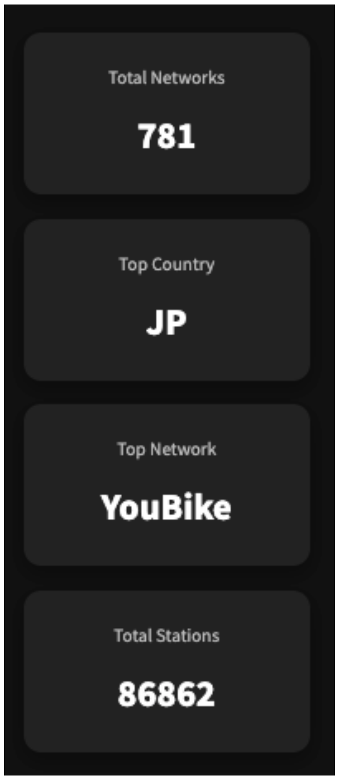

Total Networks
781
Top Country
JP
Top Network
YouBike
Total Stations
86862

Figure 3: Key summary metrics including total networks, top country, top network, and total station count.

Global Bike Network Overview & Insights

The horizontal bar chart highlights the top 10 countries by bike network count, with Italy leading (129 networks), followed by Germany, Spain, and France. This offers insight into regions with dense bike-sharing infrastructure. The adjacent donut chart shows the top 10 networks by station count, where Nextbike holds the largest share, followed by HELLO CYCLING Tokyo and YouBike. This reveals the scale and reach of dominant global operators.

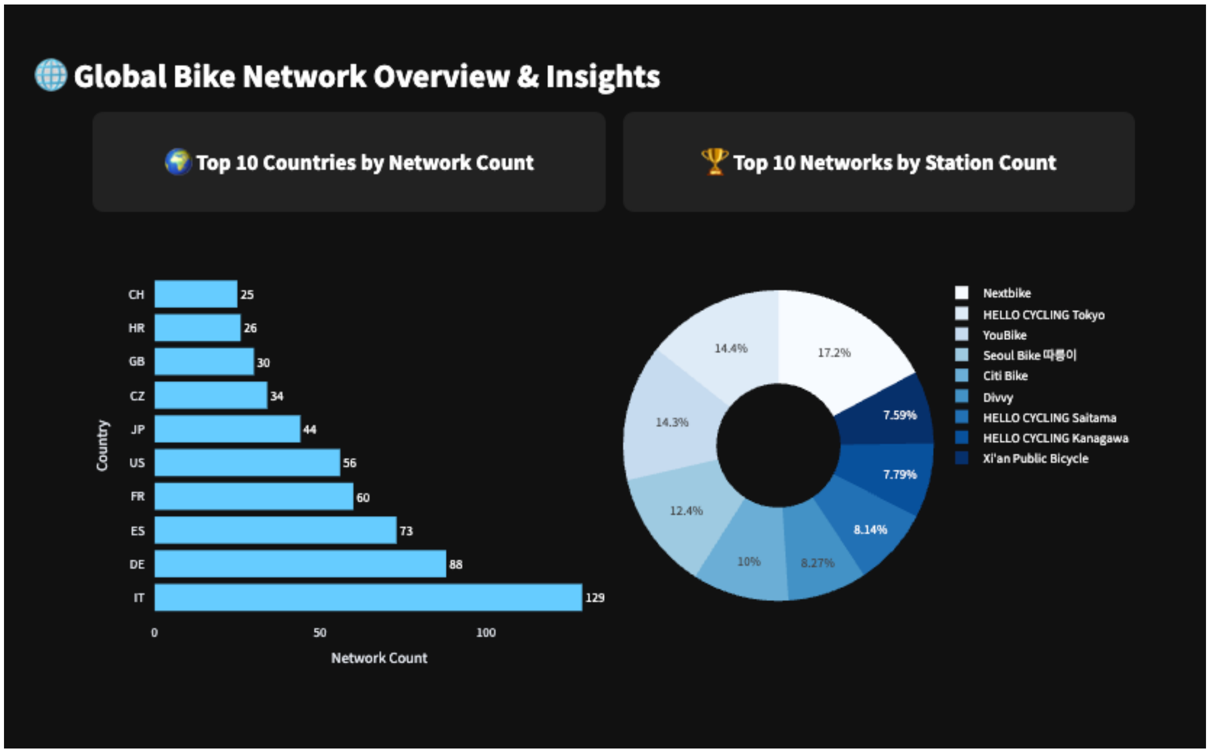
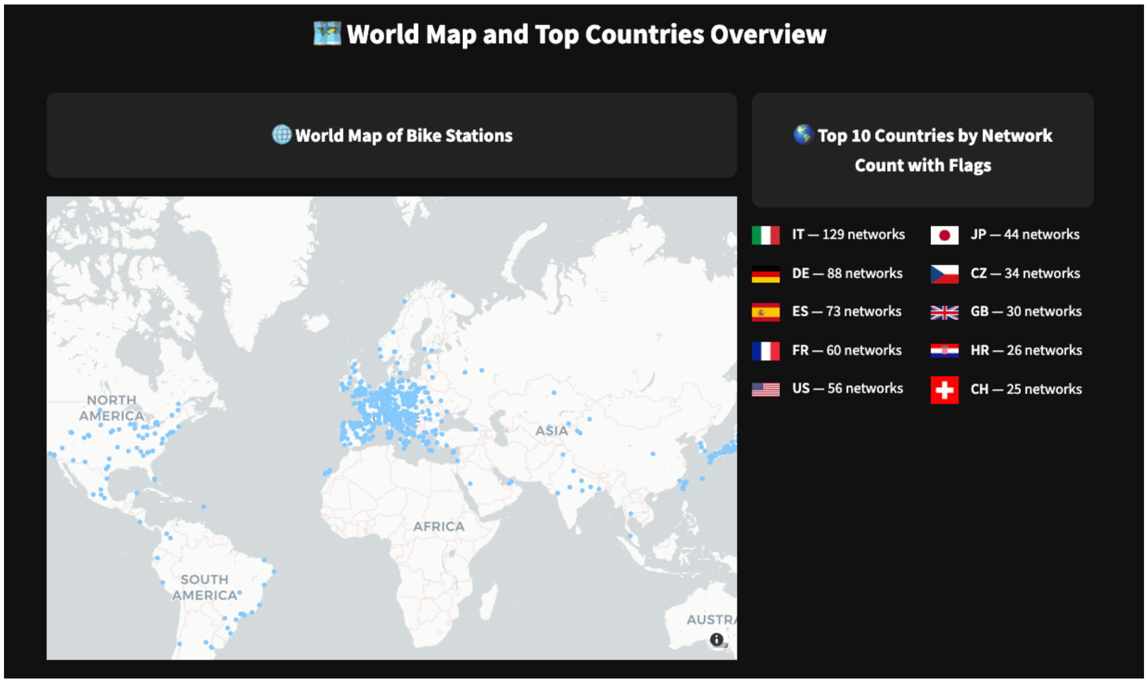


Figure 4: Dashboard view showing top 10 countries by network count (left) and top 10 networks by station count (right).

World Map and Top Countries Overview

This visualization displays a global map of bike station locations alongside a ranked list of the top 10 countries by network count. The interactive map highlights the global distribution of bike-sharing networks, with notable concentrations in Europe, North America, and parts of Asia. The accompanying list, enhanced with country flags, shows that Italy leads with 129 networks, followed by Germany, Spain, and France. This view provides both geographic and quantitative context, helping users quickly identify global hotspots for bike-sharing infrastructure.

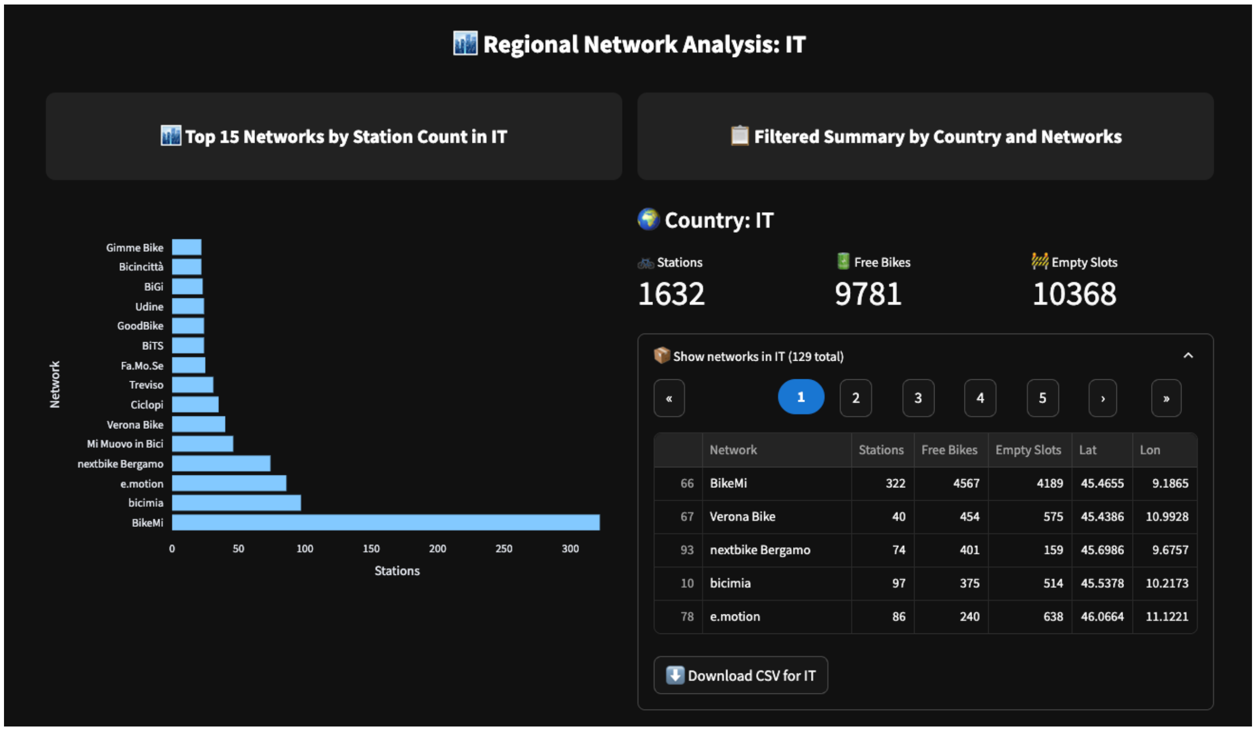

🗺️ World Map and Top Countries Overview
🌐 World Map of Bike Stations
🌎 Top 10 Countries by Network Count with Flags
NORTH AMERICA
SOUTH AMERICA
AFRICA
ASIA
AUSTRA

IT — 129 networks	JP — 44 networks
DE — 88 networks	CZ — 34 networks
ES — 73 networks	GB — 30 networks
FR — 60 networks	HR — 26 networks
US — 56 networks	CH — 25 networks

Figure 5: Global map of bike stations with the top 10 countries by network count displayed alongside.

Regional Network Analysis

This section provides a focused view of Italy's bike-sharing ecosystem, the country with the highest number of networks. On the left, a horizontal bar chart ranks the top 15 networks in Italy by total station count, with *BikeMi* leading at over 300 stations. On the right, a real-time summary displays key metrics such as the total number of stations (1632), available free bikes (9781), and empty slots (10368). Below the summary, a searchable and paginated data table shows granular information per network, including station count, real-time bike availability, and GPS coordinates. A CSV export option enables users to download filtered data for further offline analysis.



Regional Network Analysis: IT

Filtered Summary by Country and Networks

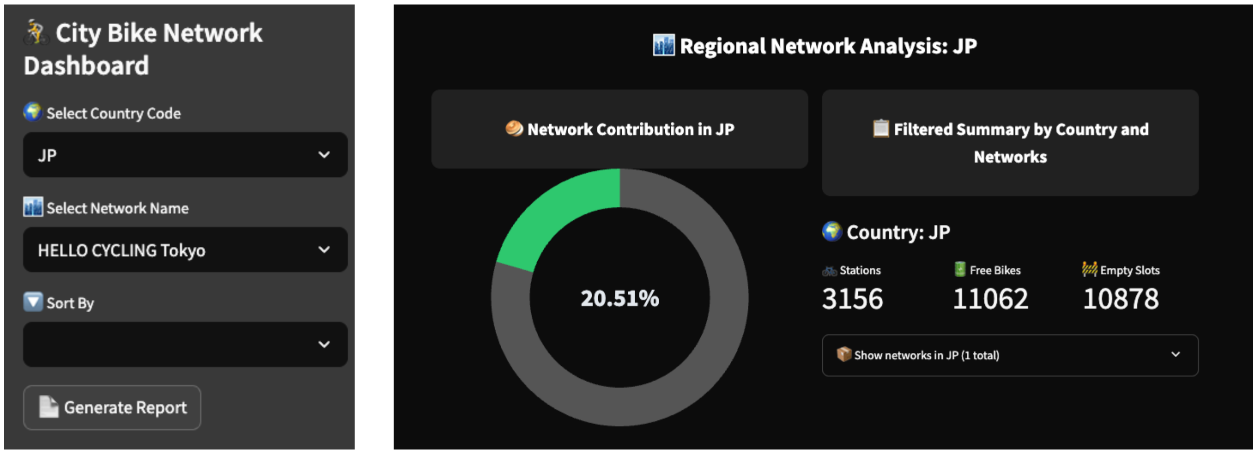
Country: IT

Stations	Free Bikes	Empty Slots
1632	9781	10368

	Network	Stations	Free Bikes	Empty Slots	Lat	Lon
66	BikeMi	322	4567	4189	45.4655	9.1865
67	Verona Bike	40	454	575	45.4386	10.9928
93	nextbike Bergamo	74	401	159	45.6986	9.6757
10	bicimia	97	375	514	45.5378	10.2173
78	e.motion	86	240	638	46.0664	11.1221

Figure 6: Top 15 networks by station count in filtered country, with real-time summary metrics and data export.

This view is displayed when a user selects both a **country code** (e.g., JP) and a **specific network** (e.g., HELLO CYCLING Tokyo) from the sidebar filters. The dashboard then narrows down the data to only that network within the selected country. All visualizations and summary statistics are updated accordingly, enabling users to analyze a single network's contribution in a localized context.



City Bike Network Dashboard

Select Country Code: JP

Select Network Name: HELLO CYCLING Tokyo

Sort By

Regional Network Analysis: JP

Filtered Summary by Country and Networks

Country: JP

Stations	Free Bikes	Empty Slots
3156	11062	10878

Figure 7: Sidebar filters showing country (JP) and specific network (HELLO CYCLING Tokyo) selected.

Error Handling

The application incorporates robust error handling to ensure a smooth user experience, even in the case of data inconsistencies or service interruptions. Key error handling strategies include:

- **API Request Failures:**
  If the CityBike API is unreachable or returns an invalid response, a clear warning message is displayed using Streamlit's st.warning(), informing users of the issue without breaking the application.

- **Missing or Malformed Data:**
  Records with incomplete fields (e.g., missing latitude, longitude, or station counts) are automatically filtered out during the data processing stage to prevent visualization or analytics errors.

- **Visualization Safeguards:**
  Charts and maps are only rendered when valid data is available. If no data matches the current filters, the app gracefully notifies the user instead of rendering empty or broken visuals.

- **PDF Report Generation:**
  The report generation process is wrapped in a try/except block to catch and display any errors that may occur (e.g., plotting issues, file permission errors). Users are notified with an st.error() message if the report fails to generate.

These measures ensure that the dashboard remains stable and user-friendly under various failure scenarios.

Deployment

The application is containerized using Docker and deployed publicly via Streamlit Cloud, enabling easy access without requiring local setup or manual dependency installation.

Live Dashboard: https://city-bike-network-dashboard.streamlit.app/

Deployment Steps (GitHub → Streamlit Cloud)

To deploy the dashboard using Streamlit Cloud, follow these steps:

1. **Push the Project to GitHub**
   Ensure your complete project is committed and pushed to a public or private GitHub repository.
2. **Log in to Streamlit Cloud**
   Navigate to https://streamlit.io/cloud and sign in using your GitHub credentials.
3. **Create a New App**
   - Click **"New App"**
   - Select the GitHub repository that contains your project
   - Choose the appropriate branch
   - Set the main file path to DashBoardUi.py
4. **Deploy the Application**
   Click **"Deploy"**. Streamlit Cloud will automatically:
   - Install dependencies listed in requirements.txt
   - Build the app environment
   - Launch the application at a public URL

Once deployed, the dashboard will be live and accessible from any device via the generated link.

Enhancements

To improve the dashboard's functionality, usability, and accessibility, several enhancements have been implemented. These additions provide richer visual insights and ensure smooth user interaction across platforms.

<u>Implemented Features</u>

- Donut Chart for Country-Wise Distribution
    - Displays the percentage share of networks by country, offering a clear comparative view.
- Downloadable PDF Reports.
    - Enables users to export key metrics and visualizations into a professional, shareable format.
- Docker-Based Deployment.
    - Ensures environment consistency and simplifies local or production setup via containerization.

<u>Planned Enhancements.</u>

- Interactive Map Visualization
    - Introduce dynamic maps with zooming, panning, and clustering features to explore bike station locations geographically.
- Responsive Design for Mobile & Tablets
    - Optimize layout and UI elements for smaller screens, ensuring usability on all devices.
- Additional Graphical Insights
    - Add more advanced chart types (e.g., line charts, heatmaps, time-series graphs) to provide deeper analytical perspectives.
- "Sort By" Functionality in Tables
    - Allow users to sort data tables dynamically by attributes like station count, free bikes, or empty slots—enhancing usability and data navigation.