Iteration 3

Mitul Nakrani, Priyanka Maan

1 Dataset Description

The dataset selected for this project is sourced from the Massachusetts Bay Transportation Authority (MBTA) GTFS feed:

`https://www.mbta.com/developers/gtfs`

This comprehensive GTFS (General Transit Feed Specification) feed provides scheduled transit information for the MBTA network, including buses, subways, and commuter rails. The dataset ZIP archive contains several standardized CSV files:

- `agency.txt`: Agency-level metadata such as name, URL, and timezone.
- `routes.txt`: Transit route information (IDs, names, route types).
- `trips.txt`: Mappings of routes to specific trips, including service and direction.
- `stop_times.txt`: Detailed sequence of stops per trip with scheduled arrival/departure times.
- `stops.txt`: Geographic coordinates, names, and IDs of stops and stations.
- `calendar.txt` and `calendar_dates.txt`: Operating schedules and exceptions.

This structured, schedule-based dataset is ideal for analyzing transit performance metrics such as on-time reliability, route frequency, and stop-level usage. The dataset has standardized CSV format that ensures easy ingestion into MySQL.

The dataset's rich structure makes it highly suitable for analysis of transit efficiency, on-time performance, and passenger service coverage. It allows to simulate and visualize operations, identify bottlenecks, and propose data-driven optimizations, directly aligning with the goals of SmartTransit.

2 Tools and Methodologies

The SmartTransit system will be developed using the following tools and technologies:

- **Python**: The core programming language for data processing, backend logic, and chatbot integration.
- **Streamlit**: A lightweight, rapid development framework for building the interactive web dashboard
- **Plotly**: Used for creating rich, interactive visualizations such as dynamic line charts and heatmaps.
- **MySQL**: The primary database for storing cleaned and structured transit data, chosen for its reliability and support for complex queries.
- **Pandas and NumPy**: For data aggregation, and basic statistical analysis.

- **Git + GitHub**: For version control, team collaboration, and codebase management throughout the project.
- **OpenAI LLM (Large Language Model)**: Integrated into the dashboard as a chatbot assistant to provide users with natural language responses to transit-related queries (e.g., "Which route is the busiest in the morning?").

These tools were chosen to ensure fast prototyping, ease of integration, support for interactive analytics, and modern user engagement through conversational AI. All together, these provide a robust foundation for delivering an impactful, data-driven transit analysis platform.

3 Preliminary Timeline

- **Week 1:** Project kickoff. Set up GitHub repository and branching strategy. Configure development environment (Python, Streamlit, MySQL). Acquire the dataset and perform exploratory data analysis to understand data structure and quality.
- **Week 2:** Design and implement normalized MySQL database schema. Write scripts to ingest and clean the data. Load data into the database and validate integrity with sample queries.
- **Week 3:** Develop core Python data processing functions to support dashboard queries (e.g., stop search, route statistics, delay calculations). Test MySQL queries for performance and accuracy.
- **Week 4:** Build initial Streamlit dashboard structure. Connect dashboard components to Python data functions. Display basic static summaries (e.g., route lists, stop locations).
- **Week 5:** Integrate Plotly to create interactive visualizations (e.g., line charts for delays, heatmaps for stop density, maps for routes). Add filtering controls (e.g., date, route, time of day).
- **Week 6:** Integrate OpenAI LLM as a chatbot assistant within the dashboard. Allow users to query transit data insights in natural language. Begin usability testing of the chatbot interface.
- **Week 7:** Conduct comprehensive testing (unit, integration, user acceptance). Refine chatbot responses and visualization interactivity based on feedback. Address performance bottlenecks.
- **Week 8:** Complete project documentation, including user manual, system architecture, and code comments. Prepare final presentation and demo of the system.

4 References

1. Massachusetts Bay Transportation Authority (MBTA) GTFS Feed: `https://www.mbta.com/developers/gtfs`
2. MBTA GTFS Documentation and Resources: `https://www.mbta.com/developers`
3. Streamlit Documentation: `https://docs.streamlit.io`
4. Plotly Python Graphing Library: `https://plotly.com/python/`
5. MySQL Documentation: `https://dev.mysql.com/doc/`
6. OpenAI API Documentation (for LLM integration): `https://platform.openai.com/docs`
7. GitHub Documentation (Collaboration and Version Control): `https://docs.github.com/en`