Software Requirements Specification For Centro Bus Predictor

Prepared by

Team E

(Diya Patel, Emma Halsey, Niva Pradhan, Vandan Patel)

Table of Contents

1. Introduction

1.1 Purpose

This document defines the software requirements for the Bus Prediction System for Centro, the public transportation service in Central New York. The system aims to collect, store, and analyze real-time bus tracking data from the Centro API to improve schedule accuracy predictions. The data will be stored in a MySQL database and accessible through a web interface that allows users to search, filter, and obtain bus route information and analyze schedule accuracy.

1.2 Document Conventions

This document follows standard Software Requirements Specification (SRS) guidelines, using a hierarchical numbering system (e.g., 1, 1.1, 1.2) for clarity. Key terms are bolded, while italics are used for emphasis. All requirements are stated explicitly without priority levels, and external references are cited with hyperlinks where applicable.

1.3 Intended Audience and Reading Suggestions

This document is intended for the following stakeholders:

- **Developers** – To understand system architecture, API integration, and database design.
- **Project Managers** – To track system scope, features, and constraints.
- **Testers** – To validate system requirements and expected behaviors.
- **Users & Administrators** – To understand system functionality and available features.

1.4 Product Scope

The Bus Prediction System is designed to enhance public transportation reliability by analyzing historical and real-time Centro bus tracking data. The primary objectives of the system are:

- **Accurate Bus Arrival Predictions** – Provide insights into schedule adherence and delay patterns.

- **Data Storage & Processing** – Store structured route and timing data in a MySQL database for analysis.

- **User-Friendly Web Interface** – Enable users to search and filter routes, dates, and analyze prediction results.

- **Extended Data Analysis** – Allow integration of additional data sources (e.g., weather conditions) for further accuracy improvements.

This project supports Centro's goal of improving public transit efficiency by leveraging real-time data insights for better decision-making.

1.5 References

The following documents and resources were referenced while preparing this SRS:

1. **SRS-template-ieee**
2. **Centro API Documentation** – Available at Centro Official Website.
3. **MySQL Documentation** – Available at MySQL Official Docs.
4. **Project Instructor Meetings & Notes** – Internal documentation from discussions with stakeholders.

2. Overall Description

2.1 Product Perspective

This product, the Centro Bus Tracker, is a predictive model of bus services offered by Centro and tailored to SUNY Oswego residents.

2.2 Product Functions

This product must perform the following major functions, as defined in detail in Section 3:

- The Web-based Interface will allow users to search, filter, and select data based on parameters including, but not limited to, route, date, and time.

- The system shall display the estimated arrival time (ETA) for a particular stop on a specific route, as defined by the user. Alongside the ETA, the system shall display the scheduled arrival time for comparison.

- The system will compare average travel times among various routes to the same or nearby (less than 0.5 miles) location and propose the quickest route to the user.
- The system shall provide users with a real-time map that displays the current position of the bus on the route.
- The interface will feature histograms and statistical analysis to help users understand bus arrival patterns, wait times, and other key transit metrics.

2.3 User Classes and Characteristics

Administrator

Important to document and outline the system for their navigation. Will possess high technical expertise, and access the system on occasion. Requires full access to the database, admin menus, and agent, connecting to the system remotely or through a direct connection.

End User

Important to meet their needs to satisfaction. Technical expertise can vary, but system usage will be frequent. Interacts with the User Interface only and connects to the site remotely through internet access.

2.4 Operating Environment

The system will operate on SUNY Oswego's Computer Science Department server, pi.cs.oswego.edu. It will rely on *Centro's* API for data, and store the collected data in a MySQL database that also runs on the pi.cs.oswego.edu server. The data will be collected from the API with the help of *Postman*, an API platform.

Stretch Goals:

If weather is implemented as a predictive factor for the ETA, the system will use *Accuweather's* API to collect Oswego weather data.

If the live map is implemented, the system will host the map from *Centro's* own website or use Google Maps' API.

2.5 Design and Implementation Constraints

- This project must be completed within 15 weeks.
- The collected data must be stored in a local MySQL database on the pi.cs.oswego.edu server. The server has a limited amount of space.
- The Centro API has a *rate limit* of requests per day.

- The system will use MySQL to query data from Centro's existing API regularly while respecting the API's data limits.
- API keys must be secured and hidden from the user interface.
- ETAs will automatically be recalculated for further accuracy after each API pull.

2.7 Assumptions and Dependencies

This project operates on the assumption that the pi.cs.oswego.edu server has the capacity to store the amount of data necessary. Additionally, the timely completion of this project is dependent on Postman's ability to automate data collection and storage from Centro's API to the MySQL database while meeting the *rate limit.*

3. External Interface Requirements

The system will interact with users, external data sources, and a database hosted on the school's **pi server** to provide bus tracking, historical data storage, and delay predictions.

3.1 User Interaction & Web Interface

- A web-based interface will allow users to:
  - Search for bus routes and stops.
  - View real-time bus locations and check estimated arrival times.
  - Access historical data on past bus delays and schedule adherence.
- Users will have the ability to filter buses based on **route number, time of day, and day of the week** to view detailed statistics and live updates.

3.2 System Communication & Data Handling

- The system will retrieve real-time bus data from the **Centro API**. The data will include:
  - **Bus ID**: Unique identifier for each bus.
  - **Route ID**: Identifier for the bus route.
  - **Current Location**: GPS coordinates (latitude and longitude).
  - **Scheduled Arrival Time**: Time the bus is expected to arrive at the next stop.
  - **Actual Arrival Time**: Time when the bus actually arrives at the stop.
  - **Bus Status**: In-transit, delayed, or on-time status.
- The database will be saved on the **pi server** account created for the purpose of this project and will be accessible to all team members.
- The system will use this data to:
  - Update real-time bus positions on the map interface.

    - Store arrival times and delays in the **MySQL database** hosted on the **pi server** for historical analysis.
    - Calculate average delays for each route and time of day to make predictions about future arrival times.
- The backend, developed using **Python (Flask or FastAPI)**, will handle:
    - Fetching data from the Centro API at intervals of **30-60 seconds**.
    - Processing and cleaning data before storing it in the database.
    - Performing data analysis to calculate average delays and generate predictions.
- RESTful API calls over **HTTPS** will ensure secure communication between the frontend, backend, database, and external APIs.
- Data caching mechanisms will be implemented to minimize unnecessary API calls and reduce server load.

4. System Features

The system will provide essential functionalities to help users plan their bus trips effectively by combining **real-time tracking**, **historical analysis**, and **predictive insights**.

4.1 Live Bus Tracking

- Fetches real-time bus locations from the **Centro API** and updates every **30-60 seconds**.
- Provides live updates on bus status (e.g., on-time, delayed, or canceled).

4.2 Historical Data Analysis

- Stores past arrival times and delays in a **MySQL database** hosted on the **pi server**.
- Allows users to check the schedule accuracy of buses by comparing the scheduled arrival times with the actual arrival times.
- Provides filters to analyze data based on **route number, time of day, day of the week**, and **specific date ranges**.
- Displays visual statistics like graphs and charts (e.g., average delays for each route over time).

4.3 Arrival Time Predictions

- Uses a simple **average delay calculation algorithm** to predict future bus delays:
    - Calculates the average delay for each route based on historical data (e.g., the **average delay for Bus 46 on weekdays between 8 AM - 9 AM**).
    - Predicts arrival times by adding the average delay to the scheduled arrival time.
- Optional enhancement: Factor in external conditions such as **weather data** using an API like **OpenWeather/Accuweather**.
- Displays the predicted arrival time on the user interface along with the real-time status.

- Updates predictions dynamically based on the most recent data from the Centro API.

4.4 Search & Filtering

- Allows users to search for bus routes by **route number, stop name, or destination**.
- Filters buses based on **time of day, route**, and **status** (on-time, delayed).
- Displays upcoming scheduled arrivals and predicted delays for selected routes.
- Supports custom filters for power users (e.g., finding the most delayed routes).

4.5 API Rate Limit Handling

- Tracks the number of API requests made to ensure compliance with **Centro's API rate limits**.
- Implements **caching** strategies to avoid redundant API calls by storing frequently requested data temporarily.
- Reduces unnecessary server load by serving cached data to users when appropriate.
- Automatically adjusts the data polling frequency if API rate limits are close to being exceeded.

5. Other Non-Functional Requirements

5.1 Safety Requirements

- Ensure that any user data collected is protected and not shared without consent.

- Clear disclaimers stating that arrival predictions may vary and users should account for possible delays.

5.2 Security Requirements

- Error messages should not reveal sensitive system details to prevent exploitation.

- Only authorized users and system components should have access to sensitive data and core functionalities.

- Historical data must be stored securely in MySQL and be accessible for analysis, without violating any privacy policies.

- The API keys and data should not be visible to the user.

5.3 Software Quality Attributes

- The software should have a user-friendly interface, allowing easy interaction for people checking bus arrival times.

- The system will only track buses within the Oswego area that include the SUNY Oswego campus center as a stop and served by Centro.
- The system should be able to make quick predictions and handle requests efficiently, even with multiple users interacting at once.
- The system must predict the bus arrival time based on real-time data fetched from Centro's API as well as the historical database.
- If the API fails to provide data, the system should display a relevant error message.

6. Other Requirements

6.1 Accessibility for Users

The system's user interface will be compatible with accessibility needs and technology, including assistive reading technology and simple formatting.

7. Appendix A: To Be Determined

1. The *rate limit* for the Centro API requests.