SOFTWARE REQUIREMENTS SPECIFICATION

for

CENTRO BUS PREDICTOR

Version 1.1

Prepared by: Canol Onar, Jack Gallagher, Noah Mintzer, James Clough, Nikki Knecht

SUNY Oswego Software Engineering

March 2, 2025

Contents

1 Introduction

1.1 Purpose

Passengers who are waiting to board a bus would like to know when it would arrive at their current stop. The goal of this Centro Bus Predictor product is to predict when a Centro bus will arrive at their stop, given the name of the stop and the direction of the route.

1.2 Intended Audience and Reading Suggestions

This SRS is intended for developers and stakeholders of the product. This document should be read in order, after reading the Clever Devices BusTime®Developer API Version 3 Guide.

1.3 Project Scope

The Centro Bus Predictor is intended for passengers waiting to board a bus. The goal of the product is to collect and analyze historical data for bus time arrivals at bus stops. The benefit of this in turn is to provide more accurate predictions to passengers using the application.
This SRS encompasses the Centro Bus Predictor system in its entirety. The system boundary consists of the users of the Centro Bus Predictor, passengers on Centro buses, HTTP requests to the API, and the physical devices in each bus, such as GPS, that provide data to the API. In addition, the context consists of the buses and bus drivers. Finally, the context boundary consists of the factors leading to changes in bus predictions: traffic, weather, etc.

1.4 References

- Clever Devices BusTime®Developer API Version 3 Guide
- MySQL 8.4 Reference Manual
- Pomegranate Documentation

2 Overall Description

2.1 Product Perspective

The Centro Bus Predictor project is a new and self-contained product.

2.2 Product Functions

The product will predict the time that a bus will arrive at a stop, along a certain route. The product will be a system and web application that retrieves live data (bus positions, predicted arrival times, etc.) from the API for the system to use and produce predictions. The application will have data stored in a database and process it to predict upcoming bus arrival times at various stops. The site will offer accessibility on a wide variety of platforms including smartphones, tablets, and laptops.

2.3 User Classes and Characteristics

- Passenger Actor: A passenger who is waiting to board a bus and wants to know when it will arrive at a given stop.

2.4 Operating Environment

- The database will be a MySQL database.
- The database will be on pi.cs.oswego.edu, a remote Linux-based server.
- The webpage will be displayed on phone or laptop.

2.5 Design and Implementation Constraints

- The team shall complete the system in 15 weeks.
- The team will consist of five team members.
- The team will have one stakeholder other than the developers.
- The team will have a budget of $0.

2.6 User Documentation

- The webpage shall have a user help page accessible by a link on the website.
- The help page shall contain FAQs regarding the usage of the webpage.

2.7 Assumptions and Dependencies

- The API will provide reliable and recent transit data as specified.
- Stable internet connectivity should be available for regular API polling.
- Users should have access to standard web browsers on their devices.

3 External Interface Requirements

3.1 User Interfaces

- The web application shall display a map image showing one or more routes.
- The web application shall display each stop on a route as an interactable button.
- The webpage shall be written in HTML, CSS, and JavaScript.

3.2 Hardware Interfaces

- The system shall display the webpage on a physical device, including a phone and laptop.

3.3 Software Interfaces

- The system will make getvehicles?, getpatterns?, getstops?, getdirections?,and getpredictions?, HTTP requests to the API.
- The system will retrieve the API responses in JSON format.
- The system will extract predicted time, vehicle id, pattern id, stop id, timestamp, and route fields from the responses.
- The system will store bus stop name and id in a dictionary for all stops along a route.
- The system will keep its own time clock in Python.
- The system will store field information from responses and time data using its time clock in tables in a MySQL database.
- The system will use the MySQL Connector or PyMySQL library to retrieve data from the MySQL database to use it in Python for data analysis.

3.4 Communications Interfaces

- The system will continuously poll the Centro API every second using HTTP data feed requests.

4 System Features

4.1 Predict Bus Arrival Based on Current Location

4.1.1 Description and Priority

The system shall predict when a bus will arrive at a particular stop on a route based on its current location. It will do so through statistical analyses of historical data. It is of high priority that this feature be executed.

4.1.2 Stimulus/Response Sequences

- A user will enter the route and stop if they wish to get on the bus.
- The system will identify the route and bus based on the user entry.
- The system will identify where the bus is currently based on Centro GPS data.
- The system will predict when the bus will arrive at the specified stop based on its current location and previous GPS data analyses.
- The system will display to the user the predicted time of arrival.

4.1.3 Functional Requirements

1. The system will provide a user interface to input stop and route information.
2. The system shall interact with the API using a Python script running continuously as a background process.
3. The system shall use previous statistical analyses regarding bus arrival times for predictions.
4. The system should be able to compute a predicted time of arrival at a stop.
5. The system will display the predicted time of arrival to the user.
6. The system shall display predictions to the user in a legible manner.
7. The system shall return the bus time provided by Centro if a prediction cannot be made.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- The webpage must be able to update its relevant information regarding bus time prediction every second due to the additional poll. This allows users to access the most updated data.

5.2 Safety Requirements

- If the system fails, the user should be warned that the prediction might not be correct.

5.3 Software Quality Attributes

- The API will be limited to 10,000 requests per day.
- The system should prevent invalid user inputs by providing a selection of routes and stops.