SOFTWARE REQUIREMENTS SPECIFICATION

for

CENTRO BUS PREDICTOR

Version 2.1

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

SUNY Oswego Software Engineering

April 9, 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, the name of the route 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 arrival time data at bus stops, ultimately providing a prediction to the user from their input. The benefit of this in turn is to provide more accurate predictions to passengers using the application.

This specification 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, such as 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 be a web application that retrieves live data from the API for the system to use and produce predictions. The application will store this data in a database and, provided stop, route, and direction, analyze it to predict upcoming bus arrival times.

2.3 User Classes and Characteristics

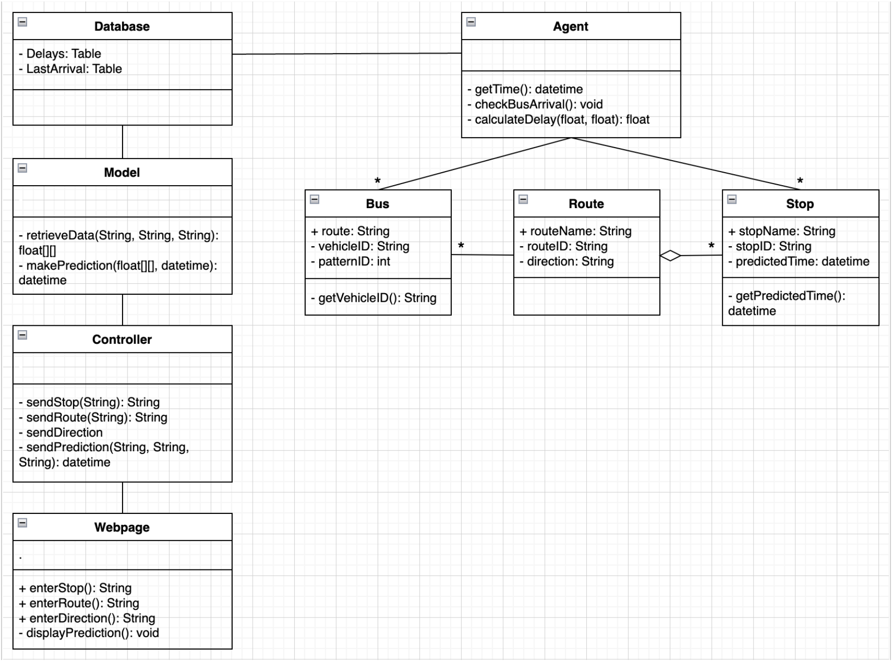
Database
- Delays: Table
- LastArrival: Table
Agent
- getTime(): datetime
- checkBusArrival(): void
- calculateDelay(float, float): float
Model
- retrieveData(String, String, String): float[][]
- makePrediction(float[][], datetime): datetime
Controller
- sendStop(String): String
- sendRoute(String): String
- sendDirection
- sendPrediction(String, String, String): datetime
Webpage
.
+ enterStop(): String
+ enterRoute(): String
+ enterDirection(): String
- displayPrediction(): void
Bus
+ route: String
- vehicleID: String
- patternID: int
- getVehicleID(): String
Route
+ routeName: String
- routeID: String
- direction: String
Stop
+ stopName: String
- stopID: String
- predictedTime: datetime
- getPredictedTime(): datetime
*
*
*
*

Figure 2.1: This is a UML Class Diagram showing how all of the classes interact with each other.

- The system will have a passenger actor, who is a passenger who is waiting to board a bus and wants to know when it will arrive at a given stop.

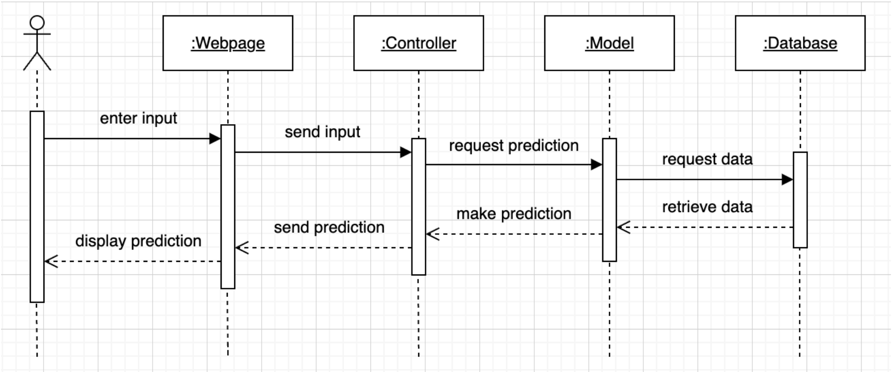

Figure 2.2: This is a UML Sequence Diagram showing how the Passenger Actor interacts with the system and is displayed a prediction.

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 agent will run in a tmux session on pi.cs.oswego.edu.

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, such as a phone, tablet, or 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 vehicle id, stop name, stop ID, route, direction, and predicted time fields from the JSON dictionaries.
- The system will use the datetime module in Python to keep track of the current time.
- The system will compute a delay upon bus arrival to a stop by subtracting the predicted time from current time.
- The system will store the field information from the dictionaries and delay together as a row in a MySQL database.
- The system will use the MySQL Connector library to put data in and retrieve data from the MySQL database.

3.4 Communications Interfaces

- The system will continuously poll the Centro API every five seconds 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 analysis of historical data. It is of high priority that this feature be executed.

4.1.2 Stimulus/Response Sequences

- A user will enter the stop, route, and direction if they wish to get on the bus.
- The system will identify the bus based on the user entry.
- The system will predict when the bus will arrive at the specified stop based on its delay to its most recent stop and historical data along its current route and direction.
- 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, route, and direction information.
2. The user interface should be straightforward for the user to interact with.
3. The system shall interact with the API using an agent, which is a Python script running continuously in a tmux session.
4. The system shall perform statistical analysis on historical bus arrival data based on the user's provided input.
5. The system should be able to compute a predicted time based on the statistical analysis.
6. The system shall display the prediction 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 five seconds due to the additional poll. This allows users to view the most recent prediction.

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 input by providing a selection of routes and stops.