User Manual

GCPS Bus Monitoring Through Kafka – Team 2

Sarah Fashinasi, Amali McHie, Tyler Hood, Alex Baker, Jeffrey Sanderson

Version 1.0

12 / 2 / 2024

Table of Contents

1. Introduction

This application enables users to monitor bus locations and speeds in real-time via an interactive map. The system features a backend for simulating and processing bus data, and a frontend for visualizing the information. This manual explains the technical setup, backend services, and frontend user interface, helping both technical and non-technical users utilize the system effectively. Users can reference this document to learn how to interact with the features of this system and train themselves on these features.

2. System Requirements

- Operating System: RedHat9 Linux Server (or compatible)
- Docker
- Python 3.8+
- Node.js 14+
- Apache Kafka 3.8.0
- Zookeeper 6.2.3
- MySQL 8.0.40
- Web browser with WebSocket support

3. Installation

1) Clone the repository:

```
git clone https://github.com/SWE-Capstone-GCPS/gcps-react.git
cd gcps-react
```

2) Install Python dependencies:

```
pip install confluent-kafka pyodbc websockets
```

3) Install JavaScript dependencies:

```
npm install
```

4) Set up your MySQL database and update the connection string in the file main.py.

5) Set up the environment variables for the database and the WebSocket server configuration.

6) Ensure you have a MySQL database set up and running with the correct schema for the Event_Instances and Asset_Telemetry tables.

7) Ensure Kafka and Zookeeper are downloaded and running on your system.

4. System Architecture

The system consists of the following components:

- Producer (producer.py): Simulates bus movements and sends events to Kafka.
- Consumer (consumer.py): Receives events from Kafka and forwards them to the frontend via WebSocket.
- Processor (processor.py): Processes raw events into a format suitable for storage and display.
- Main Application (main.py): Coordinates the consumer, processor, and database operations.
- Database Manager (database_manager.py): Handles database connections and operations for storing and retrieving bus data.
- WebSocket Server (websocket_server.py): Manages real-time communication between the backend and frontend.
- Frontend Components (App.jsx and App.cs):
  Map Interface: Displays real-time bus locations and their routes.
  Sidebar: Displays live bus data, such as speed, location, and route details.
  Frontend Styling (App.cs): Enhances the user interface with a responsive design.

5. Starting the Application

5.1. Starting the Backend Services

Start Zookeeper:

```
cd kafka
bin/zookeeper-server-start.sh config/zookeeper.properties
```

Start Kafka (in another terminal):

```
cd kafka
bin/kafka-server-start.sh config/server.properties
```

Start the main application:

```
python main.py
```

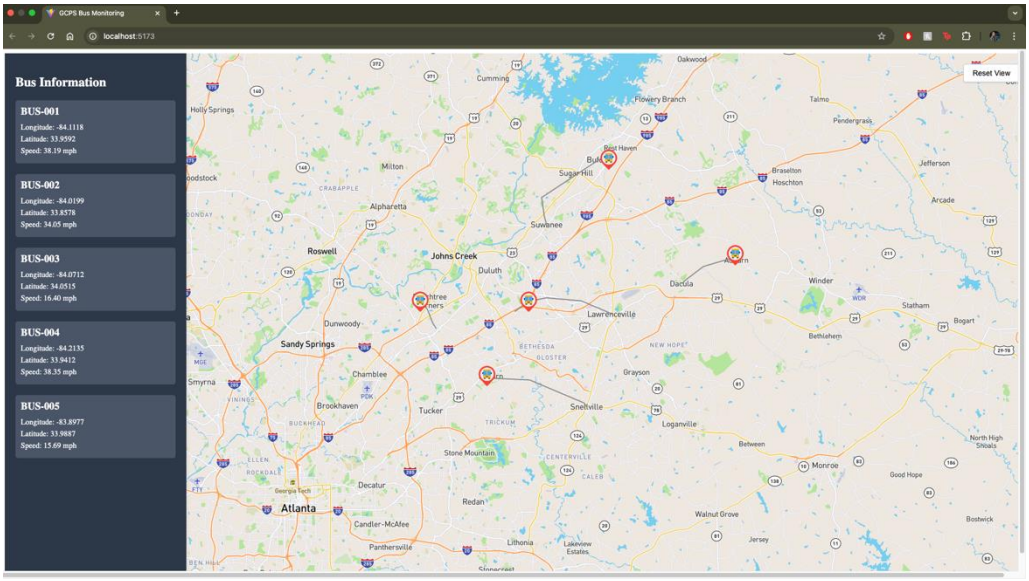
5.2. Launching the Frontend

Start the React development server:

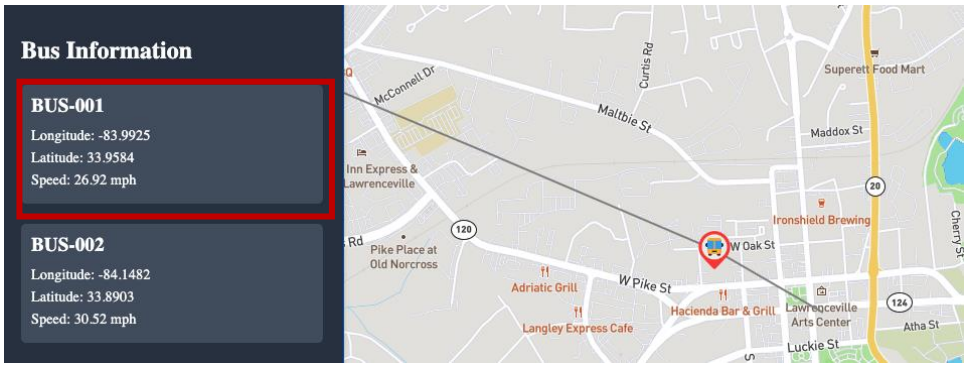
```
npm run dev
```

Open your web browser and navigate to http://localhost:3000.

6. Map Interface



Upon loading the website, the user is introduced to an interactive map. Move around the map by clicking and dragging on the main map to the right of the bus data. While hovering over the map, scroll the mouse wheel down to zoom the map out, and scroll the wheel up to zoom the map in.



On the map there are several pins with a bus icon on them. Click on any of the bus information boxes on the left-hand side to recenter the map on a bus. The map will recenter on the bus pin as it moves in real-time, and the data on the left side of the screen will change to show the live information of the bus that was selected. On the map there are also gray lines that represent the bus route paths. These paths are independent from the road paths, but they usually will overlap as they represent the expected path the bus will take to drop off or pick up students.

7. Troubleshooting

- Map does not display correctly:
    - If the map fails to load, ensure your internet connection is active and your Mapbox API key is valid.

- Data does not update:
    - Check the WebSocket connection in your browser's console.

- Database connection error:
    - Verify your MySQL connection string in main.py and that the database schema is implemented correctly.

8. Frequently Asked Questions (FAQ)

- How often does the bus data update?
    - The system updates in real-time as buses produce data and Kafka events are processed.

- Can I track multiple buses at once?
    - Yes, all GCPS buses are displayed on the map, and you can click on individual buses for details.

- Is historical bus data available?
    - No, the system focuses on real-time tracking, so only the most recent data for each bus is displayed. The system can be updated in the future to display older data, but that is not in this version of the program.