12-T3: GCPS REAL-TIME BUS MONITORING SYSTEM
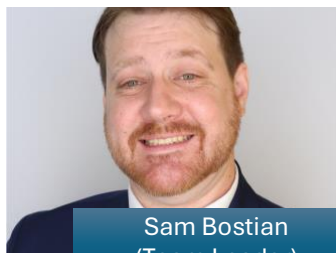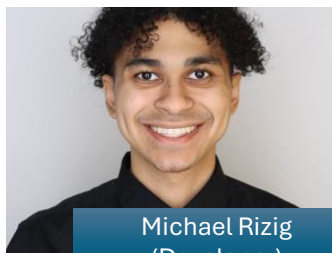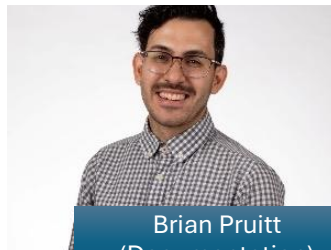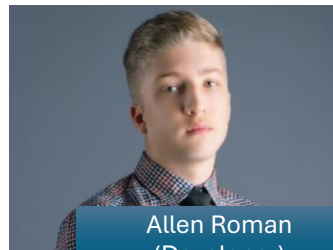
Sam Bostian
(Team Leader)

Michael Rizig
(Developer)

Charlie Mclarty
(Developer)

Brian Pruitt
(Documentation)

Allen Roman
(Developer)

Name	Role	Cell Phone / Email
Sam Bostian	Team Leader	404.555.1212 sbostian@students.kennesaw.edu
Michael Rizig	Developer	678.668.3294 mrizig@students.kennesaw.edu
Charlie McLarty	Developer/QA	470.303.9544 cmclarty21@gmail.com
Brian Pruitt	Documentation	404.207.6548 bpruitt9@students.kennesaw.edu
Allen Roman	Developer	aroman14@students.kennesaw.edu

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 OVERVIEW

This document outlines the requirements for a software application intended to improve on the current data polling and monitoring system of school bus operations for Gwinnett County Public Schools (GCPS). The current system utilizes the Samsara API to poll data every 5 seconds providing telemetry data on buses to identify potential errors and general tracking. To reduce strain on the Samsara API, GCPS plans to transition to Apache Kafka for event streaming of bus data.

1.2 PROJECT GOALS

The main objective for this project is to develop an efficient application to process data that will be generated by GCPS school buses and process them into a SQL Server enabling an easy-to-use monitoring system. Synthetic data will need to be procedurally generated to clearly test whether the application can withstand the incoming data when deployed and taking in real-time data. In addition to data processing, monitoring functionality will be developed to provide performance information as well as identify and address any bottlenecks and potential backlogs.

The goals of this project are:

- **Reduce Latency:** Transition from API polling to event streaming to minimize latency in data retrieval.
- **Enhance Data Processing:** Implement real-time data validation and storage in an SQL Server database.
- **Improve Deployment Consistency:** Use containerization (Docker/Podman) for consistent application deployment across environments.

1.3 DEFINITIONS AND ACRONYMS

Kafka: A distributed event streaming platform.

Samsara: A third-party telemetry solution used by GCPS.

SQL Server: A relational database management system developed by Microsoft.

API: Application Programming Interface.

Docker/Podman: Containerization platforms for deploying applications.

1.4 ASSUMPTIONS

Kafka will be correctly set up and configured.

SQL Server will be accessible from the Linux environment.

The Linux environment will have sufficient computing resources.

2.0 DESIGN CONSTRAINTS

2.1 ENVIRONMENT

The solution will be containerized to include the following components:

- Linux instance: serves as the base operating system for the solution
- Kafka: the system that will be connected to the Samsara API
- SQL Server: stores processed data from the Kafka event stream
- Python Application: the core application that will be consuming events, validating data, and inserting records into the SQL Server database

2.2 USER CHARACTERISTICS

Users include GCPS IT staff, developers, and database administrators. They are expected to be familiar with Kafka, SQL, and Docker/Podman.

2.3 SYSTEM

The system must be capable of processing high volumes of data in real-time, with minimal latency and secure data transmission.

3.0 FUNCTIONAL REQUIREMENTS

3.1 EVENT STREAMING FROM KAFKA

o   The system shall consume telemetry events from Kafka topics at a configurable frequency.

- The system shall log all consumed events for audit purposes.

3.2 DATA VALIDATION

- The system shall validate data according to predefined rules (e.g., valid GPS coordinates).
- Invalid data shall be logged separately for further analysis.

3.3 DATA STORAGE

- The system shall store validated telemetry data in an SQL Server database.
- The system shall provide interfaces for querying stored data.

3.4 ERROR HANDLING

- The system shall retry data processing in case of transient errors.
- The system shall alert administrators in case of persistent errors.

3.5 CONTAINERIZATION

- The system shall be deployable as a Docker/Podman container.
- The container shall include all necessary dependencies for the application to run.

4.0 NON-FUNCTIONAL REQUIREMENTS

4.1 SECURITY

- The system shall secure data transmission using encryption protocols.
- Third parties outside the designated GCPS employees should not have access to any bus data from the SQL Server or directly from the Samsara API.

4.2 CAPACITY

- There are about 2000 functional school buses for the GCPS and approximately 1700 operating at peak hours. The solution must be capable of handling a consistent near-real-time data stream to prevent data backlogging.
- The system shall support high-throughput data processing to handle the scale of GCPS operations.

4.3 USABILITY

o The system shall be user-friendly, with clear documentation and simple deployment procedures.

4.4 PERFORMANCE

o The system shall process and store data with a latency of no more than 15 seconds.
o The system shall maintain a high availability rate, with uptime of at least 99.9%.

5.0 EXTERNAL INTERFACE REQUIREMENTS

5.1 USER INTERFACE REQUIREMENTS

o A command-line interface will be provided for system deployment and management.
o Possible UI implementation may be added if time allows.

5.2 HARDWARE INTERFACE REQUIREMENTS

o The system will interact with Linux-based servers.

5.3 SOFTWARE INTERFACE REQUIREMENTS

o The system shall interface with Kafka for event streaming and SQL Server for data storage.

5.4 COMMUNICATION INTERFACE REQUIREMENTS

o The system shall use HTTPS for secure communication between components where necessary.