Project Plan

Document Number: 1

Date: 8 September, 2024

Gwinnett County Public Schools Technology and Innovation Division - Bus Monitoring through Kafka

Amali McHie

Sarah Fashinasi

Tyler Hood

Jeffrey Sanderson

Alex Baker

Gwinnett County Public Schools

Table of Contents

1. INTRODUCTION

This Project Plan outlines the development of an advanced Real-Time Bus Monitoring System for Gwinnett County Public Schools. The system is designed to enhance student safety and operational efficiency by providing real-time insights into school bus locations and statuses. Utilizing cutting-edge technologies like Apache Kafka for event streaming, Docker for containerization, and SQL Server for robust data management, this project represents a significant step forward in the district's transportation monitoring capabilities. This document also details the project's scope, stakeholders, development strategy, and deliverables, ensuring alignment with the school district's objectives and timelines.

2. PROJECT OVERVIEW

2.1 Scope

2.1.1 Identification

This project involves the development of a Kafka consumer and associated infrastructure for the Real-Time Bus Monitoring System of Gwinnett County Public Schools (GCPS). The software versions include Kafka Connector version 3.8.0, SQL Server version 2022, and Docker version 27.1.2.

2.1.2 System Overview

The Real-Time Bus Monitoring System is designed to enhance the operational efficiency of Gwinnett County Public Schools by providing real-time visibility into school bus operations. The system will consume data from the Samsara Kafka Connector, process it, and store the relevant information in a SQL Server database. This project is sponsored by the Technology and Innovation Division of Gwinnett County Public Schools. The primary users are the school district’s transportation department, with developers, support teams, and system administrators also being involved in its operation. The system will be deployed across multiple operating sites within the district, with potential future expansions.

2.1.3 Document Overview

This document outlines the scope, development plan, and key deliverables for the Real-Time Bus Monitoring System. It includes the identification of stakeholders, system overview, and high-level project details. Security considerations include ensuring data privacy and integrity, especially during data transmission and storage.

2.2 Current System or Situation

The system currently uses Kafka to poll for information every 5 seconds. While this system does provide quick updates, approximately every 11 seconds, the client wishes to update to a newer version to receive real time updates. However, we do not have access to their current set-up, so we will create everything from scratch using the Kafka connector.

2.3 Background, Objectives, and Scope

The current transportation system in Gwinnett County Public Schools lacks real-time monitoring capabilities, leading to inefficiencies and communication delays. This project's objective is to implement a real-time monitoring system that enhances the safety and operations of school bus transportation. The scope includes the

development, testing, and deployment of a Kafka consumer, data processing modules, and a bus event simulator.

2.4 Operational Policies and Constraints

The system will adhere to the operational policies set by Gwinnett County Public Schools, including data privacy regulations and operational uptime requirements. Constraints include the integration with existing IT infrastructure and following school district policies on data management and security.

2.5 Description of Current System or Situation

Currently, the school district uses a manual or semi-automated system for tracking bus operations, which lacks real-time data processing capabilities and operates in a static, regular state without modes. The new system will introduce real-time event streaming, providing a dynamic operational environment with enhanced capabilities. It will be deployed on Linux servers and will incorporate key components like Kafka consumer, SQL Server, and Docker containers. The system will link interface with the Samsara Kafka Connector and the district's internal IT systems, offering real-time data ingestion, processing, and storage within SQL Server. Designed for high throughput with minimal latency, the system emphasizes reliability, maintainability, and scalability, along with data encryption and secure communication protocols to ensure safety, security, and privacy.

2.6 Users or Involved Personnel

The system will be used by various types of users, including transportation department staff, system administrators, IT support personnel, and KSU students. It will be managed by the Technology and Innovation Division, with close collaboration from the transportation department. Users will require training to effectively operate the new system, including real-time monitoring tools and data management protocols.

3. DEVELOPMENT BACKGROUND/APPROACH

The development of the Real-Time Bus Monitoring System will be conducted on a Linux platform using Docker for containerization. The development team will primarily use Python or C# for backend development. Apache Kafka will be used for real-time data streaming, and SQL Server will handle data storage. The team consists of undergrad Software Engineer students currently attending Kennesaw State University.

3.1 High Level Estimates

- **Total Effort: 9 hours per week per person, ~450 work hours.**
- **Lines of Code: ~1,000 lines.**
- **Pages of Requirements & Documentation: 50 pages.**
- **Test Cases: Client dependent.**
- **Duration: 9 weeks (about 2 months).**

3.2 Key Contacts and Stakeholders

- **Project Sponsor**: Technology and Innovation Division, GCPS.
- **Lead Developer**: Edward Van Ness - ed.van.ness@gcpsk12.org

4. FEATURES, PRIMARY DELIVERABLES, AND EXTERNAL COMMITMENTS

4.1 Feature List

- **Event Streaming**: Real-time data intake and processing via Apache Kafka.
- **Backend Processing**: Data validation, transformation, and storage in SQL Server.
- **Simulator**: Tool to generate test data for system validation.
- **Containerization**: Docker containers for deployment.
- **Monitoring & Error Handling**: Basic monitoring tools to track data processing and system health.

4.2 Customer Deliverables

Deliverable 1: Design Document

This deliverable will contain the documentation of the project's design, including the Linux environment, database structure, methods of creating test data, and connections to the Kafka connector stream. This deliverable is due on September 22nd.

Deliverable 2: Test Document

The test document will be a deliverable that details the edge cases, methods of creating test data, the test data that will be used to check the edge cases, and contingency plans in the event test cases fail. This document will be sent to the client on November 17th.

Deliverable 3: Implementation

This deliverable will contain preliminary code and its documentation, and a user guide to the program. All files to run the project in its current state will be given to the client. Any adjustments to the project's structure and issues encountered will be detailed in the code documentation too. This deliverable is due on November 24th.

Deliverable 4: Final Submission

The final deliverable will contain the completed project, as well as source code, a setup guide, and all previous documentation, design, and test case documents. This deliverable contains all necessary documentation to understand and build off the final project given. The final submission will be given to the client on December 2nd.

5. PROJECT SCHEDULE

5.1 Major Project Milestones

Date (YYYY-MM-DD)	Milestone/ task	Deliverable	Remarks
2024-09-22	Design Document	Design documentation, meeting minutes.	-
2024-10-17	Test Document	Testing documents, meeting minutes, implementation progress report.	-
2024-10-24	Implementation	Source code, code documentation, meeting minutes, user guide, project runnable.	-

Date (YYYY-MM-DD)	Milestone/ task	Deliverable	Remarks
2024-11-2	Final Submission	Source code, code documentation, final design documentation, final testing documentation, and user setup guide.	Should contain any additional documentation the sponsor requests.

5.2 Project Status Tracking & Working Meeting Minutes

For meetings, the minutes will include attendees, weekly plans, status, possible issues, meeting length, and anything else discussed.

6. PROJECT WORK AND PRODUCT ESTIMATES

Inputs to estimation include:
1) Deliverables listed in Section 4
2) Major Milestones listed in Section 5
3) Tasks and/or sub-tasks identified in the detailed project Schedule.

6.1 Estimate Summary

The tables below summarize the product size and effort estimates:

Project	Estimate Attributes		
	Size		
WBS areas	Unit of Size	Size	Effort
Total Requirements Effort (includes feature-related **and** "other" (non-feature) Requirements work)			80 hours
Feature Related Requirements Size and Effort Totals	PAGES	15	20 hours
Total Development Effort (includes feature-related **and** "other" (non-feature) Development work)			120 hours
Feature Related Development Coding Size and Effort Totals	LOC	1000	200 hours
Feature Related Development Documentation Size and Effort	PAGES	25-50	80 hours
Total Testing Effort (includes feature-related **and** "other" (non-feature) Testing work)			10-20 hours
Feature Related Testing Size and Effort Totals	TEST CASES	50	10 hours
Feature Level Effort Total (from Feature Estimate Worksheet)			20 hours
Development Effort Total (Includes Feature Level and project level overhead for Requirements, Development, and Testing)			400 hours
Project Level Effort Total (from Project Level Effort Estimates worksheet, excluding requirements, development, and testing)			20 hours

Project Total Effort (Project Totals + Feature Totals)			**440 hours**

7. PROJECT RESOURCE REQUIREMENTS

7.1 Staffing/ Skill Requirements

Role: Team Leader
Critical Skills
Communication across all platforms including team, sponsor and teacher communication.
Skill Gaps:
I don't think there are any skill gaps for team lead all of us are good communicators and are more than familiar with KSU communication guidelines.

Role: Development Manager
Critical Skills
Programming and linking several dependencies together to create a working model. Needs to know/learn Linux, Kafka, Python, MySQL, and how to containerize them in Docker.
Skill Gaps:
Many of these programs are new to the team, so learning how they work individually and then stringing them together to get a working product will be a tall task.

Role: Planning Manager
Critical Skills
Task estimation and keeping the project moving will be team coordination. Main skills will be learning programs to not fall behind in milestones/deliverables.
Skill Gaps:
The main concern here is the significant time commitment and how little time we have over the school semester.

Role: Testing Manager
Critical Skills
Needs to know what to test for and how to test as many possible scenarios as possible. Needs to know how to test Kafka input to make sure the desired files are being sent or received.
Skill Gaps:
Learning new programs and interfaces and how data is transferred from one program to another is going to be essential for testing. We must know the correct information to test and remove any incorrect entrees.

Role: Environment Manager
Critical Skills
Finding and properly downloading all dependencies on given Linux system. The dependencies are listed below. Potential to set up a VM to experiment outside of given hardware. System Dependencies and their version should be monitored along with any settings changed to allow simple reproduction of the system.
Skill Gaps:
While we have worked on VM's before, no one has created one on Linux, and no one uses a Linux system. Learning Linux and how to install dependencies, as well as learning about these dependencies is the main skill gap.

7.2 Plan to Fill Skill Gaps

All skill gaps should be filled over the first two weeks after we get our hardware. Most skill gaps are short processes of learning and getting familiar with the layout and settings of new software and Linux. Another

software that we need to learn is Kafka and how it will connect to our program and database, as well as using docker to containerize, as none of us have experience with it.

8. DEPENDENCIES AND CONSTRAINTS

8.1 Constraints

This project lists a variety of constraints, some of which are time, resources, software and hardware availability, and staffing. We are a small group of students where none of us have direct access to Linux hardware. We are stuck in our given time interval of one semester, and the client has asked for specific software to be used so they can recreate our solution in-house if they wish to do so.

8.2 Dependencies

The dependencies in the project are all of the software the customer has requested including Kafka, Docker, Python, MySQL, and Linux. The following are descriptions of the dependencies.

Kafka also known as Samsara Kafka provides real time data to the customer in the form of Json files which we can receive from the Kafka Connector. In the context of the project, we will use generated Kafka data, and not real-world data the customer uses.

Docker is a containerization software that has been requested to be used as a simple way to transfer across devices. Docker also allows for light weight running of the application.

Python is a standard programming language. MySQL is the requested database for storing information. Linux is the operating system the clients' servers will be running and so we will develop on Linux.

9. RISK MANAGEMENT

9.1 Risk Management Strategy

To handle risks, our team will focus on a proactive approach, identifying risks in the beginning and at the different milestones of our project. Once we identify the risks, we will categorize them based on priority and the potential impact it could have on the success of the project. This helps prioritize our response efforts and allocate resources effectively. We will continuously monitor the risks we have and detect new ones as they occur. We will uphold continuous communication with the customers as we complete each milestone, making sure that risks and mitigation efforts are being effectively communicated. Finally, we will maintain updated documentation of each risk and the changes made while working on the project to keep track of the decisions made and actions taken.

9.2 Initial Risk List

This is the initial risk list; the risks are listed in priority order from top to bottom. Descriptions are provided below the table. Risks with a pre-mitigation magnitude of 2.0 or below are not listed.

Risk number	Risk Priority (H, M, L)	Likelihood of Occurrence	Risk name: brief description	Mitigation Strategy "ACCEPTED" or "MITIGATED" with pointer to plan.
1	H	Very likely	Difficulty in securing or retaining the necessary personnel, equipment, or materials.	MITIGATED
2	L	Less likely	Lack of active involvement from sponsor	ACCEPTED
3	M	Very likely	Issues with technology, such as software or hardware incompatibilities, bugs, or performance problems.	ACCEPTED

10. PROJECT CONFIGURATION AND DATA MANAGEMENT

10.1 Configuration Management

The high-level configuration plan consists of using Git for configuration and version control. The central repository will be established on GitHub to store all configuration items, including source code, Docker files, Kafka configurations, and SQL Server scripts. Configuration items include settings for the Samsara Kafka Connector and interface configurations with the district's internal IT systems, Docker containers consisting of Docker files that containerize applications and ensure continuous deployment. All changes will be managed through formal change requests like pull requests or obtaining approval from sponsors and developers. Baselines will be updated at key milestones, helping to maintain system stability and consistency. Detailed documentation will be maintained for all configuration management processes, including how to manage Kafka consumer settings, SQL Server configurations, Docker container deployments, and integration procedures.

11. PROJECT PROCESS

11.1 Software Life Cycle Model

Our project follows the Agile lifecycle approach. Our development process uses Agile Scrum methodology to ensure iterative progress and adaptability. This includes a weekly scrum meeting, weekly meeting with our sponsor and sprint review. Our development tools include Smartsheet for tracking project tasks, Git and GitHub for version control, Docker for containerization and Visual Studio Code and IntelliJ for coding and debugging. By adopting the Agile Scrum methodology and utilizing these development tools, we hope to deliver high-quality software, with the flexibility to adapt to changing requirements and continuous feedback.

12. REFERENCED DOCUMENTS

College of Computing and Software Engineering. (2024). *Industry capstone projects (IT, CS, SE)* slide 16. Kennesaw State University

13. GLOSSARY

GCPS- Gwinnett County Public Schools
LOC- lines of code
SQL- Structured Query Language
VM- virtual machine

14. CHANGE RECORD

No changes have been made to this document yet as this is the first draft.