Continuous Visualization of CyRide Through an Interactive Map

Team 22: Evan Schlarmann, Endi Odobasic, Braden Buckalew, Andrew McMahon

Advisor: Selim, Mohamed

Client: Soliman, Mohammed

Table of Contents

3 Project Plan

3.1 Project Management/Tracking Procedures

For our project, we'll be using the agile development project management style. This was decided to help adapt quickly to any new requirements needed for the project and contain any errors that come up. Our sprints will be 2 weeks long, and we'll meet weekly with our advisor and client. This consistency will ensure our sprints are productive and deadlines are met, while streamlining the development process and guaranteeing quality work by all team members. The project's progress will be documented through the Gitlab issues board; these issues will be created from our task decompositions so group members can select given tasks and will be completed with separate branches. All of the issues will be affiliated with a milestone our team aims to complete by a particular date.

3.2 Task Decomposition

Frontend (F):

1. Main Map Interface
    a. **Task:** Integrate Google Maps into the user interface and start the display over the Iowa State Campus
    b. **Task:** Ensure the user can zoom in/out and move the map around
    c. **Task:** Create a button to return the user to the starting point of the map
2. User Interface
    a. **Task:** Create the generic layout of the page with the header and footer.
    b. **Task:** Have a help button that links to a help page describing the project and what data may mean.
    c. **Task:** Have a menu to select what UE to track based on the bus.
3. Data Integration from Backend
    a. Requires the WebSocket task from the backend and the main map interface.
    b. **Task:** Create a WebSocket connection to Django and parse the data received.
    c. **Task:** Call functions that update the UI to display all the updated bus data.
    d. **Task:** Have a bus icon appear on the map where their coordinates are.
4. Bus information
    a. Requires the WebSocket task from the backend and the main map interface.
    b. **Task:** When a click occurs on a bus, display the data about the bus, including but not limited to name, route, speed, heading, latitude, longitude, Rx/Tx frequency, and UE strength.
    c. **Task:** Have a function to update the data given updates from Django.
5. Bus Location Visualization

    a. Requires the WebSocket task from the backend and the main map interface.
    b. **Task:** Given an update of data, move the bus to the new location in a smooth transition.
    c. **Task:** If a UE has no connection, turn the path red else, if connected, turn the path green.
    d. **Task:** Display the estimate for when the bus will be back in range based on data from Django.

Backend (B):

1. WebSocket creation
    a. **Task:** Create a WebSocket that allows connections to receive updates on data for the buses.
    b. **Task:** Format the data that will need to be sent in the WebSocket and how it should be parsed.
2. Connection to the UE service LibreNMS
    a. **Task:** Setup an API call for the LibreNMS service that receives the necessary data for the UE's. This data can include the name, speed, heading, latitude, longitude, and Rx/Tx frequency
    b. **Task:** Whenever data is received, it can be saved to the database to be used by the machine learning model.
3. Data processes and prediction (ML)
    a. **Task:** When the UE has no connection, use GPS, and Google Maps API to receive data about the bus and store that data in the database.
    b. **Task:** Create a Machine Learning model that uses the bus location and pathing to predict the bus movement.
4. Manage Database
    a. **Task:** Make efficient use of keys and tables to make sure that query times are efficient.

Testing (T):

1. Frontend Testing
    a. **Task:** Use a test suite to test the UI components to make sure that they load correctly and have the correct data appear. This can also test edge cases to make sure that data is formatted correctly based on what is given.
    b. **Task:** Create tests to ensure data can be parsed and errors are handled if any received data is malformed or missing.
2. Backend Testing

a. **Task:** Use a test suite to make sure that connections to all external APIs are setup correctly, connected, and receive expected results.
b. **Task:** Create tests to ensure that malformed data is handled correctly within the APIs and when stored in the database.

Server (S):

1. Download necessary applications and dependencies
    a. **Task:** Download MySQL and set up the database so that applications can connect.
    b. **Task:** Download Python and its dependencies for Django.
    c. **Task:** Download Node.js and install React.
2. Setup CI/CD
    a. **Task:** Create the CD for the server to deploy and start the application on an update from GitLab
    b. **Task:** Create CI branches for all tasks that get merged into the main branch to be tested and deployed.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestone 1 - F1, F3, B1, S1, S2
The first milestone is to have a bus icon appear on the Google Maps integration with mock data from the backend. This shows that the data retrieved about the bus locations can be displayed and updated. The milestone also has the server set up with the application so that all changes to the main branch are automatically deployed for continuous development.

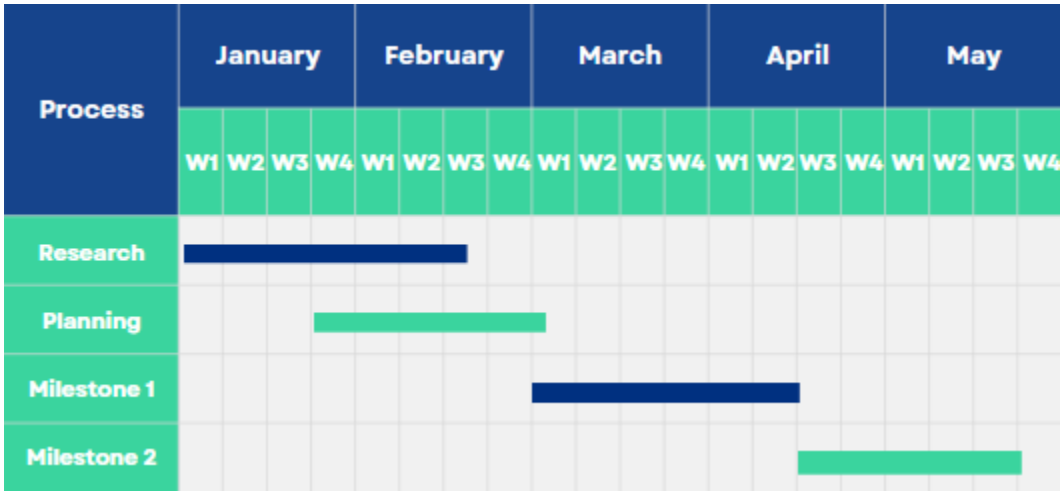
Milestone 2 - F2, F4, B2, B4
The second milestone is to retrieve data from the user equipment using the LibreNMS service and store it in the database. This data can then be retrieved by Django and sent to the front end to display the bus location updates along with other data provided. In this stage, the goal is to have a responsive frontend that takes < 0.5 seconds to retrieve updates and provides a clean bus movement transition. This milestone also covers other user interfaces that would be needed by users–but not critical to the main functionality–including a help page to help integrate users into the platform, and an about page that describes the project.

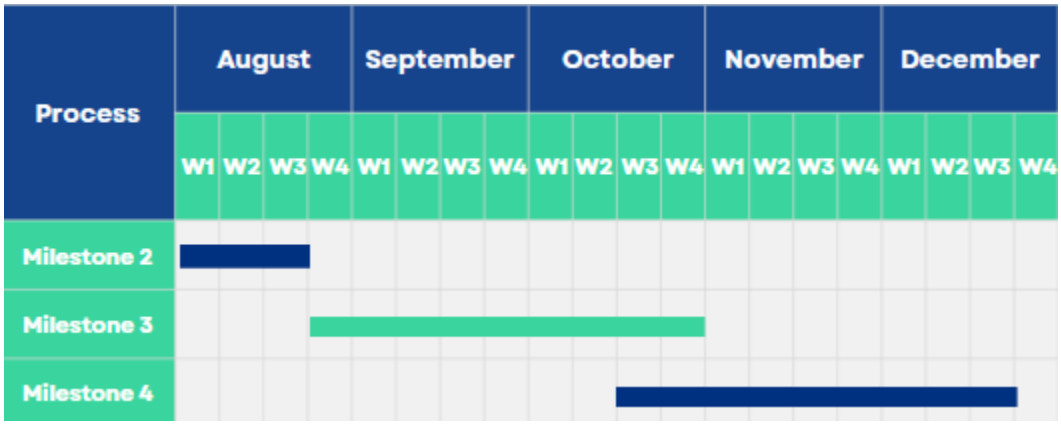
Milestone 3 - F5, B3
This milestone is important, as it completes the functionality of the project, giving users constant updates on the bus location using multiple methods of prediction. The first is the UE data retrieved when it is within range of towers to transmit data, and then machine learning uses the bus GPS coordinates to predict the bus movement. The method of prediction will be shown to

the user and give insight into when methods may change while giving accurate predictions that are close to 95% of the bus's actual location/arrival time.

Milestone 4 - T1, T2
This is a final milestone that can be achieved depending on time constraints. The goal is to ensure the application runs correctly with 100% of tests passing. The testing would cover unit, integration, system, and acceptance testing.

3.4 Project Timeline/Schedule





3.5 Risks And Risk Management/Mitigation

Possible Risks	Probability	Risk Severity	Analysis
No working user equipment	0.1	High	If there is no usable working equipment for testing, we might have

			to pivot to other tracking methods to get locations. Otherwise, there would be no way to test UE data and the system.
No user equipment on running buses	0.4	Medium	If there were user equipment on a bus running a route, we would have to drive the UE around to test data collection manually. This would require more time for development.
Inaccurate machine learning model	0.3	High	The less the ML model is used the less accurate it's going to be, so it's expected to not be as reliable in the beginning. Our team will start early on developing the model to make sure it can predict locations in the end otherwise, we won't have any time estimates for users.
Technical Issues	0.2	Medium	Many external applications and new software are being used in this project. They may cause setbacks or roadblocks for development that our team must overcome.

3.6 Personnel Effort Requirements

Task	Hours Expected	Explanation
Frontend	——---——----------——---——----	——---——----------——---——----
Main Map Interface	15	Configure Google Maps API to display the interface for Ames
User Interface	20	Make a user-friendly interface to make the application fit for everyone
Data Integration from Backend	20	Obtain the data (mock+UE) to get constant location updates to use for the bus updates

Bus information	15	Grab the data from the bus and display any information for the user, such as arrival times or any other insights
Bus Location Visualization	20	Use the Data pulled from the backend to fetch longitude/latitude locations and constantly update bus locations on the map.
Backend	—---—---—---—---—---—	—---—---—---—---—---—---—
WebSocket creation	20	To send and get real-time updates of data with the help of the backend and the database storage.
Connection to the UE service LibreNMS	25	Setting API calls to fetch and store data from UE in the database.
Data processes and prediction (ML)	35	Creating a process to handle data when there is no connection to the UE. The ML model will (to its best ability) accurately display bus movement based on past data from DB.
Manage Database	15	Creating well-defined tables, schemes, and keys to store/manage data efficiently and cleanly.
Testing	—---—---—---—---—---—	—---—---—---—---—---—---—
Frontend Testing	25	Testing all the tasks and milestones worked on iteratively ensures all are working and displaying properly.
Backend Testing	25	Like the Frontend testing, tests, tasks, and milestones regarding the backend and

		making sure they are processing data correctly.
Server	—----—----—----—----—----	—----—----—----—----—----
Download necessary applications and dependencies	7	Make sure everything is downloaded regarding the application so that the application can run as intended with no issues
Setup CI/CD	7	Allow the application to run constantly with automatic pulls and updates

3.7 Other Resource Requirements

The project requires a server to run the application and give access to individuals to view bus locations. It also requires working user equipment that will be put on a CyRide bus to test data transmission.