Project Components CyRide Visualization

Frontend:

1. Main Map Interface
   i. Shows the main screen on the website, such as the bus to select and the zooming.
   ii. Buttons, Drop-down, smaller increment zoom option.
   b. **Task:** Integrate Google Maps into the user interface and start the display over the Iowa State Campus
   c. **Task:** Ensure the user can zoom in/out and move the map around
   d. **Task:** Create a button to return the user to the starting point of the map
2. Bus Location Visualization
   i. Live bus location with a sketch of the bus moving throughout the route.
   ii. Some design tools for bus and image location updates.
   iii. Visualize the pathing when the bus is in range of base stations and when outside of range.
   iv. Give estimates of when buses will be in the range of base stations to connect to the wireless network.
   b. **Task:** Have a bus icon appear on the map where their coordinates are.
   c. **Task:** Given an update of data, move the bus to the new location in a smooth transition.
   d. **Task:** If a UE has no connection, turn the path red else, if connected, turn the path green.
   e. **Task:** Display the estimate for when the bus will be back in range based on data from Django.
3. Bus information
   i. Upon selection by the user, display the bus' name, route, speed, heading, latitude, longitude, and Rx/Tx frequency for the current bus.

        ii. Also display the UE connection strength to the base station, giving an estimate of the WiFi connection strength at the bus' location.
    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.
4. User Interface
        i. User interaction
            1. buttons, images, map drags, dropdowns, pop-ups, forms building text
    b. **Task:** Create the generic layout of the page with the header and footer.
    c. **Task:** Have a help button that links to a help page describing the project and what data may mean.
    d. **Task:** Have a menu to select what UE to track based on the bus.
5. Data Integration from Backend
        i. Use backend API to fetch the data needed to update bus locations.
        ii. Use data fetched to update the 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.

Backend:

1. WebSocket creation
    - i. Manage the WebSocket for the frontend to fetch bus location.
    - ii. requests/returns
    - **b. Task:** Create a WebSocket that allows connections to receive updates on data for the buses.
    - **c. Task:** Format the data that will need to be sent in the WebSocket and how it should be parsed.
2. Data processes and prediction (ML)
    - i. Uses real-time bus location and predicts the location of the bus.
    - ii. logic for predicting bus location based on data available.
    - **b. 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.
    - **c. Task:** Create a Machine Learning model that uses the bus location and pathing to predict the bus movement.
3. Connection to the UE service LibreNMS
    - i. This allows the backend to receive live update information about the UE location and connection to base stations.
    - **b. 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
    - **c. Task:** Whenever data is received, it can be saved to the database to be used by the machine learning model.
4. Manage Database
    - **a. Task:** Make efficient use of keys and tables to make sure that query times are efficient.

Testing:

1. Frontend Testing
    i. Create tests for the frontend that deal with the UI and the data received to mitigate edge cases that could cause errors.

    b. **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.

    c. **Task:** Create tests to ensure data can be parsed and errors are handled if any received data is malformed or missing.
2. Backend Testing
    i. Create tests for the backend to ensure data is stored/used correctly and the API can calculate correct predictions to a given degree.

    b. **Task:** Use a test suite to make sure that connections to all external APIs are setup correctly, connected, and receive expected results.

    c. **Task:** Create tests to ensure that malformed data is handled correctly within the APIs and when stored in the database.

Collaboration:

1. Communicate together
    a. Primarily discord, as we are all comfortable with using it.
2. Gitlab
    a. We will use GitLab for version control.

    b. If anything bad happens, we can easily go back and use the last version.

    c. Agile development so that we can split into pairs to work on different functional components.

    d. Use issues and branches to develop components separately from the main release branch.

Server:

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.