Continuous Visualization of CyRide Through an Interactive Map

Team 22
Evan Schlarmann - Backend/Server
Endi Odobasic - Frontend/Mapping
Braden Buckalew - Frontend/Testing
Andrew McMahon - Backend/Database
Chiran Subedi - Backend/Machine Learning

Email: sddec24-22@iastate.edu
Team Website: https://sddec24-22.sd.ece.iastate.edu

Advisor: Selim, Mohamed
Client: Soliman, Mohammed

Executive Summary

Rural communities in Central Iowa face limited access to reliable wireless connectivity due to factors like low population density, sparse infrastructure, and unpredictable weather. This lack of coverage impacts essential services, economic development, and access to education. The CyRide Visualization project addresses this issue by providing a data-driven platform to improve wireless connectivity through the collection and analysis of connectivity data, helping to develop network solutions that benefit rural areas.

The primary design goals of the project were:

- **Real-Time Data Visualization**: Create an interactive, map-based interface to display bus locations and wireless connectivity data for researchers, bus operators, and students.
- **Predictive Location Analytics**: Use historical bus location data to predict future positions when UEs (User Equipment) lose connectivity, ensuring continuous tracking even in low-coverage areas.
- **User-Friendly Interface**: Ensure the system is intuitive for researchers, CyRide operators, and Iowa State students.

The CyRide Visualization system integrates multiple technologies to meet its design goals:

- **Frontend**: The web application is built using React to provide an interactive user interface, visualizing bus routes and connectivity data on Leaflet.
- **Backend**: A Django server processes and stores data, integrating with the UE API for real-time connectivity data collection. The system uses a MySQL database to store historical bus location data and connectivity statistics.
- **Machine Learning**: Algorithms analyze past bus location data to predict future positions when UEs are out of range. This allows the system to provide continuous tracking during signal loss.
- **Real-Time Updates**: Django Channels and Daphne are used for WebSocket communication, delivering live updates on bus locations and UE connectivity status.
- **Data Security:** The system uses CORS (Cross-Origin Resource Sharing) to secure data exchange between the frontend and backend.

The system effectively addresses the needs of its users:

- **ARA Researchers**: The platform provides insights into wireless coverage gaps and bus patterns, which assist in optimizing network deployment and improving service in rural areas.

- **CyRide**: The system offers an efficient way to track bus locations in real-time, predict arrival times, and monitor signal strength, helping optimize bus routes and reduce operational costs.
- **Iowa State Students**: The app provides students with live updates on bus locations and predicts arrival times, ensuring they have accurate transportation information and improving their daily routines.

The CyRide Visualization platform has the potential to extend beyond Ames, offering insights for the expansion of wireless connectivity across rural Iowa. Future steps could involve adding more transportation routes, integrating additional data sources for coverage prediction, and refining machine learning models to enhance location forecasting. The project also sets the stage for further collaboration with ARA to optimize wireless infrastructure based on real-time data insights.

In summary, this project is a vital step toward improving connectivity and transportation in Central Iowa, providing both immediate benefits to bus operators and students, and laying the groundwork for long-term improvements in rural wireless coverage.

Learning Summary

Development Standards & Practices Used

- IEEE 29148 - Standard for Systems and Software Engineering Life Cycle Processes
- IEEE 1012 – Standard for System, Software, and Hardware Verification and Validation
- IEEE 1061 – Standard for a Software Quality Metrics Methodology
- IEEE P3123 – Standard for Artificial Intelligence and Machine Learning (AI/ML) Terminology and Data Formats
- IEEE/ISO/IEC 23026: Systems and Software Engineering

Summary of Requirements

Functional Requirements:

- Server
- Real-Time Bus Tracking with Long/Lat Coordinates
- Prediction Algorithm
- WebSocket

- Database

Resource Requirements:

- Leaflet API

Physical Requirements:

- Computer/Laptop Needed
- WiFi/Cell Signal

Qualitative Aesthetic Requirements:

- User Friendly Design
- Visual Feedback

Environment Requirements:

- Need of UE Device to Fetch Real-Time Location
- Computers For Bus Visualization Display

UI/UX Requirements:

- Minimal + Simple Interface
- Error Displays
- ToolTips
- Map Interaction
- On-Screen Displays and Overlays

Applicable Courses from Iowa State University Curriculum

- Engl 314: Technical Communication
- COMS 227: Introduction to Object-Oriented Programming
- COMS 228: Introduction to Data Structures
- COMS 252: Linux Operating System Essentials - Iowa State
- COMS 309: Software Development Practices
- SE 317: Introduction to Software Testing
- COMS 319: Software Construction and User Interfaces
- SE 339: Software Architecture and Design
- COMS 363: Introduction to Database Management Systems

New Skills/Knowledge acquired that was not taught in courses

- How to build with Django
- How to build with React and TypeScript
- Setting up a Linux server
- Machine Learning
- Mapping

Table of Contents

1 Introduction

1.1 Problem Statement

Currently, rural communities in Central Iowa have limited access to reliable wireless connections. The combination of low population density, a plethora of low-coverage areas, and unpredictable weather conditions have left rural communities in Central Iowa with little to no access to the internet and other coverage–dependent technologies.

To solve this issue, the CyRide Visualization project strives to provide some critical data to assist the implementation of wireless mobile connectivity across Central Iowa via a visually appealing predictive mapping interface. To solve this, we will collaborate with the "Agriculture and Rural Communities" team, or simply "ARA." ARA is a multi-university-wide team focused on building a platform in and around Central Iowa that enables the research and development of affordable and high-capacity connectivity to benefit rural communities and industries. Our application will utilize mobile UE devices from the ARA team stationed on vehicles around Ames to gather location-specific connectivity data and apply it to future data sets. These data sets will then be displayed, helping the ARA team develop a network solution for the rural Ames communities, and eventually, to all rural communities in Central Iowa.

1.2. Intended Users

ARA Researchers:

- Needs:
    - ARA researchers, which include our client, manage the UE devices throughout Ames. To better understand the data from each UE, they need a reliable and readable application to display the data on a map. ARA researchers also need a way to display trends and predict wireless connectivity through differing times of day and seasons.
- Benefits:
    - This project provides insights into the base station ranges and where the system cannot connect to provide accurate data in the network. These insights allow ARA researchers to build upon the system efficiently to ensure that Ames is connected to provide continuous coverage of vehicle locations. In addition, by managing the

transportation system, ARA researchers can uncover important data such as bus patterns, greater involvement, or even better route optimizations.

Cyride:

- Needs
    - ARA plans on coordinating more Bus UE's with Cyride. Cyride needs a in-house bus tracking solution to reduce costs and have access to the implementation to create new features. Need an application to store bus GPS locations and display locations to CyRide users.
- Benefits
    - This project will provide insights of the bus connectivity and signal strength throughout ames showing Cyrides Buses being implemented with ARA research. It will track the bus locations and predict when a bus will arrive at a given destination. It also gives an interactive map to show to that displays their movement over Ames.

Iowa State University students:

- Needs
    - Iowa State students rely on a strong and reliable wireless connection daily to get around campus and to get to classes on time. For this to come into play, they need accurate bus scheduling. As buses are their primary source of transportation, they may get frustrated when missing a bus because of incorrect arrival times.
- Benefits
    - With this application, we will give them access to live bus updates that give the most accurate bus tracking information and other insights of any other bus tracking application. This will help their daily routines and give them a dependable means of transportation. It will also help improve connectivity so that students may have wireless access all around campus.

2 Requirements, Constraints, and Standards

2.1 Requirements and Constraints

Functional Requirements:

- Server
    - The application requires a server to host the website. It must allow connections from users and handle all requests for the application.
- Real-Time Bus Tracking with Long/Lat Coordinates
    - Requires connection with the UE API to retrieve UE coordinates. This connection must retrieve and send locations within 4 seconds.
    - Requires machine learning to predict a UE's coordinates when the UE is not in range of a base station. The algorithm should provide predictions within 4 seconds.
- UE Pinging and Rest API
    - The application needs to ping the bus UEs to determine their connectivity and display the connectivity along their respective paths. It will also require the pinging data to determine if the UE is out of range and requires using machine learning to continue providing bus location data.
    - All UEs need an API built on the system that can be requested to fetch bus GPS data from the databases. This needs to be a REST API that can return GPS data within the 4 second real-time updates.
- Prediction Algorithm
    - Machine learning, using the GRU model, can predict with 95% accuracy where a bus is currently located and when the corresponding UE will be back in range of a base station.
    - Algorithm should be able to predict the location of the bus every 4 seconds and fill in Gaps where the UE connection is lost.
- WebSocket
    - WebSockets will provide efficient data transfers between the frontend and backend components. The WebSocket must transfer data every 4 seconds to update the UI in real time. This follows the historical data given by CyRide.
- Database
    - The project will utilize a database to store past and current data that can be used in machine learning algorithms. This requires the data to be efficiently stored in a schema that can be queried within 0.5 seconds.

Resource Requirements:

- Real-Time Data Feed Acquisition

    - The server must have enough memory to handle many requests for bus location data and machine learning algorithms so that the application doesn't suffer performance losses.
- LeafLet API
    - LeafLet is an open-source library used to build the world maps and interaction. It will be used to show the bus movement over Ames and give users the basic controls over map movement.

Physical Requirements:

- Computer/Laptop Needed
    - The development will occur on laptops. This ensures that the UI fits the size requirement during development and provides access to developmental environments.
- WiFi/Cell Signal
    - A signal connection is required to retrieve UE data for bus locations. This allows the application to access external interfaces to retrieve necessary data.

Qualitative Aesthetic Requirements:

- User Friendly Design
    - Need an intuitive design that a user can quickly grasp and use completely. Anyone can access the application, so it must be accessible to people with different technical skills.
- Visual Feedback
    - The UI display must provide users with clear feedback when using different functionalities. This feedback will be accurate to its design and clearly provide its data.

Environment Requirements:

- Need UE Device to Fetch Real-Time Location
    - A UE device is needed to test data retrieval and prediction algorithms once the UE device is out of range. The UE device will be supplied on the brown bus route for Cyride.
- Computers For Bus Visualization Display
    - Computers are needed to access the application for development and testing.

UI/UX Requirements:

- Minimal + Simple Interface
    - It will be a simple interface to reduce Navigation Complexity for the User. Will have the One UE preselected.
- Error Displays

    - The application will notify users of network or server errors, ensuring a smoother user experience by addressing issues effectively.
- ToolTips
    - The application will include tooltips and a comprehensive FAQ section to assist users in understanding the system, ensuring ease of use and accessibility for all.
- Map Interaction
    - Users can easily interact with the map, including dragging, zooming, and viewing building names, for efficient exploration and navigation.
- On-Screen Displays and Overlays
    - On-screen displays and overlays will provide additional information, such as bus routes and availability, enhancing the user experience with informative features. More specifically, the user will be able to select the bus, and see additional information about the bus through an info box such as longitude/latitude pairs, its signal, and the name of its stop.

2.2. Engineering Standards

- IEEE 29148 - Standard for Systems and Software Engineering Life Cycle Processes:
    - Provides requirements for planning documentation and specifications that will be developed for the project and may be revised during the project's life cycle.
- IEEE 1012 – Standard for System, Software, and Hardware Verification and Validation:
    - Covers processes, methods, and techniques to ensure software products meet specified requirements. This ensures that the project meets all user and client requirements once it is completed.
- IEEE 1061 – Standard for a Software Quality Metrics Methodology:
    - Guides measuring software quality attributes and assessing product quality. This ensures that the product is efficient and meets all computational constraints provided by the client.
- IEEE P3123 – Standard for Artificial Intelligence and Machine Learning (AI/ML) Terminology and Data Formats
    - This provides a standardized way for implementations of machine learning that can be used in the project.
- IEEE/ISO/IEC 23026: Systems and Software Engineering -- Engineering and Management of Websites for Systems, Software, and Services Information
    - This standard provides website life cycle requirements, including design, testing, and management. This provides a guideline to ensure a quality life cycle of the website such that every aspect has been thought out.

3 Project Plan

3.1 Project Management/Tracking Procedures

We'll use an agile development project management style for our project. This was decided to help adapt quickly to any new requirements needed for the project and contain any errors. Our sprints will be two 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 Leaflet 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 API
    a. **Task:** Setup an API call for the UE API 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 Leaflet 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

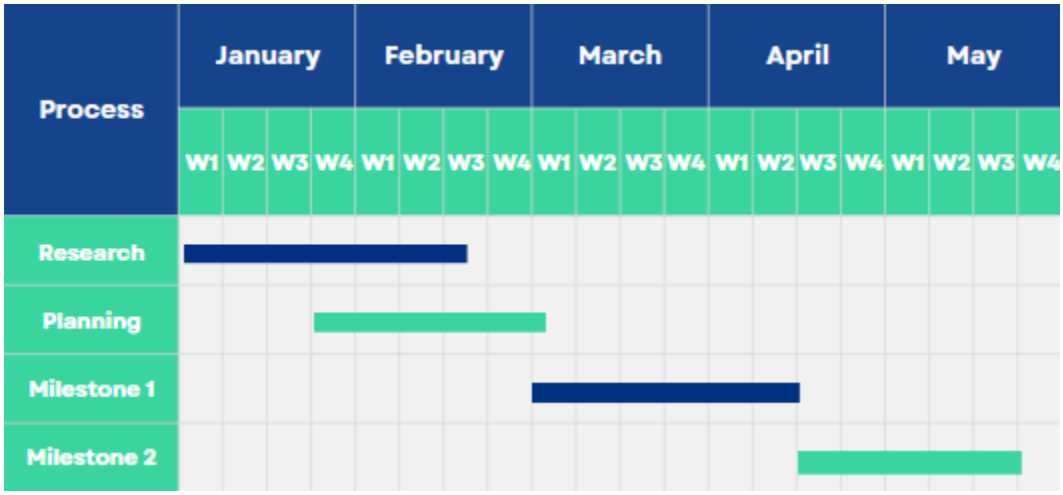
Milestone 1 - F1, F3, B1, S1, S2
The first milestone is to have a bus icon appear on the Leaflet integration with mock data from the backend. This shows that the data retrieved about the bus locations can be displayed and updated. The milestone also has the server set up with the application so that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

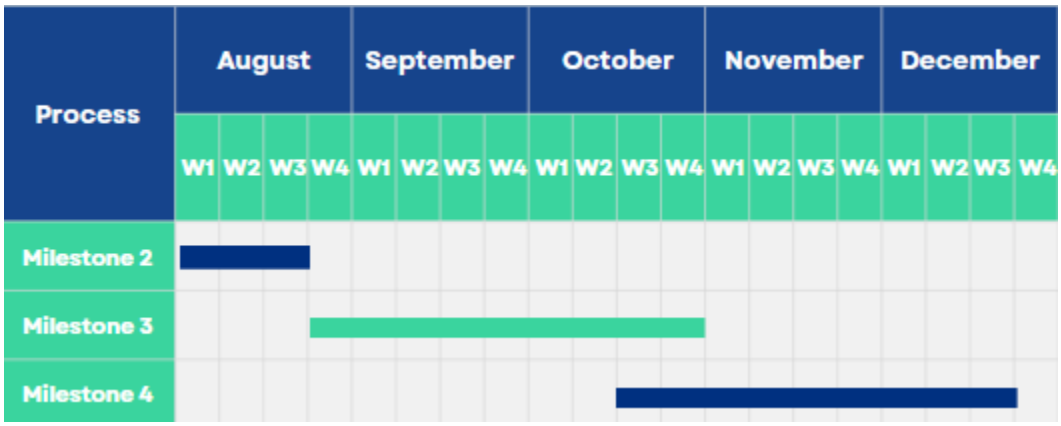
Milestone 2 - F2, F4, B2, B4
The second milestone is to retrieve data from the user equipment using the UE API service and store it in the database. This data can then be retrieved by Django and sent to the front end to display the bus location updates and other data provided. In this stage, the goal is 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 users would need–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 prediction method 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
Main Map Interface	15	Configure Leaflet 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.
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 API	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.
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.
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
Total Time	249	

3.7 Other Resource Requirements

The project requires a server to run the application and give access to individuals to view bus locations. The server chosen was an Ubuntu server provided by ISU. It also requires working user equipment that will be put on a CyRide bus to test data transmission. The client will provide and handle user equipment so our application can access the data for testing.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	Researchers will be able to gather data for mobile user equipment around Ames. This will help improve on connectivity which will benefit Ames residents in the future.	The applications increases the benefit of mobile user equipment research. It will lead to an increase in network connectivity.
Global, Cultural, and Societal	The project provides data in an efficient manner and allows any usage to further research. The project focuses on bringing connectivity to everyone to ensure they are connected.	The project is built to help researchers bring solutions to connectivity nationwide.
Enviormental	The application uses a lot of computing power to retrieve data and predict bus locations. It also requires the resources needed in building servers, user equipment, and base stations.	Powering the application requires an increase in grid production. It also requires an increase in harvesting metals to build equipment.
Economic	The project provides a way to test mobile user equipment connectivity efficiently. It allows for constant data gathering.	The product needs to be accessible and cheap so that many researches can have access.

4.1.2 Prior Work/Solutions

"The POWDER Manual," docs.powderwireless.net. [Online Manual] Available: https://docs.powderwireless.net/ [Accessed April 19, 2024].

The POWDER application tracks buses in Utah using the ARA network. This application uses the data to update the bus locations and tracks them in a given area with connectivity. Our project differs as Ames does not have total connectivity to show bus locations. The application must use machine learning and UE data to provide bus locations. This will allow ARA researchers to test mobile UE data within Ames to see the connectivity during different parts of the year. This will allow researchers to improve the network connectivity and bring better user equipment services.

4.1.3 Technical Complexity

Our project has a lot of complexity with many communicating components. For internal complexity, our application will be running and maintained on an Ubuntu server that deploys/tests the codebase. On the server, MySQL runs to store all the UE and bus data for machine learning algorithms. This database will be queried by Django which will need to decide whether to provide bus locations with the UE data or machine learning algorithms. It will then constantly send updates to React to display the received data on the Leaflet interface for the user.

This will have many parts communicating constantly to provide real times updates. The machine learning algorithms will take mathmatical thinking and implementation to get accurate models that provide bus predictions. React will need to communicate with Google APIs to retrieve the Leaflet and be able to display routes/coordinates accurately. Django also needs to communicate with UE APIs to receive UE data updates quickly.

These requirements allow the team to show our expertise in developing software applications. Many top end software standards are being used to build this application. This keeps the application modern and provides the most functionality for building the project. With this project, the team will work with full-stack knowledge and how to maintain that stack for deployment. It will take all of the team's skills to construct the final design and require learning new technologies to bring the best solution.

4.2 Design Exploration

4.2.1 Design Decisions

TechStack: React-Django-CORS-SQL:

Using the React-Django-CORS-SQL tech stack is important for this project because it combines a powerful frontend framework (React) with a robust backend framework (Django), creating efficient development and an easy user experience. CORS (Cross-Origin Resource Sharing)

improves security by controlling which domains can access resources. SQL databases provide reliability and scalability for managing data, which is essential for effectively handling real-time data transfer. This tech stack offers a comprehensive solution for building a secure, scalable, and efficient web application.

Websockets: Real-time UE Data:

WebSockets allow bi-directional communication between the server (Django backend) and the client (React frontend), enabling real-time data updates without constant polling. Unlike traditional HTTP requests, which require establishing a new connection for each request, WebSockets maintain a persistent connection, reducing overhead and latency. Websockets are well-suited for applications requiring high scalability and concurrent connections, making them ideal for simultaneously handling real-time data from multiple UE devices.

Machine Learning: Traffic Accuracy:

This is for displaying mock vehicle data if the vehicle is outside coverage of the towers, giving the users the illusion of the vehicle being on its correct route. This will be done with the data stored in the database as the bus travels across the intended routes throughout the year. In the initial stages, when first gathering data, the accuracy will not be too great as we will consider the time of year, and for each period, it will take a year to get to the next set of data to "learn" from. Because of this, it may take us a few years until it gets to the accuracy we hope for, but as we continue to gather enough data, the accuracy of our predictive algorithm continues to grow as much as it can.

4.2.2 Ideation

The identification of these technology stacks is based on common industry practices and the specific needs of web application development. Here's how and why each option was identified:

1. **MEAN Stack:**
    a. MongoDB: NoSQL database for storing data in JSON-like documents.
    b. Express.js: Web application framework for Node.js, providing a robust set of features for web applications.
    c. Angular: Frontend framework for building dynamic web applications.
    d. Node.js: JavaScript runtime environment for server-side development.
2. **MERN Stack:**
    a. MongoDB: NoSQL database for storing data in JSON-like documents.
    b. Express.js: Web application framework for Node.js, providing a robust set of features for web applications.
    c. React: Frontend library for building user interfaces.

d. Node.js: JavaScript runtime environment for server-side development.

3. **LAMP Stack:**
    a. Linux: Operating system.
    b. Apache: Web server software.
    c. MySQL: Relational database management system.
    d. PHP: Server-side scripting language for web development.
4. **RDCS Stack:**
    a. React: Frontend library for building user interfaces.
    b. Django: Web framework for building web applications in Python, providing a clean and pragmatic design.
    c. MySQL: Relational database management system.
    d. CORS: enables secure communication between the React frontend and Django backend, even when served from different domains or ports.
5. **MEVN Stack:**
    a. MongoDB: NoSQL database for storing data in JSON-like documents.
    b. Express.js: Web application framework for Node.js, providing a robust set of features for web applications.
    c. Vue.js: Progressive JavaScript framework for building user interfaces.
    d. Node.js: JavaScript runtime environment for server-side development.

4.2.3 Decision-Making and Trade-Off

Weighted Decision Matrix

Criteria	**MEAN**	**MERN**	**LAMP**	**RDCS**	**MEVN**
Learning Curve/Difficulty	Low	Low	Moderate	Moderate	Low
Familiarity	High	High	Moderate	Moderate	Moderate
Flexibility	Moderate	Moderate	Moderate	Moderate	Low
Suitability	Moderate	Moderate	High	High	Moderate
Scalability	Moderate	Moderate	Moderate	Moderate	Low
Learning Opportunity	Low	Low	Moderate	Moderate	Low
Total	3/5	3/5	4/5	4/5	2/5

The Weighted Decision Matrix reflects the summarized ideation processes we had as a group when discussing the different tech stacks we could have used for our project. Most important to us was our familiarity with the languages/frameworks and their learning curves. We wanted to make sure we were using something manageable so we came to a great compromise that everyone was closely familiar with. Then again, we didn't want it to be too easy, so we decided to pick something we could still learn from and grow as programmers. That being said, the stack that gave us the most value was the RDCS stack.

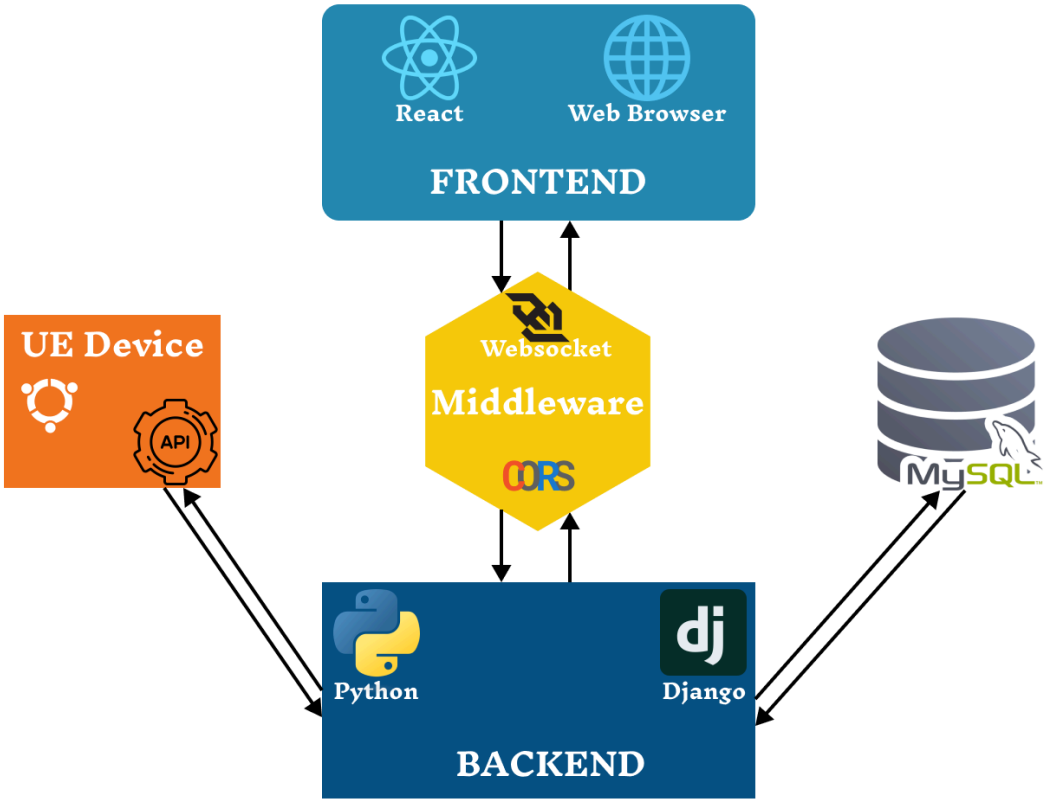
4.3 Proposed Design

4.3.1 Overview

CyRide visualization provides a web service for tracking bus routes and displaying user equipment connectivity using React. This service uses Leaflet API to display an interactive view of Ames, Iowa. Django provides all the data needed to be displayed on Leaflet, which uses multiple methods to provide accurate locations and predictions. The UE API and machine learning algorithms retrieve this data. Bus location data is stored in the MySQL database to be used in machine learning to deliver a seamless tracking experience when user equipment is out of range to transmit data.

To track bus locations, UE (User Equipment) is put onto buses and acts as a receiver while signals from data towers are transmitted around campus. This allows the application to gather longitude and latitude pairs to get the exact location of the bus by calling the UE API. When the UE is not in range of the signals, it goes into a dead state, which is where a predictive algorithm takes over to estimate where exactly the bus could be at a given time.

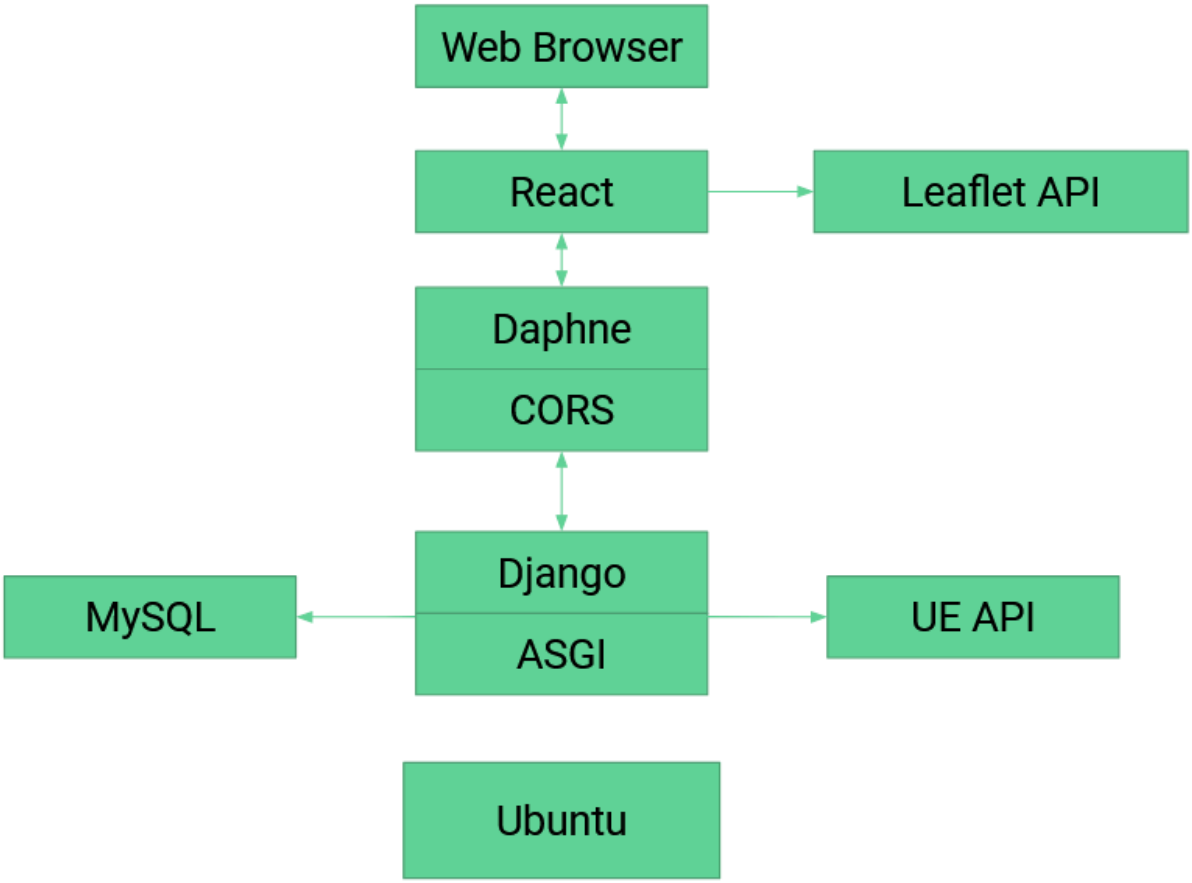
The estimation of bus locations will be done using past longitude and latitude pairs stored within the database. By using this data, with the power of basic machine learning, we can use that past data to approximate exactly where the bus might be at a given time. This allows the application to predict when UE devices will be connected to base stations. It also gives a seamless flow for the application so that buses will still be tracked even if UE connections are lost.



4.3.2 Detailed Design and Visual(s)

All data pertaining to buses is retrieved/calculated by Django. Django utilizes API calls to retrieve updated data from UE APIs for all UE's and inserts that data into the MySQL database. This relational database has a direct SQL connection with Django, where the data can later be retrieved and used in machine learning algorithms. The database will also store other information about buses that UE's collect including their direction and connectivity.

Django and React send data using Rest API calls and WebSocket connections. Django utilizes Daphne and Channels to create WebSockets connections. Daphne is a server deployment that handles HTTP and WebSocket protocol connections for an ASGI application. ASGI provides asynchronous support for programming Python on the web server. WebSocket connections are created for all necessary live updates on bus locations and their UE data. This ensures the Leaflet interface is constantly updated with the most bus data. Rest API calls are used to retrieve all other bus data that only need to be retrieved once per user request. To ensure that HTTP calls are secure, Django utilizes CORS, which receives incoming requests and checks if the origin can connect with Django.



Below are wireframes for the web browser that users would interact with to get bus location data. They show how the pathing of the bus would be displayed along a given route with insight on connectivity. A user can choose a specific route that a UE is on to receive data pertaining to those buses. The user can also select a specific bus on that route to get information about the bus location and UE connection.

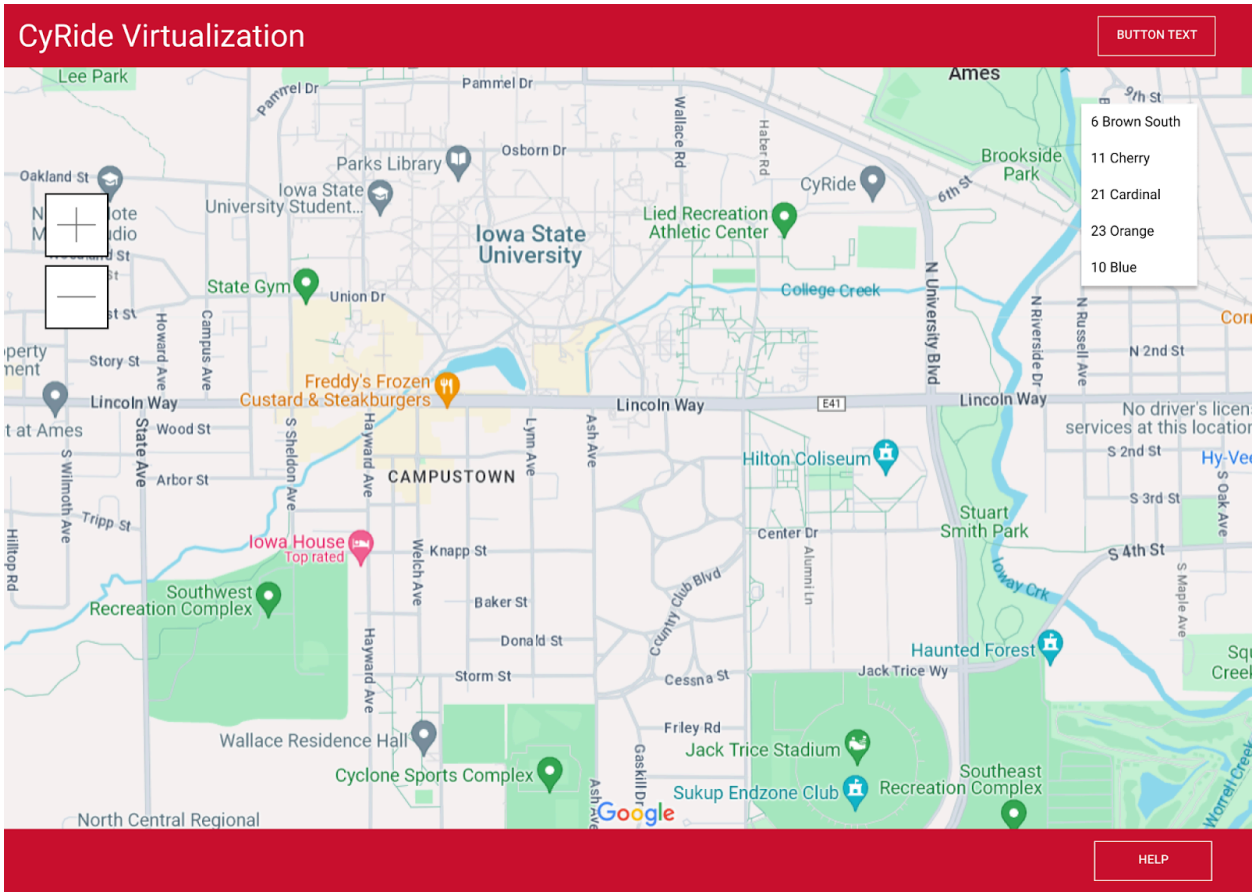


Figure 1. Default View

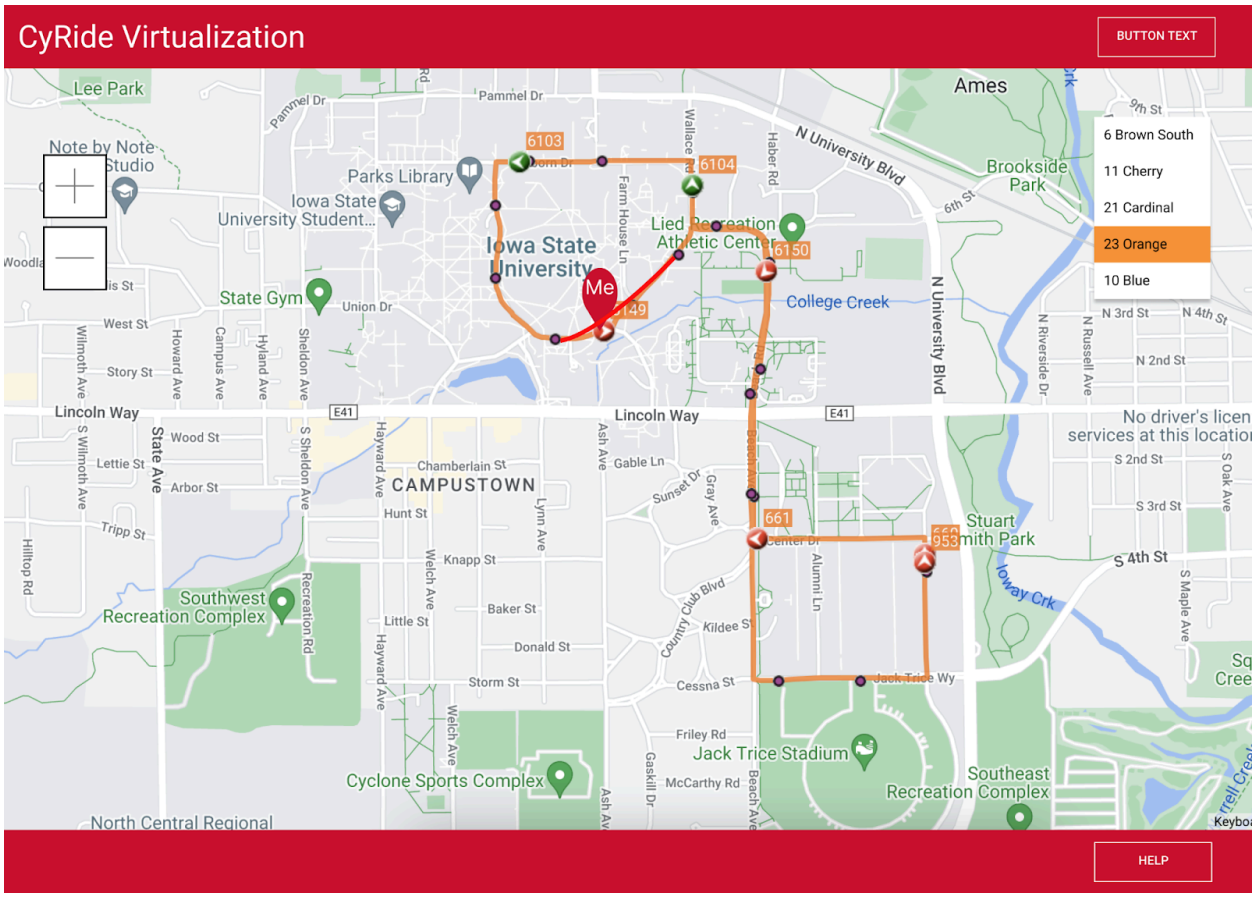


Figure 2. Selected UE Shows Bus in and out of coverage

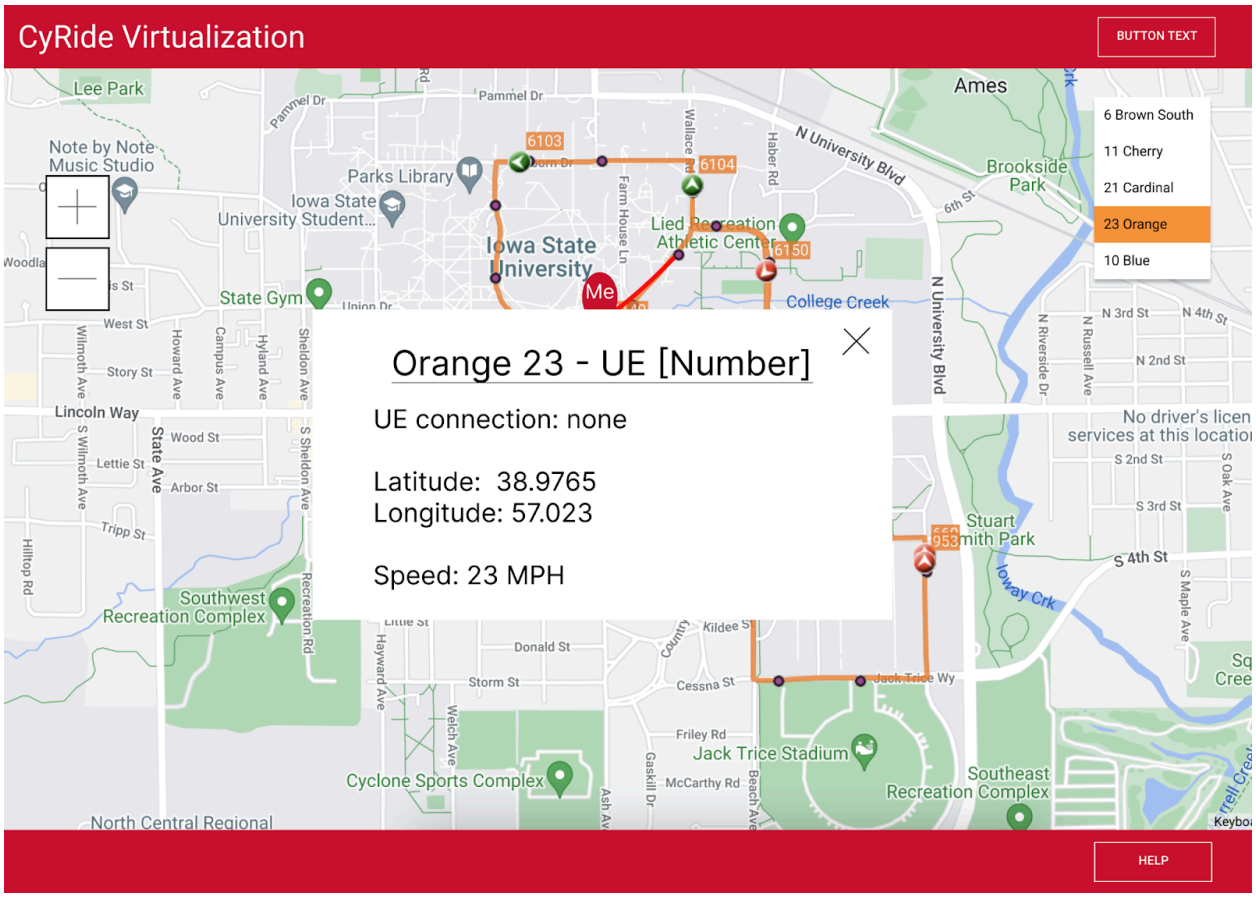


Figure 3. Shows specific metadata of bus

4.3.3 Functionality

When the system is available to users, they will be able to see all mobile UEs that are currently in use. They will be on buses tracking the routes while also giving insight into the connectivity and location prediction. Users can select a route they would like to view, which would isolate only those buses and their data. The user can go even further to select a single bus that would give information about that specific UE and its location. The route of a bus will change its color appearance for areas where the UE has no connection, and machine learning will provide location predictions. This gives users insight into when a UE will be connected again while also tracking the bus. This interaction with the user is displayed in the above Figma designs.

4.3.4 Areas of Concern and Development

The current design provides all the functionalities needed to satisfy design requirements and meet user needs. React provides a robust framework for developing user interfaces that can be translated into mobile development. Django's environment simplifies application development

and allows for easier integration of machine learning algorithms. A primary concern with this design is developing an efficient and accurate machine-learning algorithm that provides bus location data. To address this, we adapted Django, which has access to Python's vast library, to simplify development and integration.

4.4 Technology Considerations

TechStack: React-Django-CORS-SQL:

React (Frontend): Powerful Typescript library for building user interfaces. Allows developers to create reusable UI components that update efficiently when data changes. This is important for project success as it enables the development of a responsive and dynamic user interface, which is crucial for a modern web application. The downside of React is that there is a lack of documentation that doesn't cover the details of the framework. Angular is a similar componentizing framework that is harder to learn and understand but has detailed documentation.

Django (Backend): High-level Python web framework that's easy to deploy and scale. This is important for project success as Django provides a robust backend framework that simplifies the development of complex web applications. Python's versatility allows for easy integration with other technologies, such as Machine Learning. A weakness of Django is that it can be slow if the application is not properly optimized. A lack of convention in Python leads us to a configuration boilerplate that could slow down the development process. Ruby on Rails is another option with a better standard convention that is easier to develop.

CORS (Cross-Origin Resource Sharing): Used for same-origin communication between Django and React. Used to avoid Web Scraping and creating secure connections. This is important for project success as it ensures that the frontend and backend can communicate securely and helps prevent security vulnerabilities such as cross-site scripting (XSS) attacks. If CORS is configured incorrectly an attacker can look for known vulnerabilities in the CORS implementation and gain access to restricted resources. CORS is still a better option than another resource-sharing service like JSON-P, which detects errors after the data has been passed.

SQL (Structure Query Language): Django can only connect to Relational Databases, and SQL is familiar to the development team. This is important for project success as it ensures that the backend can efficiently store and retrieve data, and SQL's familiarity with the team will make development easier and more efficient. Another alternative to the project would be PostgreSQL.

4.5 Design Analysis

The current design progress is a fully functional tech stack deployed onto a remote server. This design utilizes communication between React and Django using WebSockets and the Rest API framework. Django also has a SQL connection to MySQL that automatically updates the database schema when object models are built or changed. Lastly, it has a functional map utilizing the Leaflet API, which displays location data. Currently, the deployment shows success for the implementation and will fulfill all requirements for the application. This design simplifies many aspects of development, making it easier to fulfill all user needs.

In the future an external connection will be made with the UE API to collect real data from UE's and display the locations on the UI. A machine learning model will be implemented to predicate bus locations even when UEs have no connection. This estimates when UEs will be connected and gives a seamless tracking experience. The bus locations will have to be updated smoothly in the user interface, and provide interfaces to view UE/location data of a given bus. We will also have to provide security updates by providing HTTPS support and utilizing CORS headers.

5 Testing

5.1 Unit Testing

Units being tested include individual components such as react components, server functionality, real-time bus tracking, machine learning algorithms, WebSocket communication, and database performance. Unit tests will be written using Jest for React components and Django's unittests for backend components.

5.2 Interface Testing

Interfaces in the design include communication interfaces between components (e.g., WebSocket for frontend-backend communication). The composition of two or more units/interfaces will be tested to ensure they work together correctly. Tools such as Postman or Swagger can be used for API testing.

5.3 Integration Testing

Integration testing is a crucial aspect of ensuring the smooth interaction between different components of our system. In our testing plan, we have decided to include integration tests using Jest. This decision is based on the specific needs and constraints of our project. Integration testing with Jest on the frontend will involve connecting multiple components of our React application and testing their functionality together. By including Jest in our testing strategy, we aim to ensure the reliability and functionality of our system while streamlining our testing process and allocating resources more efficiently to meet our project timeline and objective.

5.4 System Testing

System testing with Selenium involves using the Selenium framework to automate the testing of the complete, fully integrated product. Selenium allows us to simulate real user interactions with the software in a controlled testing environment that closely resembles the production environment. By automating these interactions, we can systematically navigate through all the features of the software, verifying that the end-to-end business or functional features work as expected. This approach provides us with a comprehensive understanding of the system's behavior and helps identify any issues that may arise from the interaction of different components.

5.5 Regression Testing

To ensure that new additions do not break old functionality, we rely on our testing pipeline, which runs automated tests before deploying any code changes. These tests are driven by requirements, ensuring that all critical functionalities, such as real-time data tracking and machine learning predictions, are thoroughly tested. Tools like Jest for React components and Django's unittests for backend components are used for regression testing. Overall, our testing pipeline ensures that any new code changes are thoroughly tested and do not introduce regressions in existing functionality.

5.6 Acceptance Testing

Regarding both the functional and non-functional requirements, our design requirements will be met as we are developing our milestones. We will know things are proceeding as planned through visual queues. If the data is being grabbed from the backend and the frontend is updating and processing in real-time as expected, we will have noticed that the requirements are being met and the project is working as expected. The client and advisor will be involved through their own testing of the application and making sure they use it without any issues. The app will be released to them in an alpha state and we can receive feedback if anything is not working as expected.

5.7 Security Testing

Our security testing plan includes defining the scope, identifying threats with threat modeling, and using tools like OWASP, ZAP and Nessus for vulnerability assessment and penetration testing. We conduct access control and data protection testing, analyze results, and generate detailed reports for mitigation. Regular testing and continuous improvement help us maintain a high level of security

5.8 Results

The testing approach for the project included unit testing for individual components like React and backend functionality, ensuring they worked as expected. Integration testing focused on interactions between components, excluding Selenium due to project needs. Regression testing prevented new code from breaking existing functionality. Acceptance testing involved stakeholders to confirm requirements were met. Security testing ensured the system's security and compliance with standards. Overall, the testing process confirmed the design's reliability, functionality, and security, meeting project requirements.

6 Implementation

- August 26 - September 9 | Review Work and Milestone 1
    - Make sure the team understands everything from where we left off on before break.
    - We were able to finish our first milestone and have a great start to understand the project’s functionality.
- September 9 - September 23 | UE Setup
    - Communicate with the team and the client and advisor about the UE device.
    - Make sure the we have access to the UE for data retrieval.
    - Replace the mock data to real-time data of the UE.
- September 23 - October 7 | UI Updates
    - Finish the data retrieval from the UE device.
    - Make sure that the frontend is able to call the data api in real-time and processes it live quickly.
- October 7 - October 21 | Complete Functionalities
    - Make sure constant updates and a live feed is being processed in a user friendly manner.
- October 21 - November 4 | Machine Learning
    - Get the functionality of if the bus is not in range of Data Towers for connection signals finished. This will not be started here as we used the summer time to research methods and we have continued to progressively implement them from the beginning of the semester.
    - This includes Machine Learning using previously processed data for these instances.
    - Make a way so that there is an automatic switch from live data and predictions.
- November 4 - November 18 | Testing and Touch Ups
    - Get the main functionality completed and try to tidy up a cohesive and working project
    - Testing will not only appear here since we will be testing throughout the development process, but will finish up around here and near the 100% mark.

- November 18 - December 2 | Integrate Additional Features
    - Try to get things that would help the user navigate or use the application
    - FAQ, Notifications, Pop-ups,
- December 2- December 16 | Get Presentation Together
    - Work on the final presentation.
    - Make sure everything that we did is covered.
    - What has been completed | What could not be completed | What still needs to be worked on | What needs more work

7 Professional Responsibility

7.1 Areas of Responsibility

Area of Responsibility	**Definition**	**NSPE Canon**	**SE Code of Conduct**
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence.	Perform services only in areas of their competence; Avoid deceptive acts.	Ensure that the software developed is usable, reliable, and scalable to fit our needs.
Financial Responsibility	Deliver products and services of realizable value and at reasonable costs.	Act for each employer or client as faithful agents or trustees.	Be honest and realistic when planning and stating the cost(s) of the project.
Communication Honesty	Report work truthfully, without deception, andunderstandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts.	Communicate transparently amongst each other, with the advisor & client, and with prospective users.
Health, Safety, and Well-Being	Minimize risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	Ensure the software is safe to use, and doesn't violate user safety or privacy.
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	Respect the intellectual property and rights or all team members and the client.

Sustainability	Protect environment and natural resources locally and globally.	Contribute to the advancement of sustainable development, including the well-being of present and future generations.	Strive to minimize the environmental impact by developing efficient software, and limiting its usage.
Social Responsibility	Produce products and services that benefit society and communities.	Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	Consider and factor into development the social consequences of software development and use, and promote the good and fair use of the project.

7.2 Project Specific Professional Responsibility Areas

Area of Responsibility	**Application**	**Team performance**	**Justification**
Work Competence	This area applies to our project, as it is paramount that we deliver high-quality work on time, and that all team members have expertise in their respective areas.	High	We are extremely ahead of track on the project, and have had few competence errors throughout the development process thus far.
Financial Responsibility	This area applies somewhat to our project. In using the Google API, calls can add up and end up being a pretty large expense, but we do have a budget given.	Medium	We've been doing alright in this area. We've been tracking the total cost, cost per day, and determining if these numbers make the application financially reasonable.
Communication Honesty	This area applies to our project, as its	High	We've been doing extremely well in this,

	necessary the advisor and client are aware of our work and can provide feedback as necessary.		documenting our development plans and work done throughout each week, going beyond the weekly reports outlined by the class.
Health, Safety, and Well-Being	This area doesn't apply to our project, as our web app is simply plotting and predicting data on a screen, with little user interference needed.	N/A	N/A
Property Ownership	This area applies somewhat to our project. Since the client has specific wants and needs, we need to mak sure those are accommodated.	Medium	We've been doing well in respecting he client's idea and goals with the project, and have implemented – or plan to implement – all of the suggestions given.
Sustainability	This area doesn't apply to our project too much, as we're developing a web app, which will have minimal impact on the environment.	N/A	N/A
Social Responsibility	This area applies to our project, as we have a responsibility to serve a functional product to the ARA team, which can be used to better Iowa.	Medium	Though we've been doing well with implementing the application with the user in mind, we still have some work to do on the documentation of the project to make it easily accessible.

7.3 Most Applicable Professional Responsibility Area

In all, “Communication Honesty” is our selected area that is most applicable to our project. Since the application is being built for the ARA research team, it is most important that we follow the guidelines given and listen to all advice. Likewise, it is also extremely important that we are transparent throughout the entire process of development, to ensure that the product is as described in the end.

8 Closing Material

8.1 Discussion

Thus far, we have developed a full-stack application using the React–CORS–Django–MySQL tech stack. The MySQL database is populated with mock data, which can be modified with CRUD operations in Django. The React app is able to read and display the data via CORS, which is currently displayed on a custom implementation of the Leaflet UI, which includes some specific information about our project. Our current implementation is a baseline, and will be custom-fit to adhere to the requirements and restraints outlined in the project description in the fall semester.

8.2 Conclusion

In all, we feel as a team that a solid start has been made on the project. For the fall semester, we strive to link the UE device from the ARA team to our application, and display the data in live time. The ARA team has also emphasized the importance of having this application work for all mobile UE devices, irrespective of the vehicle type it is deployed on, so, having support for all vehicles is another goal. Our last goal is to train and implement the GRU Machine Learning model and display the predictions on our application’s frontend. Achieving all of these goals will mark the completion of the CyRide Visualization Project.

To best complete these goals, our first step is to have round-tripped connectivity between our app and the mobile UE’s API. This allows us to access all of the data associated with the mobile UE, which can then be serialized and displayed on our application’s frontend. After this connectivity is established, we can then utilize the testing methods outlined in Section 6 to ensure our application works as intended with real-time data. Then, our final steps would be to implement the selected Machine Learning model, and train it to predict future data sets, followed by adding a predictivity view on our application.

8.3 References

[1] "The POWDER Manual," docs.powderwireless.net. [Online Manual] Available: https://docs.powderwireless.net/ [Accessed April 19, 2024].

9 Team

9.1 Team Members

Evan Schlarmann, Endi Odobasic, Braden Buckalew, Andrew McMahon, **Chiran Subedi**

9.2 Required Skills Sets

1. Python Development
    a. Django is built using Python and requires skills to develop the framework to retrieve data and use machine learning models.
2. TypeScript Development
    a. TypeScript was chosen as the language to construct React and forms the basis for the website communication.
3. SQL Development
    a. Even though Django provides easy development with SQL, we must know the basics to understand relationships and querying efficiency.
4. Machine Learning
    a. To provide location predictions we must be able to implement a machine learning model that is accurate.
5. React knowledge
    a. The website is built using React, so the team must be comfortable working with the framework.
6. Ubuntu environment
    a. The project is hosted on Ubuntu, requiring knowledge about the environment and all necessary commands.

9.3 Skills Sets Covered

1. Python Development
    a. Evan, Andrew
2. TypeScript Development
    a. Braden, Endi
3. SQL Development

    a. Braden, Evan, Andrew
4. Machine Learning
    a. The whole team is learning the fundamentals
5. React knowledge
    a. Braden, Endi
6. Ubuntu environment
    a. Braden, Evan

9.4 Project Management Style Adopted

Agile development was selected for the team's project management style. This allows everyone to have flexible tasks to transition quickly whenever roadblocks arise. It also ensures that the project can adapt to meet new requirements while providing a working application at the end of every sprint. The client can see functional progress toward the final project and give feedback so every piece can be refined during development.

9.5 Initial Project Management Roles

- Evan Schlarmann
    - CI/CD pipeline and Django
- Endi Odobasic
    - React and Leaflet API
- Braden Buckalew
    - React and Testing
- Andrew McMahon
    - Django and MySQL

9.6 Team Contract

Team Members:
1) Evan Schlarmann
2) Endi Odobasic
3) Braden Buckalew
4) Andrew McMahon
5) Chiran Subedi

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

- Follow the When2Meet Schedule for times when everyone is available to meet.
- Use a the best method to meet on a weekly basis
    - We meet and communicate through Discord or In-Person (whenever available).
- We all agree together for the things we should implement or work on.

- We have weekly reports and meeting minutes to keep track of what we worked on.

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
We will have a Discord server designated for our group only, and we will be able to ping/message each other in case of any questions or concerns. But also, if anyone needs help, one can easily ask in the chat for assistance or clarification. Communication with clients and our advisor will be down over email using our senior design group email address.

3. Decision-making policy (e.g., consensus, majority vote):
We will use a consensus style of decision, making sure that way everyone on the team is involved and we can work around decisions we are able to or not able to include based on requirements.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
We will have a designated Google Doc to hold all the times we meet together as a team. There will be a section for in-person meetings and another for virtual for any times we can't all make it in person.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
Make sure to be in attendance at all meetings at the time agreed upon with the team. Team members should actively participate in meetings and mention things that have been working on and what they will work on following the next meeting. Bring your own thoughts and ideas to collaborate on where the project will head.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
Individuals need to make sure to manage time efficiently in order to fulfill assigned tasks from meetings before. This is for all of the assignments that are assigned over the course of the semester. Should not wait until the last minute in case of clarification or help.

3. Expected level of communication with other team members:
Communication from team members should be open. The team should make frequent progress updates on work being done throughout the semester.

4. Expected level of commitment to team decisions and tasks:
If team members have any ideas or comments on decisions, they are encouraged to voice their thoughts. Sometimes, team members might not have other ideas they can think of, but they could build off others or contribute to the conversation in different ways.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

1) Evan Schlarmann: Team Organization
2) Endi Odobasic: Client Interaction
3) Braden Buckalew: Individual Component Design
4) Andrew McMahon: Testing
5) Chiran Subedi: Testing

2. Strategies for supporting and guiding the work of all team members:
Discuss any roadblocks that need to be solved during meetings so that all members can continue contributing to the project. Communicate possible solutions if anyone gets stuck during the week so they don't fall behind before the next meeting.

3. Strategies for recognizing the contributions of all team members:
Discuss how processes and decisions were made either by team or individuals. With the goal of talking about if, as a team or individual if they could do anything better and can learn from mistakes or different ideas.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
1) Evan Schlarmann: Done previous work in full stack development using Vue and Springboot to create new web applications. Has completed testing for multiple applications with unit, modular, and system testing. I have the most experience with Java applications and varying experience in many other programming languages.
2) Endi Odobasic: I had done work before with the MERN stack in creating web applications. Additionally, I have also done coursework involving work with C/C++, Java, SQL, Android, and Testing.
3) Braden Buckalew: Previous work with regression testing and unit testing, Full Stack Developer with Angular and .NET Framework
4) Andrew McMahon: I've worked primarily with Java and Python in backend development, but have had some exposure with JavaScript and its relative packages in frontend development. I also have experience with Shell Scripting, C, C++, and SQL.
5) Chiran Subedi: I have experience in full-stack development, UI/UX design, and building responsive web applications.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

As a team working conclusively, we will try to open the team for inclusivity so that everyone can contribute with their own amazing ideas. Also, we will make sure to communicate and recognize the work that other group members have done. We will avoid judging others' thoughts and ideas. To add, we will not criticize others who speak up that way so we don't discourage members from speaking up in the future.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
We will create a discord text channel labeled “Issues” where we can post our individual uses or group issues and use reactions to show the status of the Issue (read, resolved, need discussion). We want everyone to communicate how tasks are going so that we can help each other during the week to get everything assigned done before the next meeting. Constant communication is important to make sure everyone is up to date and providing collaboration for the team. Our discord is open for all discussions about this project if anything needs to be discussed.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:

- Grow team cohesion
- Apply individual skills to formulate a group-oriented plan of action for S E 492
- Gain experience using the Engineering Design Process
- Develop interpersonal skills to give in-depth presentations and clear communication with clients
- Learn about new technologies to optimize our project to meet necessary constraints given by our client

2. Strategies for planning and assigning individual and teamwork:
Give work out to individuals based on their talents and experience so they can contribute most efficiently. We want teamwork to be spread out evenly and make sure that everyone has a chance to work on problems they find interesting or want to learn more about.

3. Strategies for keeping on task:
Writing goals and tasks needed to get done before each group work session. Always update what work has been done for the project so that everyone is up to date on completed tasks. We will be using Gitlab Milestones to keep track of Tasks that need to be completed.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

First-time infractions will be lenient as unavoidable events may cause someone to miss a meeting or not perform on their part. We will try and communicate with the individual to see if we can help put them back on track so they can adhere to the rest of their obligations.

2. What will your team do if the infractions continue?

If one continues to not adhere to the contract then steps will be taken to connect the individual with the class professors or TAs to try and get them back on track or resolve any issues that may be occurring.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) Evan Schlarmann DATE 1/28/2024
2) Braden Buckalew DATE 1/28/2024
3) Endi Odobasic DATE 1/28/2024
4) Andrew McMahon DATE 1/28/2024
5) Chiran Subedi DATE 09/02/2024