2 Project Plan

2.1 Project Management/Tracking Procedures

We will be using SCRUM as a framework for our project's development and weekly lifecycle. Following SCRUM and the agile methodologies, we will complete our weekly status updates. Furthermore, each week after our client meeting, the team will go over issues/impediments from the previous week and address them. Development and planning work is expected to be expressed clearly in the weekly status reports and in standup after each client meeting, and the work for development should be reflected in a story on the team's Trello board. We chose SCRUM because it is the most familiar project management style, and fits closely with our goals and objectives as developers to complete the project and facilitate communication.

Our project will be using Gitlab for version control. We will also be using Trello to help track work, progress, and aid in our standup meetings. Our primary means of communication will be Discord, and secondarily we will use emails when a group member is needed.

2.2 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

Task	Description
Implement Visualization Tool Front-End	Create a user interface for dispatchers and drivers to view routes and stops
Develop UI for Web App	Create a user interface for dispatchers to view orders and routes, communicate with drivers, and input changes manually
Develop UI for Mobile App	Create a user interface for drivers to view orders and routes, communicate with dispatchers, and indicate when a package has been delivered.

Develop REST API microservices	Design and implement application functions into multiple microservices which will communicate with the frontend, db and external services.
Setup application DB	Setup db configs and communication with the API.
Setup application server	Setup server to host API and config changes.
Final Application Testing	Testing individual components of the web app, mobile application, and the microservices.

Subtasks

Implement Visualization Tool Front-End
- Display external map (i.e. Google Maps, etc)
- Overlay routes (color-coded by truck)
- Overlay stops on routes (color-coded by truck)
- Be able to hide/show specific routes

Develop UI for Web App
- Create home page
- Create visualization page
- Create communication page
- Create order overview page
- Create route update/modify page

Develop UI for Mobile App
- Create main screen
- Create visualization screen
- Create communication screen
- Create order list and stops screen
- Implement route-update notifications
- Create “package delivered” input screen

Develop API microservices
- Create communication service
- Create truck allocation service and route allocation service
- Create user account (login, registration, settings) service
- Create user order service. (New user order)
- Create user order tracking service
- Create Q&A service
- Setup microservice communication with external services (Google maps, Department of transportation etc)

Setup application DB
- Setup db connection to microservices
- Setup db configs (SQL dialect, security configs, data handling configs)

Setup application server
- Setup server to host api services
- Setup server configs

Final Application Testing

- Test the web app and the mobile application for navigation between views and other inconsistencies.
- Test the web app and the mobile application for correctly receiving data from the backend
- Test the algorithm using real time metrics.
- Test the individual microservices for their respective functions.
- Setup a testing environment to see if all the components communicate with each other flawlessly and achieve desired results.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

What are some key milestones in your proposed project? It may be helpful to develop these milestones for each task and subtask from 2.2. How do you measure progress on a given task? These metrics, preferably quantifiable, should be developed for each task. The milestones should be stated in terms of these metrics: Machine learning algorithm XYZ will classify with 80% accuracy; the pattern recognition logic on FPGA will recognize a pattern every 1 ms (at 1K patterns/sec throughput). ML accuracy target might go up to 90% from 80%.

In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprint).

Metrics of interest:

- UI and visualization tool usability
- Algorithm update speed (in response to dynamic changes)
- General algorithm efficiency

Evaluation criteria:

- UI and visualization tool usability
    - Create questionnaire for users
    - Responsiveness of UI
    - Accuracy of visualization tool
- Algorithm update speed
    - Time for changes to be returned
- General algorithm efficiency
    - Compare miles traveled, time driving, packages delivered between a brute force algorithm and our implementation
- Algorithm scalability
    - Ease of adding new drivers or dispatchers (measured in change in efficiency values and update speed as more drivers/dispatchers are added)

Milestones:

- Baseline functional UI
- Alpha UI (first round of user feedback)
    - UI responds to input in under 500 ms
    - Visualization tool at least 75% accurate
- Beta UI (second round of user feedback)
    - UI responds to input in under 250 ms
    - Visualization tool at least 90% accurate
- Polished UI
    - UI responds to input in under 100 ms
    - Visualization tool at least 98% accurate
- Algorithm can make updates in under 20 seconds
- Algorithm can make updates in under 10 seconds
- Algorithm can make updates in under 5 seconds
- Algorithm can make updates in under 1 second
- Algorithm efficiency is better than brute force approach
- Algorithm is 15% more efficient than brute force approach
- Algorithm is 25% more efficient than brute force approach
- Algorithm is 50% more efficient than brute force approach
- Doubling the number of drivers/dispatchers reduces miles traveled and time driving by 10%
- Doubling the number of drivers/dispatchers reduces miles traveled and time driving by 25%
- Doubling the number of drivers/dispatchers reduces miles traveled and time driving by 50%

2.4 Project Timeline/Schedule

• A realistic, well-planned schedule is an essential component of every well-planned project

• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

This project will entail a continuous cycle of brainstorming, research, prototyping, testing, refining, and demonstrating. We plan to have the first prototype finished by the end of week 5, after which new prototypes will be developed in 2-to 3-week intervals. With each successive prototype, we will make steps from experimenting with possibilities to refining a finished product with both the desktop and mobile ui, so that the final prototypes assembled in the spring will be a fully-functional system that meets all requirements and constraints and is able to implement the algorithm in a seamless manner.

2.5 Risks And Risk Management/Mitigation

Consider for each task what risks exist (certain performance target may not be met; certain tool may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

<u>Risks for each task:</u>

- Implement Visualization Tool on Front-End
    - The biggest risk here is that we struggle to correctly implement the Vehicle Routing algorithm into the application, or that it takes longer than originally anticipated. This could take some time to re-plan and evaluate how to handle the situation. However, we have team members that are confident in figuring out the algorithm so this is not likely to happen. Risk probability: 0.3
- Develop UI for Web App
    - The biggest risk here is correctly retrieving data from the backend. This shouldn't be a big issue because we have members experienced of Angular and React. Risk probability: 0.3
    - Another risk is that we simply run into defects in the UI that cause us to re-plan. Even with team members experienced in UI development for a particular framework, weird defects can always arise, though most of them won't take long to solve. Risk probability: 0.2
- Develop UI for Mobile App
    - The biggest risk with the mobile app UI is that it doesn't meet our standards of appearance. In other words, we may run into issues where the mobile app UI can't look exactly like the mobile version. Risk probability: 0.3
    - Another risk is in what mobile UI the team decides to use (Android vs iOS). We need to be careful in this decision so we don't run into any major issues during development due to lack of experience. Risk probability: 0.2
- Develop REST API microservices
    - Since backend development is the bridge between frontend and the DB, the biggest risk/challenge is that frontend/backend communication or backend/DB communication is not working. This would set the team back some time to focus on the issue and get it resolved, may require some re-planning. Risk probability: 0.4
- Setup application DB
    - The only risk here is the decision on whether or not to use a SQL database or no-SQL. Similar to mobile development, this decision depends on the experience of the team and will require time and effort to consider. We have members that are experienced working with SQL, so this should likely not be an issue. Risk probability: 0.1

- Setup application server
    - There's hardly any risk to just setting up the application server. The biggest risk is that the server can't run for whatever reason, which may require us to take a deeper look simply using online resources or discussion. Risk probability: 0.1

2.6 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in the total number of person-hours required to perform the task.

Task	Description	Time (person-hours)
Implement Visualization Tool Front-End	Create a user interface for dispatchers and drivers to view routes and stops	100
Develop UI for Web App	Create a user interface for dispatchers to view orders and routes, communicate with drivers, and input changes manually	80
Develop UI for Mobile App	Create a user interface for drivers to view orders and routes, communicate with dispatchers, and indicate when a package has been delivered.	80
Develop REST API microservices	Design and implement application functions into multiple microservices which will communicate with the frontend, db and external services.	100
Setup application DB	Setup db configs and communication with the API.	20
Setup application server	Setup server to host API and config changes.	100
Final application testing	Testing individual	20

	components of the web app, mobile application, and the microservices.	

2.7 Other Resource Requirements

Identify the other resources aside from financial (such as parts and materials) required to complete the project.

There should not be a need for any additional resources outside of the software and database solutions that are gotten for our project. This is due to the fact that our project is a software project, so anything besides this will not be necessary. The only thing that could be considered for this project would be android mobile devices that could be used in order to ensure that our mobile app software works correctly on different android devices and with different versions of phones. However, development tools such as android studio come with an emulator in order to remedy some of this hassle during the development process, so it is unsure at this point whether or not any android devices will be necessary.