Dynamic Assignment of Trucks to Delivery Requests

Design Document

SDMAY22-32
Client:Goce Trajcevski
Adviser: Goce Trajcevski
Joshua Heroldt - Team organization, Client interaction
Bernard Fay - Database management, Meeting Scribe
Nolan Slimp - Scrum master
Asma Gesalla - Backend documentor
Matthew Medley - Website manager, Frontend architecture design
Indrajeet Roy - Frontend documentor
Siddharth Rana - Individual component design

sdmay22-32@iastate.edu
sdmay22-32.sd.ece.iastate.edu

Revised: December 2021/Version 1

Executive Summary

Development Standards & Practices Used

For software development:

- Scrum methodology
  - https://scrumguides.org/scrum-guide.html
  - https://www.scrum.org/resources/professional-scrum-developer-glossary
- IEEE 610.12, Standard Glossary of Software Engineering Terminology
- IEEE 1540: Software Risk Management

For software testing:

- IEEE 1012: A standard for Software Verification and Validation.
- IEEE 1061: A methodology for establishing quality requirements
- IEEE 1008: Unit testing standard

For working with coordinate systems:

- GPS Coordinates will use the UTM or WGS84 format for representing geolocated points
- Mercator projection and Map matching have no agreed-upon standards, so we will follow conventional projection formulas

Summary of Requirements

- Set of trucks and delivery requests
- For each truck
  - Initial location
  - Delivery location (target)
  - Goods being transported
  - Capacity of the truck (weight of goods that can be carried)
  - Load (amount of goods being carried/transported on the truck)
- Generate route for each truck
- Based on the route: Estimate truck location at any given point of time
- Cater to the dynamic updates:
  - New pickup/delivery request

    - Broken truck at any given time
- Reassign the rest of the trucks from the fleet as a result of the dynamic updates
- UI requirements
    - Dispatcher (Desktop) UI
    - Mobile app for drivers of trucks
    - Intuitive for both
- Notifications to both drivers and dispatchers
    - New route
    - Customer updates if a delivery is delayed
- Constraints
    - Response time (Seconds to a minute of response time for dynamic updates)
    - Assuming the availability of road network maps and other traffic distribution data (traffic density) -> Needed for any assignment (both initial and dynamic)
    - Economics:
        - Minimize delivery delay as a result of a dynamic update
        - Minimize idle time of trucks
    - Resource requirements
        - Need a server to be running constantly to host the database and requests as well as running the assignment algorithm
        - Android mobile device
        - Visualization tools/frameworks

Applicable Courses from Iowa State University Curriculum

- COMS 228
- SE 319
- COMS 309
- SE 339
- COMS 311
- SE 329
- COMS 363

New Skills/Knowledge acquired that was not taught in courses

- React
- Mapbox

Table of Contents

1 Team 7

1.1 Team Members 7

1.2 Required Skill Sets for Your Project 7

1.3 Skill Sets covered by the Team 7

1.4 Project Management Style Adopted by the team 8

1.5 Initial Project Management Roles 8

2 Introduction 9

2.1 Problem Statement 9

2.2 Requirements & Constraints 9

2.3 Engineering Standards 10

2.4 Intended Users and Uses 10

3 Project Plan 17

3.1 Project Management/Tracking Procedures 17

3.2 Task Decomposition 18

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria 18

3.4 Project Timeline/Schedule 20

3.5 Risks And Risk Management/Mitigation 22

3.6 Personnel Effort Requirements 23

3.7 Other Resource Requirements 24

4 Design 24

4.1 Design Context 24

4.1.1 Broader Context 24

4.1.2 User Needs 25

4.1.3 Prior Work/Solutions 25

4.1.4 Technical Complexity 26

4.2 Design Exploration 27

4.2.1 Design Decisions 27

4.2.2 Ideation 27

4.2.3 Decision-Making and Trade-Off 27

4.3 Proposed Design 28

4.3.1 Design Visual and Description 28
4.3.2 Functionality 30
4.3.3 Areas of Concern and Development 33
4.4 Technology Considerations 34
4.5 Design Analysis 35
4.6 Design Plan 36
5 Testing 36
5.1 Unit Testing 36
5.2 Interface Testing 37
5.3 Integration Testing 39
5.4 System Testing 39
5.5 Regression Testing 39
5.6 Acceptance Testing 40
5.7 Security Testing (if applicable) 40
5.8 Results 40
6 Implementation 42
7 Professionalism 44
7.1 Areas of Responsibility 45
7.2 Project Specific Professional Responsibility Areas 47
7.3 Most Applicable Professional Responsibility Area 48
8 Closing Material 49
8.1 Discussion 49
8.2 Conclusion 49
8.3 References 50
8.4 Appendices 50
8.4.1 Important Resources 50
8.4.2 Team Contract 50

List of figures/tables/symbols/definitions

Figure 2.4.1. Single Destination Use Case 11

Figure 2.4.2. Multiple Destinations Use Case 12

Figure 2.4.3. Multiple Destinations with Multiple Routes Use Case 13

Figure 2.4.4. Route Recalculation Use Case 14

Figure 2.4.5. Multiple Warehouses with Multiple Routes and Multiple Destinations Use Case 15

Figure 2.4.6. Truck Reallocation on Truck Break Down 16

Table 3.2.1. Task Decomposition 18

Figure 3.4.1. Gantt Chart 21

Table 3.6.1. Task-Effort Decomposition 23

Table 4.1.1.1. Responsibility Considerations 24

Figure 4.3.1.1. System Architecture 28

Figure 4.3.2.1. Initial Routes Calculated by Algorithm 31

Figure 4.3.2.2. Pink Truck Breaks Down 32

Figure 4.3.2.3. Load Divided Between Blue and Orange Trucks. Routes Recalculated Accordingly. 33

Figure 5.8.1: Unit testing 41

Figure 5.8.2: Testing plan 41

Table 7.1.1. ACM Areas of Responsibility 45

Table 7.1.2. Differences in ACM and NSPE code of ethics 46

1 Team

1.1 Team Members

Joshua Heroldt

Bernard Fay

Nolan Slimp

Asma Gesalla

Matthew Medley

Indrajeet Roy

Siddharth Rana

1.2 Required Skill Sets for Your Project

1. Database skills
2. API skills
3. Web development skills
4. Mobile application development skills
5. Inter-process communication
6. Program efficiency skills

Enumerating more on the specific requirements for this project

- Develop a matching algorithm that either falls under the classical or evolutionary approach to solving a vehicle routing problem
- Ability to work with front end applications to develop a web application as well as a mobile application
- Database skills that allow for multiple tables to be modified and accessed in real time
- API skills in order to get real-time traffic data from a city in the United States
- Some manor of connection between the applications and the developed algorithm in order to issue real-time notifications to multiple parties that might be using the app
- Making the UI intuitive for multiple types of users

1.3 Skill Sets covered by the Team

1. Database skills
    a. Nolan Slimp
    b. Siddharth Rana
    c. Matthew Medley
    d. Bernard Fay
2. API skills
    a. Joshua Heroldt

    b. Siddharth Rana
    c. Matthew Medley
    d. Bernard Fay
3. Web development skills
    a. Joshua Heroldt
    b. Nolan Slimp
    c. Matthew Medley
4. Mobile application development skills
    a. Joshua Heroldt
5. Inter-process communication
    a. Indrajeet Roy
    b. Asma Gasella
    c. Bernard Fay
6. Program efficiency skills
    a. Joshua Heroldt
    b. Bernard Fay

1.4 Project Management Style Adopted by the team

The team will be using a scrum management style that is an offshoot of the typical agile methodology. This is because we want to emphasize the importance of regular communication with the team in order to make sure everyone knows the status and problems related to the project. This way if any team member is stuck it will allow for an easy way for other group members to offer advice and help to anyone if their skill sets match the problem currently being faced by the team. This will work especially well for our group seeing how many different subteams will need to be created to work on the various different aspects of the project.

1.5 Initial Project Management Roles

- Joshua Heroldt
    - Team organization
    - Client interaction
- Bernard Fay
    - Database management
    - Meeting Scribe
- Nolan Slimp
    - Scrum Master
- Asma Gesalla
    - Backend documentor
- Matthew Medley
    - Team Website Manager
    - Frontend architecture design
- Indrajeet Roy
    - Frontend documentor
- Siddharth Rana
    - Individual component design

2 Introduction

2.1 Problem Statement

Develop a system that will enable different classes of users to participate in route assignments for delivery trucks. The fundamental algorithm question we are trying to solve is how to assign a truck from a given fleet to a new request, or how to re-assign truck(s) to respond to dynamic changes in traffic or if a truck breaks down. The rest of the tasks will involve implementing the algorithmic solutions, the user interface, and integrating map data with request data to generate routes.

2.2 Requirements & Constraints

In the system, there will be a set of trucks and delivery requests that base all of our constraints. Each truck will have an initial location, delivery location, goods being transported, capacity of the truck, and current load. Then, the system must generate the optimal route for each truck. Based on the route, the user must be able to estimate the location of the truck at any time, and know its remaining distance to a delivery location. The route must be able to update to any new pickup location and delivery requests (**constraint**). It also needs to update if any truck breaks down on route at any given time (**constraint**). Based on these two constraints, dynamic updates will be made to reassign the rest of the trucks from a given fleet. These dynamic updates must be made in less than a minute response time (**constraint**). The user interface will consist of a dispatcher web application and a mobile application for the drivers of each truck. Notifications will be sent to both the drivers and dispatchers when given a new route or customer updates for delayed deliveries. The system will also be efficient, and must minimize route time for dynamic updates, idle time of a truck, and initial assignments (**constraint**). This project assumes we have access to road network maps and other traffic information; we will rely on this data and it is necessary to assign trucks (**constraint**).

- Set of trucks and delivery requests
- For each truck
  - Initial location
  - Delivery location (target)
  - Goods being transported
  - Capacity of the truck (weight of goods that can be carried)
  - Load (amount of goods being carried/transported on the truck)
- Generate route for each truck
- Based on the route: Estimate the location of truck at any given point of time
- Cater to the dynamic updates:
  - New pickup/delivery request
  - Broken truck at any given time
- Reassign the rest of the trucks from the fleet as a result of the dynamic updates
- UI requirements

    - Dispatcher (Desktop) UI
    - Mobile app for drivers of trucks
    - Intuitive for both
- Notifications to both drivers and dispatchers
    - New route
    - Customer updates if a delivery is delayed
- Constraints
    - Response time (Seconds to a minute of response time for dynamic updates)
    - Assuming the availability of road network maps and other traffic distribution data (traffic density) -> Needed for any assignment (both initial and dynamic)
    - Economics:
        - Minimize delivery delay as a result of a dynamic update
        - Minimize idle time of trucks
    - Resource requirements
        - Need a server to be running constantly to host the database and requests as well as running the assignment algorithm
        - Android mobile device
        - Visualization tools/frameworks

2.3 Engineering Standards

For software development:

- Scrum methodology
    - https://scrumguides.org/scrum-guide.html
    - https://www.scrum.org/resources/professional-scrum-developer-glossary
- IEEE 610.12, Standard Glossary of Software Engineering Terminology
- IEEE 1540: Software Risk Management

For software testing:

- IEEE 1012: A standard for Software Verification and Validation.
- IEEE 1061: A methodology for establishing quality requirements
- IEEE 1008: Unit testing standard

For working with coordinate systems:

- GPS Coordinates will use the UTM or WGS84 format for representing geolocated points
- Mercator projection and Map matching have no agreed-upon standards, so we will follow conventional projection formulas

2.4 Intended Users and Uses

The primary beneficiaries of our project are customers, truck dispatchers, and truck drivers. While not directly used by the customers, the project will have the general impact of ensuring timely deliveries. The project will be more directly used by dispatchers as it will aid them in deciding initial routes for trucks as well as making decisions and adjustments in the case of changing circumstances (traffic, new orders, truck breakdowns, etc.). This will benefit the dispatcher by reducing the stress of unpredictable circumstances and having to make quick decisions. Lastly, the project will be used by truck drivers to receive their assignments and any changes that occur throughout the day due to traffic, new orders, breakdowns, etc. This will benefit truck drivers by reducing the amount of time they spend making deliveries by optimizing routes, reducing waiting times for updated assignments, and minimizing the amount of time spent making deliveries.

POTENTIAL APPLICATION USE CASES

1. BASE USE CASE

   The base use case is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, the order route is between the cargo origin point and a single destination point (in comparison to multiple destination points) and external factors such as traffic, vehicle malfunction are not present.

   A customer's input order is allocated to a dispatcher via the allocation algorithm and the dispatcher then relays the order to a truck driver. Communication between the dispatcher and the truck driver is facilitated by the web and mobile platforms communication/notification system. The truck driver's route from the order origin pick up point to the destination point will be determined by the routing algorithm. Post order delivery to destination, the truck driver notifies the dispatcher, who subsequently notifies the customer that the order has been successfully delivered to the destination point, as per the customer's order input.

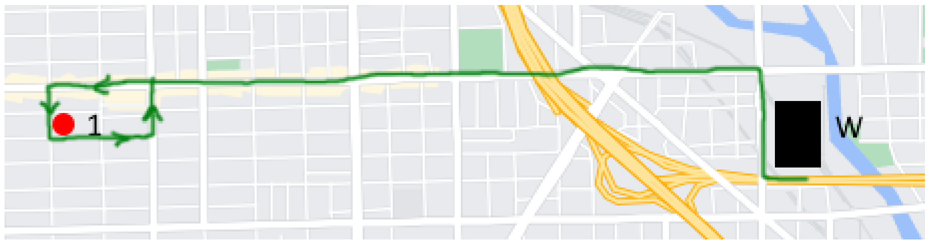


Figure 2.4.1. Single Destination Use Case

   Given warehouse W and order 1, the algorithm will generate the depicted route and present it to the dispatcher. The dispatcher will then inform the truck driver of the order and the route to be taken through the web application. The truck driver will receive this information through the mobile application.

2. MULTIPLE DESTINATION USE CASE

   The multiple destination use case is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, the order route is between the cargo origin point and multiple destination points and external factors such as traffic, vehicle malfunction are not present.

A customer's input order is allocated to a dispatcher via the allocation algorithm and the dispatcher then relays the order to a truck driver. Communication between the dispatcher and the truck driver is facilitated by the web and mobile platforms communication/notification service. The truck driver's route from the warehouse to each of the delivery locations will be determined by the routing algorithm. Post order delivery to the final destination point, the truck driver notifies the dispatcher, which subsequently notifies the customer that the order has been successfully delivered to the destination point, as per the customer's order input.

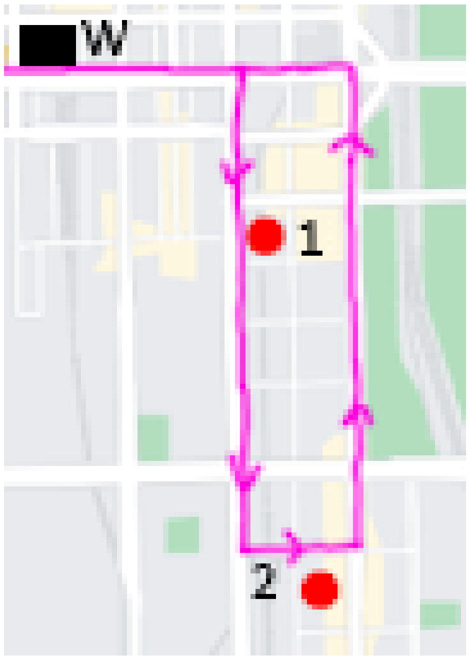


Figure 2.4.2. Multiple Destinations Use Case

Given warehouse W, orders 1 and 2 at the locations depicted, and one truck, the algorithm would output the above route given the locations of the orders. The dispatcher will then inform the truck driver of the route to be taken through the web application. The truck driver will receive this information through the mobile application.

3. MULTIPLE DESTINATION WITH MULTIPLE ROUTES USE CASE

The multiple destination use case is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, the order route is between the cargo origin point and multiple destination points and external factors such as traffic, vehicle malfunction are not present.

A customer's input order is allocated to a dispatcher via the allocation algorithm and the dispatcher then relays the order to a truck driver. Communication between the dispatcher and the truck driver is facilitated by the web and mobile platforms communication/notification service. The truck driver's route from the warehouse to each of the delivery locations will be determined by the routing algorithm. Post order delivery to the final destination point, the truck driver notifies the dispatcher, which subsequently notifies the customer that the order has been successfully delivered to the destination point, as per the customer's order input.

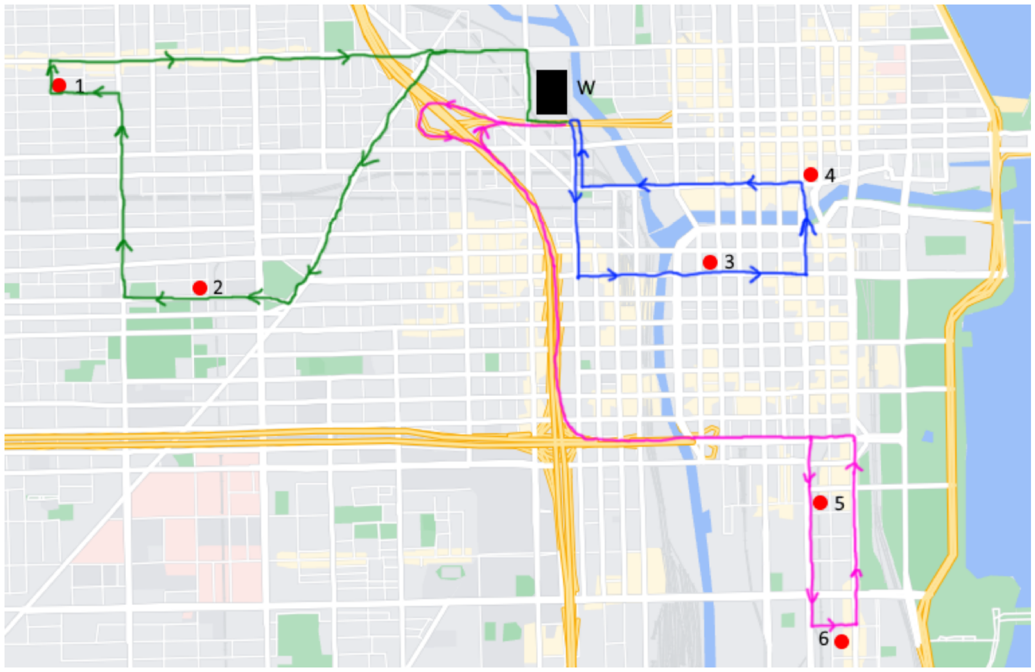


Figure 2.4.3. Multiple Destinations with Multiple Routes Use Case

Given warehouse W, orders 1, 2, 3, 4, 5, and 6 at the locations depicted, and three trucks, the algorithm would output three routes, green, blue, and pink, given the proximity of the orders, expected delivery times, and load balancing. Each of the three trucks would be assigned to one of the routes for the day, starting and ending at the warehouse.

4. ROUTE RECALCULATION USE CASE

The route reallocation use case is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, the order route is between the cargo origin point and destination points and external route blocking or route inefficiency factors such as traffic, road construction, route obstacles are present.

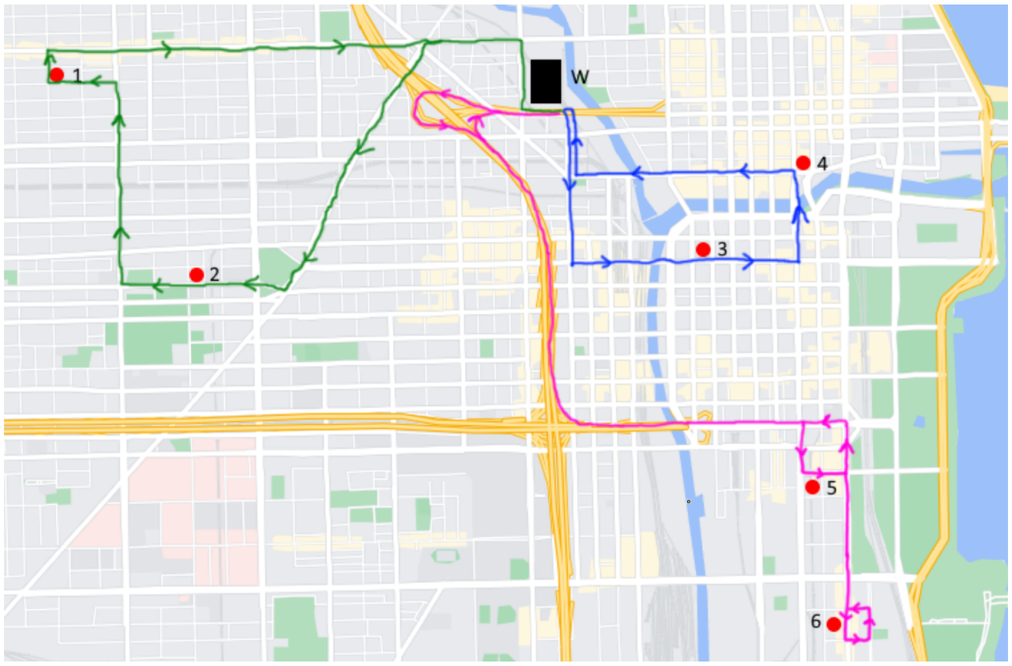


Figure 2.4.4. Route Recalculation Use Case

This example is similar to the example presented in the previous use case. However, upon delivering order 5, a backup is reported along the initial planned route to deliver order 6. The route allocation algorithm recalculates the route to ensure timely delivery of the customer's order and reduce the amount of time spent delivering orders for the truck driver. This recalculated route will be given to the dispatcher who will inform the truck driver through the web application and the truck driver will receive the updated route in the mobile app.

5. MULTIPLE WAREHOUSES WITH MULTIPLE ROUTES AND MULTIPLE DESTINATIONS

The multiple destination use case is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, the order route is between the cargo origin point and multiple destination points and external factors such as traffic, vehicle malfunction are not present.

A customer's input order is allocated to a dispatcher via the allocation algorithm and the dispatcher then relays the order to a truck driver. Communication between the dispatcher and the truck driver is facilitated by the web and mobile platforms communication/notification service. The truck driver's route from the warehouse to each of the delivery locations will be determined by the routing algorithm. Post order delivery to the final destination point, the truck driver notifies the dispatcher, which subsequently notifies the customer that the order has been successfully delivered to the destination point, as per the customer's order input.

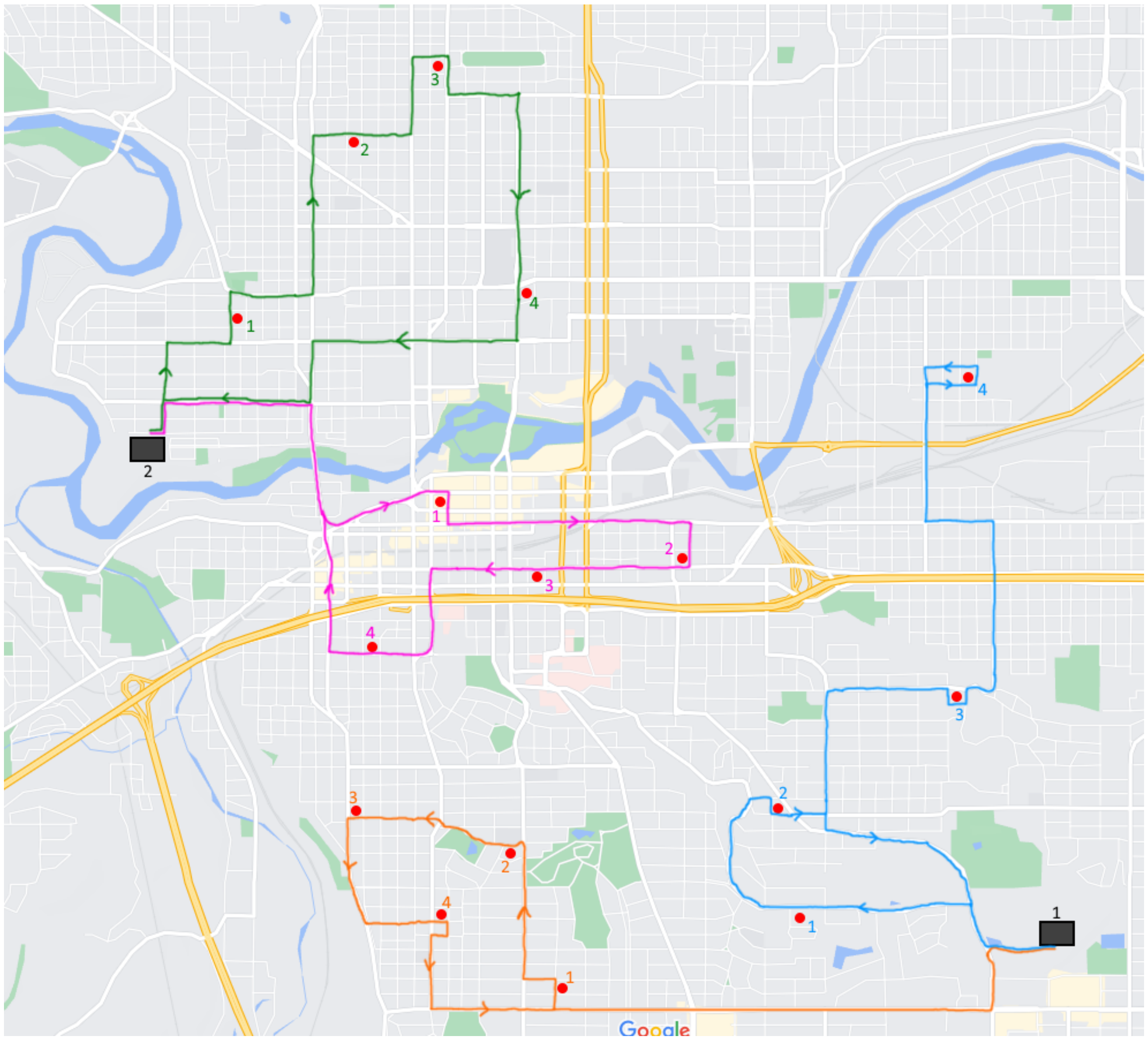


Figure 2.4.5. Multiple Warehouses with Multiple Routes and Multiple Destinations Use Case

Given two warehouses, 16 orders at the locations depicted, and four trucks, the algorithm would output four routes, green, pink, orange, and blue, given the proximity of the orders, expected delivery times, and load balancing. Each of the four trucks would be assigned to one of the routes for the day, starting and ending at the warehouse.

6. TRUCK REALLOCATION USE CASE 1

The truck reallocation use case 1 is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, a route has been allocated between specific cargo origin point and destination points, the truck has already picked up the customers cargo and external truck related inefficiency factors such as vehicle malfunction, flat tire or even driver related factors such as ill-health are present.

A customer's input order is allocated to a dispatcher via the allocation algorithm and the dispatcher then relays the order to a truck driver. Communication between the dispatcher and the truck driver is facilitated by the web and mobile platforms communication/notification system. The truck driver's route from the warehouse to each of the delivery locations will be determined by the routing algorithm. External factors such as the truck breaking down on a route may occur. The broken down truck driver will communicate to the dispatcher via the application's communication service. The truck allocation algorithm will then assign other truck(s) to the deliveries of the broken down truck. But, the newly allocated trucks' routes' destination point will be the location of the broken down truck. After receiving the cargo from the broken down truck, the newly assigned truck(s) will have the additional customer order destinations included in its new route calculated via the route allocation algorithm. Post order delivery to the final destination point, the truck driver notifies the dispatcher, which subsequently notifies the customer that the order has been successfully delivered to the destination point, as per the customer's order input.

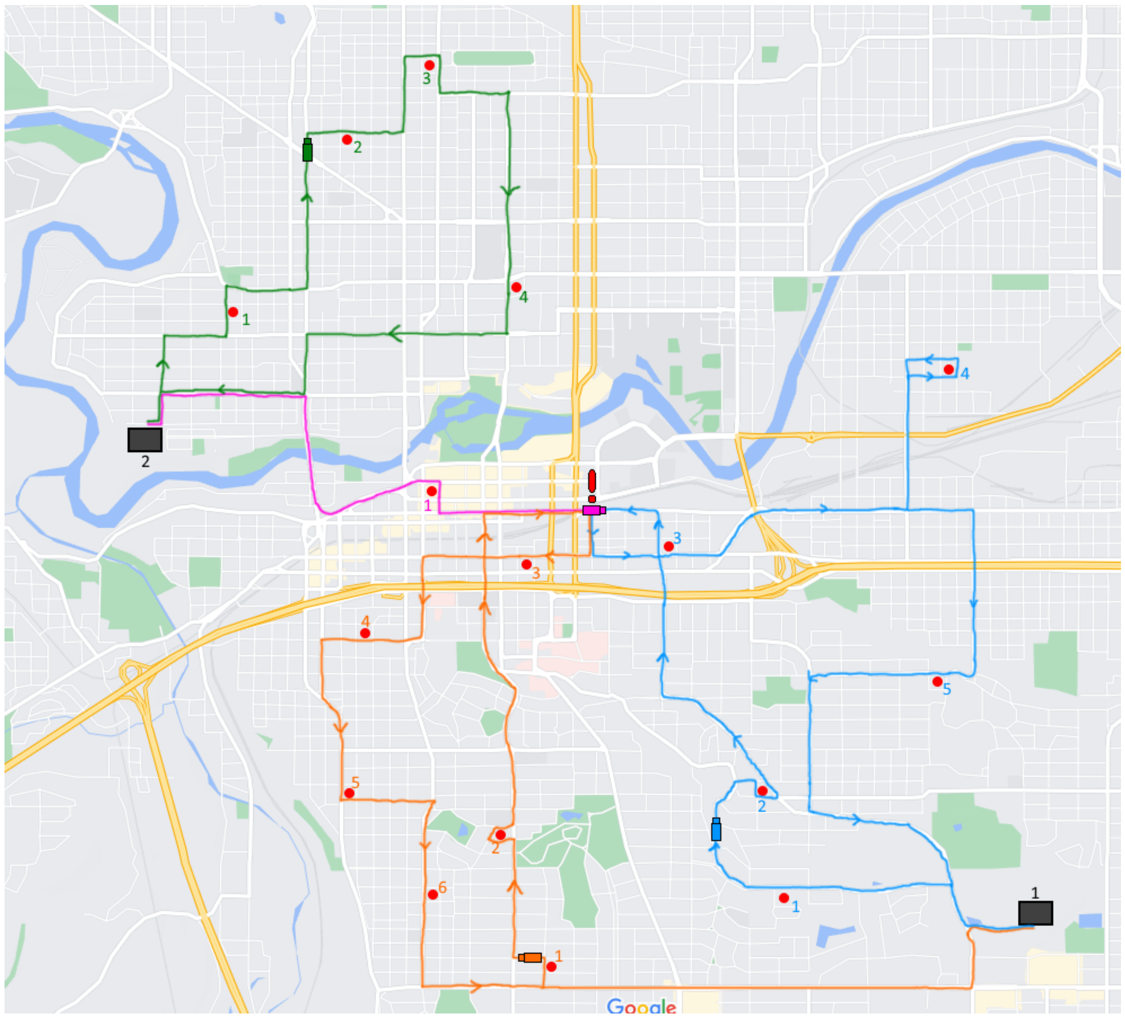


Figure 2.4.6. Truck Reallocation on Truck Break Down

This example is similar to the example presented in the previous use case. However, shortly after delivering its first order, the pink truck breaks down. The route allocation algorithm recalculates the routes of the remaining trucks based on current location and available capacity. The recalculated route will be given to the dispatcher who will inform the appropriate truck drivers, in this case the orange and blue truck drivers, through the web application and the truck driver will receive the updated route in the mobile app.

7. TRUCK REALLOCATION USE CASE 2

The truck reallocation use case 2 is formulated on the assumptions that all 3 actors (Truck drivers, Truck dispatchers and customers) are involved, a route has been allocated between specific cargo origin points and destination points, the truck has not yet picked up the customers cargo and external truck related inefficiency factors such as vehicle malfunction, flat tire or even driver related factors such as ill-health are present.

A customer's input order is allocated to a dispatcher via the allocation algorithm and the dispatcher then relays the order to a truck driver. Communication between the dispatcher and the truck driver is facilitated by the web and mobile platforms communication/notification service. The truck driver's route from the warehouse to each of the delivery locations will be determined by the routing algorithm. External factors such as the truck breaking down prior to leaving the warehouse may occur. The broken down truck driver will communicate to the dispatcher via the application’s communication service. The truck allocation algorithm will then assign other truck(s) to the deliveries of the broken down truck. As the broken down truck has not picked up the customer's order cargo yet, the newly allocated truck will directly be assigned the route to the customer order pick up point, instead of the broken down truck location. However, if all trucks are currently in use, the deliveries for the broken down truck will be reassigned and spread amongst the functional trucks. Post order delivery to the final destination point, the truck driver notifies the dispatcher, which subsequently notifies the customer that the order has been successfully delivered to the destination point, as per the customer's order input.

3 Project Plan

3.1 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.

3.2 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.

Table 3.2.1. Task Decomposition

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.

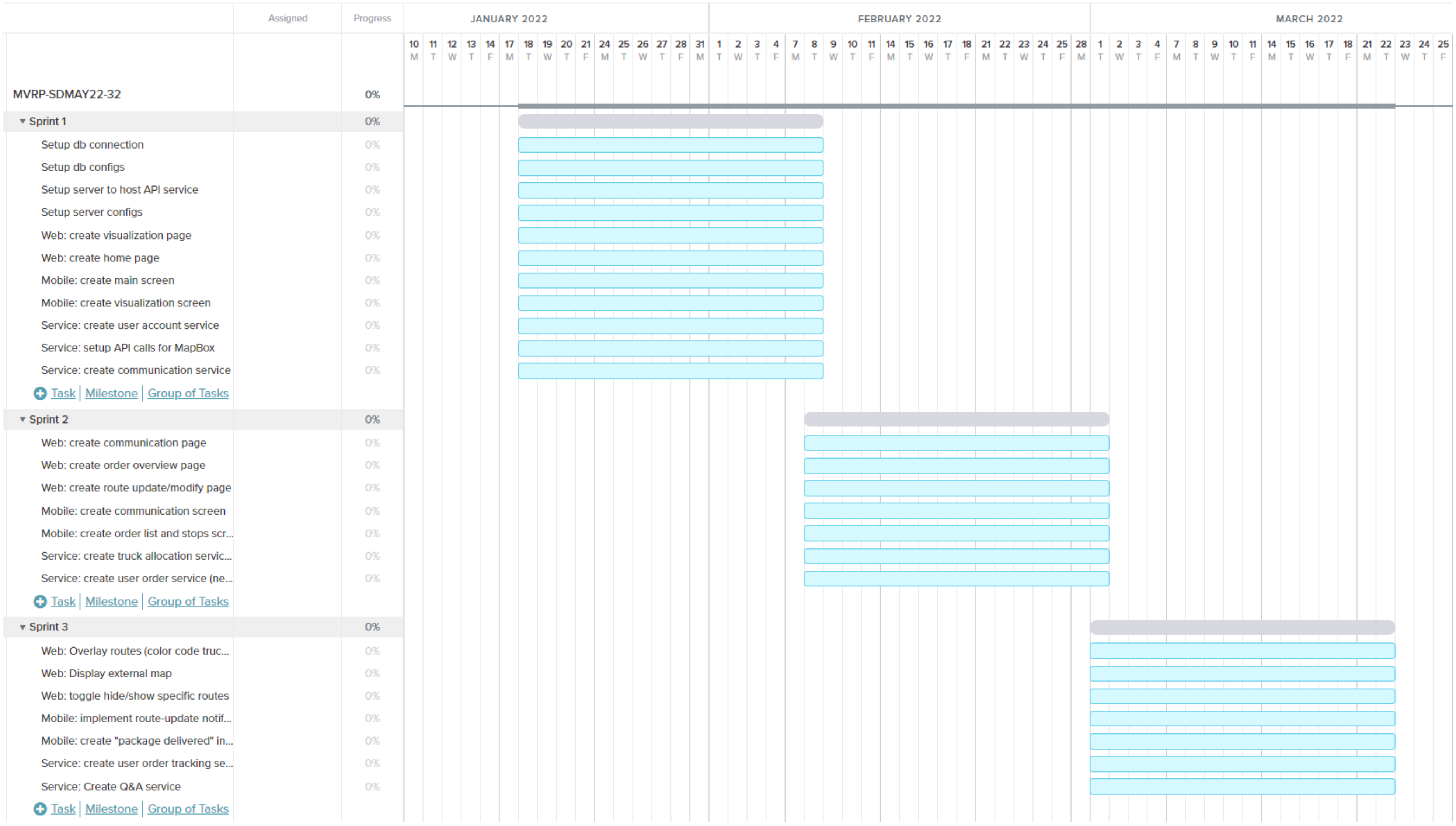
3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

- 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 speed improvements
        - 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
    - Scaling the input space provides minimal decrease in performance
        - 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%

3.4 Project Timeline/Schedule

(Gantt chart on next page)



	Assigned	Progress
MVRP-SDMAY22-32		0%
▾ Sprint 1		0%
Setup db connection		0%
Setup db configs		0%
Setup server to host API service		0%
Setup server configs		0%
Web: create visualization page		0%
Web: create home page		0%
Mobile: create main screen		0%
Mobile: create visualization screen		0%
Service: create user account service		0%
Service: setup API calls for MapBox		0%
Service: create communication service		0%
▾ Sprint 2		0%
Web: create communication page		0%
Web: create order overview page		0%
Web: create route update/modify page		0%
Mobile: create communication screen		0%
Mobile: create order list and stops scr...		0%
Service: create truck allocation servic...		0%
Service: create user order service (ne...		0%
▾ Sprint 3		0%
Web: Overlay routes (color code truc...		0%
Web: Display external map		0%
Web: toggle hide/show specific routes		0%
Mobile: implement route-update notif...		0%
Mobile: create "package delivered" in...		0%
Service: create user order tracking se...		0%
Service: Create Q&A service		0%

Figure 3.4.1. Gantt Chart

- First implementation will be completed after the completion of three Sprints
- Each Sprint will be 3 weeks with weekly standup meetings
- Unit testing will be added for each development task
- Integration testing and improvements will be made after the completion of this development schedule
- Sprint 1: January 18th - February 8th
- Sprint 2: February 8th - March 1st
- Sprint 3: March 1st - March 22nd

3.5 Risks And Risk Management/Mitigation

Risks for each task:

- 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

3.6 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 components of the web app, mobile application, and the microservices.	20

Table 3.6.1. Task-Effort Decomposition

3.7 Other Resource Requirements

There should not be a need for any additional resources outside of the software and database solutions that are used 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.

4 Design

4.1 Design Context

4.1.1 Broader Context

The broader context is situated in the domain of transportation and delivery. Specifically, we consider a fleet of delivery trucks, a set of orders for goods from stores with given locations and a set of warehouses where those goods are stored and loaded into the trucks. Given an information about a road network (map) with corresponding distances/travel times and the capacity of load for each truck in the fleet the objective of the work is A. utilize some of the existing techniques for assignments of trucks to destination B. Develop novel solutions to the problem of having one of the trucks break down(re-distributing its load) C. Implement an interactive system that can manage users and assignments.

Relevant considerations related to our project:

Area	**Description**	**Examples**
Public health, safety, and welfare	This project will aim to use algorithmic efficiency to benefit the public health by cutting down the time trucks and other fleet vehicles are on the road.	Tries to optimize the efficient delivery of goods(welfare). However, the findings can be indirectly used in public health for routing ambulances. Additionally with efficiently routing trucks we can lessen the probability of trucks coming into contact with pedestrians so public safety will increase.
Global, cultural, and social	Countries that rely on heavy amounts of consumerism will be allowed to continue with this practice more through the use of our product.	Our project is not affected by any specific societal context such as nationality, ethnicity and so on. However, it does reflect an impact on

		the efficiency of transportation and delivery.
Environmenta l	This project will have a deal of impact on any industry that deals with the production of gasoline or diesel fuel. This is due to the nature of efficiency that our project aims to create for a fleet of trucks.	Minimizing fuel consumption and the emission of a fleet of trucks. This will allow for less greenhouse gases overall to be released companywide for whoever utilizes our product.
Economic	Economics benefits from our project can fall under two categories. The first is the savings on fuel cost that businesses can experience due to our product. The second is the maximization of profits that businesses can generate from their current fleet.	System implementation will be deployable either on standard desktop with minimum installation overhead or on popular mobile devices (e.g., android). This product will allow any company that uses it to optimize costs of delivery and balance it with reassignment of deliveries.

Table 4.1.1.1. Responsibility Considerations

4.1.2 User Needs

- Customers
    - Each customer needs to be able to access the menu where he/she can place the requested products because they want to get products delivered and restocked as soon as possible.
- Warehouses/Dispatchers
    - Each warehouse/dispatcher needs to be able to summarize all the requests from the customers, check the availability of the products in the local facility, and determine the assignment of trucks to delivery locations so that they can keep track of and be on top of orders that are coming in from customers.
- Truck Driver
    - Each truck driver needs to be able to login and see the assigned route and notify the completion of a delivery at a particular time and location(products and quantities) to the warehouse that it was assigned from because they want to be able to work as efficiently and safely as possible.

4.1.3 Prior Work/Solutions

The VRP has been written about a lot. Broad nature or specific characteristics of the problem. Previous research includes that done by Nasser A. El-Sherbeny [3], Wang, et. al. [5], and Cappanera, Requejo, and Scuttelà [2].

In his research, El-Sherbeny focused on the various exact methods, heuristics, and metaheuristics that can be used to solve a VRP with time windows (VRPTW) [3]. While he investigates a large variety of methods for finding an optimal solution to a VRPTW, the methods are approached from a theoretical perspective rather than a practical one. As a result, the true optimality of the methods presented may not be accurate when transitioned to a real-world application.

In their research, Wang, et. al. expand upon the VRPTW problem by including simultaneous delivery and pickup and optimizing their solution based on a 5 part multiobjective function (MO-VRPSDPTW) [5]. The two methods they focus on are local search and a memetic algorithm. While the research comes from a theoretical perspective, they openly acknowledge that the optimality of the two methods may not translate well when implemented in a practical solution.

Lastly, Cappanera, Requejo, and Scuttelà focus on the skill VRP, an extension of the traditional VRP that focuses on delivering human services instead of goods [2]. What makes their research interesting is their specific focus on situations where one individual does not have all of the skills necessary to complete a job. As a result, multiple "deliveries" must be made to the same customer.

In comparison, what separates this project is the setting that it considers, which is reacting to the breakdown of a truck and properly executing the reassignment of the routes to the rest of the fleet so that

A. The goods from the broken truck(s) will be delivered to their destinations, and
B. It will be done in an optimal manner.

As a result, the focus of the above research on the VRPTW problem does not address the same problem as we seek to address. While many of the aspects are shared between our project and previous research, the end goals diverge greatly when problem constraints are considered. This proves to be an advantage to us as customer-specific time windows for delivery increase the difficulty of finding an optimal solution. Additionally, our focus on the situation where a truck breaks down and routes need to be recalculated part way through is not addressed in any of the above research. This is a shortcoming of the above research and will likely prove to be a disadvantage to us if the solution isn't immediately obvious.

4.1.4 Technical Complexity

There are three kinds of novelties in this project:

1. Algorithmic: We will solve the very specific problem of reassignment of trucks. This is being done through an existing application called MapBox. MapBox will be performing a MultiVehicle Routing Algorithm that we can use for the assignment of trucks.
2. System-wide: We will develop, implement, and deliver a system for managing assignments of trucks to delivery locations which can be accessed and used by all kinds of entity classes that participate in this scenario(customers, warehouses, and drivers). Across our project, we will do integration testing to ensure all components are functioning properly.
3. Technical complexity: Defining proper test cases for an "optimal route" and evaluation procedures. Our project also focuses on routing trucks with a capacity constraint, and the

potential for vehicles to break. This creates additional complexity as a rerouting will have to occur for broken vehicles. Additionally, a new vehicle must have the capacity to be able to assist a broken vehicle with deliveries.

4.2 Design Exploration

4.2.1 Design Decisions

Use the use cases to see the initial system architecture.

1. Location (Austin, Texas, Chicago, Illinois, Seattle, Washington)
2. Traffic Density and map (A city's department of transportation w/ API ?)
3. Which dbms to use? (MySql vs GravityB)
4. Interface module (Which type of phone?)
5. Front end Framework (React vs Angular)
6. Backend communication (Spring)
7. Database Management System (MySQL Workbench, Postgres, MSMS)

Subject to changes

4.2.2 Ideation

(7) Considered dbms:

1. MySql Workbench
2. Microsoft SQL server management studio
3. PostgreSQL
4. Oracle SQL Developer
5. Toad for SQL Server
6. Neo4j
7. Gravity

4.2.3 Decision-Making and Trade-Off

Because of the relational nature of MySQL workbench and its ability to be used easily on backend systems with the use of its specialized drivers we are selecting MySQL workbench for this project. It also has the added benefit of being able to be run and accessed from multiple different systems without having data loss. Additionally, this project does not require anything more complex. Should the complexity increase in the future, we will re-evaluate this decision

4.3 Proposed Design

4.3.1 Design Visual and Description

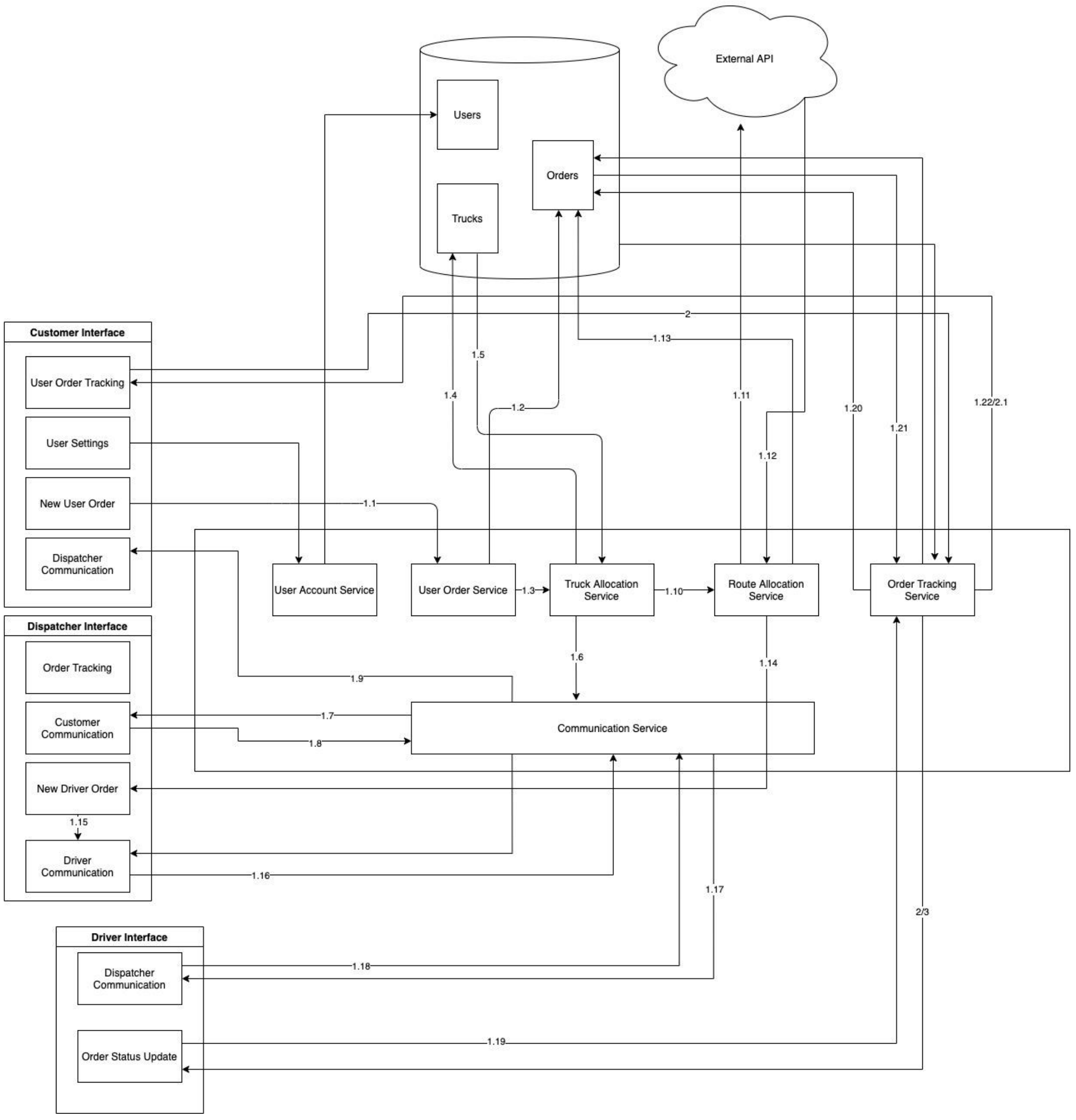


Figure 4.3.1.1. System Architecture (See Appendix for higher resolution version)

Components

- Customer/dispatcher/driver interfaces
- DB (User, Truck and Order tables)
- API (Account service, order service, order tracking/update service, route allocation service, truck allocation service, communication service)
- External API service

1.1 A customer can place a new order via the new user order component of the interface which will be handled by the user order service.

1.2 The user order service will perform CRUD operations with the db, to persist the order information for later use.

1.3 The order information is passed to the truck allocation service from the user order service.

1.4/1.5 The truck allocation service fetches truck information from the db on the basis of the new order param inputs and allocates a truck to the order.

1.6/1.7/1.8/1.9 Communication service establishes and handles communication between the assigned truck dispatcher and customer.

1.10 Order updated with truck information now passed to route allocation service, which will determine the most efficient route for the truck.

1.11/1.12 Truck Allocation service calls external API (Google Maps, Department of transportation) for information which will be consumed by the route allocation algorithm to determine the most efficient route.

1.13 Route allocation service performs CRUD operation to update route parameters of the order in the db.

1.14 Order with truck and route information is passed to the dispatcher.

1.15/1.16/1.17/1.18 Communication service establishes and handles communication between the assigned truck dispatcher and assigned truck driver. Dispatcher passes route information to the driver.

1.19 Post order complemention, via Order status update component of the driver UI, the driver notifies the customer which is handled by the order tracking service.

1.20/1.21/1.22 The order tracking service performs CRUD operations to update the order status in the db. The updated order status is forwarded to the customer.

2/2.1 A customer can track the status of their order or cancel their order via the User order tracking component of the UI. The requests will be handled by the order tracking service, which will fetch the order from the db and relay it to the user. If the order is cancelled, the driver will be notified via the order status component of the driver UI.

3. User account settings can be accessed via the user settings component of the customer UI. The requests will be handled by the user account service, which will perform CRUD operations with the users table in the db.

4.3.2 Functionality

Functional requirements:

1. Truck drivers should pick-up locations.
2. Truck drivers should be able to deliver to the picked location.
3. Find the nearest truck in case of any breaks.
4. Find the set of closest trucks that can take care of the load that the truck has
5. Set of orders and trucks w/ given capacity.

Non-functional requirements:

1. App should have access to all truck drivers databases.
2. App should have enough information about all warehouses locations, type of loads and capacity.

Our design functions by taking the orders in the system and the associated delivery locations (red dots in the below figures) and generating an optimal set of routes (denoted by green, pink, orange, and blue lines in the below figures) based on the available number of trucks and warehouse locations(denoted by grey boxes in the below figures). Routes will start and end at the same warehouse and trucks will only deliver goods from a single warehouse. The result of this step is shown in Figure 4.3.2.1. Once the routes have been decided and assigned, the truck drivers will be notified via the UI system of their route for that day. In our specific case, we are addressing the instance in which a truck breaks down. Suppose the pink truck breaks down at the location depicted in Figure 4.3.2.2. The UI system will be used to notify the central system that the pink truck has broken down Based on the locations of the other trucks, their load capacity and the remaining delivery locations, two trucks are assigned to take over the remaining deliveries for the pink truck and their routes are recalculated. In this instance, the blue and orange trucks are identified as the optimal choices and new routes for these two trucks are generated as depicted in Figure 4.3.2.3. The orange and blue truck drivers are then notified of their updated routes via the UI system. This system meets all of the listed functional and non-functional requirements sufficiently.

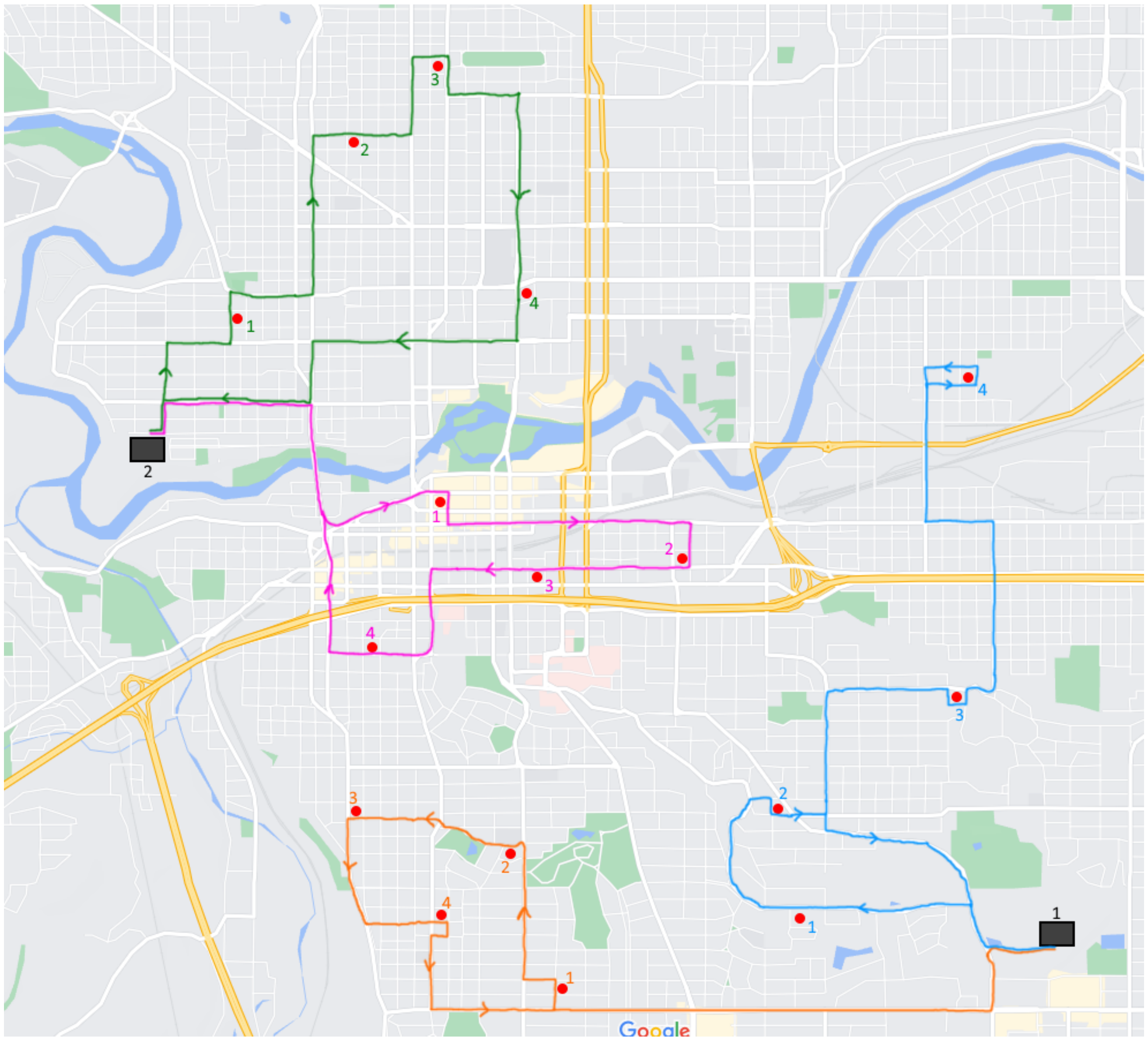


Figure 4.3.2.1. Initial Routes Calculated by Algorithm

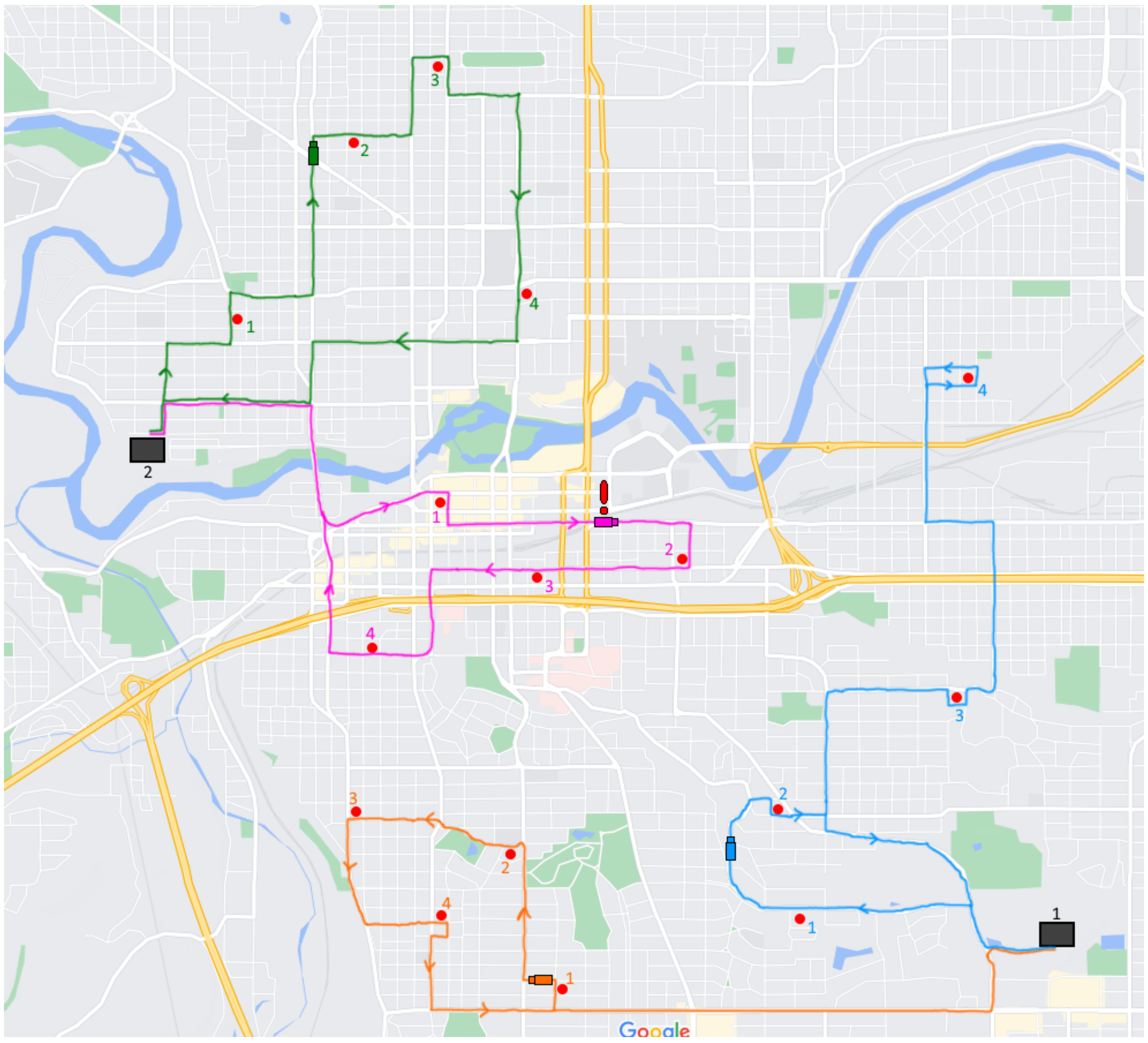


Figure 4.3.2.2. Pink Truck Breaks Down

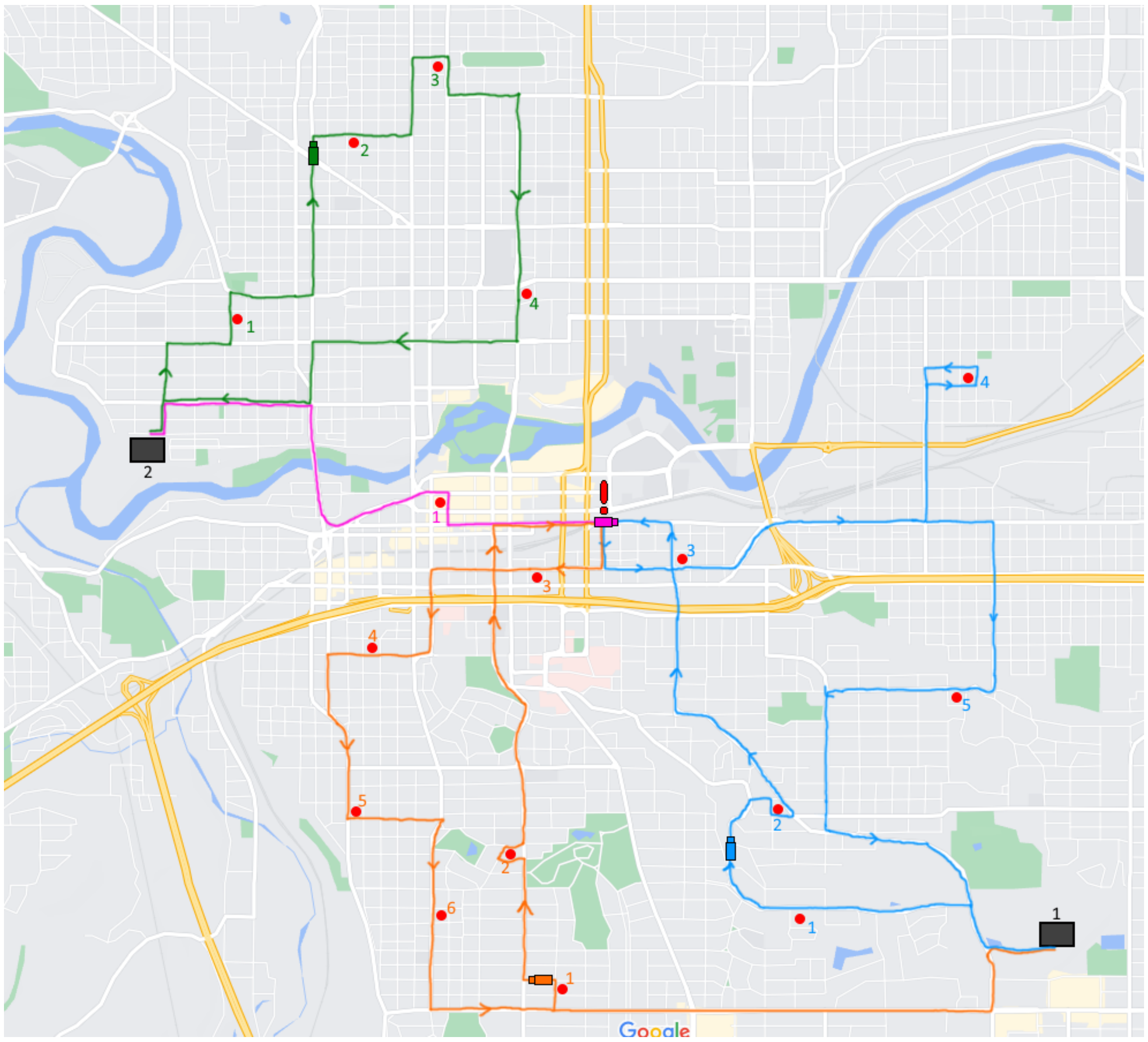


Figure 4.3.2.3. Load Divided Between Blue and Orange Trucks. Routes Recalculated Accordingly.

4.3.3 Areas of Concern and Development

Primary concerns for delivering a product/system that addresses requirements and meets user and client needs:

1. For our current design, we have a better sense of how we want to structure our backend services as opposed to our frontend UIs. This lack of description in the UI design can raise a possible concern because it can leave multiple interpretations to team members, making the project structure less organized. In the upcoming week, this issue will be addressed by the team.
2. Another possible concern is the usage of MapBox external API. MapBox will be crucial for our applications success and is needed for visualization and vehicle routing.

Immediate plans for developing the solution to address those concerns?

1. We plan to hold multiple team meetings to tackle this concern, so everyone in the team could get a single and clear understanding of how the UI design should be implemented, so it is not open to interpretation.
2. We plan to tackle this problem by collectively sharing our understandings of the mapbox api's. We plan to use widely available resources on the internet to educate ourselves about the api and use tutorials to get a basic simulation working as soon as possible.

What questions do you have for clients, TAs, and faculty advisers?

1. Our faculty adviser/client and TA have been very helpful in answering any questions we have. At this point in time we are satisfied with the direction of our project. Should any questions arise, we will not hesitate to ask them.

List other functional requirements. List how the concerns are met with the current system. Map them to how different components interact with each other to meet the requirements.

1. Our concern of not having a clear UI design could be addressed by using our trello board for looking up the tasks. This current system would be efficient in assigning tasks, checking up on progress, and viewing the progression to a final product.

4.4 Technology Considerations

API development

The main technology chosen for API development is the Spring framework.

Strengths:

Spring provides several additional libraries which support a range of functions which can be implemented in the application.

Spring provides a powerful abstraction to JEE specifications such as JDBC and JPA.

Spring provides declarative support for caching, validation, transaction, and formatting.

Spring ensures fast performance, as it ensures efficient startup, shutdown and optimized execution.

Weaknesses:

Development in Spring may be complex at times, as the learning curve for working with JPA and JDBC is quite high.

Mobile client development

The main technology chosen for mobile client development is the Android platform.

Strengths:

Due to the various methods of measuring and controlling application performance, the Android platform allows for greater versatility and scalability in development.

The open source nature of Android application development provides a large number of choices to open source libraries and third party API's to choose from during development.

Compatibility of the Android platform with multiple API's and libraries is really important, as it does not require developers to write custom middleware or libraries in order to port non-compatible libraries to the Android platform.

Web client development

Web development will be taking place through the use of the React js framework. This was done both due to team knowledge of the framework, but also due to market resources surrounding React.

Strengths:
React allows for customization of pages as well as flexibility, since there are preset parts that you can use but also customize one they are in pages.

React also is very well documented which will allow for errors and problems to be resolved more easily.

React is also simple and scalable, so it can allow us to create and demo ideas faster than using a non-framework based approach.

Weaknesses:

Some members of the web client team are not as familiar with React as they are with standard HTML, however this problem will be remedied by other team members' knowledge of the framework, as well as the amount of resources available to learn React.

4.5 Design Analysis

To this point we have spent most of our time brainstorming, researching, and deciding on the best software tools that we will be using for our design. We have been reviewing some past research and evaluating which pieces will result in the most success for our project. This digging has been fruitful, mostly due to the documentation that past researchers have done. As a result, the focus of the research that we looked at does not address the same problem as we seek to address. While many of the aspects are shared between our project and previous research, the end goals diverge greatly when problem constraints are considered. This proves to be an advantage to us as customer-specific time windows for delivery increase the difficulty of finding an optimal solution. Additionally, our focus on the situation where a truck breaks down and routes need to be recalculated part way through is not addressed in any of the above research. This is a shortcoming of the previous research that will likely prove to be a disadvantage to us if the solution isn't immediately obvious. Overall, our design is very solid and much better than the initial delivery apps.

4.6 Design Plan

The design plan is to complete every development and testing subtask listed in (3.2). The use-cases listed will be implemented across three separate 3-week sprints (Figure 3.4.1). The use-cases, requirements, and constraints were gathered through meetings with our client. In our design architecture (Figure 4.3.1.1.), there is a series of services. The Route Allocation Service will be responsible for route allocation response time using MapBox. The Truck Allocation Service will be responsible for the Truck objects, which given a capacity field, will be responsible for our truck capacity requirement. The Communication Service will be responsible for informing any users, admin or truck driver, of any necessary updates (ex. broken trucks, new orders) regarding deliveries. Functionality and requirements are further detailed in the previous section 4.3.1.

5 Testing

For our team we aim to test a multitude of different scenarios that connect with our use cases and requirements. This will cover not only the database and the external api's, but also the backend connectivity as a whole. These cases will allow us to be sure that as a whole the backend for the system is functioning properly and is able to run the algorithm and generate routes from given data both from the api's and the database. While these backend unit and system tests are occurring the frontend mobile and desktop applications will also be tested. These tests will align with use cases regarding the various types of users that our program can have and how they would go about using the program. The challenge for this testing model is testing the connectivity of the frontend and backend. Since this project has only one backend that is going to be deployed to two different frontend platforms, the challenge will be coming up with a comprehensive amount of tests that are able to be used to verify use cases in both the mobile and desktop application.

5.1 Unit Testing

The individual units that exist within our program are the customer interface, the dispatcher interface, the driver interface, the communication service, the database, and also the use of external api that covers the algorithm for the program. These interfaces contain units within them that will all be individually tested to ensure that the use cases for each user are working correctly. This will be done through different ways though depending on which part of the program we are dealing with.

For frontend testing we are looking at using Selenium or Appium in order to simulate a specific user running and working within the frontend. This will allow for tests to be run that follow the information pathway that is developed for that specific use case and see if the workflow that the user would follow in fact works within the scope of the frontend. Additionally for individual frontend pages field placement and style will be checked to ensure that text fits within input fields and additionally that pages have proper connectivity to each other within both frontend applications.

For the backend of the project the database will be tested in a few different ways. Firstly we will run a series of queries and check the results in an attempt to see what commonly used queries do to the data within the database. These queries will follow the flow of use cases that were developed and will simulate the data that would be needed in order to run the assignment algorithm. These tests and queries can either be run through a program such as MySQL workbench or through a terminal.

For the backend algorithm code and the api's these will be tested through the use of following use cases with static data numbers. Standard expectations for the algorithm will be developed such that we can have expected results from these static data numbers and use them to verify use cases for different assignment scenarios. Additionally the api's will be tested to ensure that we are able to reliably use them to collect data and information that is needed for the assignment algorithm. These unit tests will follow the standard unit testing framework and will be done in Junit or the equivalent testing environment.

5.2 Interface Testing

The interfaces present in the current iteration of the application architecture are the customer interface, dispatcher interface and driver interface. The customer interface will be used by the customer to interact with the application functions and interact with order dispatchers via the communication service. The dispatcher interface will be used by the dispatcher to interact with application functions and interact with the customers and drivers associated with the orders. The driver interface will be used by drivers to interact with application functions and interact with dispatchers associated with orders.

In the current iteration of the architecture, interfaces are being treated as separate components, and access to the components are determined by user role. Shared interfaces such as user login and registration are shared across all users, but other interface components are user role dependent. Ex: Customer users can only interact with the customer interface and will be limited to interaction with the customer interface, and not the dispatcher and driver interface.

Access to specific API services (specific application function) is also dependent on user role and subsequently user interface. Ex: Http calls to the new order service are only allowed from the customer interface.

Interface testing will mostly be based on the black box testing model. Black box testing is the most suited model for interface testing as the goal of this testing method is not to dig deep into the code, going through the application's internal functioning, but to interact with the UI, test the end user functionality, and make sure that every input and output of the system meets the specified requirements.

The tools under consideration to be used for the mobile UI testing are Robotium and Espresso.

Robotium is an android Testing framework to automate test cases for native and hybrid applications. Robotium can be used to create strong automatic GUI testing cases for Android applications.

Espresso is a UI test framework that can be used to create automated UI tests. Espresso tests run on an actual device or emulator and behave as if an actual user is using the app (i.e. if a particular view is off screen, the test won't be able to interact with it).

- Customer Interface testing scenarios :
  - Customer mobile UI will actively allow user touch input such as swipes or clicks, in order to test functional status of UI components such as button on click listeners and swipe listeners.
  - Customer web UI will actively allow user input such as mouse clicks and scrolls, in order to test functional status of UI components such as buttons on click listeners and scroll listeners.
  - Customer mobile UI will redirect to the correct page on user input such as swipe or click, in order to test navigation between screens/pages.

- New order form will allow users to input data into form fields, and run UI side validations on the form field data.
- Messaging page will actively update and persist message data between user and order dispatchers.
- Messaging page will correctly redirect to user messages on user click input, in order to test messaging page navigation.
- Order tracking page/screen will respond to user order information requests with correct data, in order to test backend service functionality and frontend json response handling and formatting.
- Truck allocation page will display updated and accurate truck information on new user order input. Automated page navigation between new order form and truck allocation page will be tested.

- Dispatcher Interface testing scenarios :
  - Dispatcher mobile UI will actively allow dispatcher touch input such as swipes or clicks, in order to test functional status of UI components such as button on click listeners and swipe listeners.
  - Dispatcher web UI will actively allow user input such as mouse clicks and scrolls, in order to test functional status of UI components such as buttons on click listeners and scroll listeners.
  - Dispatcher mobile UI will redirect to the correct page on user input such as swipe or click, in order to test navigation between pages.
  - Messaging page will actively update and persist message data between order dispatchers, users and truck drivers.
  - Messaging page will correctly redirect to user messages on dispatcher click input, in order to test messaging page navigation.
  - Order tracking page/screen will respond to dispatcher order information requests with correct data, in order to test backend service functionality and frontend json response handling and formatting.
  - Route allocation page will display update and accurate route information on the google map component on the page. Calls to the google maps api and page map component updation will be tested.

- Driver Interface testing scenarios :
  - Driver mobile UI will actively allow user touch input such as swipes or clicks, in order to test functional status of UI components such as button on click listeners and swipe listeners.
  - Driver mobile UI will redirect to the correct page on user input such as swipe or click, in order to test navigation between screens/pages.
  - New order form will allow users to input data into form fields, and run UI side validations on the form field data.
  - Messaging page will actively update and persist message data between order dispatchers and truck drivers.
  - Messaging page will correctly redirect to user messages on driver click input, in order to test messaging page navigation.
  - Order tracking page/screen will respond to driver order information requests with correct data, in order to test backend service functionality and frontend json response handling and formatting.
  - Route allocation page will display update and accurate route information on the google map component on the page. Calls to the google maps api and page map component updation will be tested.

5.3 Integration Testing

One critical path to test will be a customer User correctly see its orders. This will require the user to view their orders correctly in the web application by requesting information from the Order Tracking Service. The Order will be in a database and has to have been set from the Truck and Route Allocation Services. Additionally we will need to test the cycle starting from the Dispatcher Interface. The dispatcher web application will most importantly need driver orders which will be dependent on the Route Allocation Service, then the Route Allocation Service will be using an external API to process traffic and map data. Also, a Driver will use the Driver Interface in the form of a mobile application, this will be dependent on the Communication Service and Order Tracking Service. These three are our critical integration requirements because of our primary use cases. A Customer would need to track their orders, a Dispatcher would need to submit new orders, and a Driver would need to view its orders assigned.

One critical aspect we will assess is the response time from customer order to assignment. This will involve an event from the customer web application which calls the user order service, truck allocation service, route allocation service, map API, and calls our vehicle routing algorithm.

We will test this by performing a new user order from our front end web application, in Chrome Developer Tools we will be able to view the response time of this response which must use all of the services listed previously.

5.4 System Testing

For system testing strategy is to focus on interaction between all parts of the system as much as possible, and individual tests should be less plentiful and more general. Starting with unit tests, they are still important for system level testing but don't need to be as plentiful or specific. For example, a set of unit tests on a file in the customer interface can probably be narrowed down to a single test that covers the general functionality of that file. In terms of interface testing scenarios, I think it's most important to zoom in on scenarios that cover as much of the entire system as possible. An example would be the order tracking scenario in the dispatcher interface, where the page responds to driver order information requests. As stated above, this test is important because it tests both service functionality on the backend and json response handling/formatting on the frontend. And then integration tests are made for interaction between all parts of the stack, so most of those can be kept and considered part of the system-level testing.

No new tools need to be added for these tests, but rather a combination of the same tools that are used for unit and interface and integration testing. Unit test files should still be made in the code editors that we build the app in. Interface tools should use both dev tools on the frontend and postman on the backend. And then integration testing will use a combination of these depending on the functionality being tested.

5.5 Regression Testing

The most critical of features to our program is the reassignment algorithm, so ensuring that new changes do not break this will be key when looking at regression testing. The reason this is so important is that the algorithm is what will be the key part of the project that all other components need to interact with, so any requirement for the project is directly tied to the correctness of this algorithm. To ensure that the reassignment algorithm does not break upon changes being made, firstly we will use the standard data set that was developed for unit testing to verify that with new changes the assignment algorithm still outputs the expected results. Once we can verify that the assignment algorithm works as expected then we can move on to automated tests on both the mobile and desktop applications in order to verify that data and requests are correctly being sent to the backend. This will narrow down the scope of any problems that arise to one specific frontend platform rather than having to work through both in the event of any bugs. Additionally these tests will allow us to verify if any problems are occurring with the database if specific known queries are used to retrieve information such that the result of what these queries return is known. The new features or implementations that we foresee that may cause issues is the use of a larger fleet of trucks or warehouses, which might cause issues within the database, or adding additional ui changes. The regression testing in these cases needs to verify the response time of the program as adding data such as this or new ui pages should not alter the response time of assignment in any drastic way. The only thing that would be a drastic change to the core of our project would be a change to the reassignment algorithm itself. This new algorithm would force our regression testing to follow a different method of testing each component of both the frontend and backend with every part except the new algorithm. This would ensure that all of the other components are functioning properly independently and together. From there the connections between the new algorithm and each individual component would need to be tested and then finally automated tests like the ones mentioned earlier would be used to verify that prior use cases that were satisfied with the program are still functioning properly and as expected.

5.6 Acceptance Testing

Our approach to acceptance testing is to incorporate it into our sprints as we are using the agile approach. Before every sprint, we will lay out the goals to be met in regard to what the client wants and what the specifications state. At the end of each sprint we will review if these goals were met and if the appropriate functional and non-functional requirements are being met. When possible, we will verify functional requirements are being met through dry end-to-end testing as stated in the requirements section (from the requirements document). As for the rest of the functional and the non-functional requirements, those will be verified through regular meetings with the client. Consistent communication with the client will ensure that we remain on track and the final round of acceptance testing goes smoothly.

5.7 Security Testing (if applicable)

While security is not a major concern for this project, we will use basic admin tools for the databases to ensure that a user is registered and has password secured access to the system. Additionally UI fields will be protected so that no code or other malicious user can use the system.

5.8 Results

We don't have any results ready since we haven't tested anything yet. But we have a list of unit testing that we will be using in order to test our project.

Testing units will be used	Use case test
JUnit	Testing Api's to unsure that we are able to use them in collecting our data.
Selenium or Appium	Used to simulate a specific user running and working within the frontend.
MYSQL Workbench or Terminal	Used to simulate the data that would be needed in order to run the assignment algorithm.

Figure 5.8.1: Unit testing

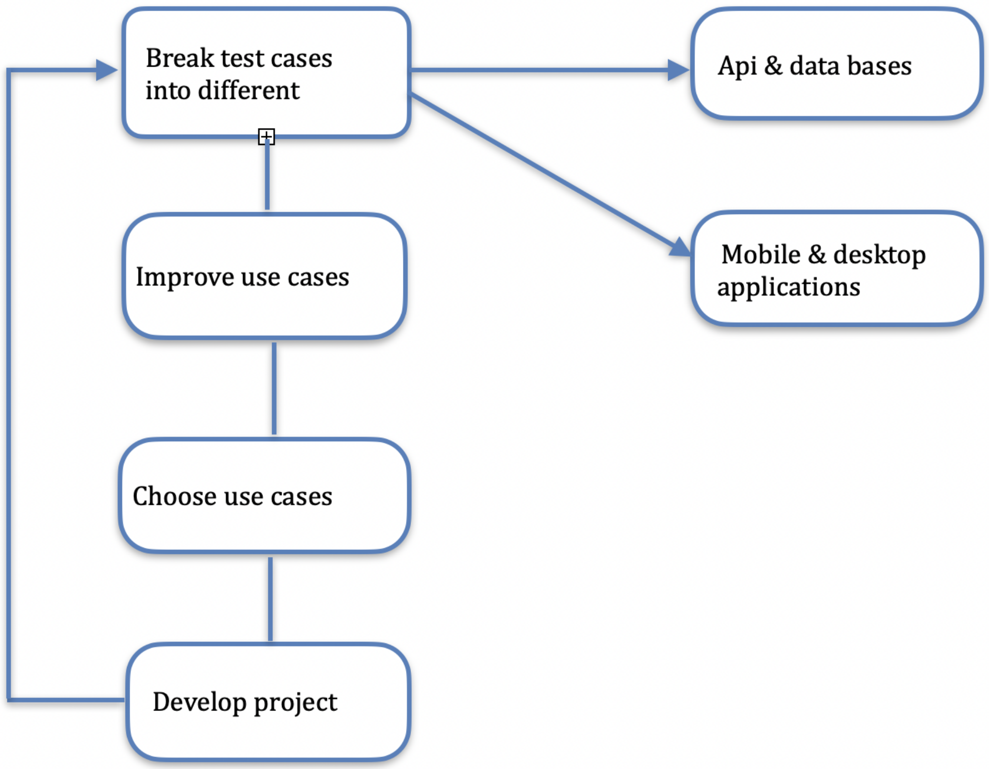


Figure 5.8.2: Testing plan

Our testing plan follows the test Driven Development software development process. In this process, requirements for the project are broken down into different individual test cases. This is so that everything that is created so far can be measurable. After this, code is redesigned for each test case so that the test case passes. Once the developer does this for a number of requirements, the tests are refactored and the process is completed again. This is so the tests get more detailed and the code gets much more secure.

6 Implementation

DB implementation

- The SQL dialect to be used is MySQL. The tables which will persist data will be orders, users and trucks.
- The relationship between users and orders will be one-to-many(1:M), as customer users can have multiple orders, but each order can only have a single user assigned as the order owner. Similarly, driver users can be assigned to multiple orders, but each order can only have a single driver user assigned.
- The relation between trucks and orders will be one-to-many(1:M), as trucks can have multiple orders, but an order can only be assigned to a single truck.
- The relation between driver users and trucks will be one-to-many(1:M), as a driver user can drive multiple trucks, but a truck can only be assigned to a single driver.
- The relation between dispatcher users and trucks will be many-to-many(M:M), as an available truck can be assigned to multiple dispatcher users and a dispatcher user can manage multiple trucks.

API implementation

- The API consists of multiple microservices. The API will be developed using Spring Boot.
- The API implementation will be based on the MVC (Model-View-Controller) architecture. The implementation will be broken down into models/entities, controllers and services.

UI Implementation

- The UI for the web app will be implemented using React (HTML, Javascript, CSS)
- The UI for the mobile app will be implemented using Android Studio (Java)
- Each of the three types of users (customer, dispatcher, driver) will have their own landing page, settings, and navigation to access what they need based on our design
- Both versions of the UI will have consistent CSS styling
- Both versions will utilize data binding with frontend components
- Both versions will communicate with the same backend APIs mentioned above

ENTITIES

Each User entity will consist of data attributes:

1. name
2. address

3. location
4. role

Each Truck entity will consist of data attributes:

1. make
2. color
3. weight
4. fuel capacity
5. cargo capacity
6. availability
7. driver
8. dispatcher
9. license plate no.

Attributes such as fuel capacity and cargo capacity will be used in the truck allocation service, as these parameters will be used in the allocation formula, which compares the order entity weight and amount attribute to the trucks capacity and ability to most efficiently and securely transport the cargo.

Each order entity will consist of data attributes:

1. cargo pickup origin location
2. cargo destination location
3. weight
4. number of items(if applicable)
5. type of cargo
6. order owner
7. assigned truck
8. route reallocated(if applicable)
9. truck reallocated(if applicable)

The assigned truck attribute will maintain a relationship with the driver and dispatcher user entities, which will allow order entities services to get access to an order's associated driver and dispatcher information. The origin location and destination location data will be consumed by the route allocation service.

The route reallocated and truck reallocated boolean variables will assist in keeping track of whether an order was fulfilled without any irregularities such as obstacles or obstructions. This persisted data can potentially be used to further optimize API services and algorithms so the number of route reallocation or vehicle reallocation is reduced.

SERVICES

The majority of services will be implemented using the Spring framework and will perform multiple functions such as DB CRUD operations and algorithm execution, with the exception of few services which will be implemented using external APIs such as route mapping and communication.

Communication, account, and order tracking services will be accessible to all users.

1. Communication service will potentially be implemented using an external API such as Firebase Cloud Messaging or Twilio Programmable Messaging API.
2. Account service will be a ground up internal implementation. This service will allow users to perform account management actions such as changing personal information.
3. Order tracking service will be a ground up internal implementation. This service will allow users to track the status of active orders, such as the order cargo condition and the location of the cargo.

New order service will only be accessible to customer users.

4. New order service will be a ground up internal implementation. This service will allow customer users to place new orders.

Order management service is only accessible to customer, driver and dispatcher users.

5. Order management service will be a ground up internal implementation. This service will allow customer, dispatcher and driver users to actively track and update the status of orders.

The truck allocation and route allocation services are not directly accessible by users, but instead are called by other services.

6. Truck allocation service will be a ground up implementation. The service will essentially assign trucks on the basis of an algorithm which consumes order and truck params and outputs the truck which will ensure the most efficient and safe delivery of the cargo.

CONTROLLERS

Each service has its own respective controller which will service http requests from clients with json data responses. Http request/response handling with the client will potentially be handled via implementation using Retrofit or OkHttp libraries. Image files will be handled using Picasso library.

Controllers will handle request/response filtration on the basis of user role. Each request to the API will consist of user information such as username and user role. A user with the role designated as customer will only be able to receive 200 http responses from the new order service, as the request has gone through successfully. But a user with the role designated as driver will receive a 401/403 http response, as drivers are not allowed to make new orders.

7 Professionalism

This discussion is with respect to the paper titled “Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment”, *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Area of Responsibility	Description	ACM addresses
Work Competence	Doing high quality and effective professional work	2.2 Acquire and maintain professional competence - taking responsibility for maintaining competence, continuing to learn and meet high standards.
Financial Responsibility	Products provide value for their reasonable cost	Responsibilities to Employers: professionals should not have any conflict of interest and be loyal to their employer to provide value.
Communication Honesty	Progress is communicated honestly to stakeholders. No deception.	2.6 Parties have the obligation and responsibility to keep that party properly informed and involved for that work.
Health,Safety, Well-Being	Ensure safety and health	Moral Responsibility - responsibility is shared for safety of a product by all individuals collaborating
Property Ownership	Respecting other's information	1.2 Avoid harm to others, including their property. 1.5 honor property rights including copyrights and patents.
Sustainability	Protect the environment	1.1 1.2: computing professionals shall design and develop systems alert to any environmental damage - avoiding harm to any unwanted environmental impacts.
Social Responsibility	Product provides some value to society	2.7/3.2: Improve public understanding of computing and its consequences - share technical knowledge with the public.

Table 7.1.1. ACM Areas of Responsibility [1]

Area of Responsibility	Differences between ACM and NSPE
Work Competence	ACM 2.2 focuses on the general high standards regarding professional competence which include reflective analysis, ethical challenges, and upgrading skills. Meanwhile NSPE 2.2 states that an engineer should not partake in work and projects that are not a part of their field or they are not qualified for. It focuses on the qualifications of the engineer and questions their competence for a particular project.ACM 2.6 is closely related to NSPE 2.2
Financial Responsibility	ACM states that professionals should not have any conflict of interest and be loyal to their employer to provide value. What NSPE 2.4 states differently from ACM is that engineers should not accept financial compensation for a project from multiple parties. Engineers shall not solicit or accept financial compensation for work they are responsible for.
Communication Honesty	ACM 2.6 says that parties have the obligation and responsibility to keep that party properly informed and involved for that work. NSPE says that engineers shall advise their clients or employers when they believe a project will not be successful and that they shall not attempt to attract an engineer from another employer by false or misleading pretenses.
Health, Safety, Well-Being	ACM mentions that responsibility is shared for safety of a product by all individuals collaborating. NSPE section 2.1 addresses the health, safety aspect in a much broader way. It includes that if engineers' judgment is overruled under circumstances that endanger life or property. Engineers should not permit the use of their name businesses that they believe have engaged in fraudulent activities.
Property Ownership	ACM section 1.2 states that avoid harm to others, including their property and 1.5 states that honor property rights including copyrights and patents. NSPE only briefly mentions that an engineers' designs and data referring exclusively to an employer's work are the employer's property.

Sustainability	ACM section 1.1 1.2 addresses that computing professionals shall design and develop systems alert to any environmental damage - avoiding harm to any unwanted environmental impacts. NSPE section 3.2.4 only briefly discusses sustainability as compared to ACM.
Social Responsibility	ACM 2.7/3.2 instructs to improve public understanding of computing and its consequences - share technical knowledge with the public. NSPE treats social responsibility in a much broader way. It states that engineers should not omit a material fact. they should prepare technical articles for the press. They are encouraged to participate in civic affairs.

Table 7.1.2. Differences in ACM and NSPE code of ethics [1], [4]

7.2 Project Specific Professional Responsibility Areas

Work Competence - HIGH

Work Competence is a major part of our project as individuals and teams who accept personal and collective responsibility for gaining and sustaining professional competence are essential. We as a team would develop and enhance our skills to meet the project requirements to deliver an optimal product. Doing solo research and attendance at team and client meetings is essential to showcase work competence. Our team encourages and facilitates these activities. Our team is at a High rating for work competence because it is absolutely essential for the development of our deliverables.

Financial Responsibility - MEDIUM

While financial responsibility is not a major part of our project, it is applicable in a few regards. The primary aspect is ensuring that we do not knowingly implement a suboptimal solution when a better solution is possible and realistically doable. Implementing a suboptimal solution would not be providing our best work and would be depriving our mentor of a more desirable final product. Additionally, such a solution could require a more expensive server in order to be efficient. Such an expense would not be financially responsible when optimizations could realistically be made to bring the cost down.

Our team is currently at a medium rating for financial responsibility, mainly because we have not done much in regard to actual implementation. As such there haven't been any decisions that carry financial weight so claiming high rating would not necessarily be fair, but it is something that is on our mind as our project centers on optimization so we are always trying to get the most out of as few resources as necessary.

Communication Honesty - HIGH

Communication honesty will be a very important part of our project. Considering that we're a large group, consistent organization of roles and assigned tasks is critical in preventing discredit of intellectual property. In addition, we need to be careful in honoring the property rights of any software we research and use that's not ours. In terms of trust, a large team like ours needs to have trust that we can each complete the tasks that we assign for ourselves. And then finally, it's important that we do our best not to discriminate or tolerate discrimination on the basis of race, sex, religion, age, disability, national origin, or other such factors.

Health, Safety, Well-Being - MEDIUM

Health and safety are definitely important for our project to manage the risks and protect our user's products. As many other projects or workplaces, users need to feel safe when they are using a certain product. Health and safety are not only to ensure that we gain our users' trust, but it can also lower any extra cost that might accrue due to any damage or loss for the products.

Since we haven't done any actual implementation yet, I would say our group is on the medium range for this responsibility. We have been mostly working on research and deciding on what platforms we will be using, but health and safety is very important and will be considered when we start implementing our design.

Property Ownership - HIGH

We as a team are cognizant of not harming individuals or corporations in terms of unjustified physical or mental injury, unjustified destruction or disclosure of information, and unjustified damage to property, reputation, and the environment. It is of HIGH importance to our team. We are devoted to applying the best practices everywhere needed. If a conflict of property rights arises, we shall work towards resolving it in a non-hostile manner. If we may need to use or incorporate any individual's or corporation's property, we shall do it in the most ethical manner possible.

Sustainability - MEDIUM

Sustainability is directly tied to our goal. The more optimized the initial routes and new routes are, the less time the trucks will be one the road. This means less carbon emissions from the trucks. Additionally, if our algorithms are optimized, they will run for shorter periods of time, meaning less electricity is used. So, our project by definition is geared toward sustainability.

The reason for the medium rating is that we have not currently done much work that is directly tied to these related processes. Most of our effort has been on research and selecting the external APIs, tools, and languages that will be used to implement our design. Our rating for this category should increase to high once development has started.

Social Responsibility - MEDIUM

As computer professionals, it is our team's responsibility to share technical knowledge with the public, foster awareness of computing, and encourage understanding of computing. It is our responsibility to communicate with individuals outside our group and the communication should be clear, respectful, and welcoming. Important issues that shall be discussed include the impacts of computer systems, their limitations, their vulnerabilities, and the opportunities that they present. Our team accepts the associated social responsibilities that includes reducing harm to the public and raising awareness of the influence of our product in the relevant areas and businesses.

7.3 Most Applicable Professional Responsibility Area

Sustainability on professional responsibility is both very important to our project and one that we have been conscious of, be it indirectly. This responsibility is important to our project because it is one metric by which we could measure success. The more optimized our algorithm, the less electricity and gasoline will be used. We have demonstrated responsibility in this regard by exploring several possibilities for optimizing our routing algorithm. We have looked into implementing algorithms from scratch as well as using existing external APIs and tools. Optimization has been a priority during this process, but so has ease of implementation. Both of these are correlated to sustainability because while a highly optimized algorithm will definitely lead to short routes and thus higher sustainability, if the implementation is difficult and requires much longer to thoroughly implement and test, the net gain might not be as high as anticipated. Additionally, there is no guarantee that an algorithm implemented from scratch that is theoretically more efficient will be more efficient in practice. In such an instance, it is likely better to go with an existing implementation that has been tested and/or optimized by several other developers. This particular issue has impacted our project by pushing us to go with an existing service in Mapbox because it is proven and we are less likely to waste resources implementing an algorithm that may or may not be better.

8 Closing Material

8.1 Discussion

As most of this semester has been dedicated to creating a concrete development plan, minimal time has been given to developing the assigned product. While we do have a rough prototype, it does not come close to satisfying the requirements outlined in this document. However, we have thoroughly planned out how we will meet all of the requirements next semester. The steps we will take to satisfy the requirements can be found throughout this document, primarily in section 3.4 with the Gantt chart and section 4.4 where we outline the proposed technologies and steps for implementing our design.

8.2 Conclusion

The project goal for our team is to implement a truck delivery app that will solve the problem of assigning a truck from a given fleet to a new request, or how to re-assign truck(s) to respond to dynamic changes in traffic and requests. This design should meet and satisfy every user's need for this app. Currently we have done our research and we decided on what software functionality we will be using in developing our app. Overall, our team has made a significant process on our project this semester and we are planning to start working on our implementation starting from January and have it done in May. Next semester we are aiming to create a working truck allocation algorithm, create UI pages and experiment with react, and verify existing solutions that we found from previous research that we have looked at while we were trying to find the best solution for our design. Whether or not these goals will be achieved is to be seen. We believe our work in designing and planning our solution will lead to each goal being sufficiently met or exceeded. Similarly, alternate designs that would have led to better results will not be known until we attempt to implement the solution outlined in this document.

8.3 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

[1] Association for Computing Machinery, "Code of Ethics," *Code of Ethics*, 2018. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: 05-Dec-2021].

[2] Cappanera, P., Requejo, C., & Scutellà, M. G., "Temporal constraints and device management for the Skill VRP: Mathematical model and lower bounding techniques", *Computers & Operations Research*, vol. 1024, no.1, Dec. 2020, doi:10.1016/j.cor.2020.105054

[3] El-Sherbeny, N. A., "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods", *Journal of King Saud University - Science*, vol. 22, no.3, pp. 123-131, doi:10.1016/j.jksus.2010.03.002

[4] National Society of Professional Engineers, "Code of Ethics for Engineers", *Code of Ethics for Engineers*, July 2018. [Online]. Available: through Canvas Professionalism assignment.

[5] Wang, J., Zhou, Y., Wang, Y., Zhang, J., Chen, C. L., & Zheng, Z., "Multiobjective Vehicle Routing Problems With Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms", *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp., 582-594, doi:10.1109/tcyb.2015.2409837

8.4 Appendices

8.4.1 Important Resources

Link to higher resolution version of Figure 4.3.1.1. System Architecture:
https://drive.google.com/file/d/1gKmW9odJ6XZqK3dwpKFoIE1DHZKHngbK/view?usp=sharing

Mapbox api:

https://docs.mapbox.com/api/navigation/directions/

https://docs.mapbox.com/api/navigation/map-matching/

https://docs.mapbox.com/help/glossary/geocoding/

https://docs.mapbox.com/api/navigation/

8.4.2 Team Contract

Team Members:

1) Joshua Heroldt

2) Nolan Slimp

3) Bernard Fay

4) Matthew Medley

5) Indrajeet Roy

6) Asma Gesalla

7) Siddharth Rana

<u>Team Procedures</u>

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

The current time and location for regular team meetings will be on discord following the TA meeting on Fridays at 1:30-2:00. This time will be allocated to weekly scrum sessions where each team member will give an update on the progress that has been made for the week.

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

Communication updates are to be given in two forms. Primary communication will occur through discord with notifications being sent to all group members when there are events or meetings occurring related to the team. A second method of communication is email, where if anyone has not made contact with the team for an event or meeting an email reminder will be sent out to the specific members of the team who were unresponsive.

3. Decision-making policy (e.g., consensus, majority vote):

Consensus will be the main method of reaching policy decisions within the group. This will be done for all whole-group related topics, while when smaller teams are created for different sections of the project those individual teams will be able to reach decisions by consensus that they think are beneficial for their small group and can bring those decisions to the entire team for discussion if they see fit.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

Notes and important information for ta and faculty meetings will be at the discretion of individuals to make during or following the meetings. However any notes that are made will be compiled and put into the google drive folder so that any team member who was absent from the meeting will be able to see what they missed. Meeting minutes will be kept by the attendees of the meetings in their notes sheet and the time will be recorded once notes are compiled into the single notes document for said meeting date on google drive.

<u>Participation Expectations</u>

1. Expected individual attendance, punctuality, and participation at all team meetings:

All team members are expected to attend and be on time for all meetings unless otherwise specified. If someone is unable to attend, advance notice is preferred, but life happens and we understand that advance notice isn't always possible. Everyone is encouraged to participate in team meetings if there's something you want to say, but no one will be required to speak.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

Everyone is expected to have their assigned work completed by the predetermined deadline and to the level specified when that work was assigned. Deadlines will be determined prior to assigning work to prevent deadlines from continually being pushed back. In the case issues arise or the assignment requires more work than expected, the assignment will be broken up into smaller assignments and new deadlines will be given for the smaller assignments.

3. Expected level of communication with other team members:

Everyone is expected to communicate when significant issues arise with any assigned work. Regular communication and progress updates are highly encouraged. If someone isn't communicating for an extended period of time, the team will reach out and make sure that everything is going okay. This should be done first through the discord server or through private messaging, and secondarily through the iowa state email.

4. Expected level of commitment to team decisions and tasks:

Team members are strongly encouraged to comment on all decisions and tasks. Since roles are being chosen by team members and responsibilities and team decisions are being made by consensus, once a decision has been made the team should be able to fully commit to said decision. The same can be said for tasks that team members are assigned. It is crucial that each team member not only communicate with others about their tasks and if any help is needed, but engage with all tasks that are assigned with their full commitment.

<u>Leadership</u>

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, scrum master, database management, team website manager, main documentor, ect.):

- Joshua Heroldt
  - Team organization
  - Client interaction
- Bernard Fay
  - Database management
  - Meeting Scribe
- Nolan Slimp
  - Scrum Master
- Asma Gesalla
  - Backend documentor
- Matthew Medley
  - Team Website Manager
  - Frontend architecture design

- ❖ Indrajeet Roy
  - ➢ Frontend documentor
- ❖ Siddharth Rana
  - ➢ Individual component design

2. Strategies for supporting and guiding the work of all team members:

We as the team want to create a strong culture of open communication and the space where anyone can ask questions. Even though roles are being assigned everyone should be free and able to aid anyone else in the group if they are struggling and this is encouraged among team members. Additionally the group will aim to use common coding practices in an effort to mirror industry standards with regards to developing and implementing the project.

3. Strategies for recognizing the contributions of all team members:

Each team member will have their contributions recognized through weekly meeting checkups with each group member. This will allow everyone to be on the same page with how the project is progressing and allows each member to share their contributions to the project with everyone else.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

- Nolan and Matthew have worked on web applications and have skills in React and Angular.
- Josh has worked with real-time web applications that interact with large sets of data
- Bernie and Siddharth have worked with backend development (Spring Boot) and are proficient in designing and implementing algorithms.
- Indrajeet has worked with the android platform and Spring framework, and has internship experience in production android application development.

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

- After standup on Fridays, a discussion will take place to contribute new ideas and keep on track.
- Having all projects be available for everyone to look at
- Having all research and code be in a shared location so that everyone can comment or view it
- Creating channels for specific small groups so that if ideas are needed small group members can collaborate with other team members
- If needed idea brainstorming sessions can be done during normal meeting times following standup so that outside perspectives can be shared

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?)

- If there is a conflict between members, it will be forwarded to the team leader and addressed in standup. There, the team will collaborate and resolve the issue.

- Emails or private messages can be sent to members outside of the student in question's small group and brought up by said outside member if the student in question is not comfortable speaking out themselves

<u>Goal-Setting, Planning, and Execution</u>

1. Team goals for this semester:

- Create a project plan that will be able to be implemented fully by another team without any communication with our initial team.
- Aim to learn and interact with new forms of coding that are more closely related to actual industry work
- Learn how to use new api's and technologies that we have never worked with before

2. Strategies for planning and assigning individual and team work:

- Split the team up into different small groups so that the project can be chunked up into more manageable sections
- Having the scrum master create stories so that specific sections of the project become more manageable
- Make work fall under one specific small group if possible, so that members of that small group can collaborate with each other in order to used their shared skills to attempt to solve the problem

3. Strategies for keeping on task:

- Assign small goals for groups to work on
- Rotate/decide among small groups who is best suited for what tasks

<u>Consequences for Not Adhering to Team Contract</u>

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

Infractions will be dealt with in a tiered structure. Firstly the student in question will be contacted by the group and a compromise will be attempted to be worked out. If this fails or if the student is unresponsive then the next level of intervention will be to contact our TA in order to issue a stronger warning. Finally if this fails then the professors will be contacted in order to issue a punishment to the student in question.

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

If infractions become a common occurrence each infraction will be clearly documented as to leave no doubt that the student in question has broken the rules of this document. These infractions will be compiled and presented to the student in question along with the TA and professor being contacted to request for aid in resolving the issue. This will only occur if infractions become a common occurrence and the team has been unable to resolve the issues with the member in question.

****

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) Joshua Heroldt	DATE 9/19/2021
2) Nolan Slimp	DATE 9/19/2021
3) Bernard Fay	DATE 9/19/2021
4) Indrajeet Aditya Roy	DATE 9/19/2021
5) Siddharth Rana	DATE 9/19/2021
6) Matthew Medley	DATE 9/19/2021
7) Asma Gesalla	DATE 9/19/2021