SDMAY22-32

Multiple Vehicle Routing Problem with Broken Trucks

Nolan Slimp – Scrum Master

Joshua Heroldt - Client interaction lead

Indrajeet Aditya Roy - Frontend documentor

Bernard Fay – Meeting Scribe

Asma Gesalla - Backend documentor

Matt Medley - Team website manager

Siddharth Rana - Individual component design

Goce Trajcevski - Faculty Advisor

Project Vision

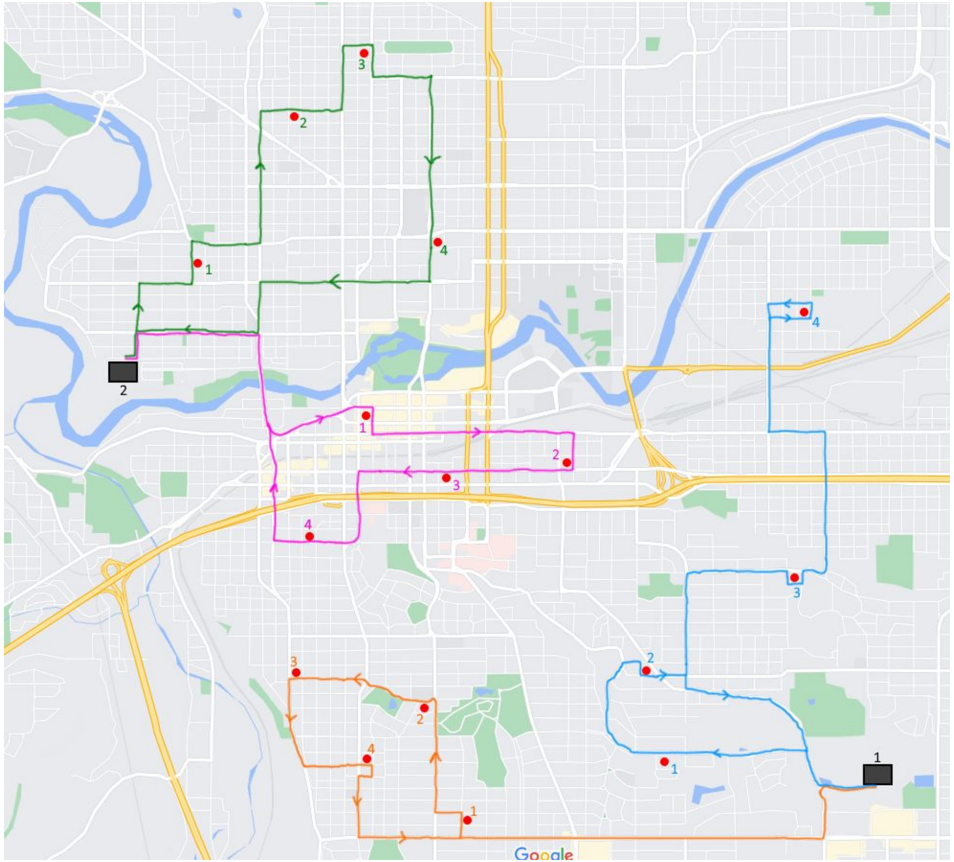
- Aims to solve the multiple vehicle routing problem
    - Constrained by the condition that a vehicle breaks down mid route
    - Will contact different types of users upon a vehicle problem
- Aims to use temporal distribution of traffic parameters in order to get more realistic estimates of route travel times
- Different UI's for different types of users
- Dynamic routing done at both the onset of a route and when a breakdown occurs
- Algorithm will account for the loads of all trucks in the area of a broken down truck

Who does this project impact?

- General public health impact
  - Allows for more efficient transit for ambulances
  - Less of a risk for accidents with pedestrians
- Environmental impact
  - Emissions from a fleet of trucks will be drastically lessened
- Economic impact
  - Corporations will be able to maximize profits with their existing fleet without any other changes

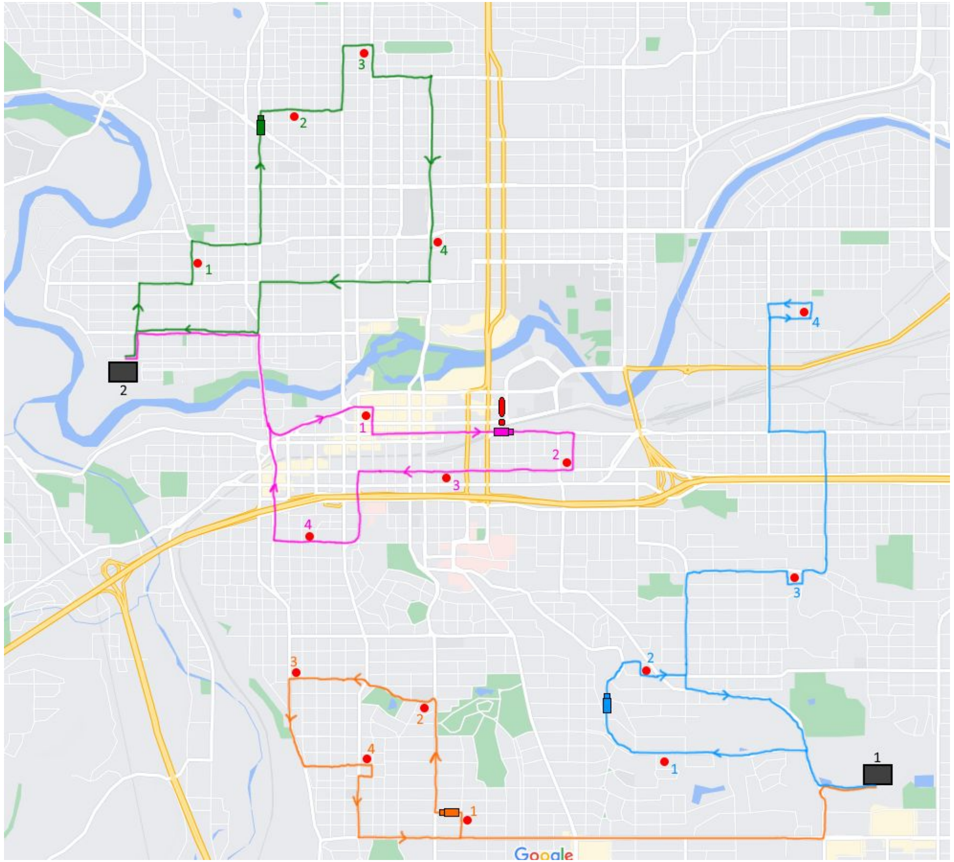
Use case (Route reallocation sketch)

- 2 warehouses with 2 routes each



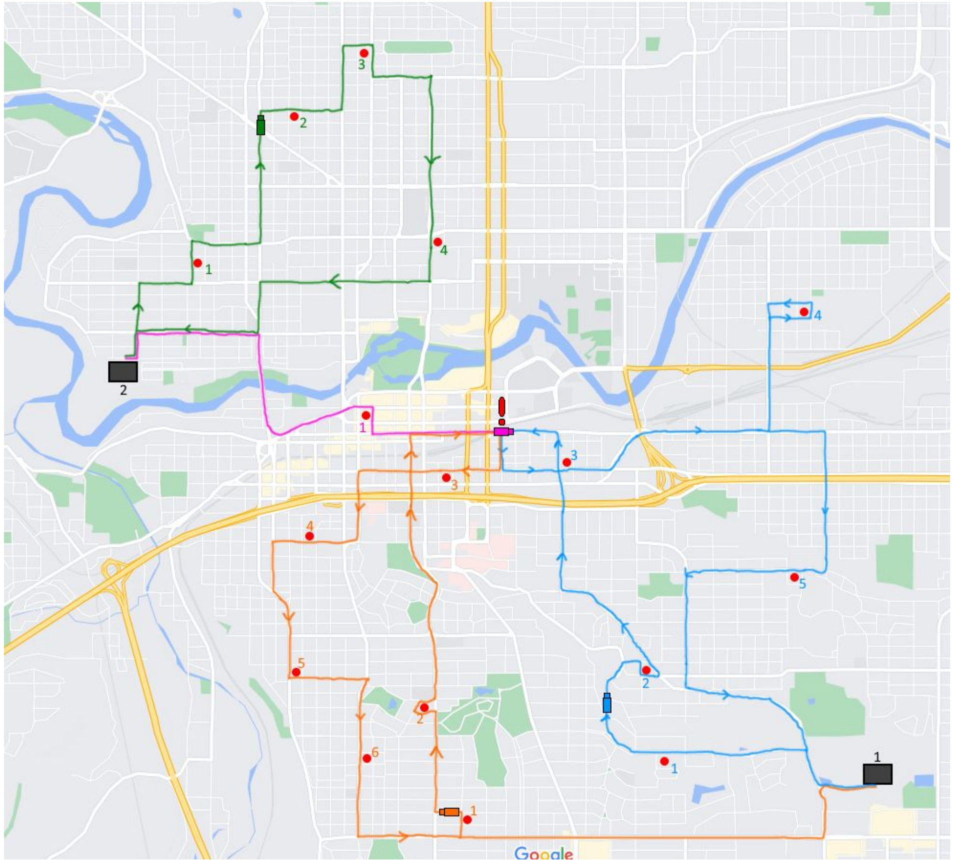
Use case (Route reallocation sketch)

- 2 warehouses with 2 routes each
- Pink truck breaks down



Use case (Route reallocation sketch)

- 2 warehouses with 2 routes each
- Pink truck breaks down
- Orange and Blue trucks are allocated to pick up and deliver remaining load from Pink truck



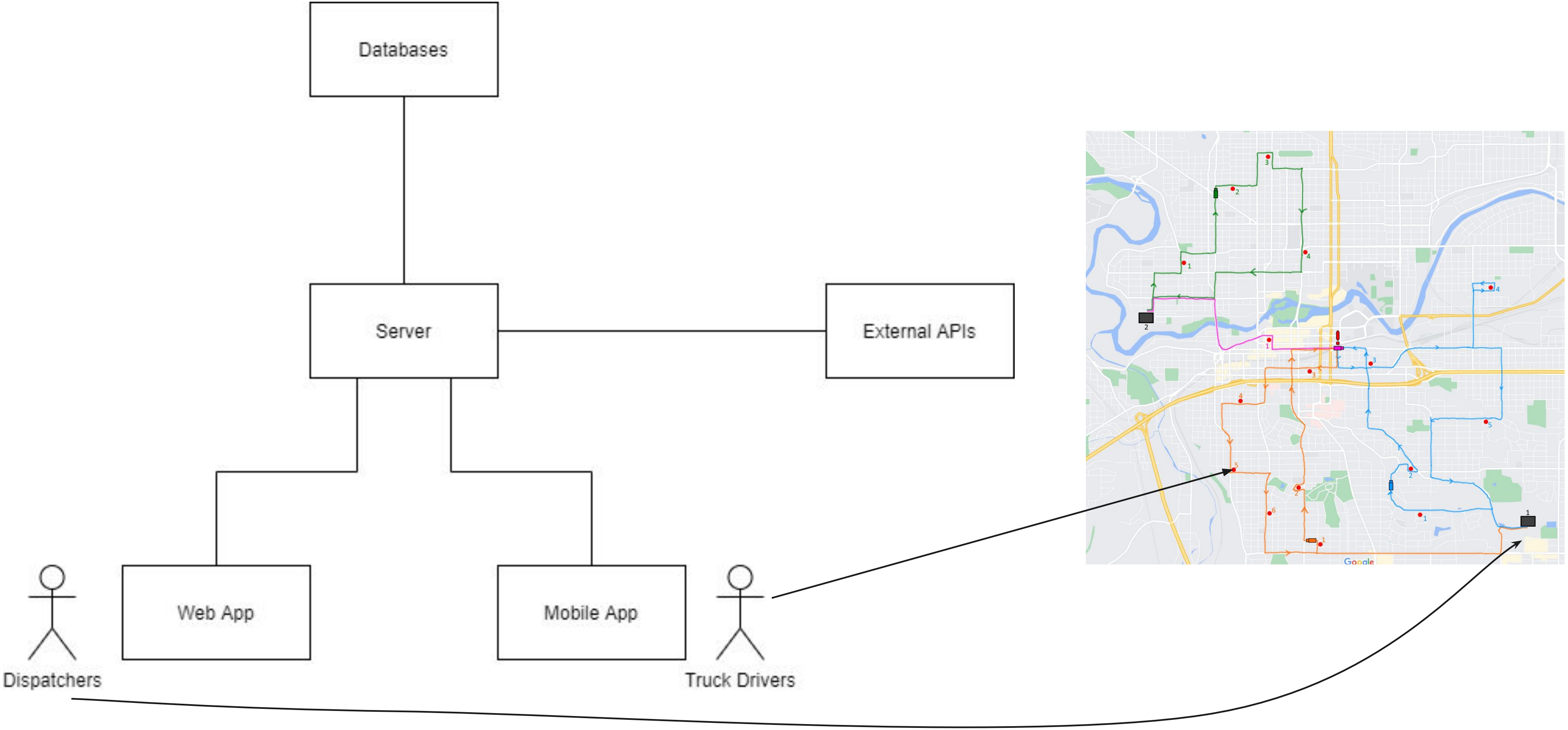
Functional Requirements

- Functional requirements
  - Each truck stores:
    - Initial location
    - Delivery location
    - The current load
  - Algorithm requirements
    - Initial truck routing
    - Estimate truck location
    - Reassign trucks when breakdown occurs
  - UI Requirements
    - Desktop and mobile app
    - Display notifications
  - Constraints
    - Response time
    - Route allocation based on traffic parameters

Other Requirements

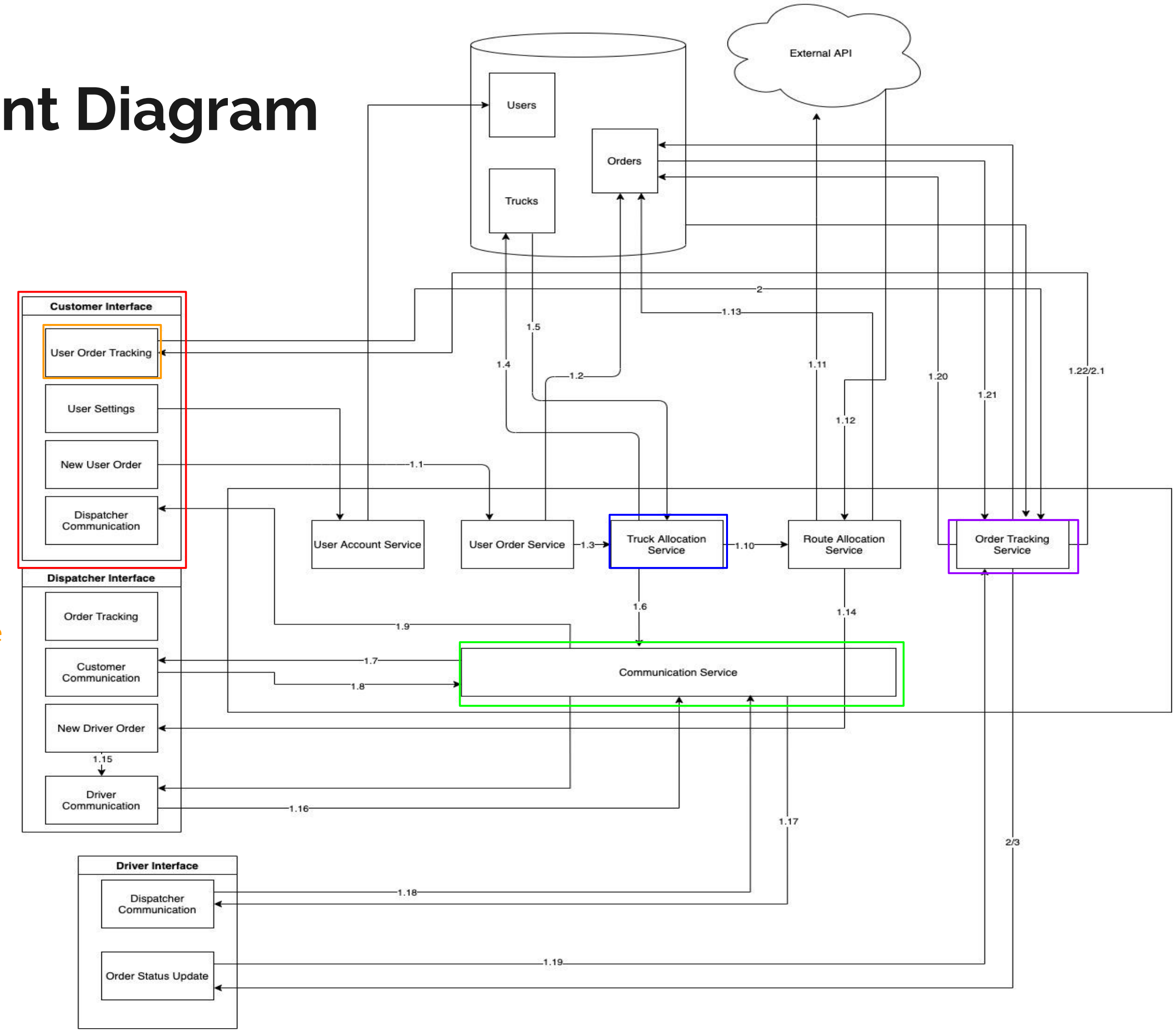
- Economic requirements
  - Minimize:
    - Delivery delay when a truck breaks down
    - Idle time of trucks
  - Maximize
    - The amount of goods delivered
- Resource requirements
  - A constantly running server (database, requests, algorithm)
  - Android mobile devices (trucker mobile app)
  - Visualization tools/frameworks (desktop and mobile)

Conceptual Design Diagram (High level)



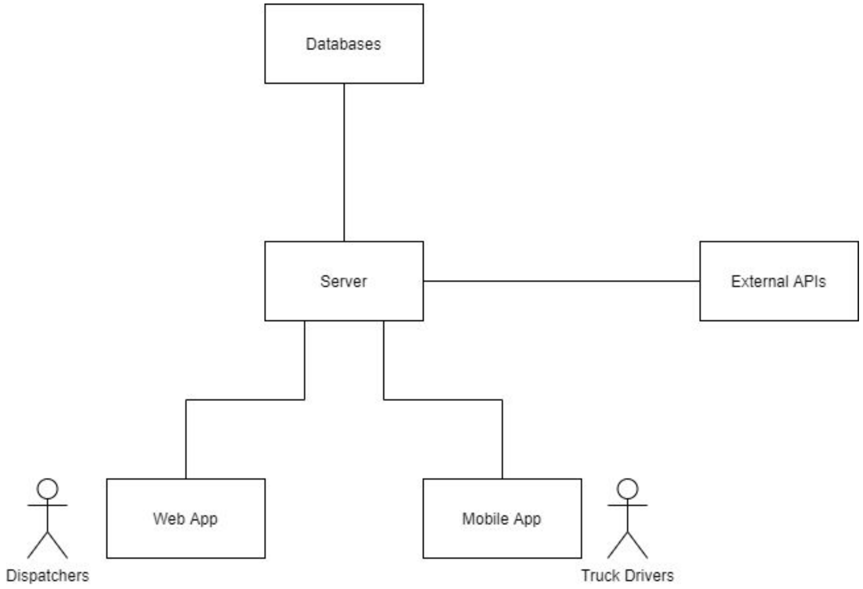
Detailed Component Diagram

- A customer can place a new order
- The truck allocation service fetches truck information and allocates the truck
- Communication service establishes and handles communication
- A customer can track their order via the User order tracking component
- The requests will be handled by the order tracking service,



Components and Modules

- Customer/dispatcher/driver interfaces
- DataBase (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



Research 1: Foundations

Algorithm 1 MOLS

archive $A = \emptyset$
generate several nondominated solutions to initialize A /*Initialization*/
while running time $\leq$ maximum computation time **do**
  x = randomly select a solution from archive A
  for $obj = 1$ to 5 **do**
    perform objectivewise local search $LS_{obj}(x)$ /*Objectivewise local searches*/
    update archive A using neighbor solutions generated during the objectivewise local search. /*Archive updating scheme*/
  end for
end while
return A

- MOLS significantly better in Solomon Benchmark
  - 56 instances, MOMA better in one
- MOMA outperforms MOLS more often in real life
  - 45 instances, MOLA better in 12

Wang, J., Zhou, Y., Wang, Y., Zhang, J., Chen, C. L., & Zheng, Z. (2016). Multiobjective Vehicle Routing Problems With Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms. IEEE Transactions on Cybernetics, 46(3), 582-594. doi:10.1109/tcyb.2015.2409837

Algorithm 4 MOMA

Archive $A = \emptyset$
generate Q uniformly distributed weight vectors $\Lambda^1, \cdots, \Lambda^Q$, where $\Lambda^i = (\lambda_1^i, \cdots, \lambda_5^i)$ /*Decomposition*/
for $i = 1$ to Q **do**
  compute the Euclidean distance between each pair of weight vectors and get the T closest weight vectors to each weight vector. Set the neighborhood $B(i) = i_1, \ldots, i_T$.
  initiate x^i
end for
while stopping criterion is not met **do**
  for $i = 1$ to Q **do**
    choose p, q randomly from $B(i)$
    $o = crossover(x^p, x^q)$ /*Crossover Operator*/
    if $\exists obj \in \{1, \ldots, 5\}, \lambda_{obj}^i == 1$ **then**
      $x' = LS_{obj}(o)$ and update archive A /*Objectivewise local searches: Algorithms 2 and 3*/
    else
      $x' = LS_{\Lambda^i}(o)$ and update archive A /*Decomposition-based local search: Algorithm 5*/
    end if
    for each $j \in B(i)$ **do**
      if $g^{ws}(x'|\Lambda^j) \leq g^{ws}(x^j|\Lambda^j)$ **then**
        $x^j = x'$
      end if
    end for
  end for
end while
return A

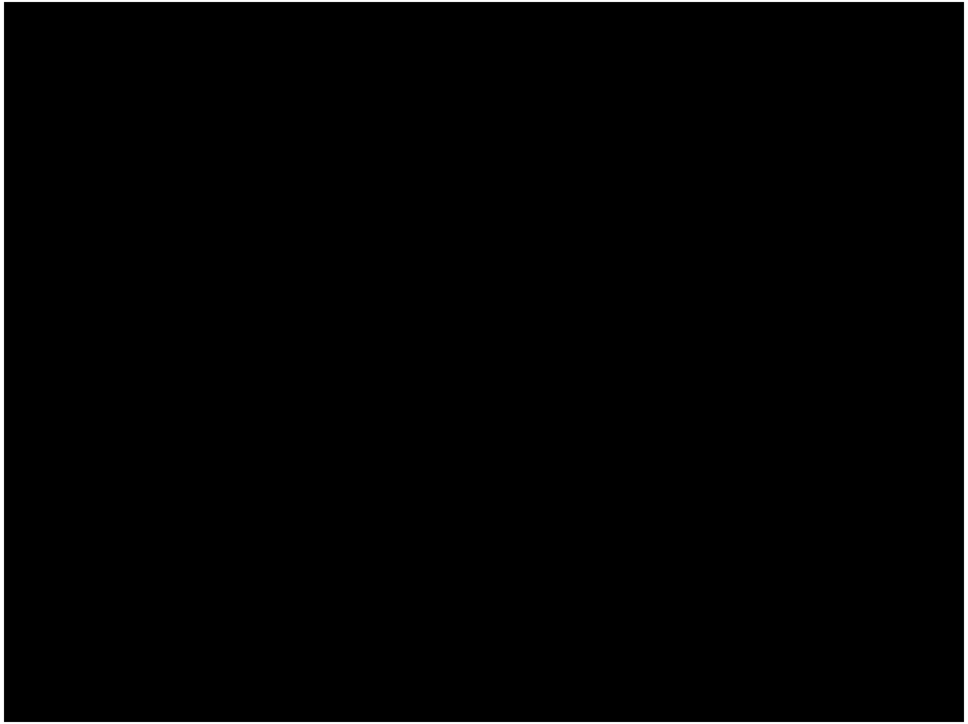
Research 1: Foundations

- General
  - https://github.com/CUTR-at-USF/awesome-transit
    - Collection of everything related to transport and maps
- Visualization
  - https://docs.mapbox.com/api/maps/
- Algorithms
  - https://github.com/valhalla/valhalla
    - Routing engine
  - https://optimoroute.com/load-planning/
    - Load balancing
  - https://ieeexplore.ieee.org/document/7945429
    - Map matching
  - https://developers.google.com/optimization/routing/vrp
    - Google's tools
  - https://github.com/dominictarr/dynamic-dijkstra
    - Dynamic shortest path algo

Research 2: Technology and Frameworks

- Frontend
  - Web App UI: React, selenium
  - Mobile UI: Android Studio, appium
- Mapbox
  - Enables our project to route vehicles in any city, taking into account closed roads and traffic density
  - Can use multiple vehicles with starting and ending locations
  - Endpoints for viewing map and routes in our web application
  - Also allows for geocoding of addresses that are entered into the UI from the customer
- Backend
  - Java Spring
  - Postman
  - JUnit
- Database
  - MySQL workbench

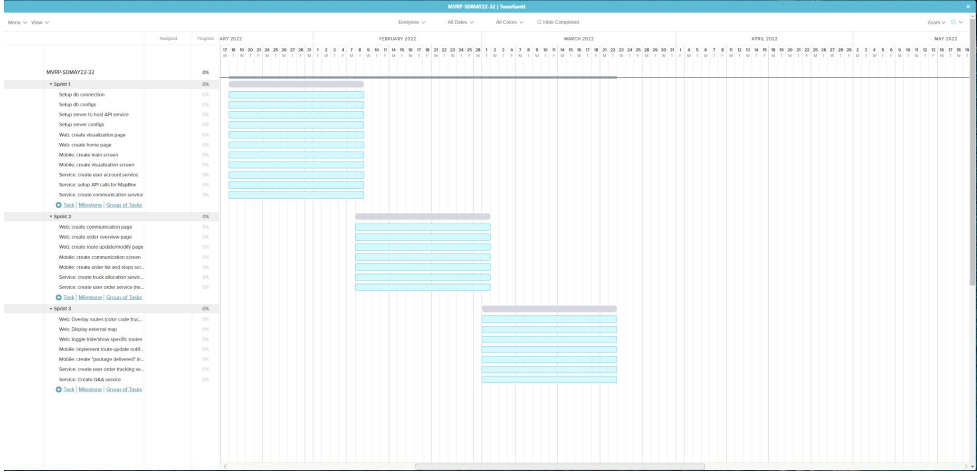
Prototype Implementation



Design Complexity

- Truck capacity constraint
  - Multiple trucks can reroute based on capacity
- Broken truck requirement
  - Vehicles are routed to deliver the goods from the broken truck
- Service to client - reliant on many modules
  - Increased complexity as communication travels through many components

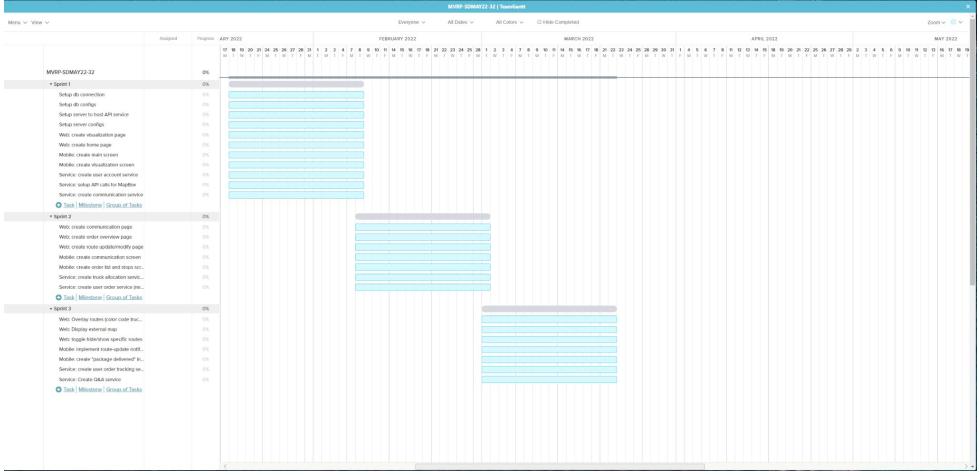
Project Plan (Development)



- Main tasks that need to be accomplished:
  - Implement Visualization Tool Front-End
  - Develop UI for Web App
  - Develop UI for Mobile App
  - Develop REST API microservices
  - Setup application DB
  - Setup application server
  - Final Application Testing

- Metrics of interest:
  - Frontend response time
  - Algorithm update speed (time it takes to get a new route)
  - General algorithm efficiency (database queries, calls to the api, ect.)

Project Milestones



- Baseline functional UI
- Alpha UI (first round of user feedback)
- New prototypes will be developed in 2-to 3-week intervals.
- The first prototype finished by the end of week 5
- Polished UI
  - UI responds 100 ms
  - Visualization tool 98% accurate
- Algorithm updates in under 20 sec -> 1 sec

Test Plan

- Multiple types of tests will be run both in tandem with development
  - Unit testing
    - Backend testing: set of static inputs
    - UI testing: end user testing and automated tests
    - Database testing: list of important queries
  - Interface testing
    - Various scenarios to match use case
  - Integration testing
    - Customer order to assignment path will be followed
  - System testing
    - Combination of prior tests with sample set of data
  - Regression testing
    - Making sure the algorithm continues to work with test data
    - Compare response time and correctness against expected results

What's next?

- Begin development in January
  - First goal: working truck allocation algorithm for initial allocation
- Create UI pages and experiment with React
  - Understand the tools we are working with better
- Verify existing solutions that will be used in the project
  - Using dynamic shortest path

Team Member Contribution

Nolan Slimp – Scrum Master

- Created Trello board and aided in shortest path algorithm research

Joshua Heroldt - Client interaction lead

- Created and spoke on lightnings talks and researched existing MVRP solutions

Indrajeet Aditya Roy - Frontend documentor

- Research implementation solutions and aided in software architecture designs. Created architecture diagrams.

Bernard Fay – Meeting Scribe

- Reviewed existing research solutions. Reviewed and submitted team documents.

Asma Gesalla - Backend documentor

- Worked on the weekley documents and spoke on the lighting talks.

Matt Medley - Team website manager

- Created prototype application and demonstration video

Siddharth Rana - Individual component design

- Spoke on the lightning talks and the youtube video