Project 1: Optimal Delivery

Group Members: Shawn, Mustafa, Anuj, Ernest, Ankita

Sprint 1

Requirements Engineering

Functional Requirements

1: Trucks should be assigned to visit all the destination points D, starting from a pickup point P
2: The combined length of the delivery routes should be the smallest possible
3: 2 possible cases regarding available trucks T: ● T < D ○ At least one truck will have to visit more than 1 delivery point ○ It is possible that they will also have to return from a delivery point to the pickup point ● T >= D ○ Designate 1 truck to 1 pickup point & 1 delivery point

Non-functional Requirements

Ease of Use	1: Each of the trucks start their routes at a pickup point
Safety	2: The routes should make allowances for petrol refills
	3: Routes should be such that they minimize chance of crashes
	4: Routes should be such that they reduce the number of right turns that the truck takes since right turns are more dangerous for trucks that are long and loaded
Speed	5: The algorithm should be performant
	6: The designed route should be produced quickly and manageable as deliveries have to be operated according to timed schedule
Reliability	7: Should minimize ROCOF & maximize MTTF

Usability Requirements

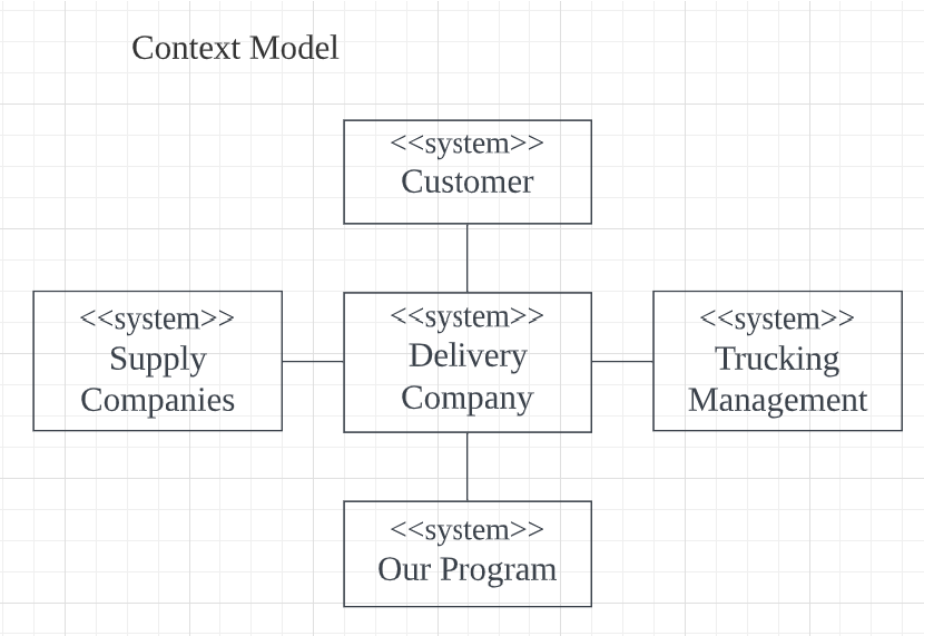
1: Service should be provided through SaaS

Requirements Elicitation

1: Interview truck drivers regarding duration & distance they normally travel per day & how much weight they carry
2: Research needs and expectations of Amazon (how they deal with numerous orders/shipments/returns

System Modeling

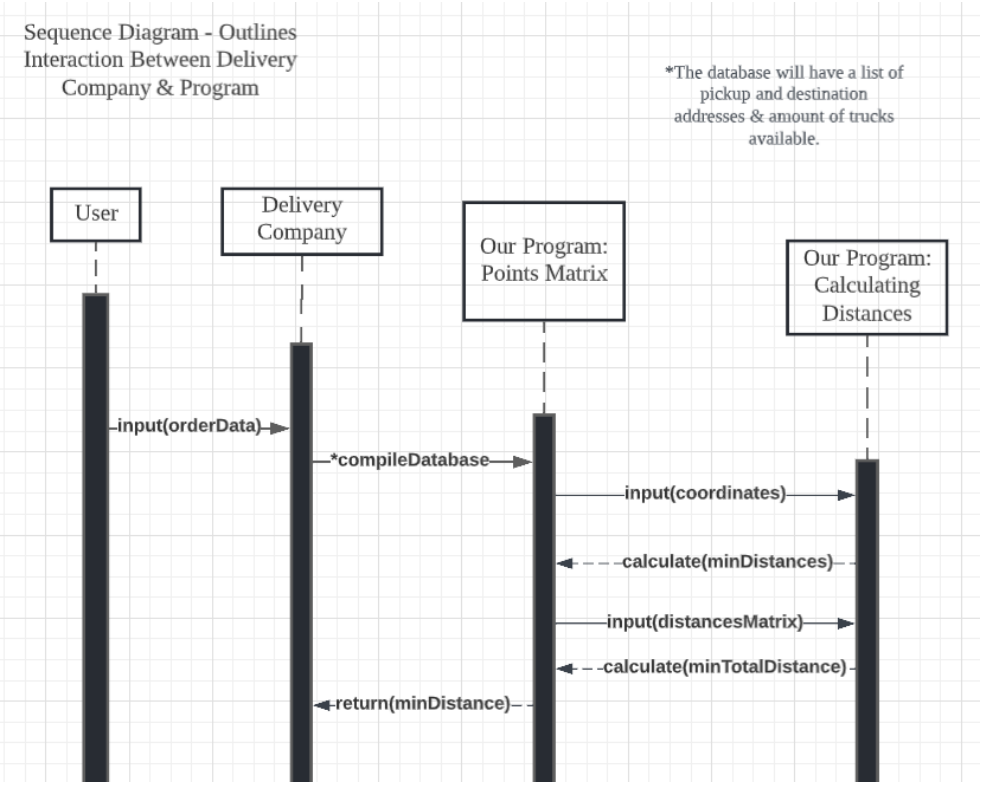
- The models illustrate the theoretical environment which is covered by the whole delivery operation system.



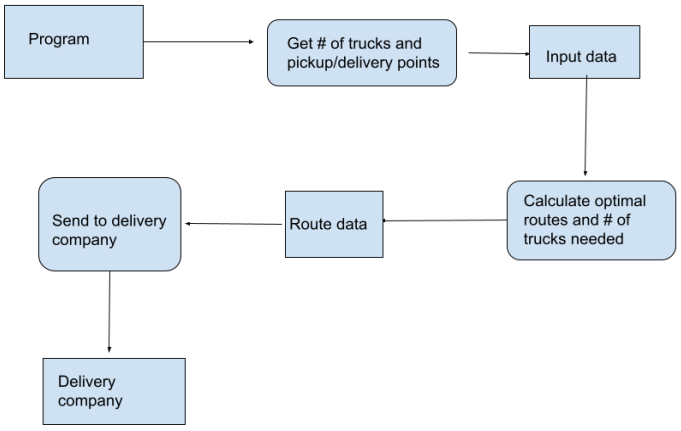
Use Case Diagram

*The consumer can also make changes to his order such as deleting, updating, adding, and changing data. The database is assumed to be updated by the delivery company accordingly.

**The database of addresses is not used in Sprint 1 code as the program outlines the basic algorithm to calculate the shortest distance which can then be implemented in different situations. The interaction between the consumer and the delivery company shows a theoretical case.

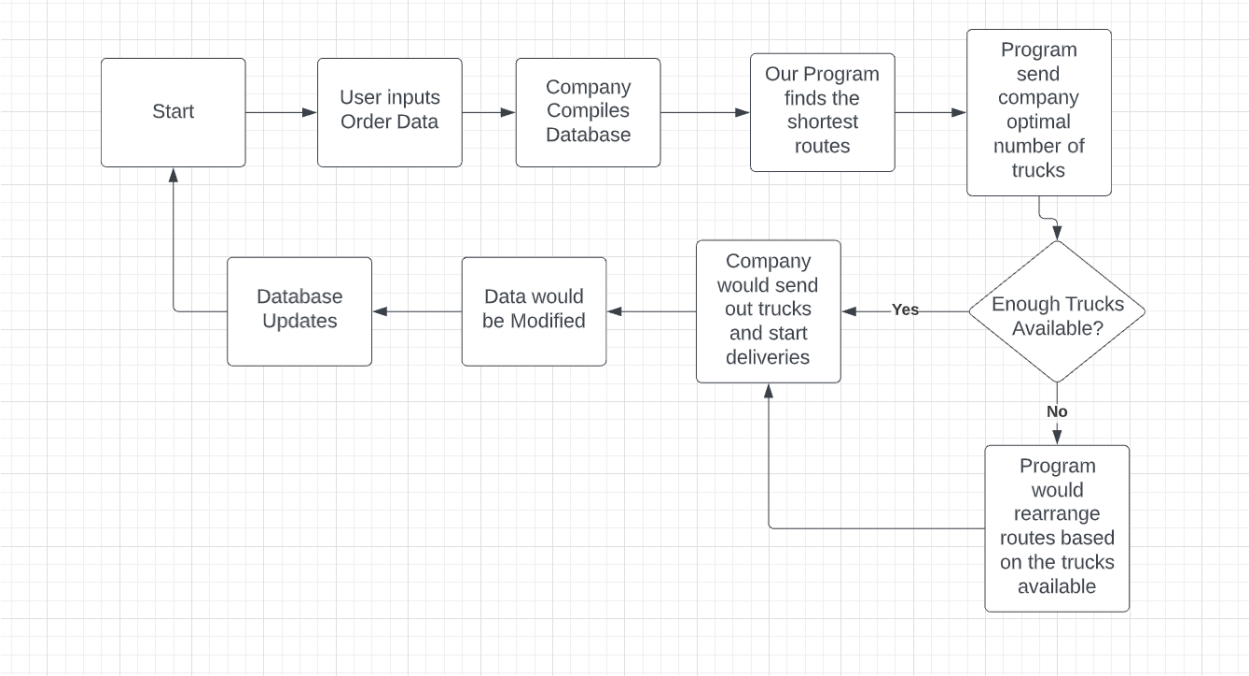


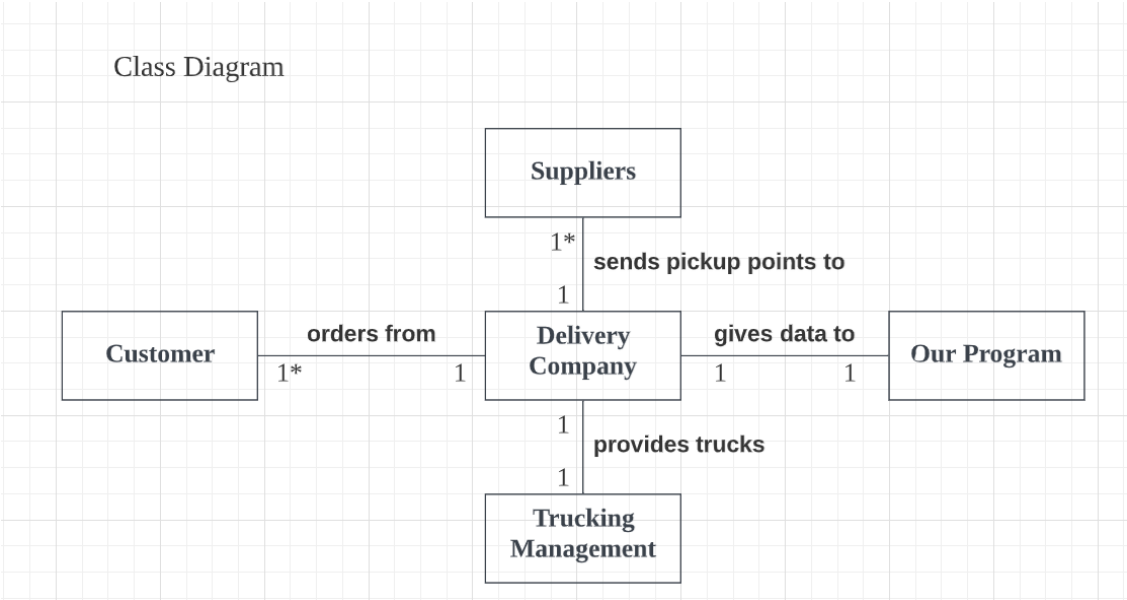
Activity Diagram



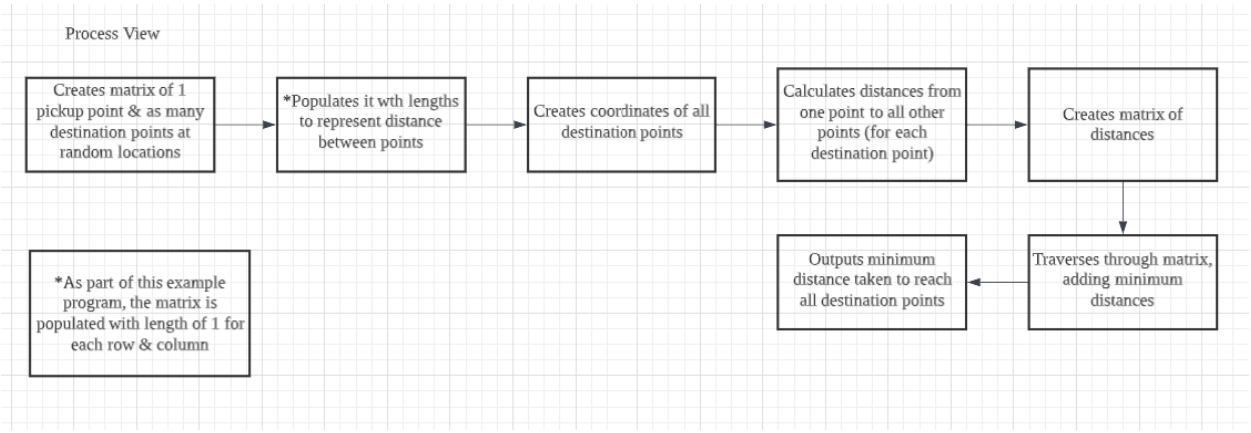
State Diagram

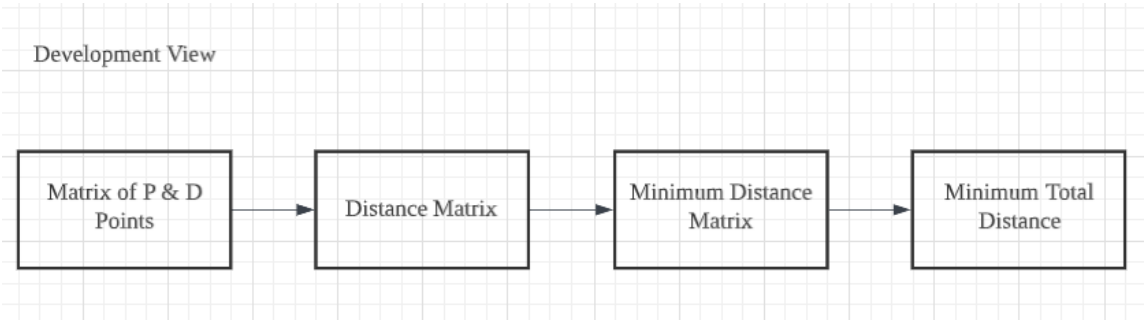
*The diagram shows a specific case of when the trucking management part of the system will influence change of states in the entire system.





Architectural Design





Specification

User Requirements

1. Routes should be assigned for each truck
2. Routes should provide stops along the routes at which trucks can fill up gas

System Requirements

1. The total distance traveled across all trucks should be minimized
2. Rate of fault occurrence should be minimized and the mean time to failure should be maximized
3. The probability of a crash should be minimized
4. Number of right turns should be minimized
5. Routes should be calculated quickly by the program
6. Service is provided through software as a service

Implementation

- The file for Sprint 1 code is “bestRoute.java”.
- We decided to first solve the problem of finding minimum distance independent from the system environment containing the different classes (Delivery Company, Truck Management, etc.) so as

to give a simplified algorithm which illustrates the basic method for calculating distances between points

- This is because the actual problem (the Vehicle Routing Problem) is much broader and complex and doesn't yet have a fully functional solution
    - This brute force algorithm helps us understand the foundational calculations needed to connect destination points to each other
- The code is designed to simulate a particular instance of random destination points and one pickup point. The locations are plotted on a 2D matrix.
- The pickup point P is at the "origin" or [0,0] (row 0, column 0) and the destination points are represented by letters which are situated at different row and column coordinates. The other locations are populated with '1's and which represent the distance from each coordinate to another. This matrix can be edited with different distances and the points can be rearranged as well depending on a desired scenario.
- To find the minimum distance needed to travel to each destination point, we chose to implement 2 functions: one which creates a distance matrix and one which uses the distance matrix to calculate the total minimum distance.
    - The distance matrix shows all the distance from one point to another. The first row shows all the distances from the source pickup point to all the destination points. The second row shows all the distances between the first destination point and all the other destination points. The third row shows all the distances between the second destination point and all the other destination points, and so on.
    - The minimum route function traverses through the distance matrix to calculate the minimum distance needed to go to all the points (starting from the pickup point). The minimum of each of the distances of each row is found and added to the sum total.
    - Then, the minimum cost (minimum distance needed to travel) is outputted.

Testing

- The output file is "bestRouteOutput.java".
- The testing used for this algorithm applies for one case of pre-defined lengths and a plotted matrix of location points. Multiple cases are not taken into account here.
- Testing shows that a distance matrix showing distances between all points is successfully created and that the minimum distance to travel between all points is found.

Evaluation

- While the algorithm is good for showing a bare outline of how a complex system spanning a large area of locations can be reduced to a maintainable matrix that is easy to visualize and traverse.
- However, this program doesn't fulfill the functional requirements of providing any route that the trucks can follow. Moreover, the program assumes that there is only one truck which isn't realistic.
- Moreover, the program isn't suited to be implemented by the delivery company since it doesn't take any input addresses.

- The second sprint should deal with these issues by making it accessible to a delivery company. It should utilize a database and use realistic addresses and distances in the calculations.

Sprint 2

Requirements Engineering

- The requirements specified in Sprint 1 apply. New requirements are specified for Sprint 2.

Functional Requirements

1: The time taken to travel the total distance should be specified.
2: Algorithm should assume variable number of pickup points (not only 1 as in Sprint 1)

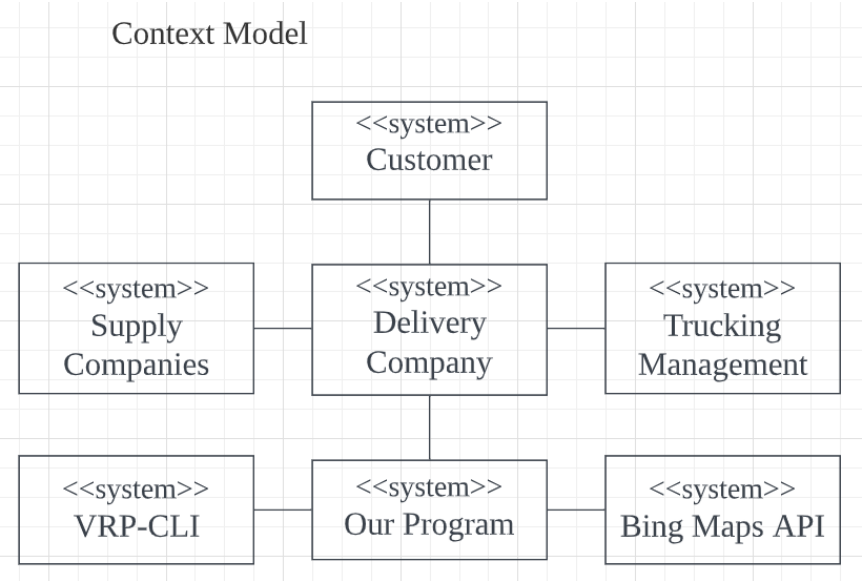
Non-functional Requirements

Safety	1: Truckers have maximum amount of time they can travel daily so total time based on route should take that into account so as to not upset scheduled deliveries
	2: Trucks cannot hold over the standard weight - cannot assume that a truck can store all goods required to visit all destination points so either: ● Route should include multiple pickups from same pickup point (back & forth delivery) ● Deploy more trucks to store for all deliveries for simultaneous dropoff
Speed	3: Routes should be calculated well in advance to accommodate new orders which might change scheduled pickups & drop offs

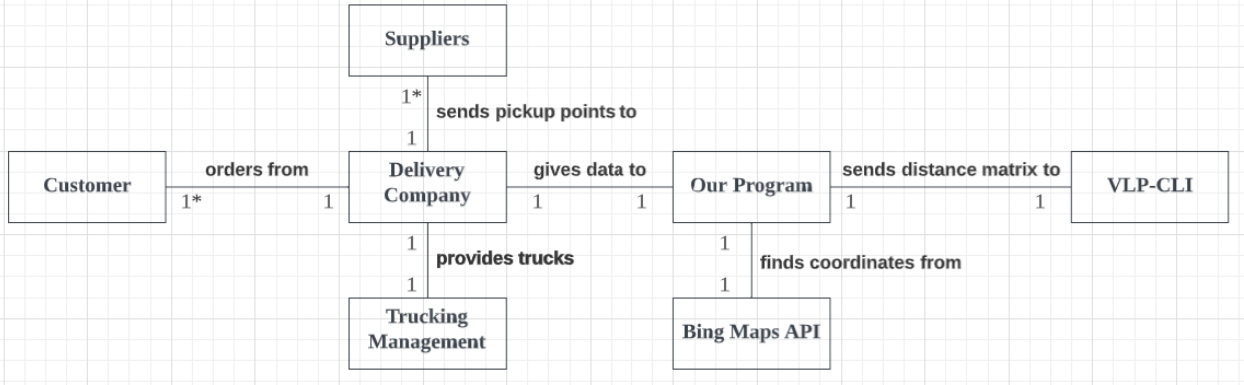
System Modeling

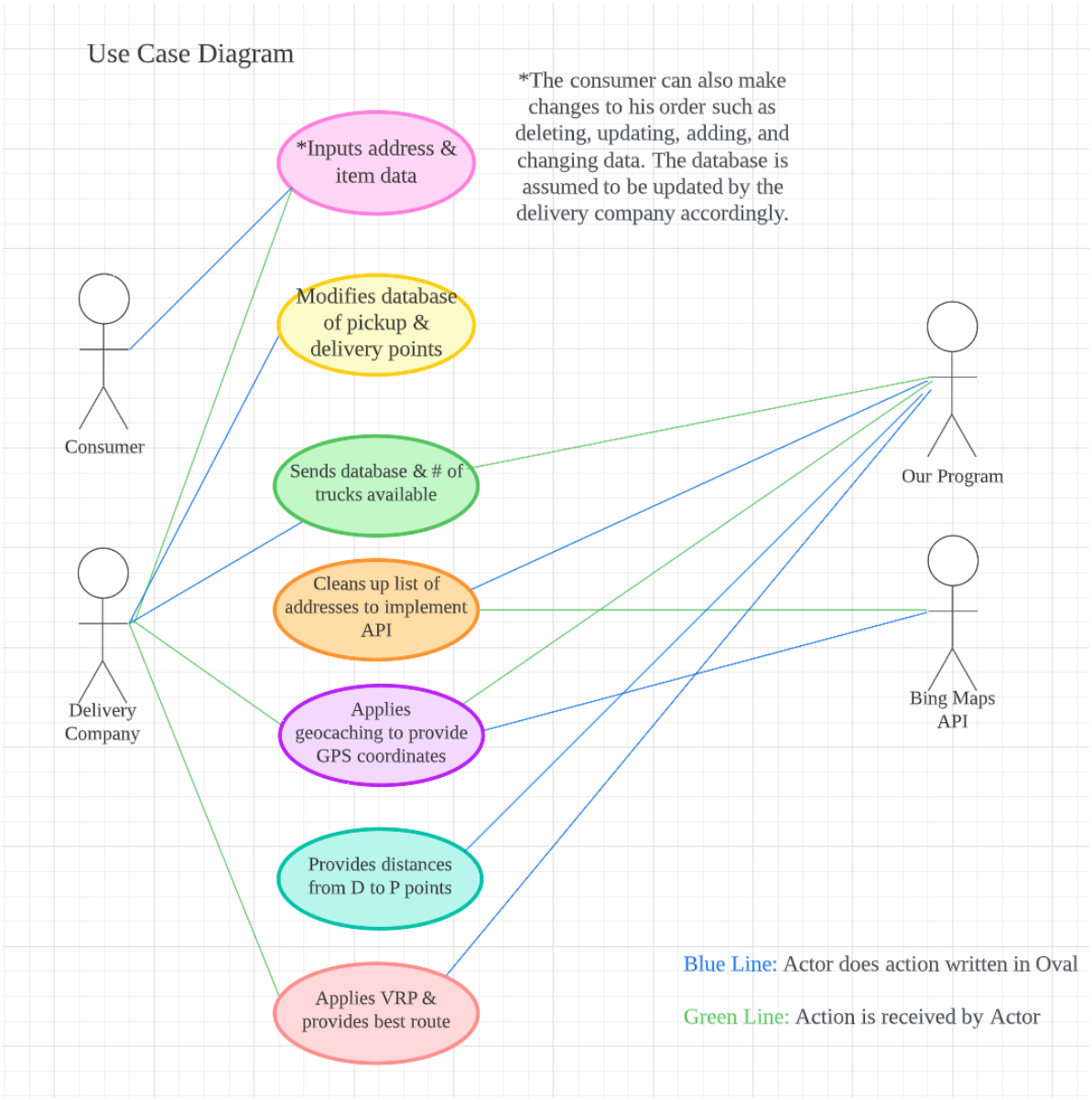
- The system contains more classes in the environment as our program will use information from existing directories which supply address coordinates and algorithms which provide broader solutions to the route planning problem.

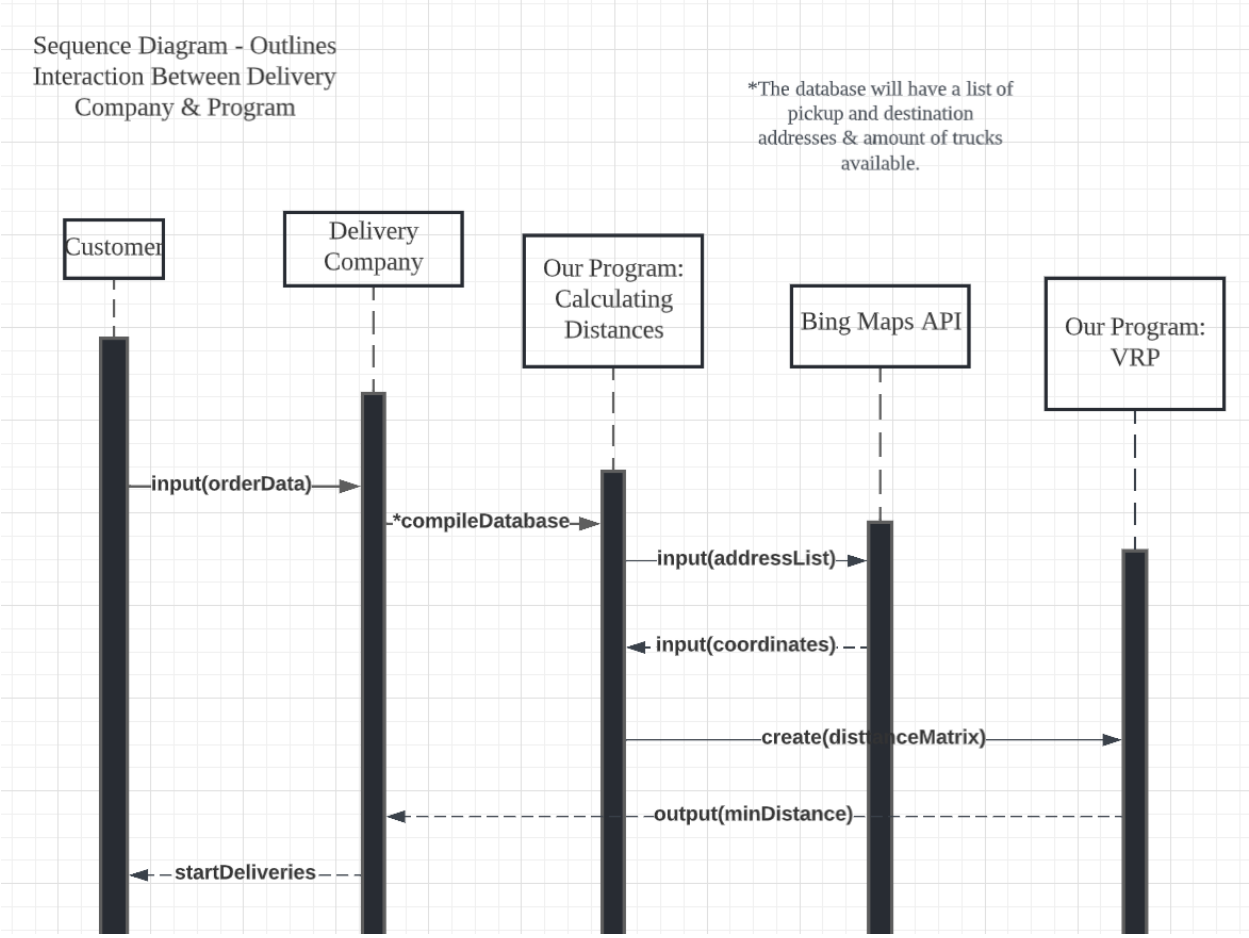
Context Model

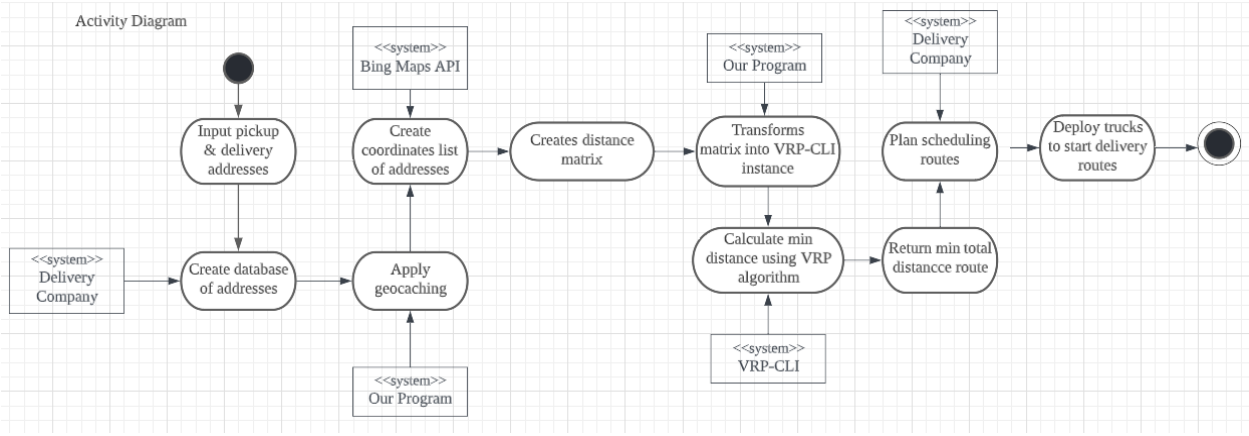


Class Diagram




Use Case Diagram
*The consumer can also make changes to his order such as deleting, updating, adding, and changing data. The database is assumed to be updated by the delivery company accordingly.
*Inputs address & item data
Modifies database of pickup & delivery points
Sends database & # of trucks available
Cleans up list of addresses to implement API
Applies geocaching to provide GPS coordinates
Provides distances from D to P points
Applies VRP & provides best route
Consumer
Delivery Company
Our Program
Bing Maps API
Blue Line: Actor does action written in Oval
Green Line: Action is received by Actor



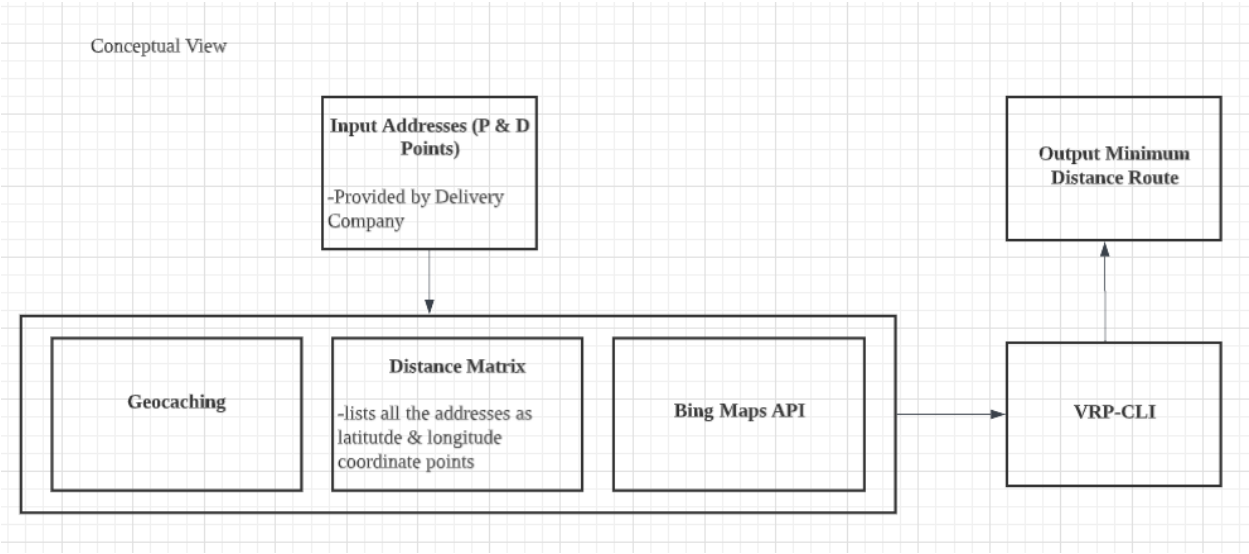


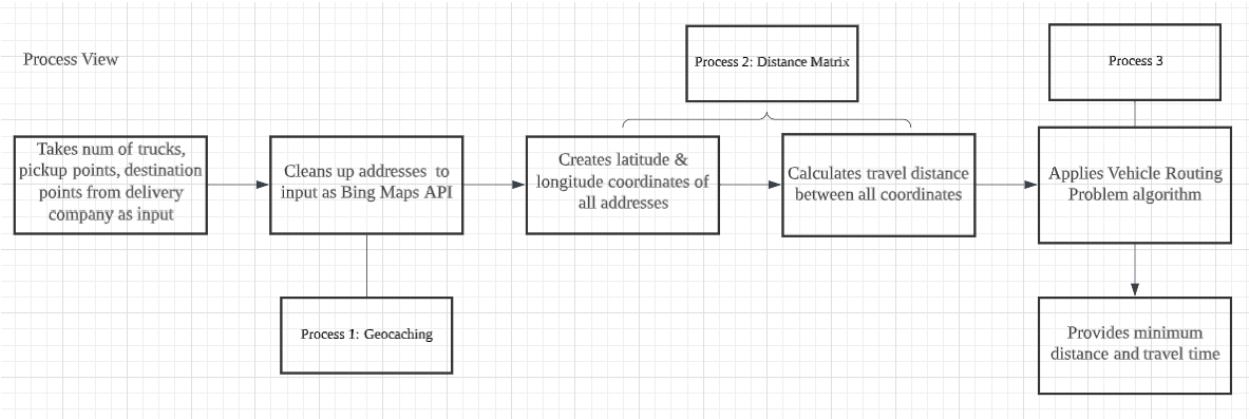
Architectural Design

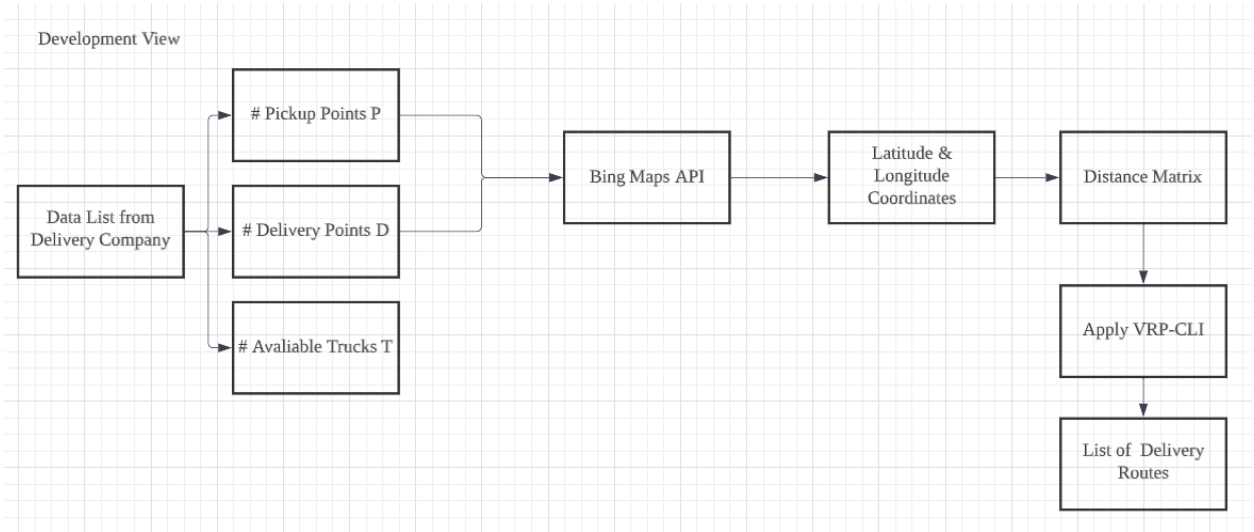
- The architectural pattern followed is pipe and filter architecture
    - The original data that is given by the delivery company - number of trucks, list of pickup point and delivery point addresses - is transformed through series of processes which

implement the Bing Maps API and the Vehicle Routing Problem (VRP) algorithm to create a list of coordinates and "flows" through eventually to create a matrix of distances.

- As this is a sequential system which processes the inputs in multiple ways to generate a final output based on the data transformation, our system most closely follows the pipe and filter pattern.







Specification

In addition to Sprint 1:

User Requirements

1. The total time taken for all trucks to travel their routes should be an output

System Requirements

1. Trucks have maximum amount of time they can travel in a day and that should be taken into account when calculating the total time
2. Trucks have a maximum weight that they can hold so they may have to take multiple trips from the same pickup point
3. Routes should be calculated well in advance so that there is time to account for last minute orders
4. Algorithm should work as intended for multiple and variable number of pickup points

Implementation

- NP - Complete problem known as MDVRP (Multi-Depot Vehicle Routing Problem) which finds the minimum distance needed to travel to all the destination points
    - It's variant of the more common problem VRP (Vehicle Routing Problem)
    - This isn't fully functional because according to MDVRP solution, the number of trucks available is not limited
    - This problem has still not yet been solved fully
- In Sprint 1, we used a brute force algorithm which makes up an independent point matrix and manually calculates the minimum distance by traversing through the entire matrix and calculating individual distances
    - However, this isn't really untenable, even for small cases, such as when P ~ 20, because destination points are variable and distances are variable as well and the routes connecting each location cannot be limited to only horizontal & vertical movement
- The conclusion is that software reuse is necessary to cover a problem of this complexity
- Below are our brainstormed ideas on how to implement:
    - Map Data:

        - OSRM
        - **Bing Maps** (Free tier allows 2500 = 50x50, up to 25000/month)
        - Openlayers
        - (Not GMaps, costs money)
    - VRP solvers:
        - ~~Concorde~~ (TSP, not same)
        - ~~Optaplanner~~ (Way too much coding required)
        - ~~VRP spreadsheet solver~~ (Offline…)
        - VRPy
    - Test Database:
        - Not many MDVRP instances
        - CVRPLib
        - VRP-REP
    - Language: Python - easier than Java for multiple processes
        - Send GET request to bing maps api for geocaching
        - Send GET request to bing maps api for distance matrix
        - Format pickup, destination, distance matrix for vrp-cli
        - Execute vrp-cli solver

Dependability

- Availability
    - The project has to be made available at all times since delivery systems work 24/7 and changes to deliveries and additions of pickup and delivery points can be made at any given time.
    - Our code allows for constantly changing input, meaning that the databases provided by the delivery company can be modified and the program will still apply since geocaching can be applied to all addresses in the database.
- Reliability
    - The system will be able to consistently find the routes of the trucks that minimize distance every time it is run.
- Safety
    - The whole delivery system will have to take into account the safety standards which have been set to protect the needs of truckers. The program itself has to provide correct routes on time to allow truckers ample time to make all the stops so they don't have to exceed their normal daily travel time. Also, the program should, in effect, allow for situations where trucks do not have to carry weight that exceeds regulations. Therefore, the visits to pickup stops should not be excessive or should come after a few delivery points have been visited. ALthough this is beyond the scope of the program which only focuses on finding the shortest route, in a real-life scenario, all these safety factors should be considered.

Project Management - Managing People

- The optimal delivery system contains classes of people from independent organizations:
    - Supply company
    - Trucking management
    - Customers
    - Our Program (which combines Bing Maps API & VRP-CLI directories)
    - Delivery company
- Time is a shared concern for all of these classes and so it is really important to manage schedules of different people so that everyone's expectations are matched as the whole operation has to be coordinated:
    - The supply company has to know how much time they have to supply certain products and bring them to their main storage pickup spaces
    - The trucking company needs to know how many truckers should be made available, how many places they would have to visit, how many days they will be on the road, and by what day should everything be delivered.
    - The customers give input on address and shipment information. They also have influence over the scheduling of orders since they can decide to fast track orders, change their orders, and cancel orders.
    - The delivery company will communicate with all these classes and compile the database of addresses based on everyone's input.
    - The main criteria on whether the delivery system is successful is whether the orders can be delivered by the right times to the customers.
        - Therefore, the routes that each truck takes should not only be optimized but also keep in mind the times/days by which the shipment must be delivered to each destination point. In this project, the only focus was minimizing the distances and the destination based on the desired timing of the orders was not taken into consideration. This would have to be considered if coordination is expected so that all orders are met. The routes should be designed to go to the destinations which expect their orders before other destinations first. It is also not expected that all the existing destination points be visited over one period of time.
- Along with maintaining open communication, we should look into satisfying everyone's needs. For instance, the comfort of the truck drivers must be considered:
    - The objective of this project is to calculate routes that are optimally the shortest. However, if we were to consider routes that offer for more suitable driving, then (as mentioned in the non-functional requirements), the routes would be redesigned to consider locations where there are enough petrol stations and hotels. Also, we would consider the landscape and type of roads. For example, it is known that taking right turns is more dangerous for trucks so we would consider routes which reduce the number of right turns each truck has to take.
        - Modifying the route planning algorithm to consider the needs of the truckers would probably be more motivating and increase their input capability.
        - Also, in a real-life situation, truck drivers might have rights on how much they are made to drive at a time.

    - Another factor to consider in relation to the trucks is the amount of weight they carry. For this project, it is assumed that all the trucks have unlimited capacity to take on all the loads at once and ship them to the corresponding destinations.
        - This is why the routes that are designed assume that the trucks can store unlimited products from a specific pickup point or even multiple pickup points at a time and visit delivery points.
        - Realistically, however, the trucks probably cannot be overloaded above the standard limit.
        - Moreover, all the pickup points cannot be treated as if they carry all the same products. They have to be differentiated according to the type of products they supply so trucks would have to discriminate between each pickup point.

Advanced SWE Topics - Software Reuse

- The problem of calculating the distance between locations has been tackled before such as OSRM, Bing Maps, Google Maps. We decided to use Bing Maps due to allowing for free requests under a given threshold (P+D <=50) and for their convenient RESTful API, with access to geocaching and distance matrices.
- TSP (Traveling Salesman Problem) has been extensively well studied, and there are many state-of-the-art algorithms available. However, our specific problem (Multi-Depot Vehicle Routing Problem) is not as extensively well studied. We found many implementations (Concorde, Optaplanner, VRP Spreadsheet Solver, VRPy, A-MDVRP) that were unsuitable for our application. Furthermore, there weren't many MDVRP 'instances' (aka test cases), meaning we needed to generate our own.
- We decided to use vrp-cli (https://github.com/reinterpretcat/vrp) as it is under the Apache License, and as it can handle our specific variant (MDVRP) and larger instances within our given constraints (P+D <=50)
- Furthermore, vrp-cli is extendable to other problem variants (the capacitated variant MDCVRP, MDVRP with time windows, etc) that can be used to handle some of the nonfunctional requirements.

Testing

- The files "bingmapsapi.py" and "problemJSON.py" contain the algorithm for using the Bing Maps API to create coordinates for each address given in the "sprint2test.txt" file, creating a distance matrix, and instantiating it as an input which can then be used to apply the vrp-cli solution algorithm.
- The sprint2test file contains 5 different test cases with different numbers of trucks, pickup points, and delivery points allotted for each test case. For the sake of simplicity, we only used addresses located in New Jersey.
- The "main.py" takes in the input test cases files so VRP can be applied. The "sprint2testsolns" folder contains the solution files for each test case, detailing the routes.

Evaluation

- The solutions are included in the files solutions{i}.json. At the top of each file is the statistics of the tour found– distance is the distance in meters, duration is the time in seconds. Below the statistics is the tour found, containing the locations of the destinations points in the order traveled and the amount of time/distance traveled to that point.
- The software reuse allowed us to produce more realistic outputs which fulfill the functional requirements and provide more statistical data about each point in the route. Although it wasn't expressly stated in the project description, providing information such as the time it takes and distance from each point to another is useful for the truckers.
- Sprint 2 involves higher complexity as compared to Sprint 1 since it utilizes advanced software engineering concepts such as software reuse which allows for the Vehicle Routing Problem to be implemented. Using the Bing Maps API to do geocaching and get accurate coordinate locations for all the addresses also makes the code more reliable and usable for a real delivery company.
- Overall, Sprint 2 shows improvement from Sprint 1 in that more requirements are fulfilled and the program becomes more dependable and provides more resources to the delivery company. The algorithm shows a higher level of complexity and knowledge of the problem at hand. It also fits more into the whole system environment since it isn't independently conceived like Sprint 1 code is. It uses services which are available to the other classes in the system and also takes in input from the company requesting the service.