**Project Idea:**

Our project idea is to solve the **Travelling Salesman problem,** this problem appears in Companies that have transportation system and want to minimize the cost of delivering the orders, and this is done by providing the correct order in which distributers should deliver orders.

**Problem Significance:**

**Problem Definition**: The traveling salesman problem is a common NP hard problem and consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one (e.g. the hometown) and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip.

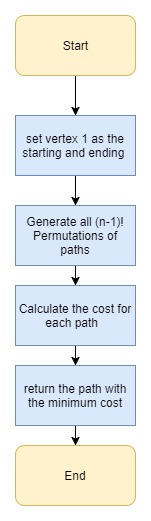
**The Problem variants:**

1. **The TSP with precedencies or time windows**: A Travelling Salesman Problem with Allocation, Time Window and Precedence Constraints (TSP-ATWPC) is considered. The TSP-ATWPC occurs as a sub problem of optimally sequencing a given set of port visits in a real bulk ship scheduling problem, which is a combined multi-ship pickup and delivery problem with time windows and multi-allocation problem. Each ship in the fleet is equipped with a flexible cargo hold that can be partitioned into several smaller holds in a given number of ways, thus allowing multiple products to be carried simultaneously by the same ship.
2. **Symmetric (STSP):** The TSP is symmetric if, for every pair of cities i and j, the distance from i to j is the same as the one from j to i.
3. **Asymmetric (ATSP):** The TSP is asymmetric if, the distance for going from a point to another may be different of the returning distance.
4. **The price collecting TSP:** In the TSP, the salesman has to visit a set of cities while minimizing the length of the overall tour. In the PCTSP, each city has a given weight and penalty, and the goal is to collect a given quota of the weights of the cities while minimizing the length of the tour plus the penalties of the cities not in the tour.
5. **The online TSP:** The number of requests n is not known to the online server. Requests are revealed to the online server at their release dates.
6. **Bus, truck, vehicle routing:** Asks "What is the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers?". It generalizes the well-known travelling salesman problem (TSP).
7. **Edge/arc & node routing with capacities:** The CARP aims to find a set of vehicle trips with minimum cost, such that each trip starts and ends at a depot node v0 2 V, each required edge.is serviced by a single trip, and the total demand for any vehicle does not exceed a capacity Q.
8. **The symmetric and asymmetric m-TSP:** (m-TSP) is a generalization of the TSP in which more than one salesman is allowed. Given a set of cities, one depot, the objective of the mmTSP is to determine a set of routes for mm salesmen so as to minimize the total cost of the mmroutes. The mmTSP is a relaxation of the vehicle routing problem (VRP); if the vehicle capacity in the VRP is a sufficiently large value so as not to restrict the vehicle capacity, then the problem is the same as the mmTSP.

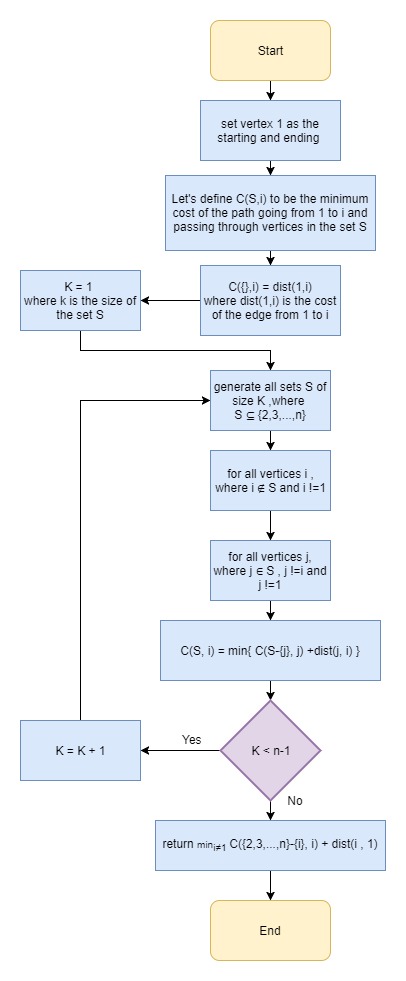
**Motivation:** Assume that you are the driver of a delivery vehicle with a certain set of stops that need to be made each day. How would you determine the order in which to make the stops? If you are interested solely in distance, you could create a graph of the transportation network and weight each edge as the distance of the roadway it represents, allowing a solution to the Traveling Salesman Problem (TSP) to determine the shortest route.

**Problem Solution:** Travel Salesman Problem has more than one approach to solve it e.g.: Exact Approach, Greedy Approach, Heuristic Approach and Genetic Algorithm Approach.

1. **Exact Approach:**
   * **Brute Force :**

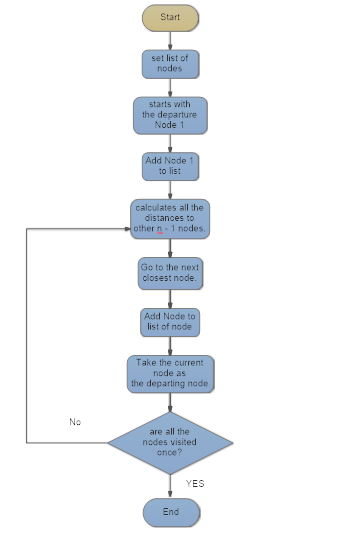


* + **Dynamic Programming (Held Karp Algorithm):–**



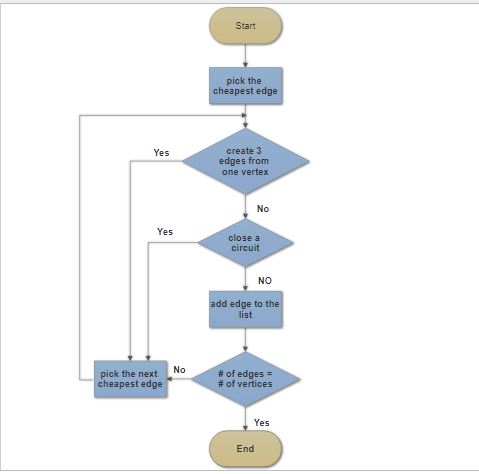
1. **Greedy Approach:**

* **Nearest Neighbor Algorithm :**

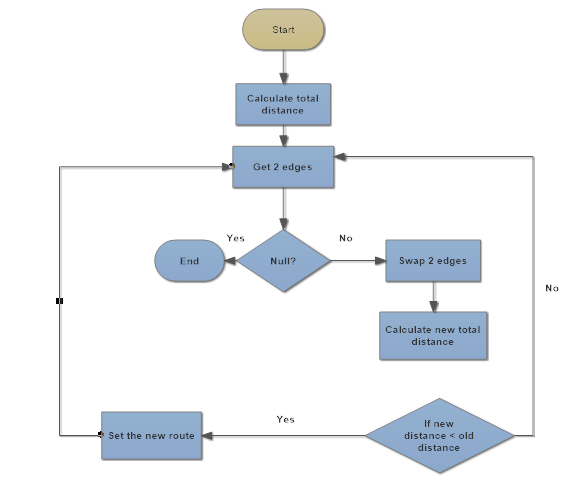
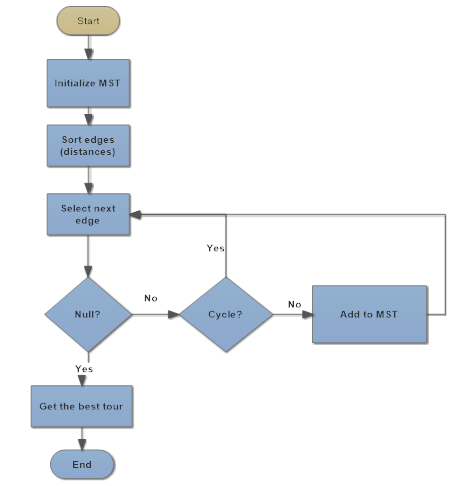


1. **Heuristic Approach:**

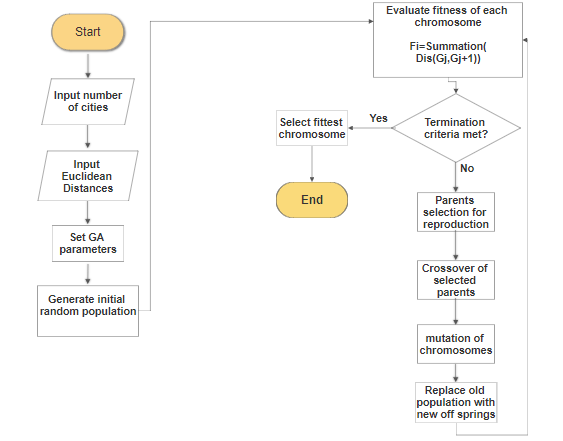
* **Cheapest Link :**



* **Depth First Tree Tour :**

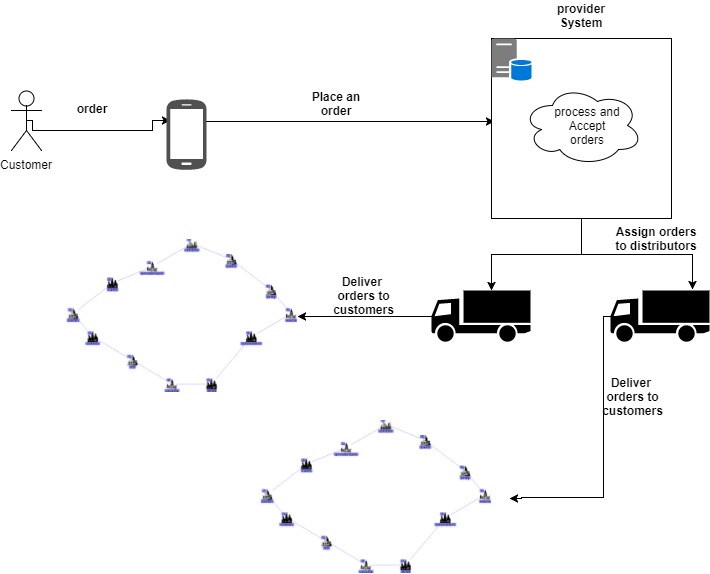


1. **Genetic Algorithm Approach:**

****

**System Analysis and Design:**

**System Architecture**

****

**Stakeholders:**

1. **Customers:** Are External-Operational stakeholders, who make the orders and this stakeholders will interact with the system via Android application.
2. **Provider:** is Internal-Operational stakeholder, who receives the orders and distributes them to the distributors according to the address of the customers, this stakeholder will interact with the system via web application.
3. **Distributers:** is Internal-Operational stakeholder, receives the orders that they should deliver and ask the system to get the optimal route to follow, this stakeholders will interact with the system via Android application.

# **Functional Requirements:**

## **1-Customer:**

* Order any Product including quantity and location.
* Cancel order before distributor takes it.
* Update orders in cart.

## **2-Provider:**

* Reject or accept an order.
* Assign carts to distributor.
* Add distributor.
* Add admins.

## **3-Distributor:**

* View orders.
* Get/request the Order of delivering the orders based on the distance (The optimal path to go).
* View the path to the current fulfilled order.
* Confirm order delivery.

# **Non-Functional Requirements:**

## **Security:**

* The System has a form of protection by applying authorization, so any unauthorized access to the system is denied
* Avoid SQL injection

## **Performance:**

* Login must be completed in less than 3 seconds
* Peak load 200 user every hour
* Provider assign orders to distributor in less than 10 s

## **Reliability:**

* The system has to be 100% reliable

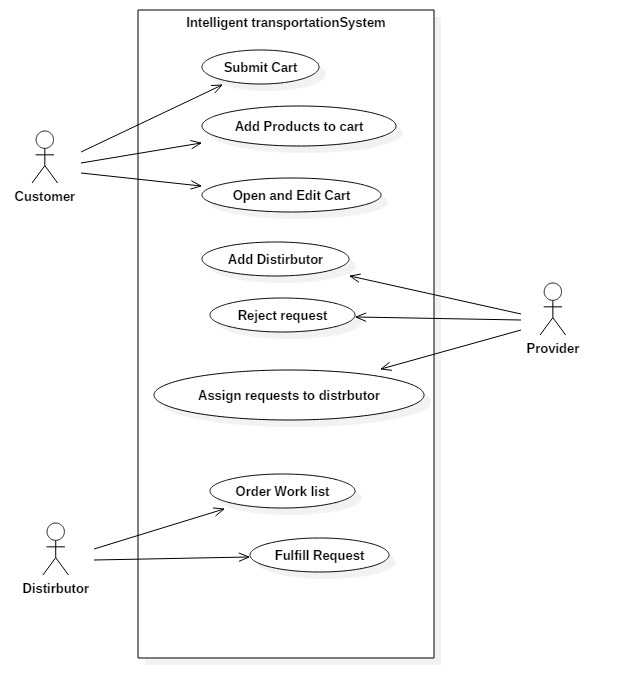
## **Availability:**

* The system will be available 24/7

## **Usability:**

* The customer can easily order any products with any quantities

**Use Case Diagram:**



**Use Case Tables:**

1. **Customer:**

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 1\_Customer | |
| Use Case Name: | Add Products to Cart | |
| Actors: | Customer | |
| Pre-conditions: | 1- Logged in as Customer | |
| Post-conditions: | 1-A list of items is selected | |
| Flow of events: | **User Action** | **System Action** |
| 1-User clicks on choose items to order |  |
|  | 2-System displays the list of available items to be order |
| 3- Customer selects items to order |  |
| 4- The customer then clicks finish to save the order and add it to the cart |  |
|  | 5- System receive the list and save it |
| Exceptions: | **User Action** | **System Action** |
| 1-the list of items is empty and customer press finish |  |
|  |  | 2-System notifies him that there are no items selected. |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 2\_Customer | |
| Use Case Name: | Submit Cart | |
| Actors: | Customer | |
| Pre-conditions: | 1. Logged in as Customer 2. A list of items is selected | |
| Post-conditions: | 1. A list of orders in the cart is send to the System (provider) | |
| Flow of events: | **User Action** | **System Action** |
| 1-Customer press on submit cart button |  |
|  | 2-Sytem shows total price and invoice of orders |
| 3-Customer Presses Confirm request to send request |  |
| 4-or the customer presses cancel to cancel the request |  |
|  | 5- System receive the request and send it to the provider |

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 3\_Customer | |
| Use Case Name: | Open and Edit Cart | |
| Actors: | Customer | |
| Pre-conditions: | 1. Logged in as Customer 2. A list of items is selected | |
| Post-conditions: | 1. List of Products items in cart is updated and saved | |
| Flow of events: | **User Action** | **System Action** |
| 1-user press on open cart button |  |
|  | 2-System shows the products already added in the cart, info and their quantities |
| 3- Customer checks the items and the invoice |  |
| 4- then decides wither to edit the list by changing a quantity or removing products from the cart |  |
| 4-The customer then click on save to save the edited list of products in the cart |  |
|  | 5- the System then updates the list in the cart and save it in the DB |

1. **Distributer:**

|  |  |  |
| --- | --- | --- |
| Use Case ID: | 1\_Dist | |
| Use Case Name: | Order Work List | |
| Actors: | Distributor | |
| Pre-conditions: | 1- Logged in as distributor. | |
| Post-conditions: | 1-Starts orders fulfillment. | |
| Flow of events: | **User Action** | **System Action** |
|  | 1-System displays the work list (Requests) to be fulfilled. |
| 2- Distributor asks to order the work list according to his current location. |  |
|  | 3-System checks number of orders in the work list and decide which algorithm to run. |
|  | 4- System Displays the new ordered work list. |
| 5- Distributor views the ordered work list and starts fulfillment. |  |
| Exceptions: | **User Action** | **System Action** |
| 1-Distributor asks to order the work list and no orders are assigned to him. |  |
|  |  | 2-System notifies him that there are no orders assigned to him. |

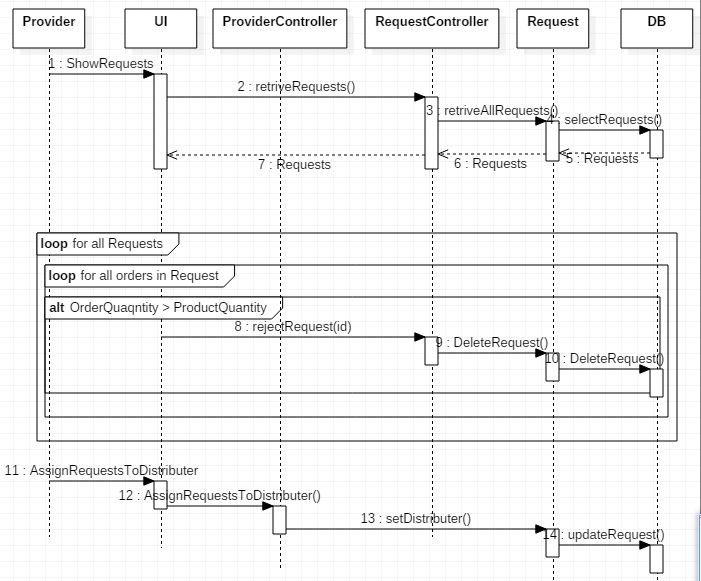
|  |  |  |
| --- | --- | --- |
| Use Case ID: | 2\_Dist | |
| Use Case Name: | Fulfills Request | |
| Actors: | Distributor | |
| Pre-conditions: | 1- Logged in as distributor. | |
| Post-conditions: | 1-Finishing request fulfillment | |
| Flow of events: | **User Action** | **System Action** |
| 1-Distributor asks the directions for the coming request. |  |
|  | 2-System displays the directions. |
| 3- Distributor reaches to the desired requester location and fulfills his request. |  |
| 3- Distributor checks on the request as fulfilled request. |  |
|  | 4- System saves the time in which the request was fulfilled. |
| 5- Distributor asks for the next request to be fulfilled. |  |

**3-Provider:**

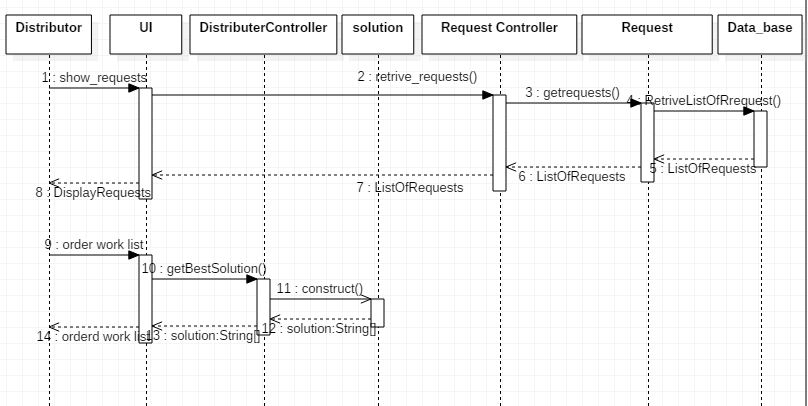
|  |  |  |
| --- | --- | --- |
| Use Case ID: | 1\_Provider | |
| Use Case Name: | Assign Requests List | |
| Actors: | Provider | |
| Pre-conditions: | 1- Logged in as Admin. | |
| Post-conditions: | 1-Assign requests to distributers. | |
| Flow of events: | **User Action** | **System Action** |
|  | 1-System retrieves all requests that arrive from customers. |
| 2-Admin rejects all requests that are greater than the available capacity. |  |
|  | 3-System delete all request that are greater than the available capacity |
|  | 4-System shows the new list of requests. |
| 5- Assign requests to distributers. |  |
|  | 6-System group requests by their location and assign them to the appropriate distributors. |
|  |  | 7-System send request to distributer |

**Sequence Diagram:**

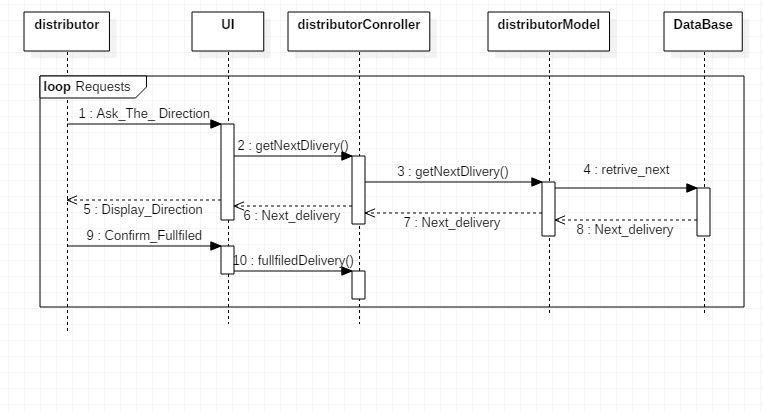
1. **Assign request to distributers:**



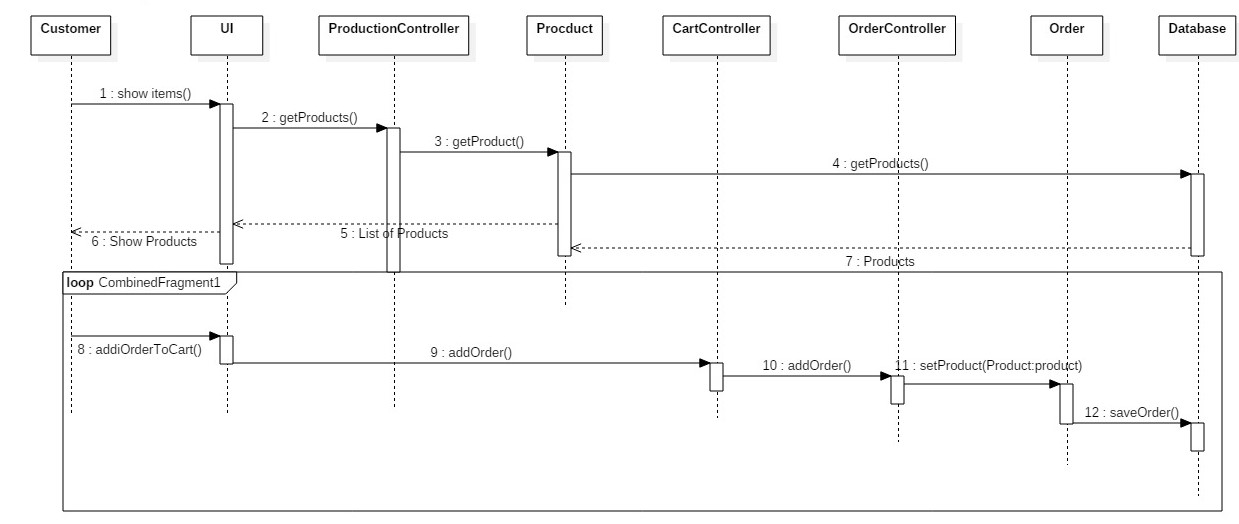
1. **Order Work List:**



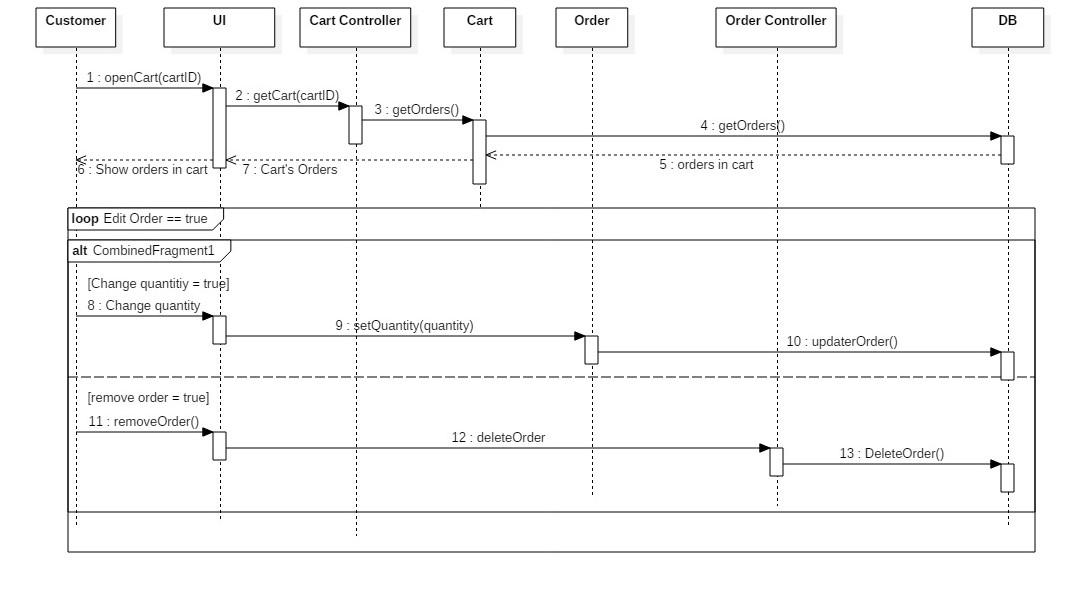
1. **Fulfill Order:**



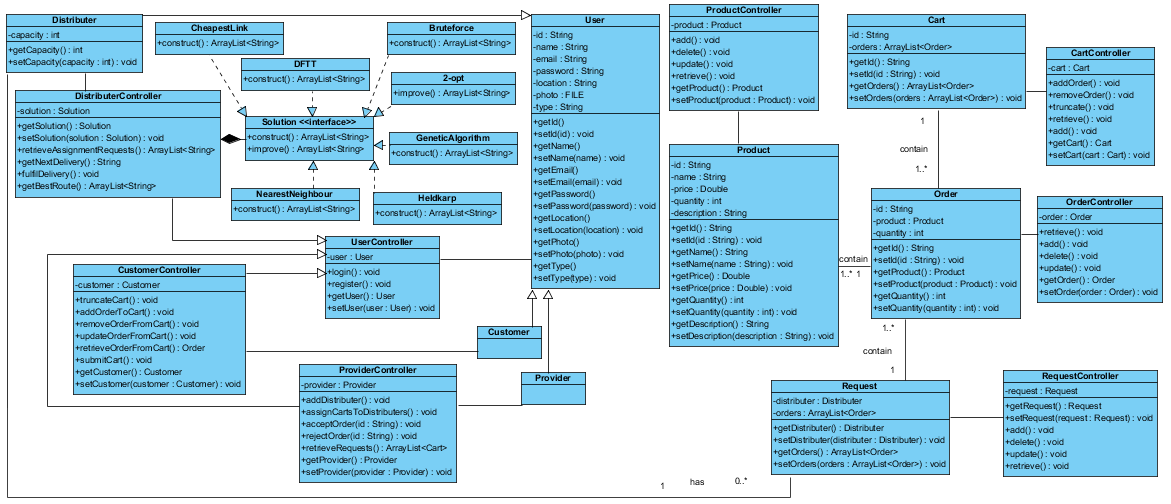
1. **Add products to cart:**



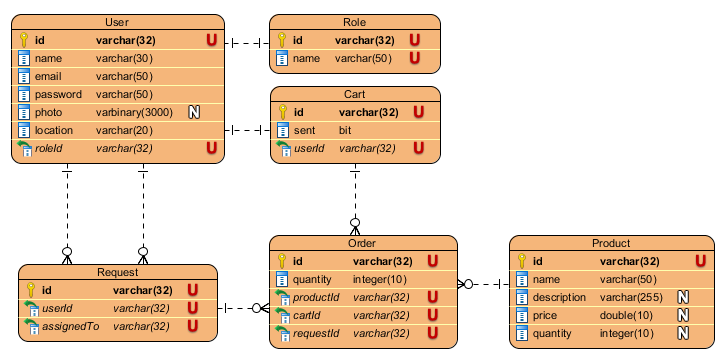
1. **Open and edit cart:**



**Class Diagram:**

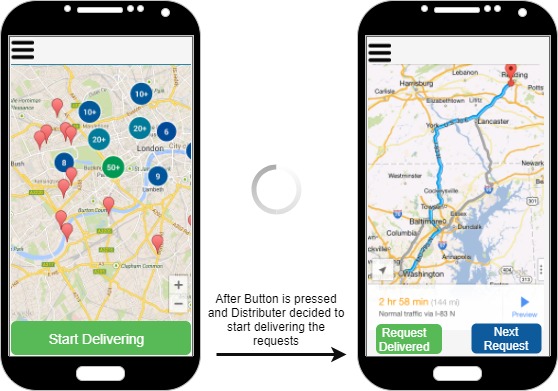


**ERD (Entity Relationship Diagram):**

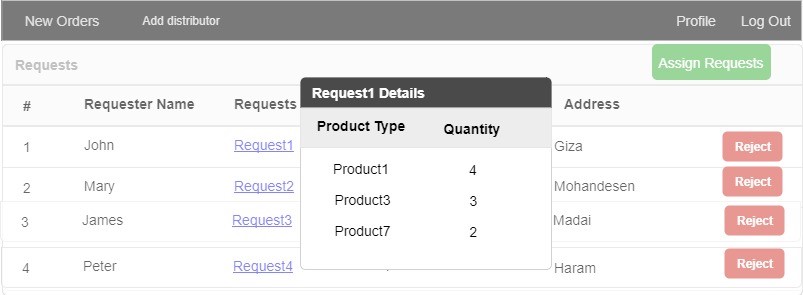
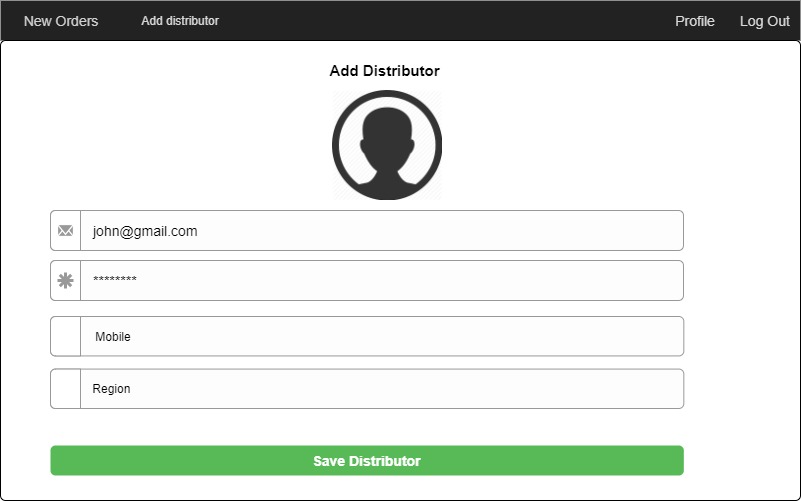


**Prototype:**

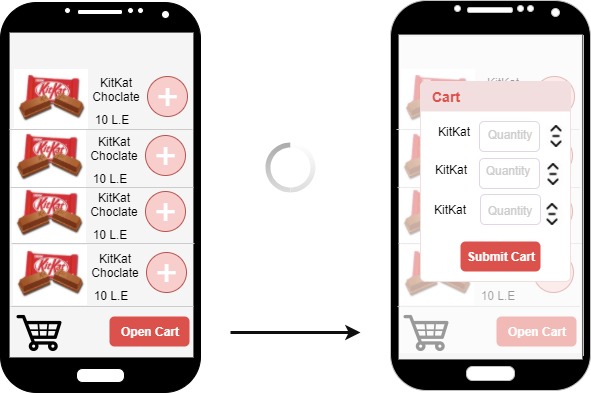
1. **Distributer:**



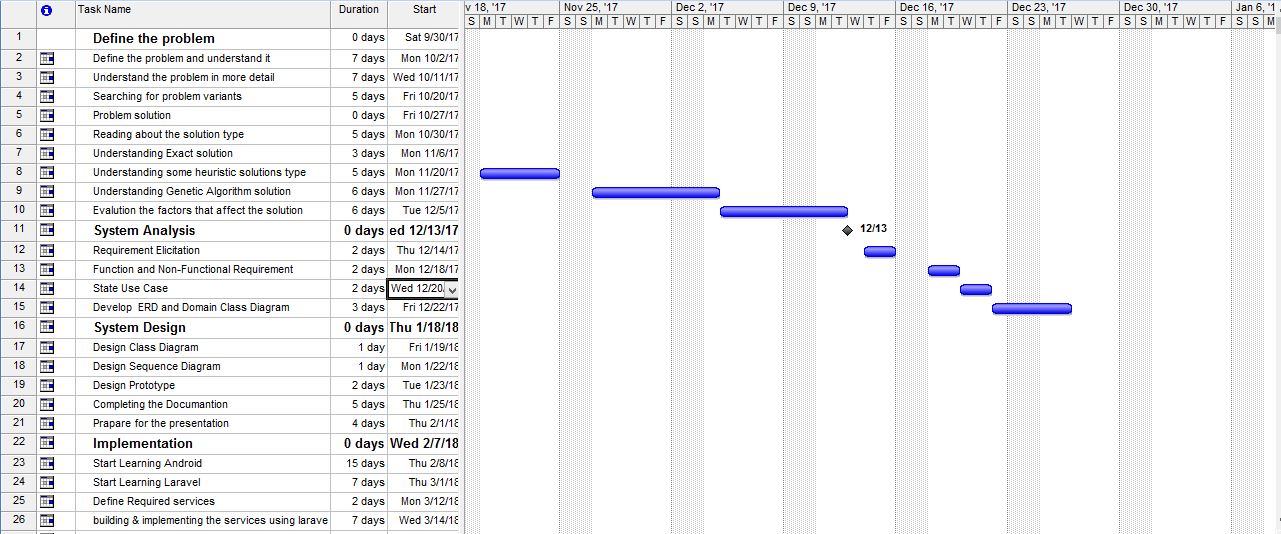
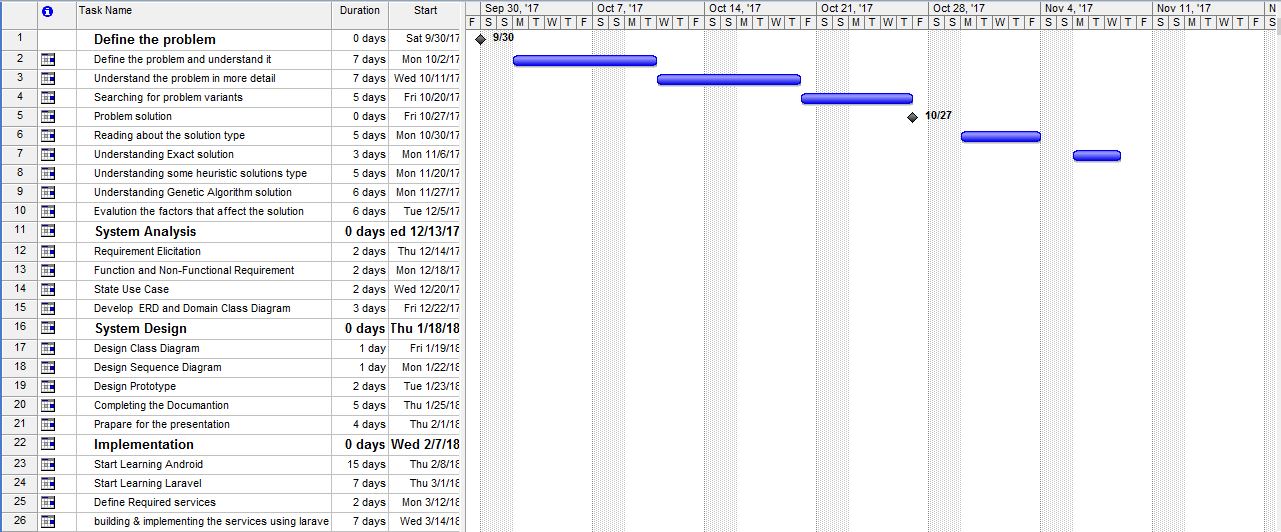
1. **Provider:**

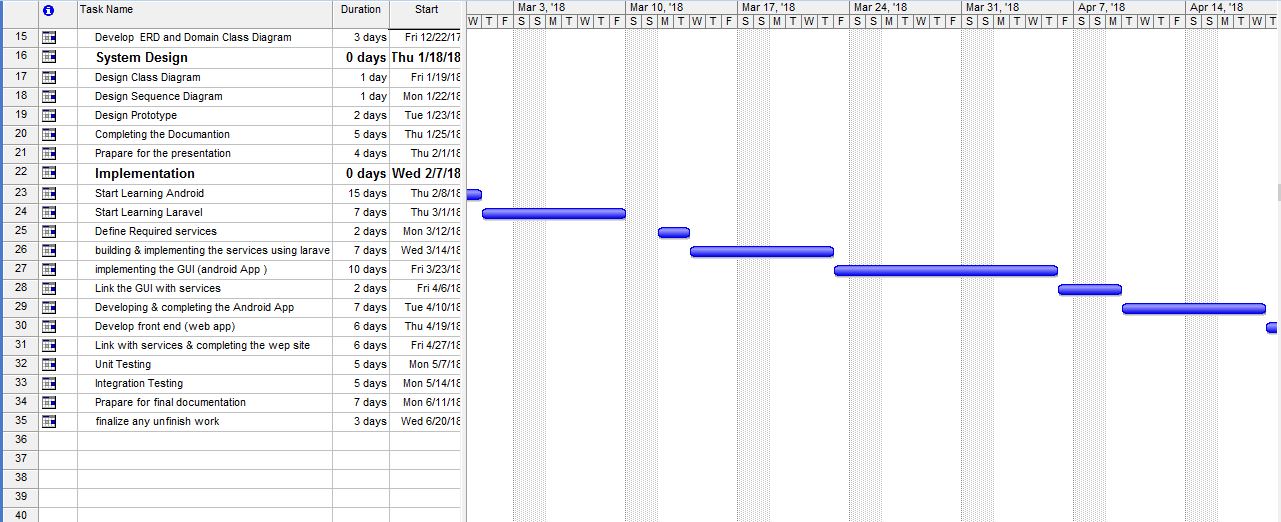


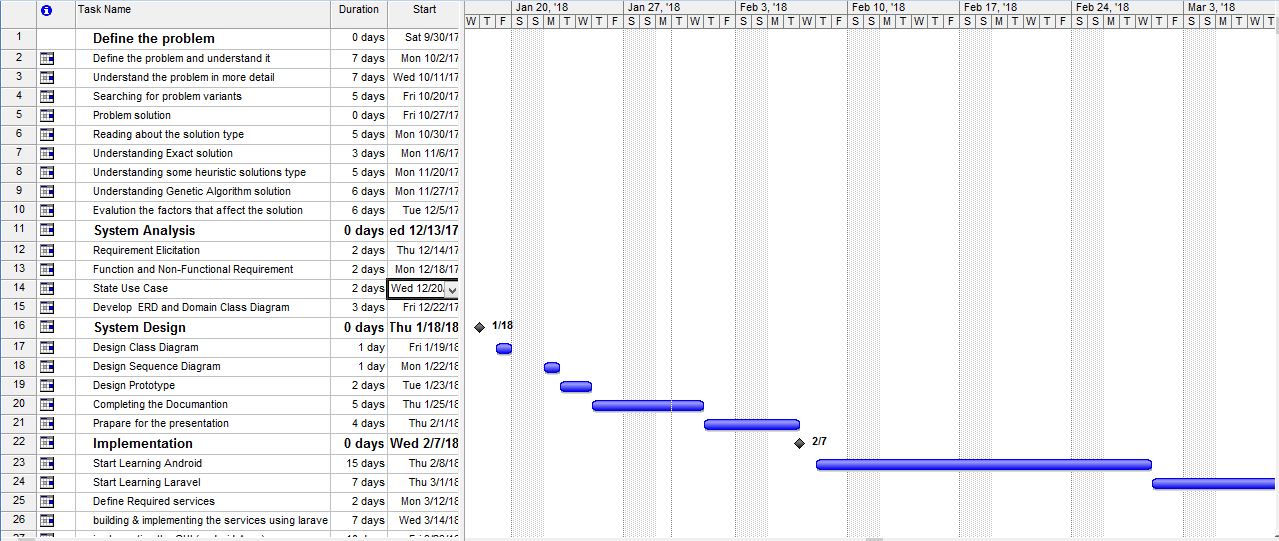
1. **Customer:**

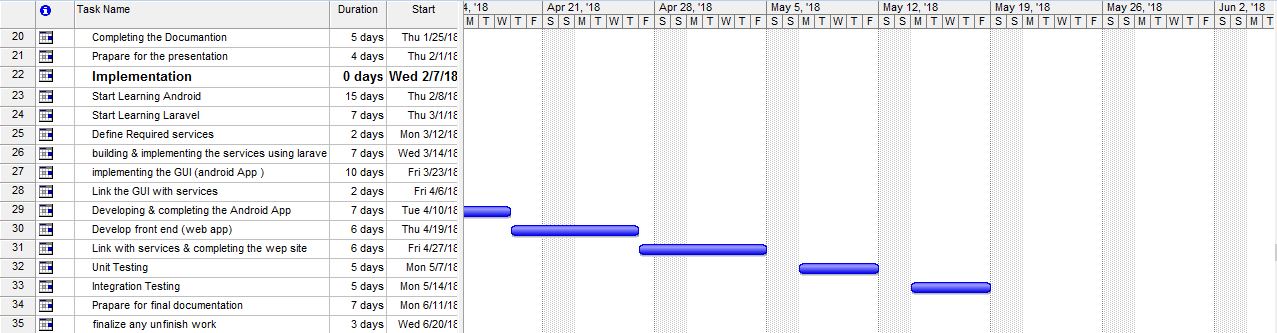


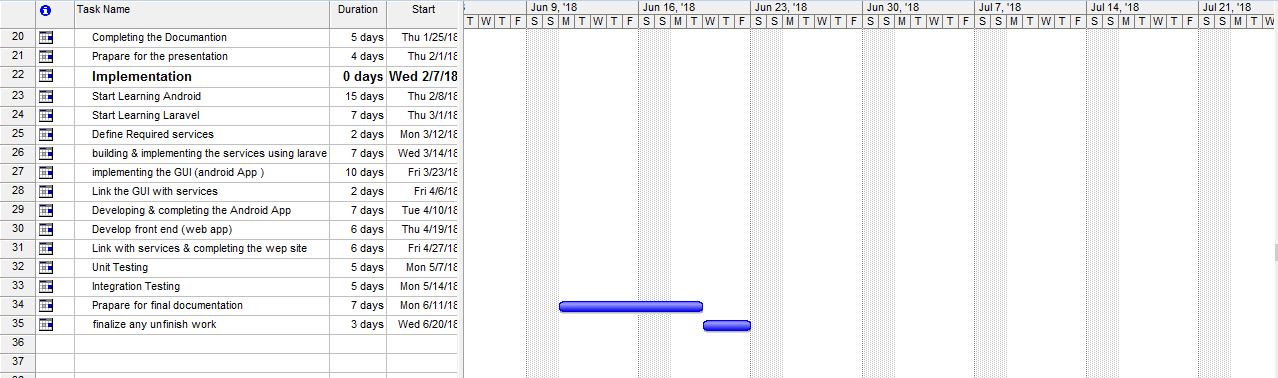
**Gantt chart:**











# **Conclusion**:

We can see now that this project will help a lot by saving money, time and efforts for the companies that have a transportation system especially the ones with large transportation systems., as the application will choose the path with the most minimum distance approximately instead of travelling longer distances, and this is done by using one of the previously mentioned.

In this project we considered one of the factors that affect the path to choose which is the distance between different requests which is constant factor, there are other factors that taking them into consideration will make better results like traveling time, traffic, the speed limit and the types of the streets chosen whether they are highways, freeways or small streets, taking all these factors well help in making the idea more effective, but it will make the project more complex ,so we can consider them later on as a better upgrade to the project for latter versions.