Intelligence Transportation System

Dr.Emad Nabil*
Doctor in cairo university, egypt

Omar alkubati
20140377
omer.alqubati@gmail.com

Mahmoud Mosaad
20140256
mahmoudmosaad50@gmail.com

Ziad Aamer
20140341
ziadaamer9@gmail.com

Hameed
20140375
hameed203083@gmail.com

Aseel
20140367
asseel7723@gmail.com

February 21, 2018

*Thank you for help us to make this project done

Contents

1 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 salesmans should deliver orders.

2 Problem Significance

2.1 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.

2.2 The Problem variants

2.2.1 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.

2.2.2 Asymmetric (ATSP)

The TSP is asymmetric if, the distance for going from a point to another may be different of the returning distance.

2.2.3 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.

2.2.4 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.

2.2.5 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).

2.2.6 Edge/arc and 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.

2.3 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.

2.4 Problem Solution

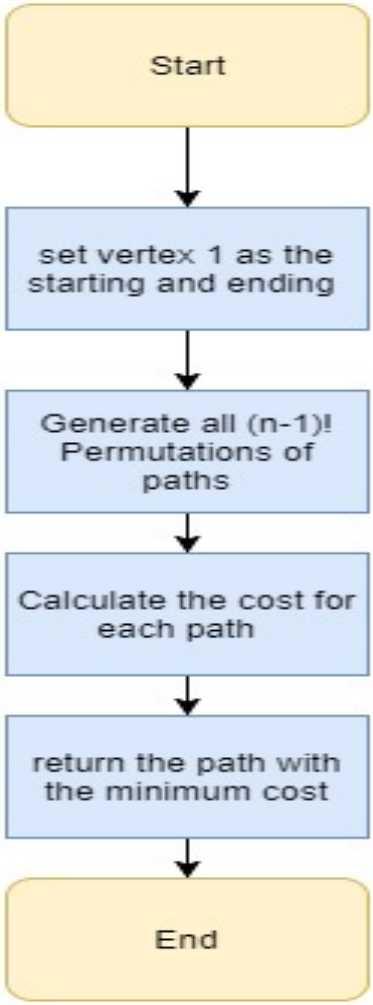
Travel Salesman Problem has more than one approach to solve it.
e.g: Exact Approach, Greedy Approach, Heuristic Approach and Genetic Algorithm Approach.

2.4.1 Exact Approach

The solution is optimal but very slow.

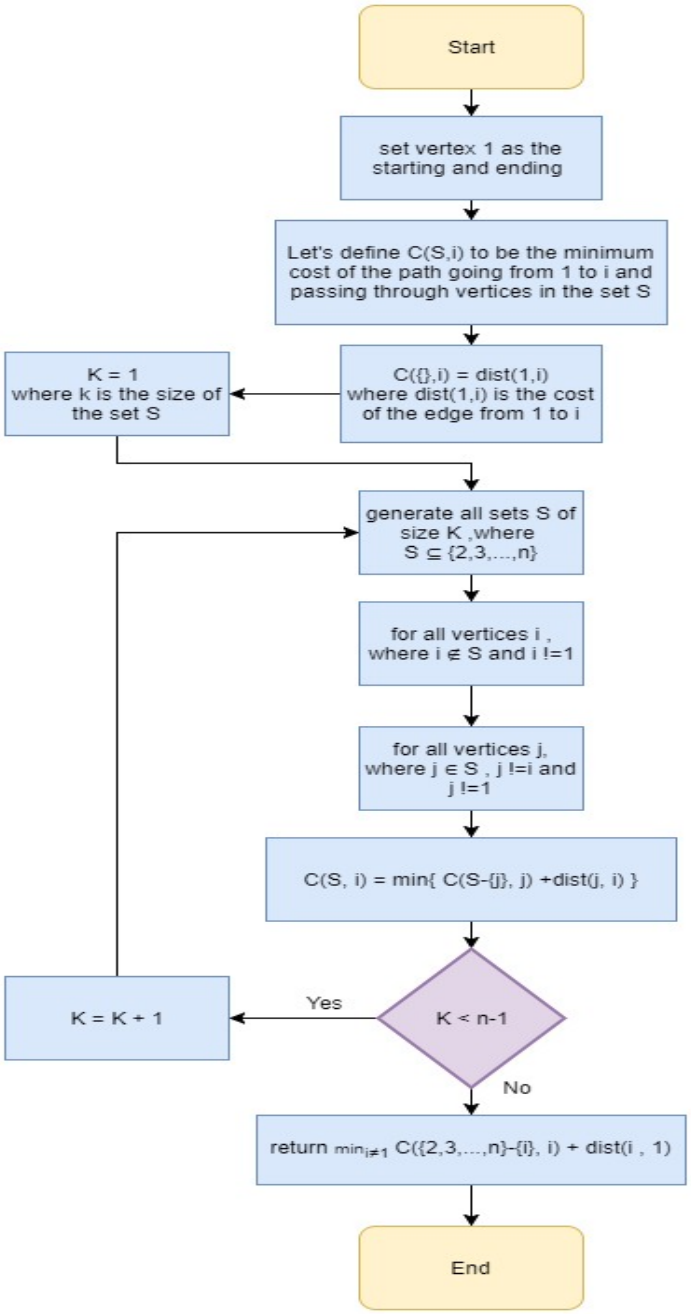
2.4.1.1 Brute Force

naive algorithm that generate all possible paths choose the minimum cost path but we will not use it because it has a very high Complexity O(n!) .



2.4.1.2 Dynamic Programming (Held Karp)

Every sub-path of a path of minimum distance is itself of minimum distance. we will use this algorithm in exact approach as its complexity is $O(2^n n^2)$.



Assuming we run the algorithm on a 10^10 instructions per seconds processor.

Feasible 

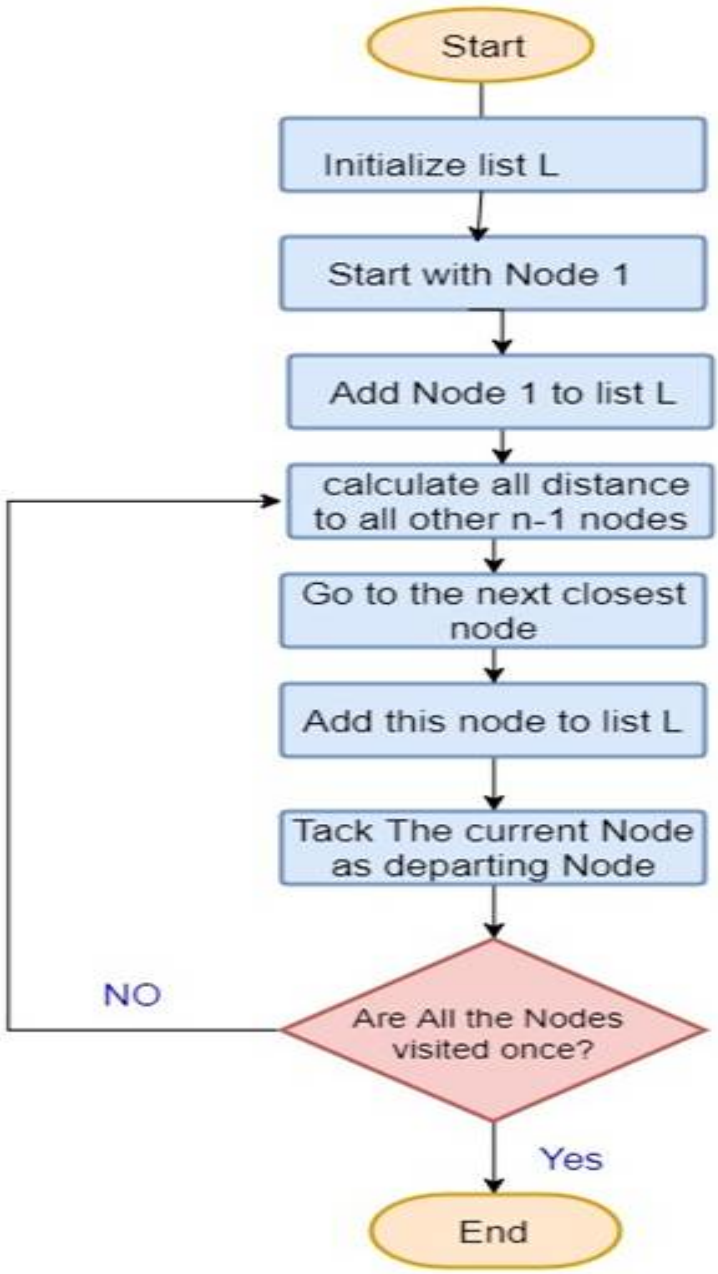
#Of Cities	Time
10	0.00001024 Seconds
20	0.04 Seconds
25	2 Seconds
30	1.6 Minutes
35	1.16 hours
40	2.03 Days
50	8.9 years !!

2.4.2 Heuristic Approach

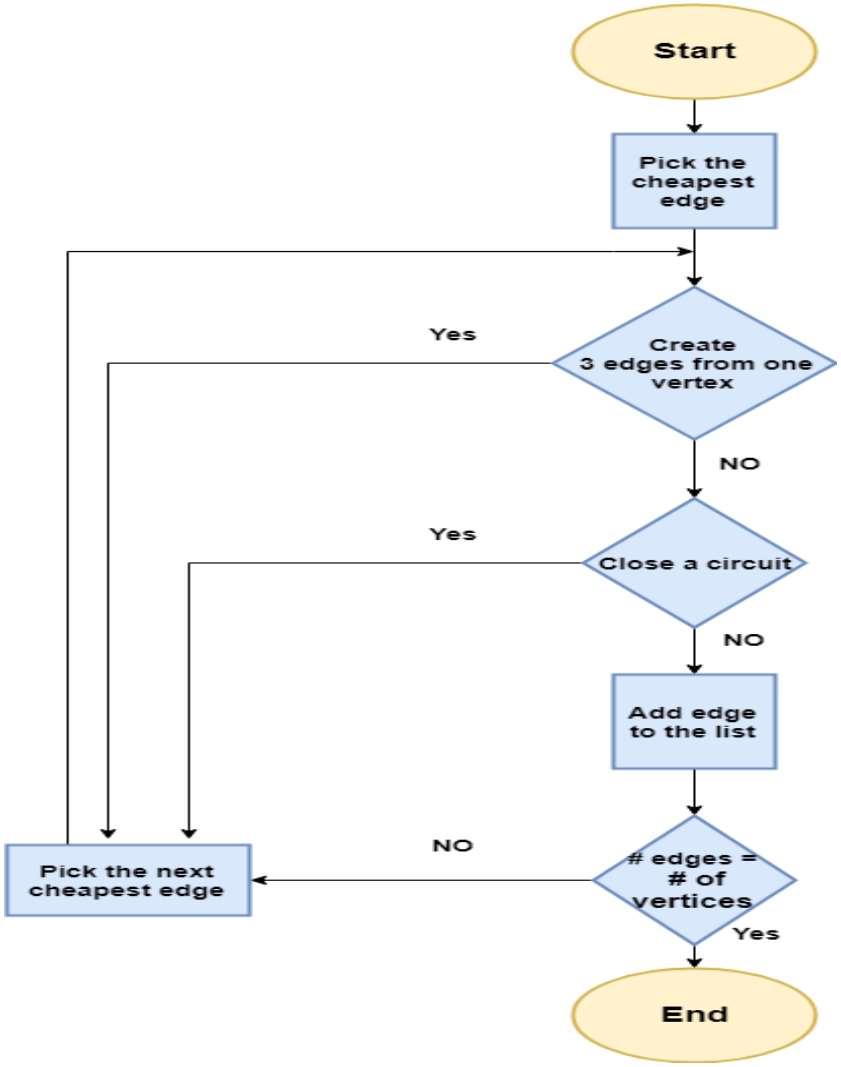
- **Approximation: Solving the TSP optimally takes to long, instead uses approximation algorithms, or heuristics and can get good solutions but may not optimal.**
- **Tour Construction: Tour construction algorithms have one thing in commmon, they stop when a solution is found and never tries to improve it.**
- **Tour Improvement: Once a tour has been generated by some tour construction heuristic, we might wish to improve that solution. There are several ways to do this, but the most common ones are the 2-opt or k-opt local searches. Their performances are somewhat linked to the construction heuristic used.**

2.4.2.1 Nearest Neighbor Algorithm (Construction)

An algorithm that choose the best choice that rechable at current state. Complexity: O(n log(n))

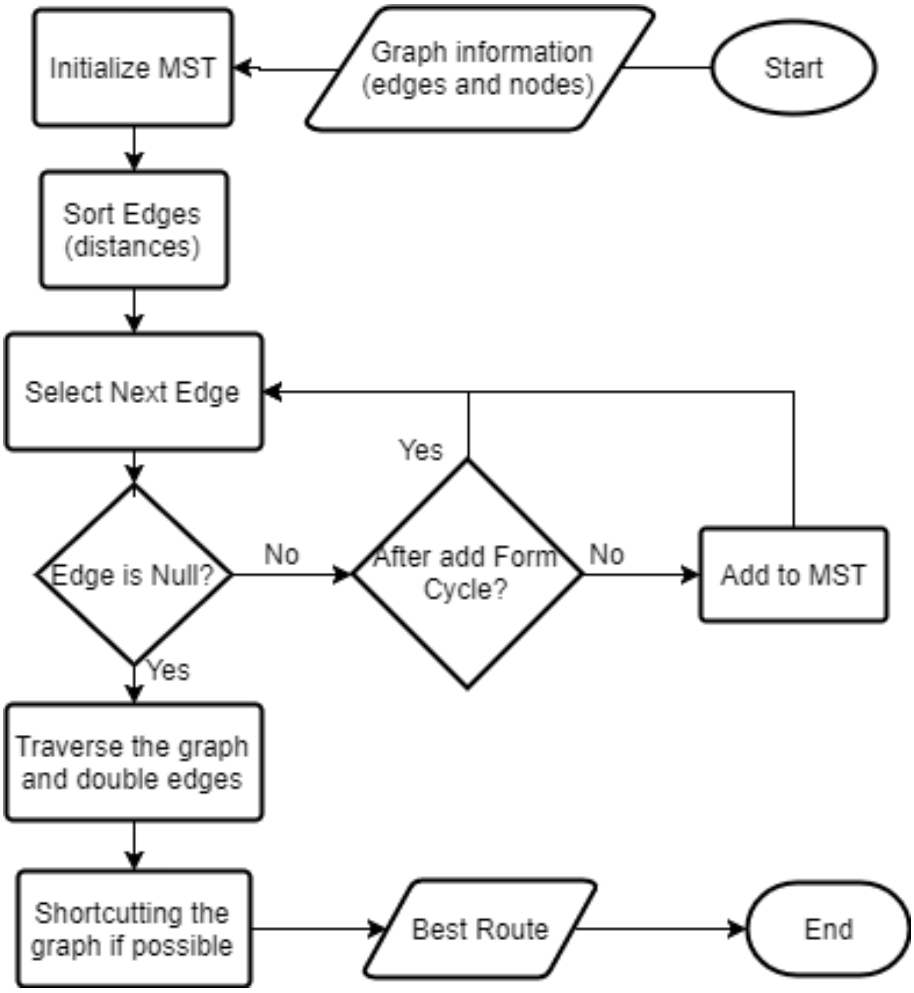


2.4.2.2 Cheapest Link (Construction)

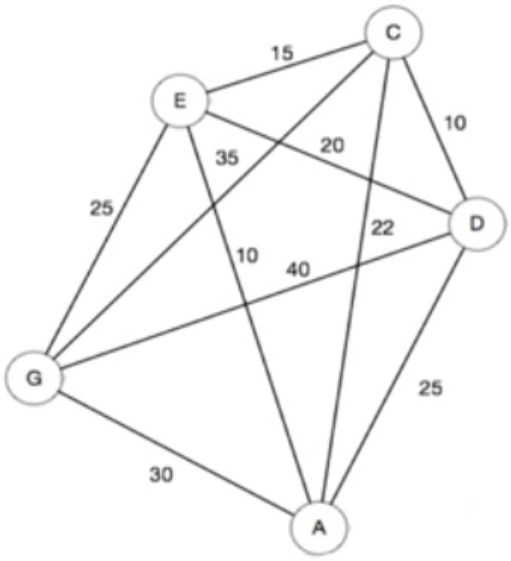


2.4.2.3 Depth First Tree Tour (Construction)

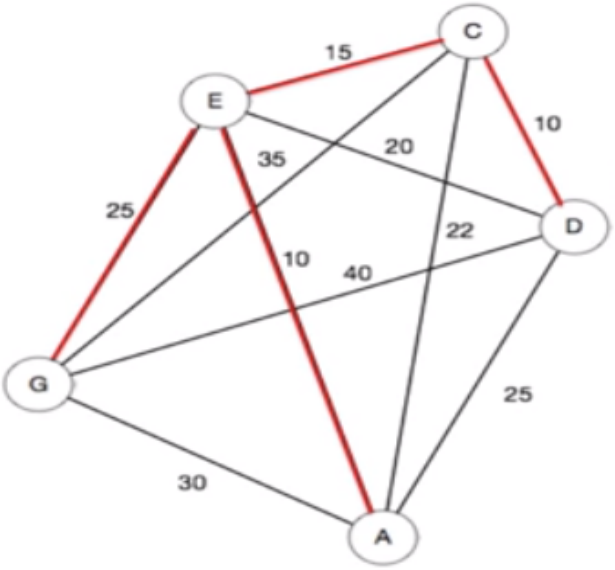
This algorithm (DFTT) based on minimum spanning tree (MST) or kruskal's algorithm, the length is exactly twice of the MST's weight, MST weight is not more than length of optimal tour, skipping visited nodes along the DFTT and apply Triangular Inequality, tour length at most twice of optimal length. This flowchart explain all the process.



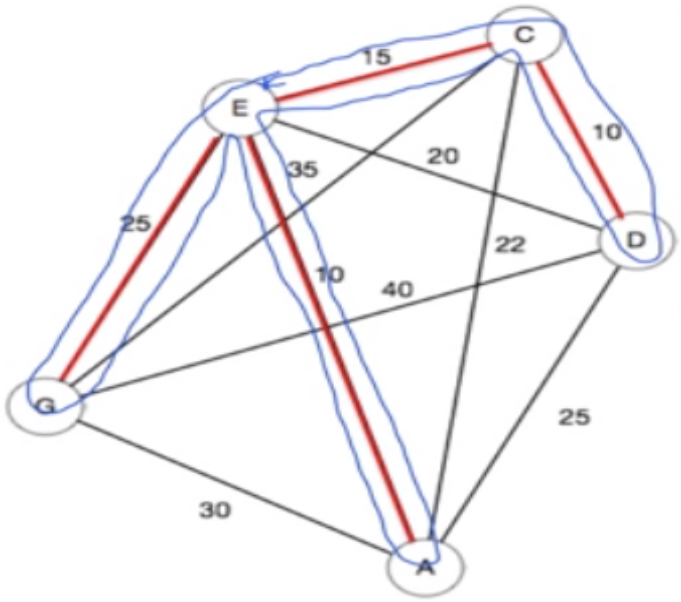
Graph Example :



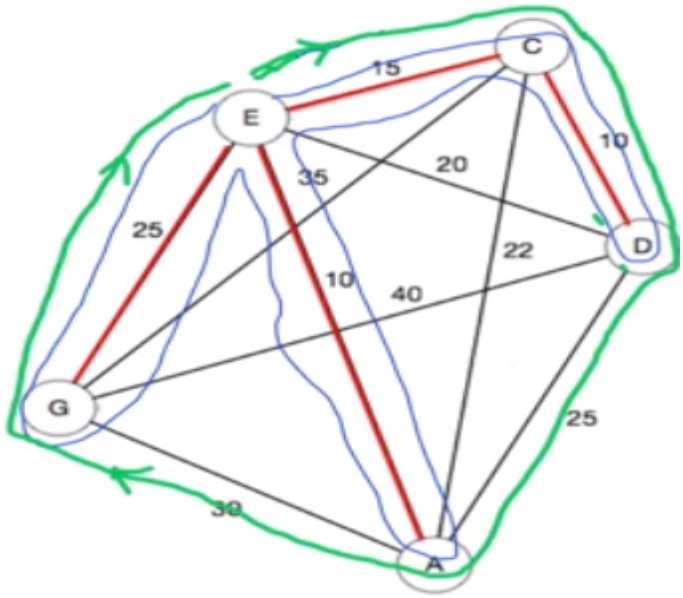
First get the MST of the graph like in the figure (in red).



Traverse the graph along MST edges and double the edges (in blue).

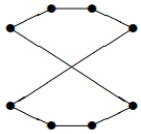


Last thing is to apply triangular inequality to get the best tour (in green).

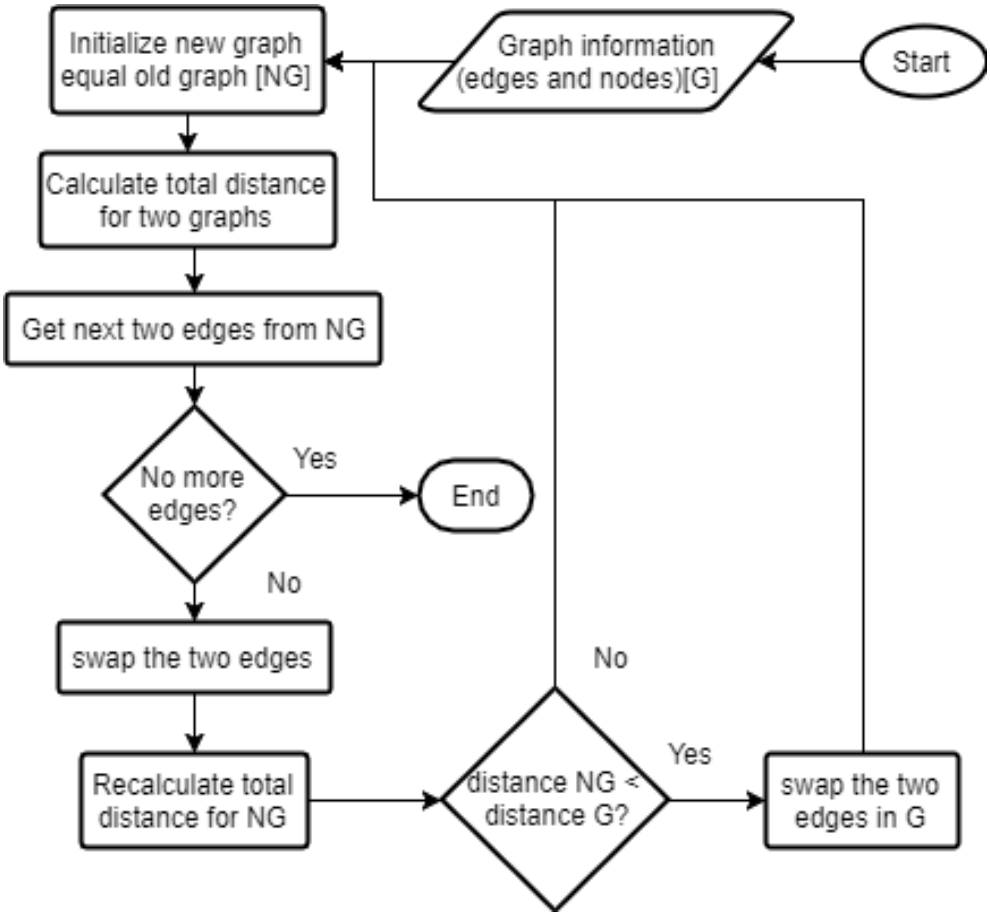


2.4.2.4 2-opt (Improvement)

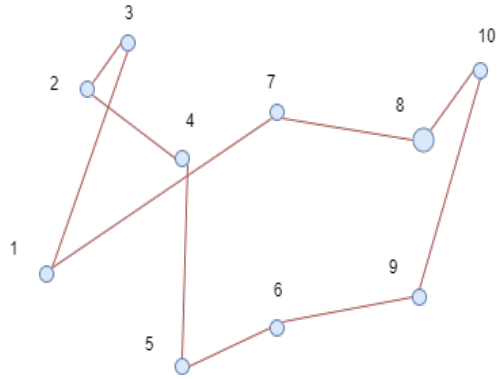
The 2-opt algorithm basically removes two cross over edges from the tour, and reconnects the two paths created. There is only one way to reconnect the two paths so that we still have a valid tour. We do this only if the new tour will be shorter. Continue removing and reconnecting the tour until no 2-opt improvements can be found. The tour is now 2-optimal. this figure explain cross over 2-opt can solve it.



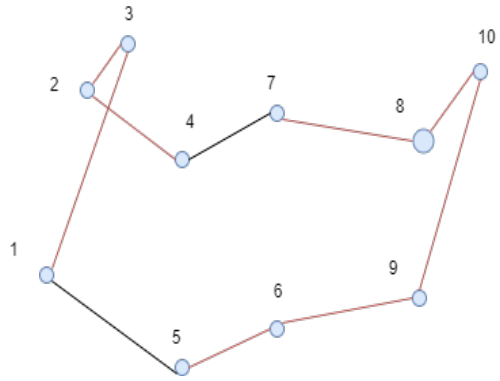
This flowchart explain all the process.



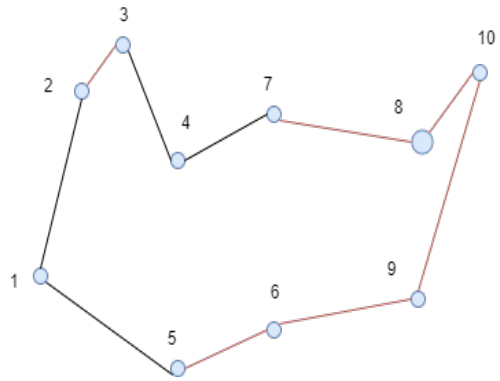
Look for an improvement obtained by deleting two edges and adding two edges.



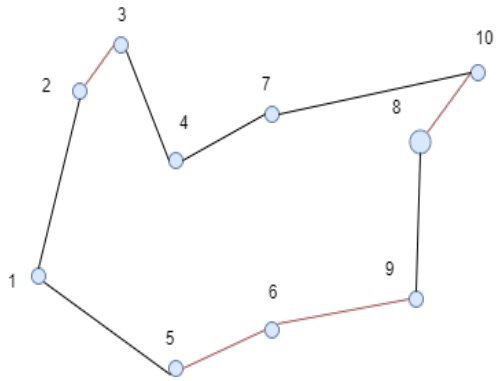
Deleting arcs (4,7) and (5, 1) flips the subpath from node 7 to node 5.



Deleting arcs (1,3) and (2, 4) flips the subpath from 3 to 2.



Deleting arcs (7,8) and (10, 9) flips the subpath from 8 to 10.



2.4.2.5 Christofides algorithm (Construction)

Solutions is guranteed to be within a factor of 3/2 of the optimal solution O(n^3). The best approximation ratio that has been proven for TSP but it has the constrain that the edges must and obey the triangle inequality. It provides a very good heuristic solutions, but unfortunately it is impractical because of the constraints it has.

2.4.3 Meta Heuristic Approach

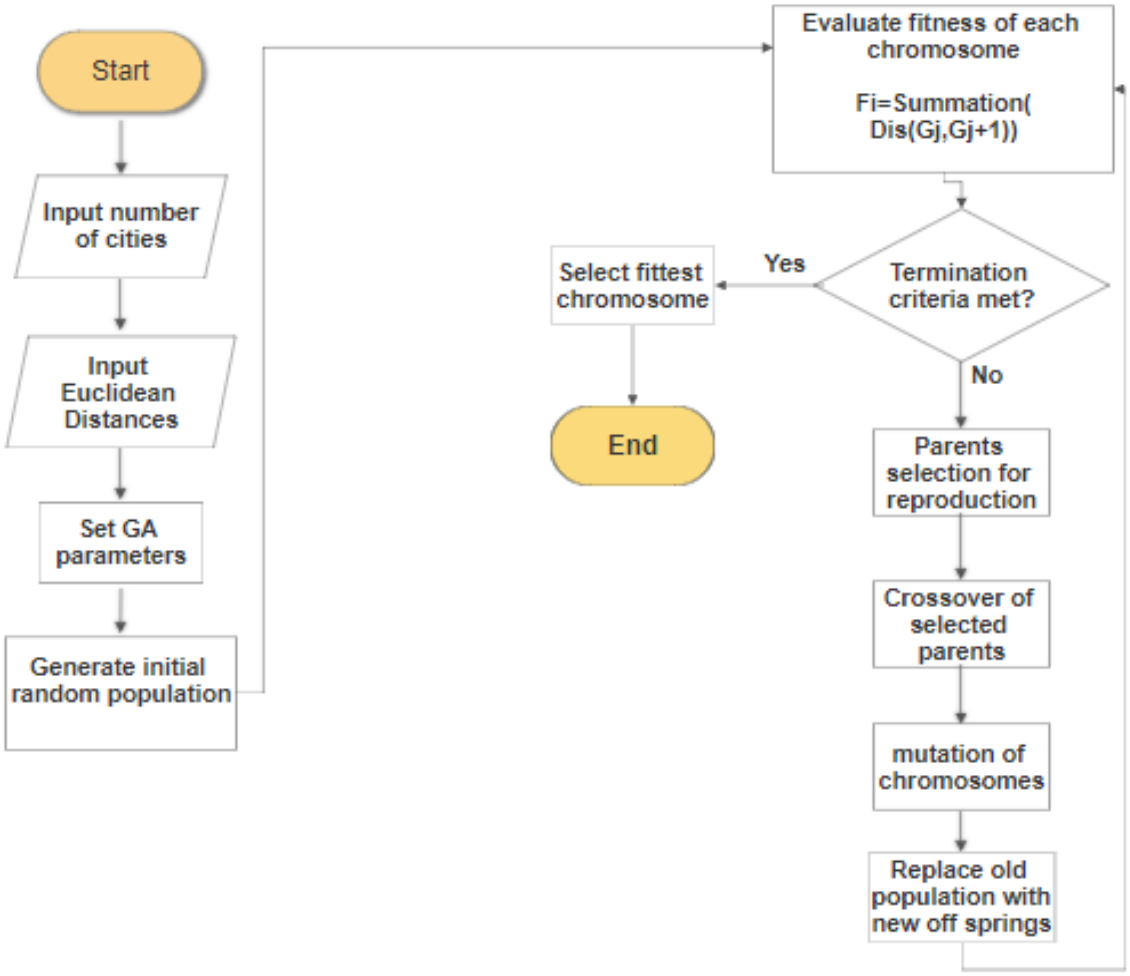
same as Heuristic, but Meta-Heuristic is problem independent and can be applied to a wide range of problems.

2.4.3.1 Genetic Algorithm

Fitness function:

$$F_i = 1/\sum_{i=1}^{n} Distance(G_i, G_{i+1})$$

Complexity: O(n^3).



2.5 TSP Solver Modules

- Clustering module.
- Route creation module.

2.5.1 Clustering module

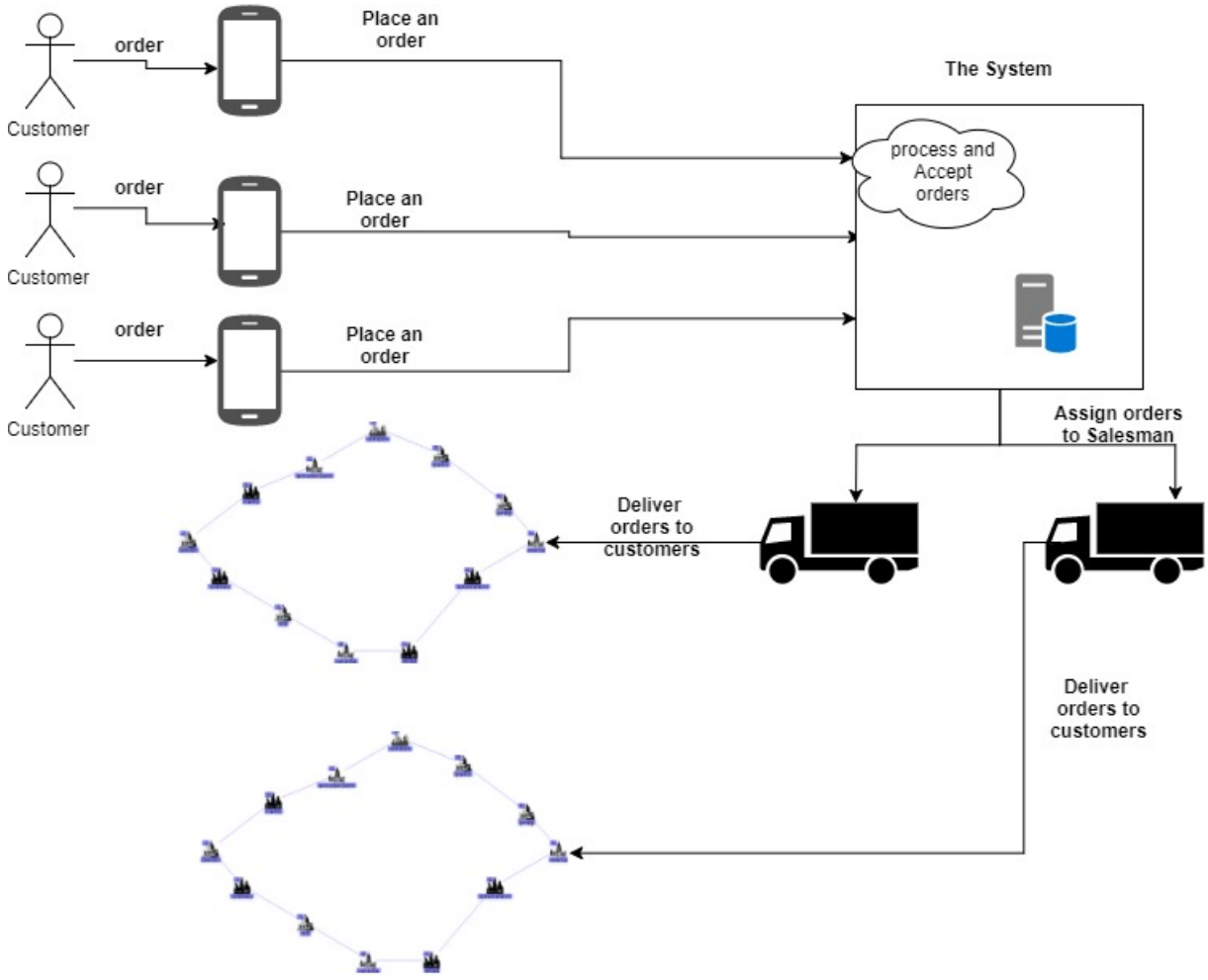
- Automation, e.g: K-Means.
- Clustering by region.
- Manual.

2.5.2 Route creation module

If number of cities >= 21 then Run Exact Algorithm to get best route
Else Run all heuristic and meta heuristic algorithms and return the best route resulted from all algorithms

3 System Analysis and Design

3.1 System Architecture



3.2 Stakeholders

A stakeholder can be internal or external.

3.2.1 Customer

Are External-Operational stakeholders, who make the orders and this stakeholders will interact with the system via Android application.

3.2.2 Admin

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.

3.2.3 Salesman

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.

3.3 Functional Requirements

3.3.1 Customer

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

3.3.2 Admin

- Reject or accept an order.
- Run TSP solver.
- Add salesman.

3.3.3 Salesman

- View requests.
- View path.
- Confirm request delivery.
- Check availablility.

3.4 Non-Functional Requirements

3.4.1 Security

- The System has a form of protection by applying authentication and authorization, so any unauthorized access to the system is denied.
- Avoid SQL injection.
- Avoid XSS.
- Avoid rainbow tables.

3.4.2 Performance

- Login must be completed in less than 3 seconds.
- Peak load 200 user every hour.
- Admin assign orders to salesman in less than 10 seconds.

3.4.3 Reliability

- The system has to be 100% reliable.

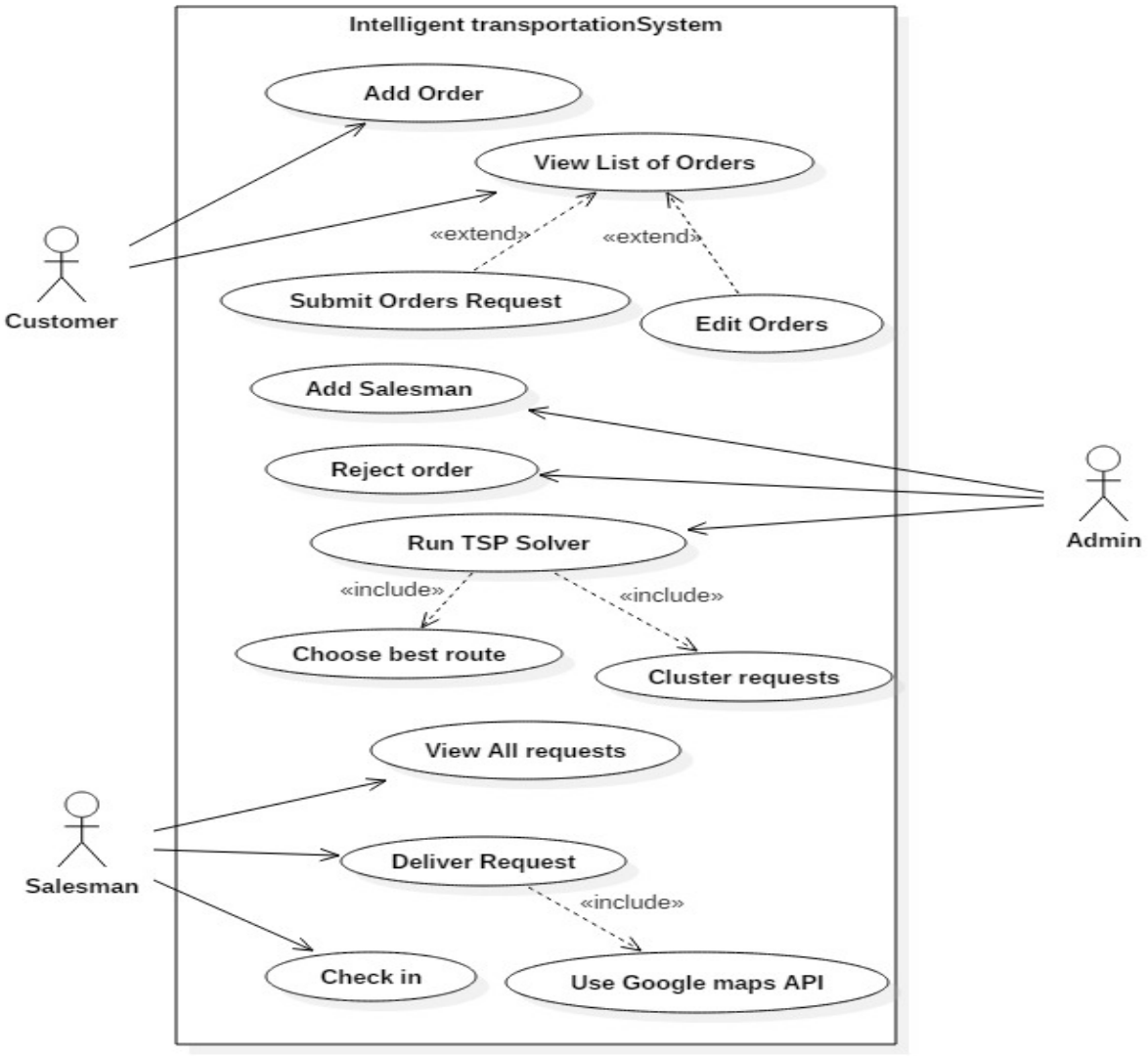
3.4.4 Availability

- The system will be available 24/7 and making backups.

3.4.5 Usability

- The customer can easily order any products with any quantities with easy UX.

3.5 Use Case Diagram



3.6 Use Case Tables

3.6.1 Customer

Use Case ID:	1_Customer	
Use Case Name:	Add an Order	
Actors:	Customer	
Pre-conditions:	1- Logged in as Customer	
Post-conditions:	1-An Order is added to the List of Customers' Orders	
Flow of events:	**User Action**	**System Action**
	1- User clicks on view items to order	
		2-System displays the list of available items to be order
	3- Customer selects an item and can choose its quantity or leave it (default 1)	
	4- The customer then clicks on order button to add the order to the orders list	
		5- System receive the order and its quantity, and then save it in the order list
Exceptions:	**User Action**	**System Action**
	1-The customer selects a negative, zero quantity or choose a very large number	
		2-The system print that the quantity selected is not available

Use Case ID:	2_Customer	
Use Case Name:	Submit Request	
Actors:	Customer	
Pre-conditions:	1- Logged in as Customer 2- The list of Orders is already selected and opened	
Post-conditions:	1- The list of orders is submitted to the System (Admin)	
Flow of events:	**User Action**	**System Action**
	1-Customer press on submit Orders button	
		2-System shows total price and invoice of orders, then show confirmation button
	3-Customer presses confirm request to send the request	
	4-The customer can press cancel instead to back to the orders list again	
		5-In case of confirming the request the system receive the request and send it to the Admin
Exceptions:	**User Action**	**System Action**
	1-The customer submit an empty list of orders	
		2-System print that there are no items in the list to submit

Use Case ID:	3_Customer	
Use Case Name:	Edit Orders	
Actors:	Customer	
Pre-conditions:	1- Logged in as Customer 2- The list of Orders is already selected and opened	
Post-conditions:	The list of orders is updated and saved	
Flow of events:	**User Action**	**System Action**
	1-Customer presses on edit button	
		2-System shows the products already added in the orders list, their info and quantities
	3- Customer checks the items	
	4- then decides whether to edit the list by changing a quantity or removing items	
	4-The customer then click on save to save the edited list of orders	
		5- System then will update the list and save it in the DB
Exceptions:	**User Action**	**System Action**
	1-The customer selects a negative, zero quantity or choose a very large number	
		2-The system print that the quantity selected is not available

3.6.2 Salesman

Use Case ID:	1_Salesman	
Use Case Name:	Deliver Request	
Actors:	Salesman	
Pre-conditions:	1- Logged in as Salesman .	
Post-conditions:	1-Finishing request fulfillment	
Flow of events:	**User Action**	**System Action**
	1- Salesman asks the directions for the coming request.	
		2-System displays the directions by using Google Map API.
	3- Salesman reaches to the desired customer location and delivers his request.	
	4- Salesman checks on the request as delivered request.	
		5-System returns the next request location.

Use Case ID:	2_Salesman	
Use Case Name:	Check in	
Actors:	Salesman	
Pre-conditions:	1- Finished All requests assigned to him.	
Post-conditions:	1-available to deliver new requests.	
Flow of events:	**User Action**	**System Action**
	1- Salesman goes to inventory.	
	2- Salesman check in when arrives the inventory.	
		3-System adds the salesman to the available salesmen list.
		4- System assigns requests to be delivered by this salesman.

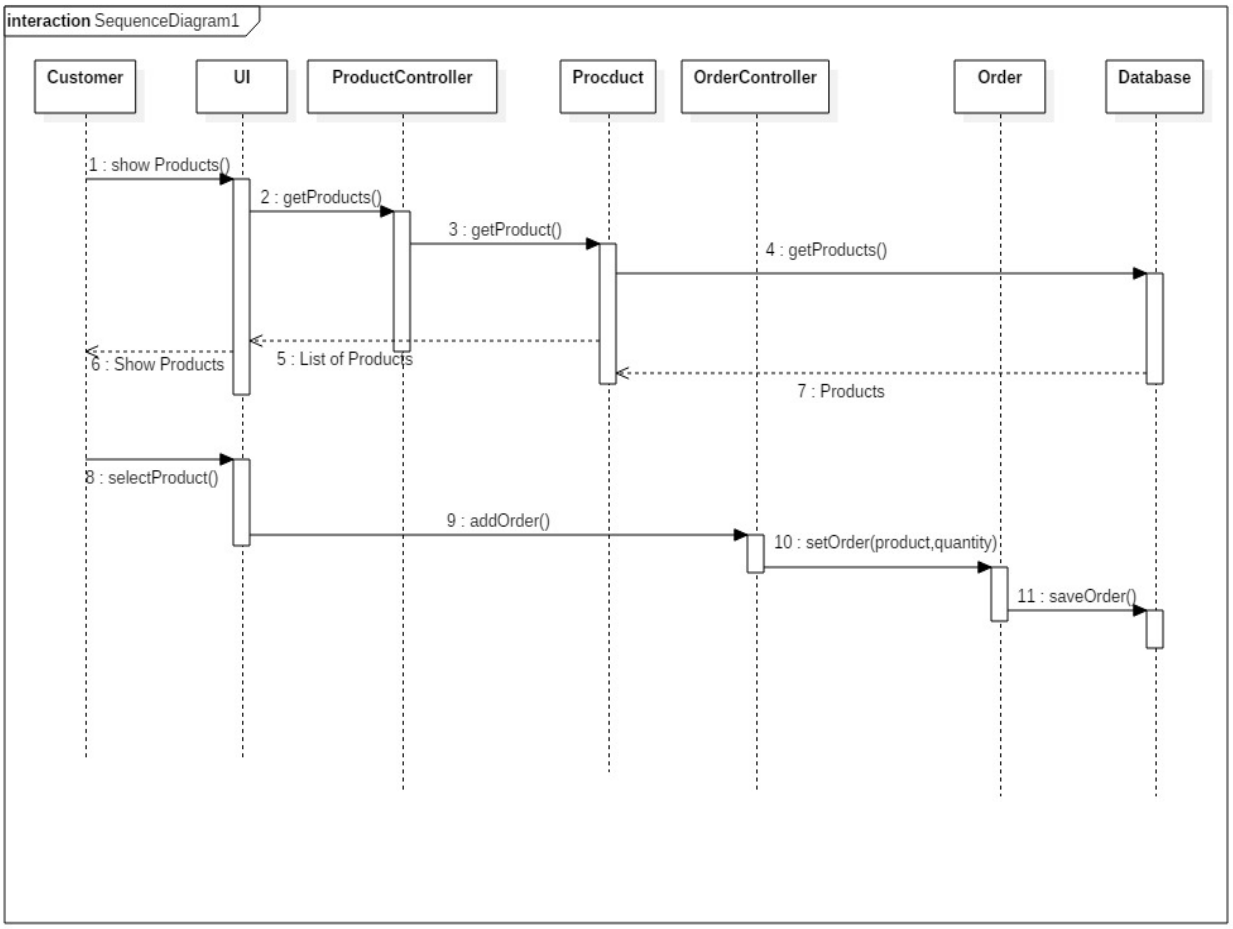
3.6.3 Admin

Use Case ID:	1_Admin	
Use Case Name:	Reject Requests	
Actors:	Admin	
Pre-conditions:	1- Logged in as Admin.	
Post-conditions:	1-Reject Requests that are greater than capacity	
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.

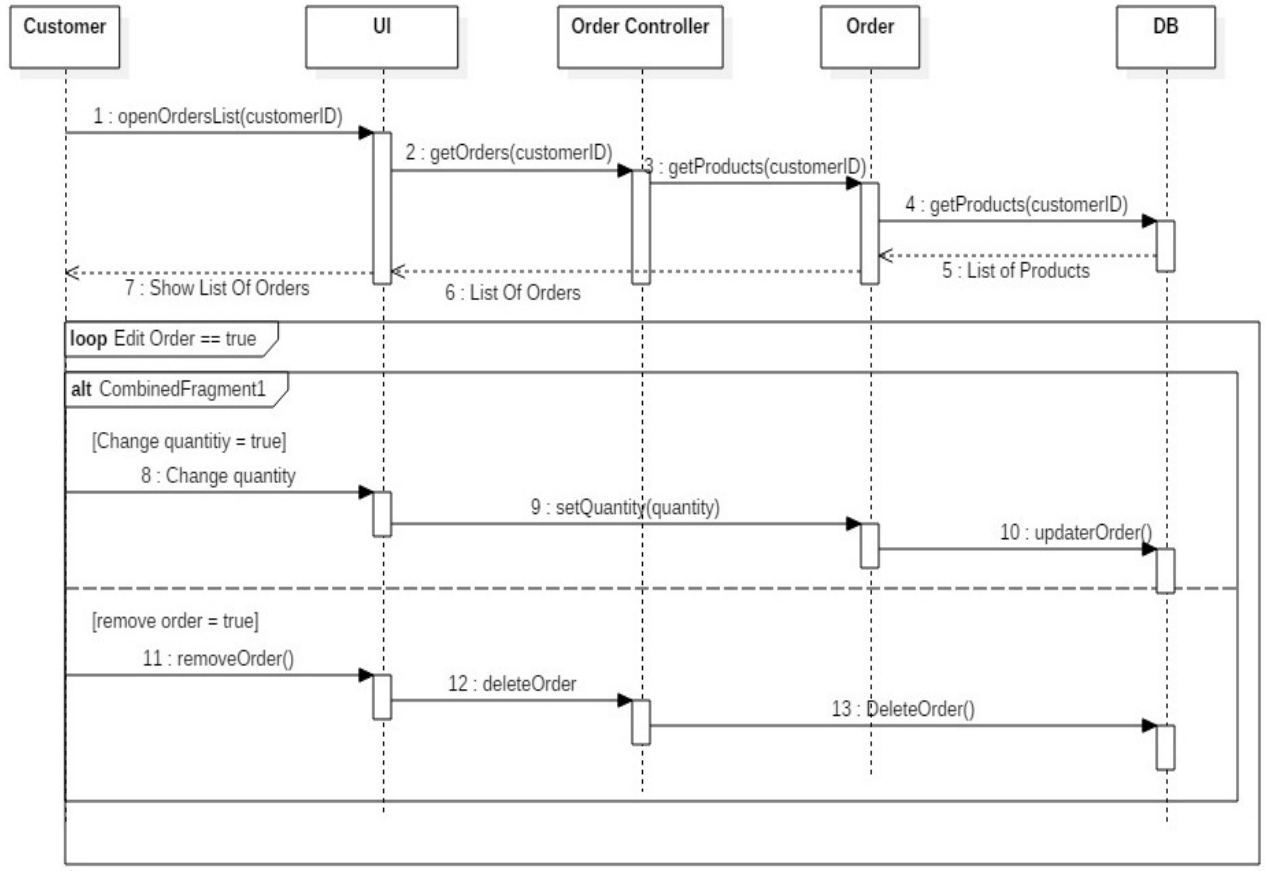
Use Case ID:	2_Admin	
Use Case Name:	Run TSPSolver	
Actors:	Admin	
Pre-conditions:	1- Logged in as Admin.	
Post-conditions:	1-Assign Route To Salesman	
Flow of events:	**User Action**	**System Action**
	1- Admin Run TSP Solver	
		2-TSPSolver Cluster requests automatic clustering using K-Mean algorithm
		3-TSPSolver chooses the best route for each salesman based on number of requests.

3.7 Sequence Diagram

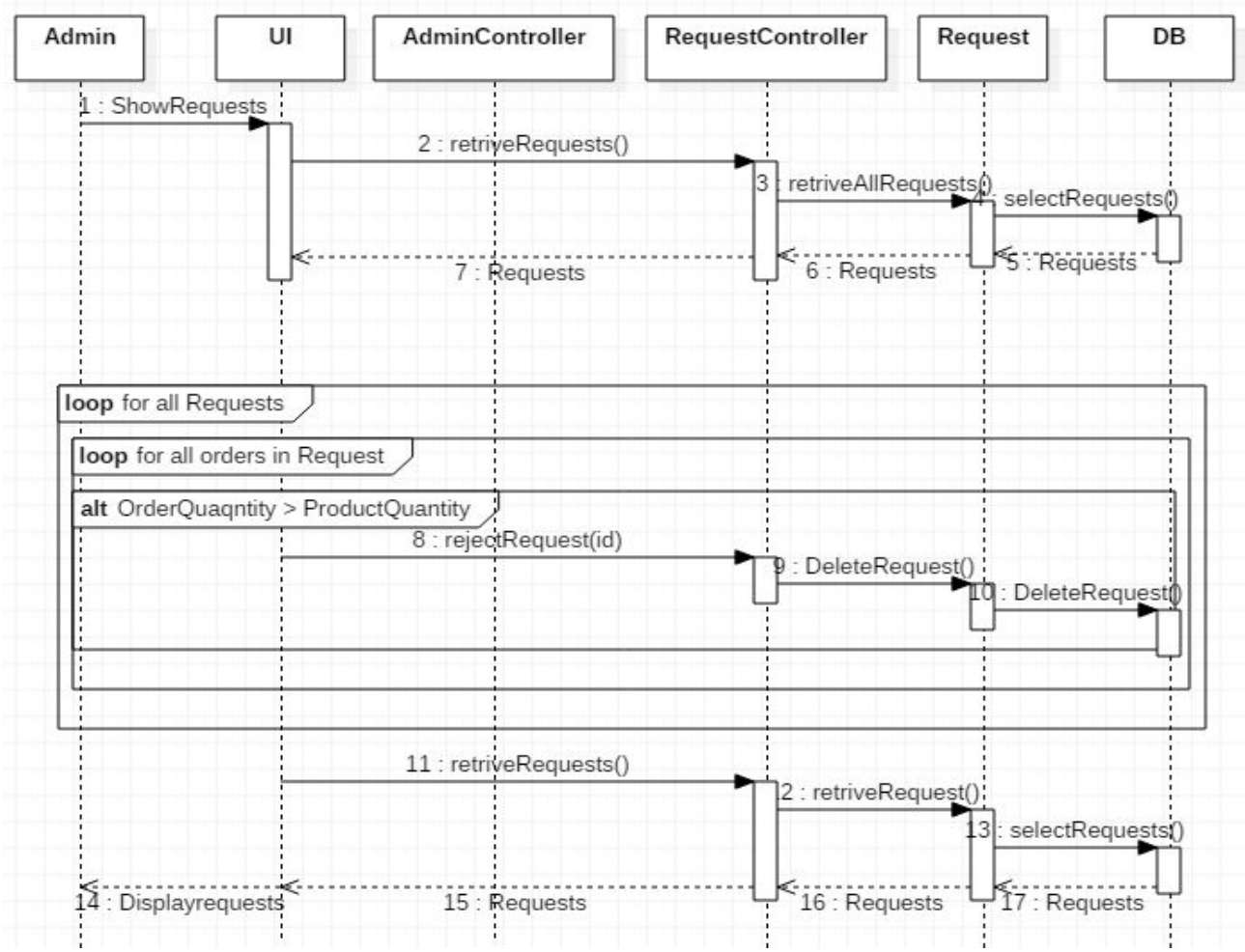
3.7.1 Add Order



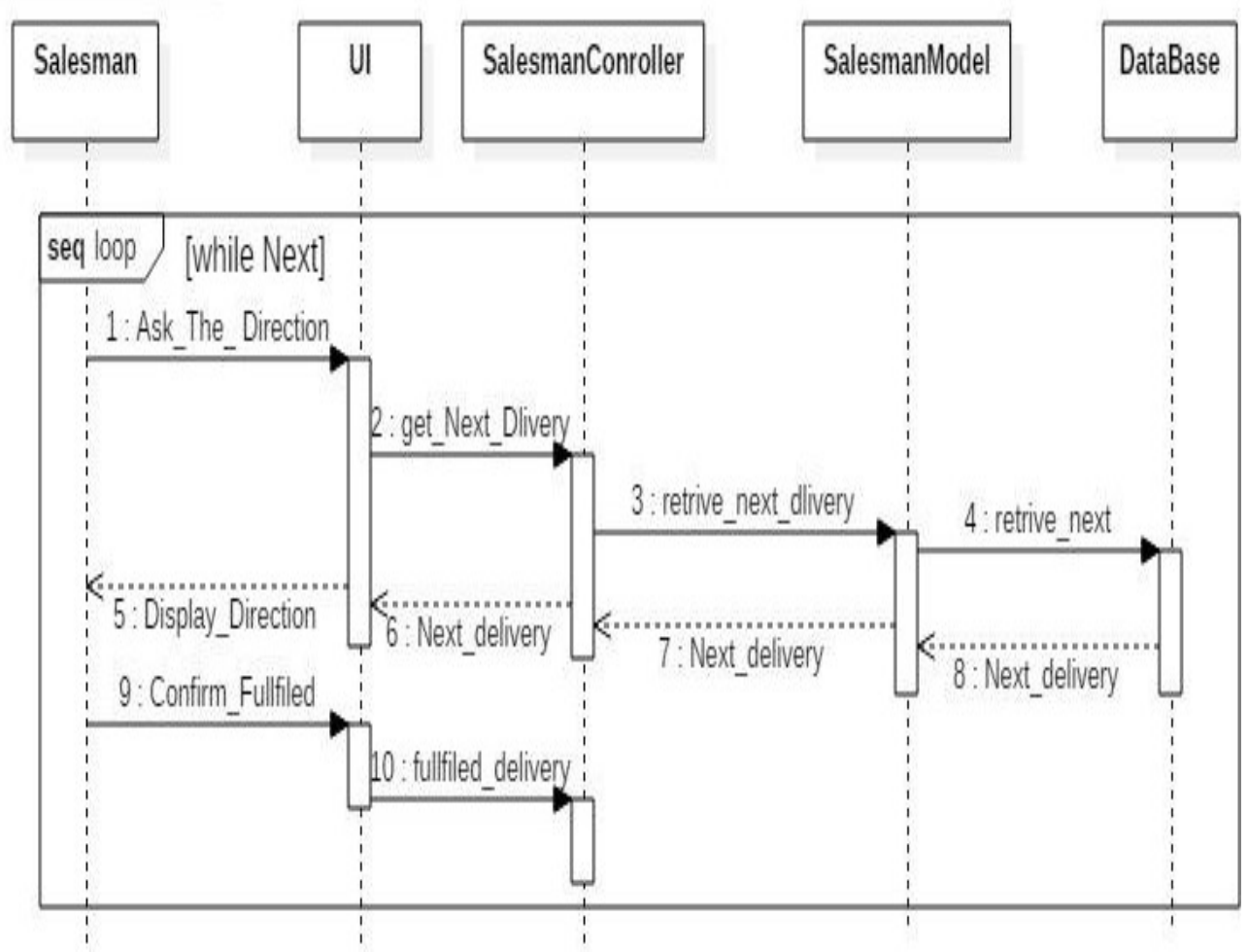
3.7.2 Edit orders



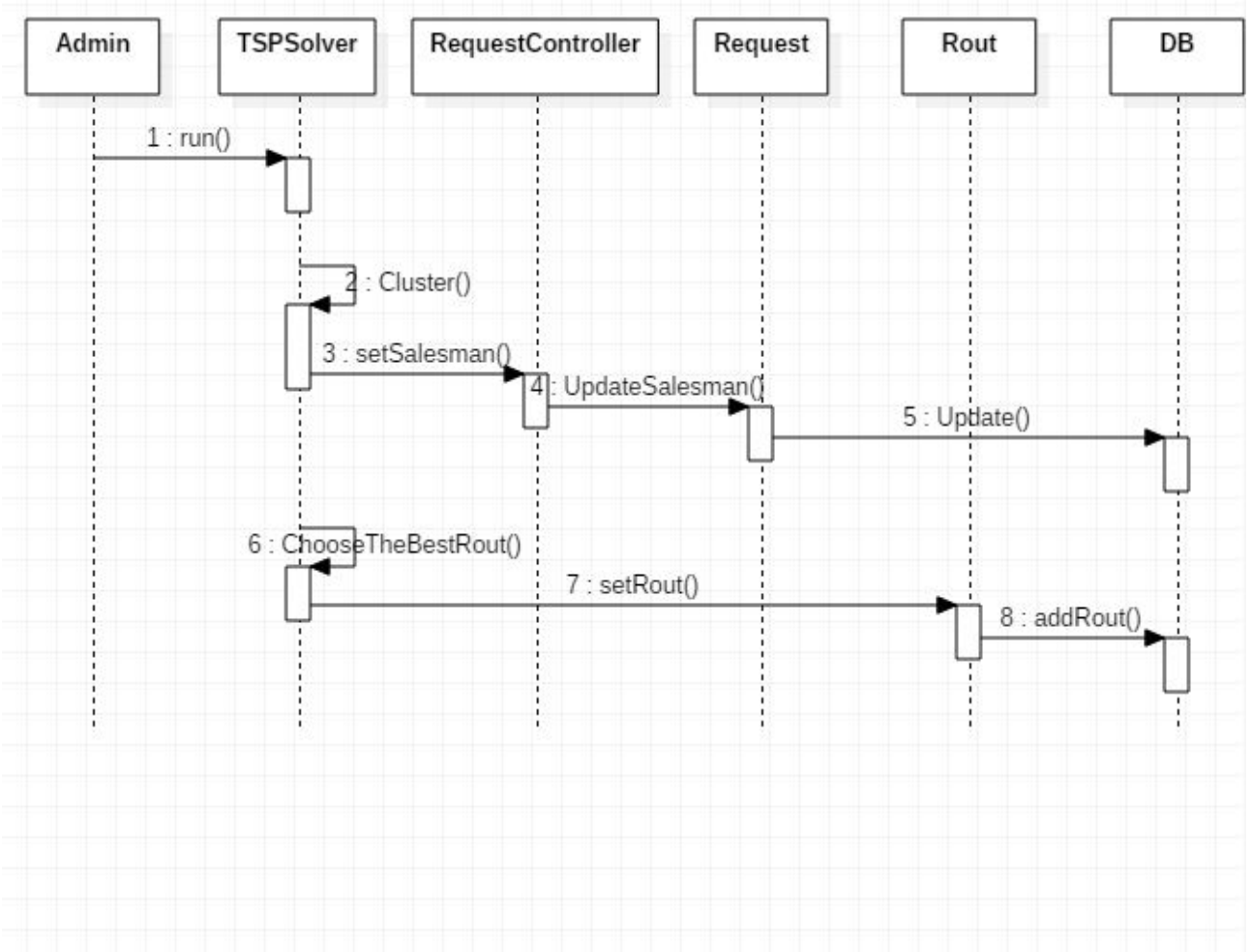
3.7.3 Reject Requests



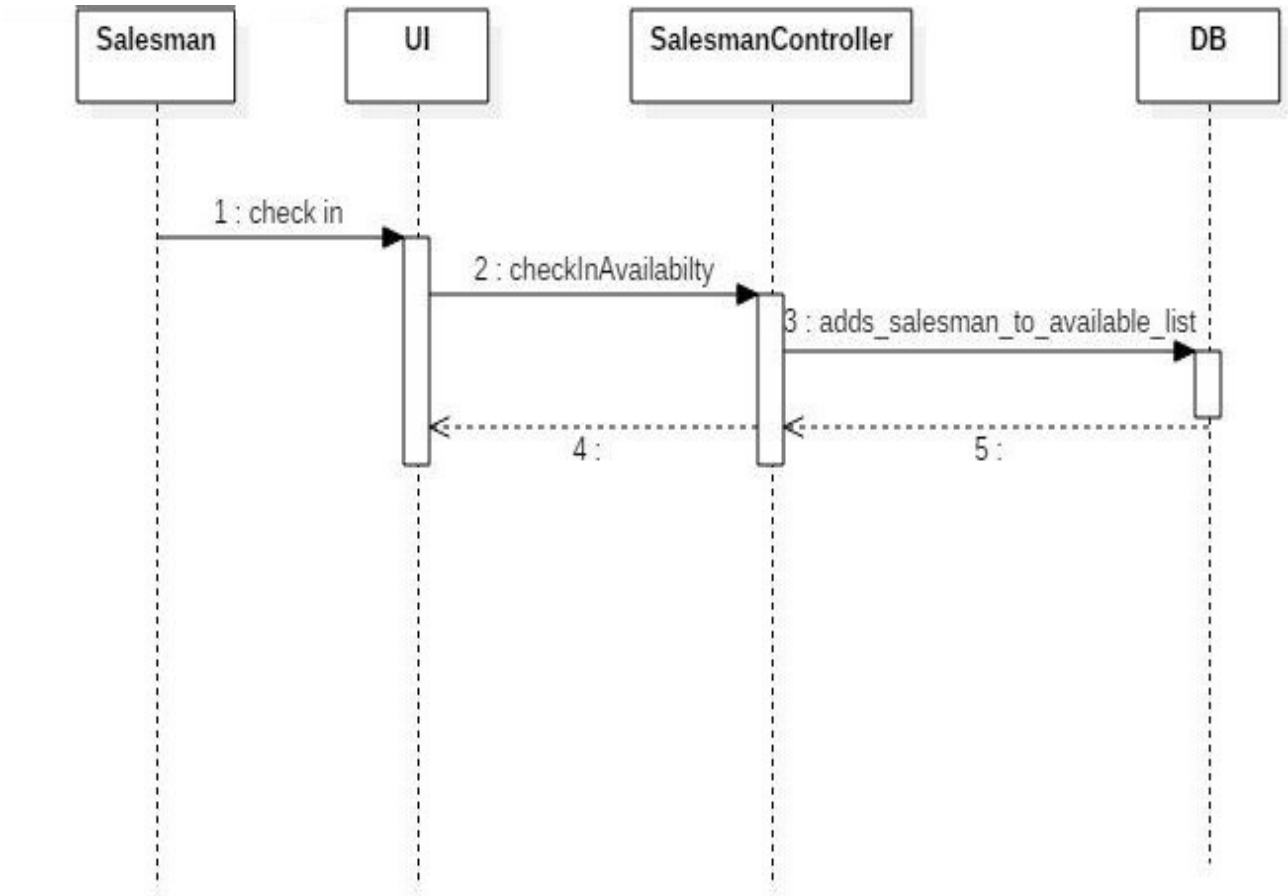
3.7.4 Deliver Order



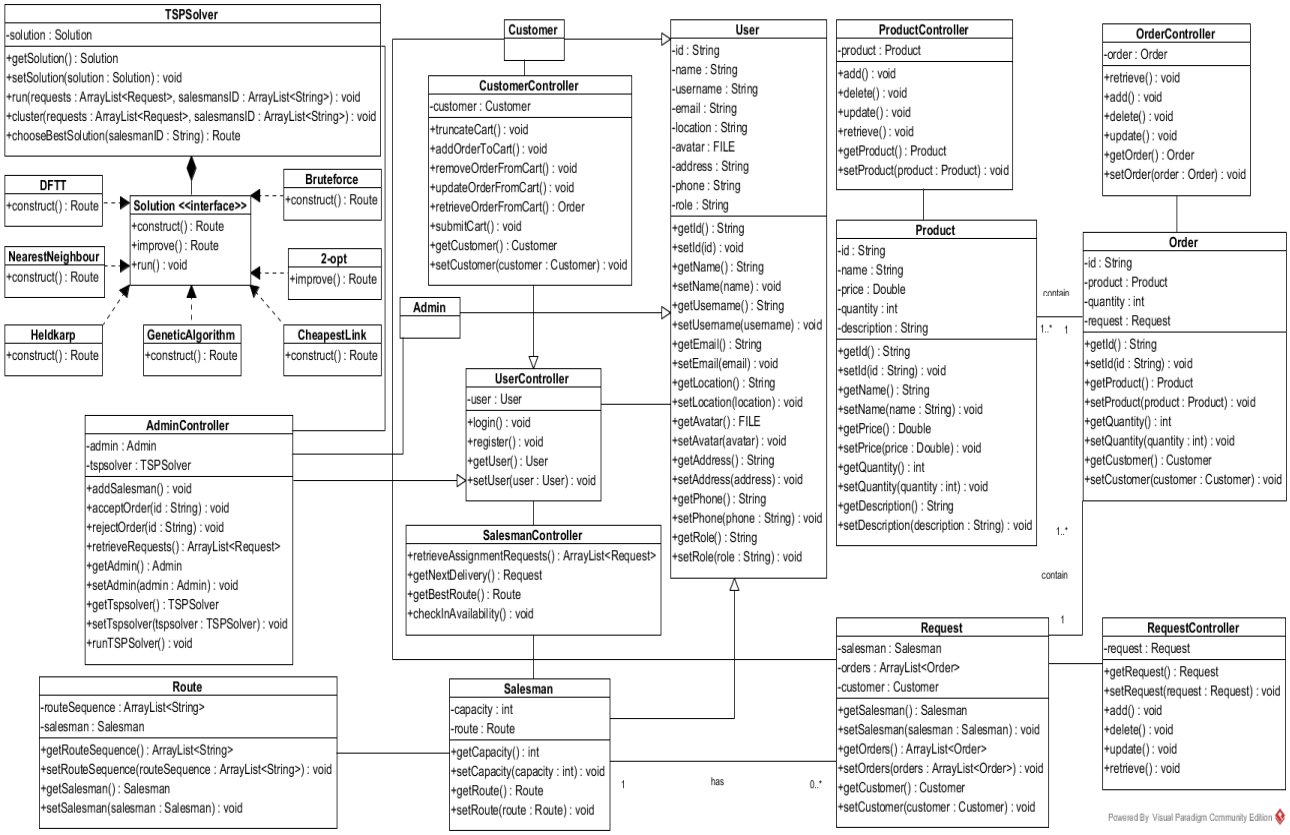
3.7.5 Run TSP Solver



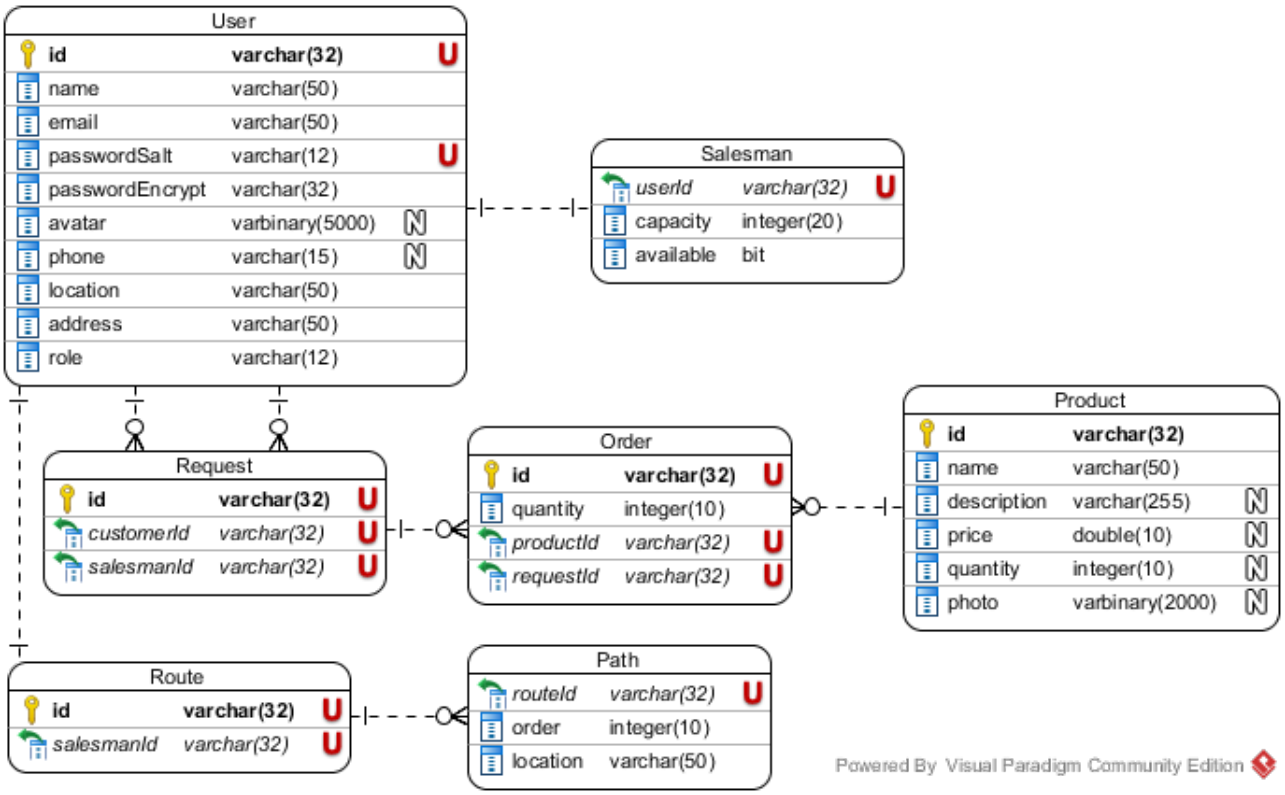
3.7.6 Check In Availability



3.8 Class Diagram

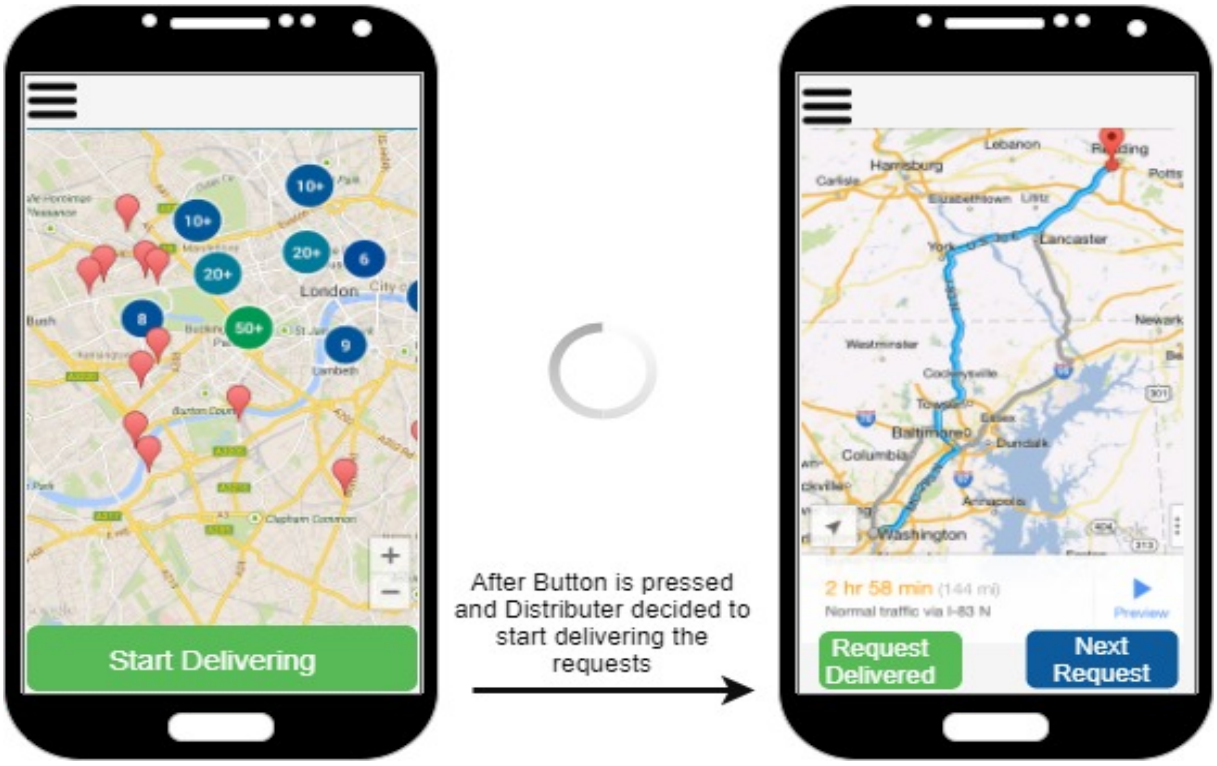


3.9 ERD (Entity Relationship Diagram)

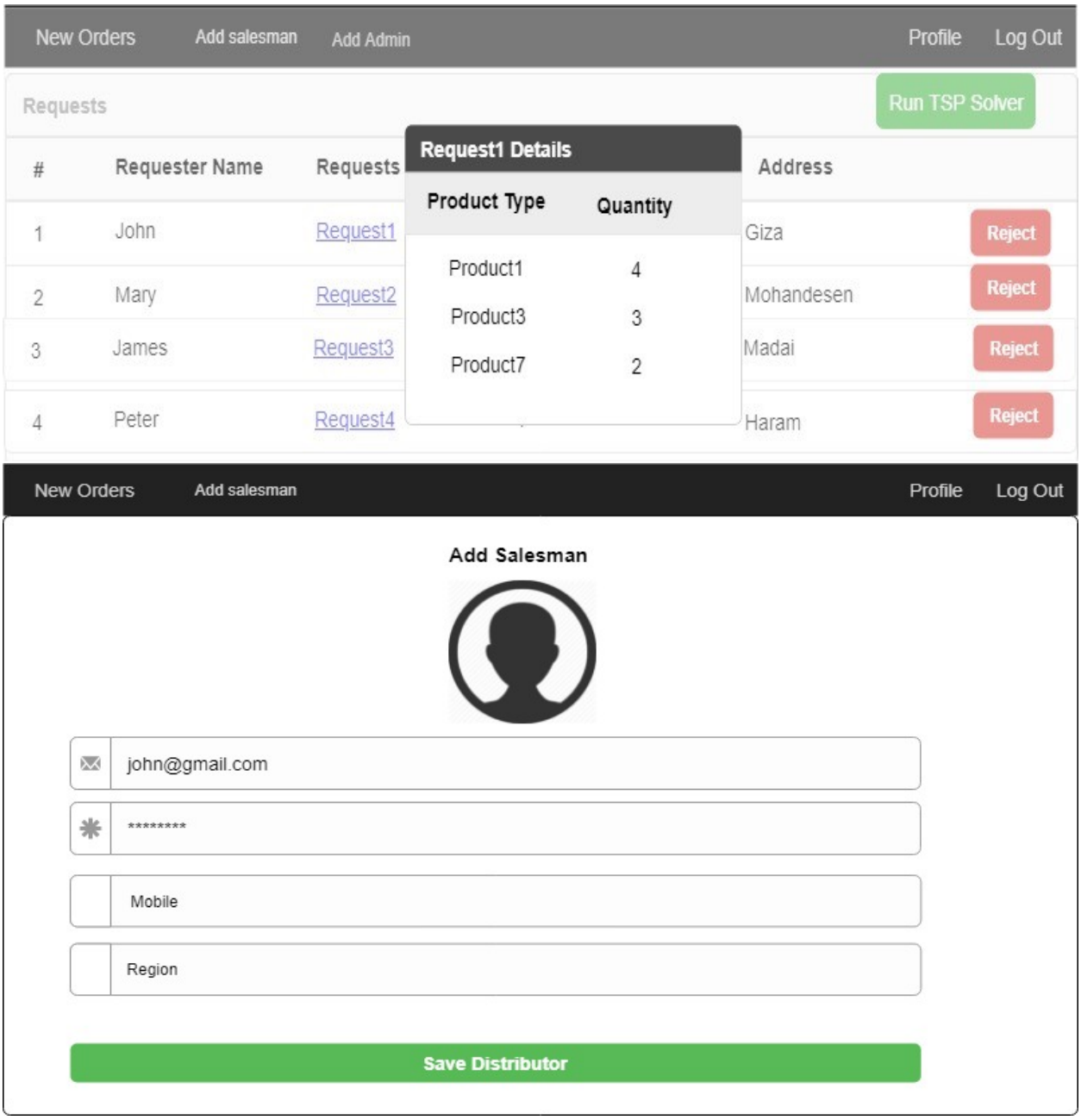


3.10 Prototype

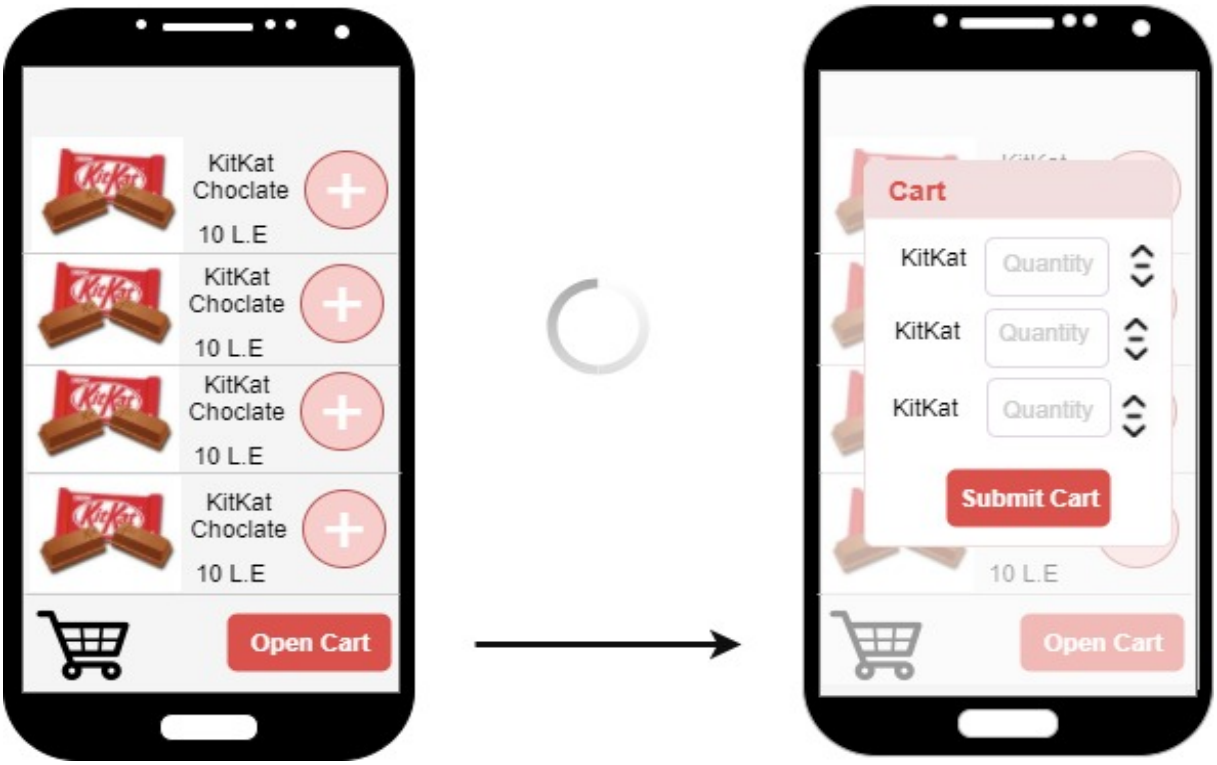
3.10.1 Salesman



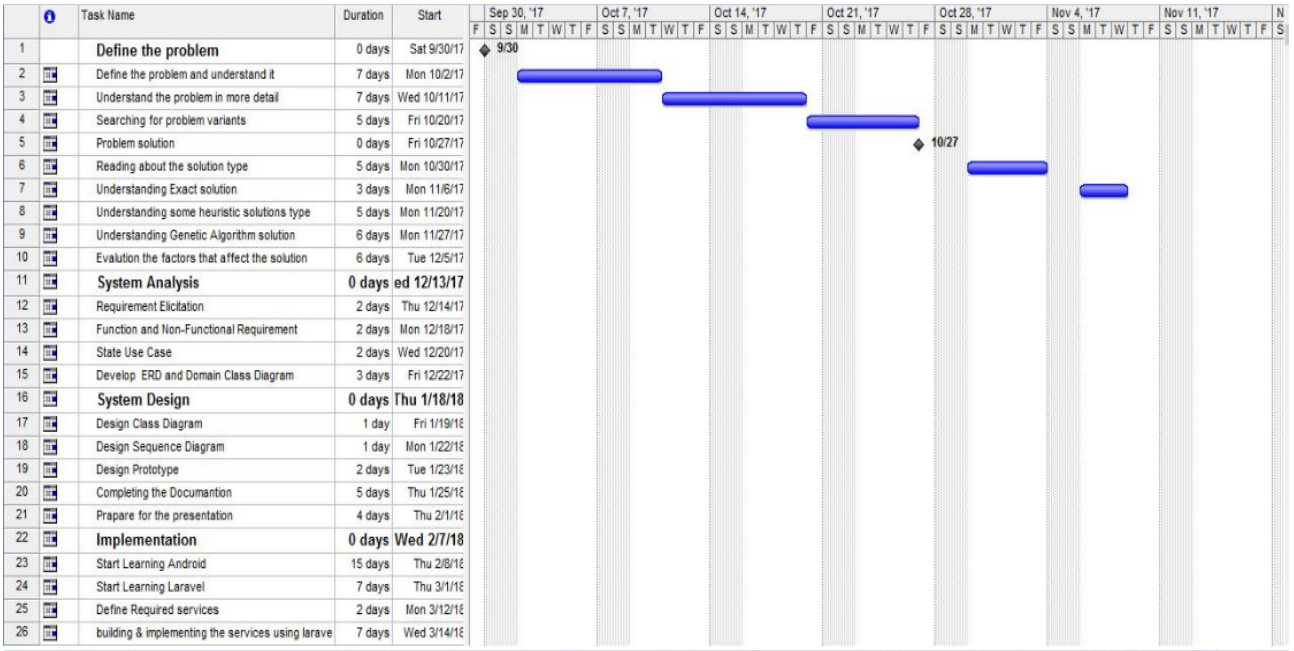
3.10.2 Admin


New Orders
Add salesman
Add Admin
Profile
Log Out
Requests
Run TSP Solver
Requester Name Requests Address
1 John Request1 Giza Reject
2 Mary Request2 Mohandesen Reject
3 James Request3 Madai Reject
4 Peter Request4 Haram Reject
Request1 Details
Product Type Quantity
Product1 4
Product3 3
Product7 2
New Orders
Add salesman
Profile
Log Out
Add Salesman
john@gmail.com
****
Mobile
Region
Save Distributor

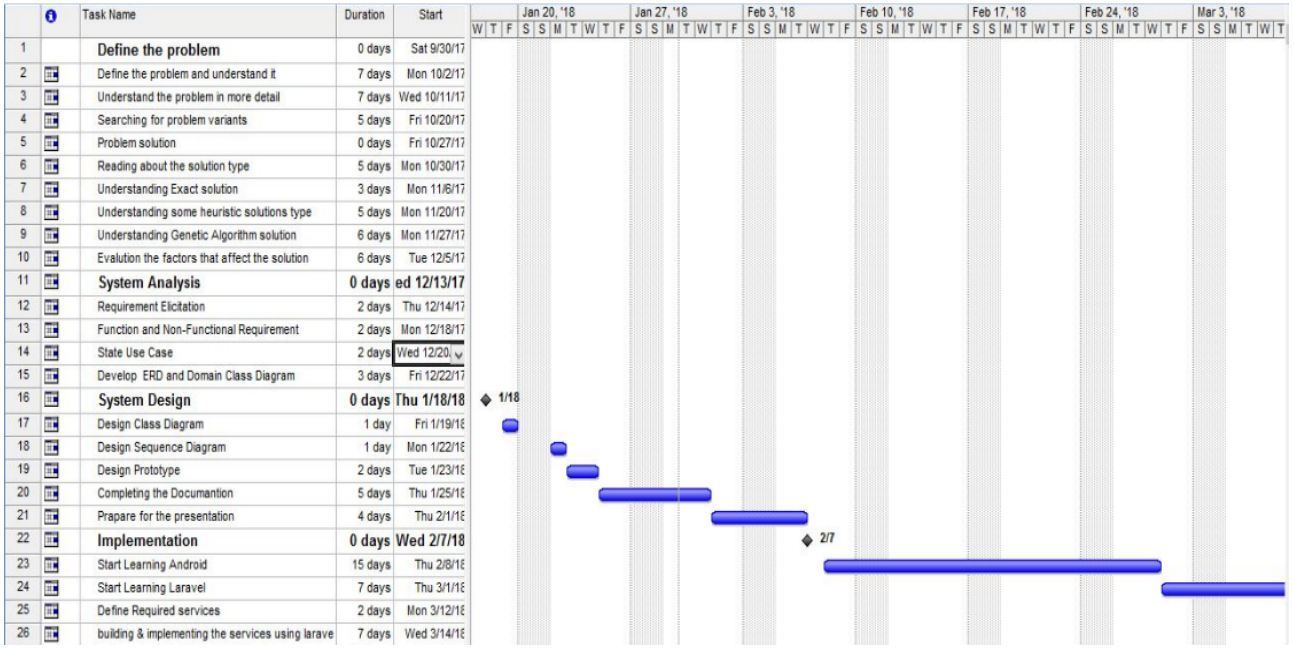
3.10.3 Customer



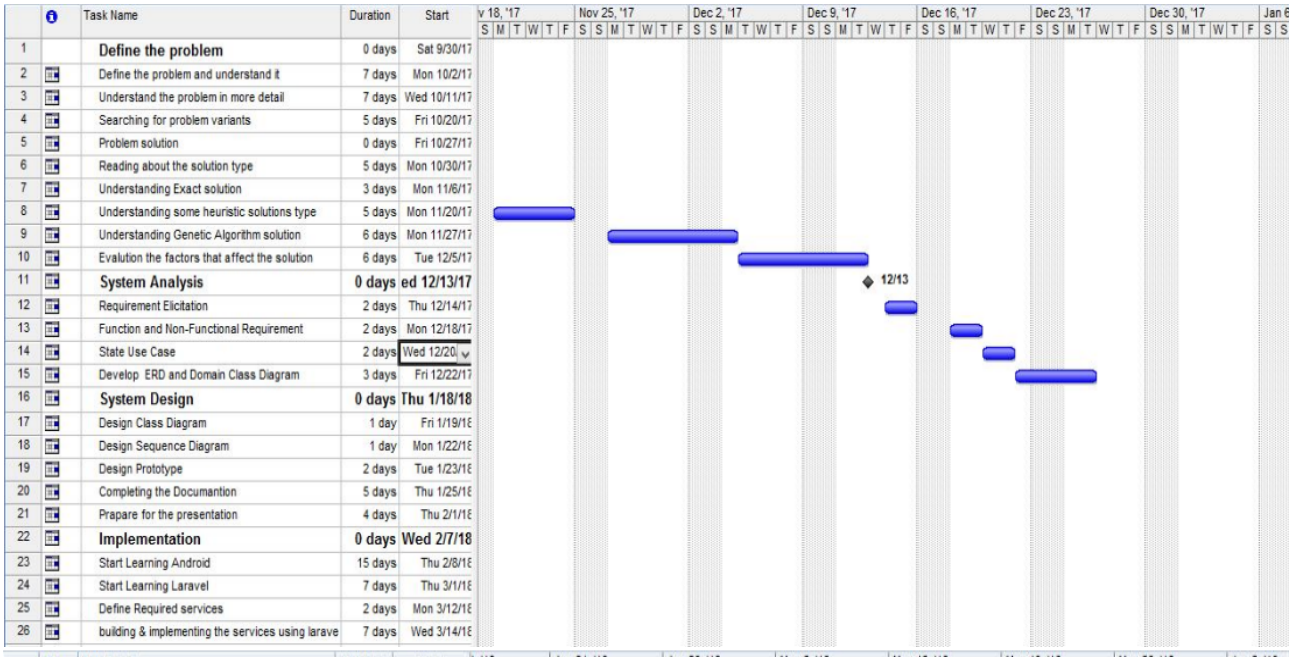
3.11 Gantt chart



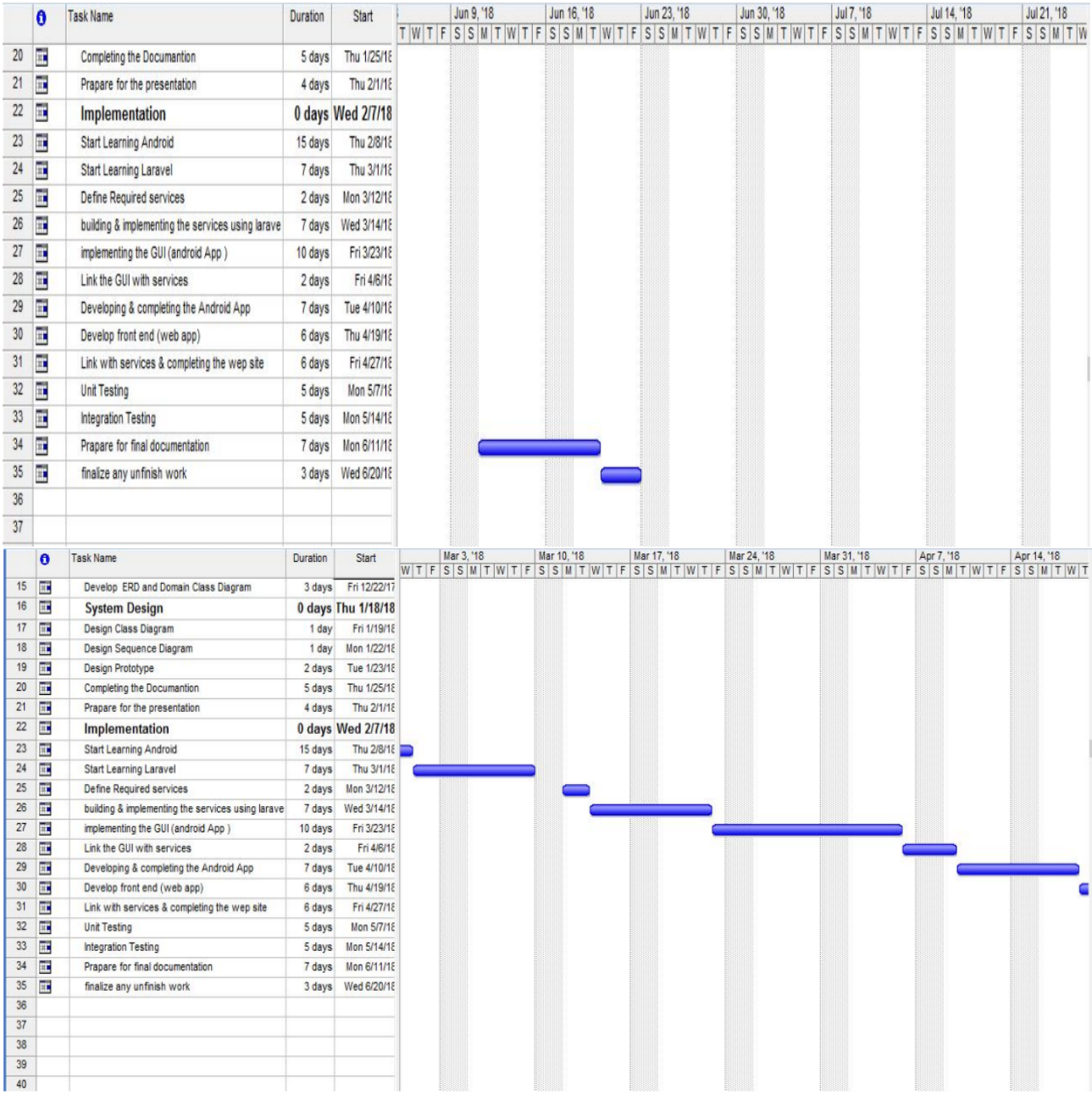
	Task Name	Duration	Start
1	**Define the problem**	0 days	Sat 9/30/17
2	Define the problem and understand it	7 days	Mon 10/2/17
3	Understand the problem in more detail	7 days	Wed 10/11/17
4	Searching for problem variants	5 days	Fri 10/20/17
5	Problem solution	0 days	Fri 10/27/17
6	Reading about the solution type	5 days	Mon 10/30/17
7	Understanding Exact solution	3 days	Mon 11/6/17
8	Understanding some heuristic solutions type	5 days	Mon 11/20/17
9	Understanding Genetic Algorithm solution	6 days	Mon 11/27/17
10	Evalution the factors that affect the solution	6 days	Tue 12/5/17
11	**System Analysis**	**0 days**	**ed 12/13/17**
12	Requirement Elicitation	2 days	Thu 12/14/17
13	Function and Non-Functional Requirement	2 days	Mon 12/18/17
14	State Use Case	2 days	Wed 12/20/17
15	Develop ERD and Domain Class Diagram	3 days	Fri 12/22/17
16	**System Design**	**0 days**	**Thu 1/18/18**
17	Design Class Diagram	1 day	Fri 1/19/18
18	Design Sequence Diagram	1 day	Mon 1/22/18
19	Design Prototype	2 days	Tue 1/23/18
20	Completing the Documantion	5 days	Thu 1/25/18
21	Prapare for the presentation	4 days	Thu 2/1/18
22	**Implementation**	**0 days**	**Wed 2/7/18**
23	Start Learning Android	15 days	Thu 2/8/18
24	Start Learning Laravel	7 days	Thu 3/1/18
25	Define Required services	2 days	Mon 3/12/18
26	building & implementing the services using larave	7 days	Wed 3/14/18



	Task Name	Duration	Start
1	**Define the problem**	0 days	Sat 9/30/17
2	Define the problem and understand it	7 days	Mon 10/2/17
3	Understand the problem in more detail	7 days	Wed 10/11/17
4	Searching for problem variants	5 days	Fri 10/20/17
5	Problem solution	0 days	Fri 10/27/17
6	Reading about the solution type	5 days	Mon 10/30/17
7	Understanding Exact solution	3 days	Mon 11/6/17
8	Understanding some heuristic solutions type	5 days	Mon 11/20/17
9	Understanding Genetic Algorithm solution	6 days	Mon 11/27/17
10	Evalution the factors that affect the solution	6 days	Tue 12/5/17
11	**System Analysis**	**0 days**	**ed 12/13/17**
12	Requirement Elicitation	2 days	Thu 12/14/17
13	Function and Non-Functional Requirement	2 days	Mon 12/18/17
14	State Use Case	2 days	Wed 12/20/
15	Develop ERD and Domain Class Diagram	3 days	Fri 12/22/17
16	**System Design**	**0 days**	**Thu 1/18/18**
17	Design Class Diagram	1 day	Fri 1/19/18
18	Design Sequence Diagram	1 day	Mon 1/22/18
19	Design Prototype	2 days	Tue 1/23/18
20	Completing the Documantion	5 days	Thu 1/25/18
21	Prapare for the presentation	4 days	Thu 2/1/18
22	**Implementation**	**0 days**	**Wed 2/7/18**
23	Start Learning Android	15 days	Thu 2/8/18
24	Start Learning Laravel	7 days	Thu 3/1/18
25	Define Required services	2 days	Mon 3/12/18
26	building & implementing the services using larave	7 days	Wed 3/14/18



	Task Name	Duration	Start
1	**Define the problem**	0 days	Sat 9/30/17
2	Define the problem and understand it	7 days	Mon 10/2/17
3	Understand the problem in more detail	7 days	Wed 10/11/17
4	Searching for problem variants	5 days	Fri 10/20/17
5	Problem solution	0 days	Fri 10/27/17
6	Reading about the solution type	5 days	Mon 10/30/17
7	Understanding Exact solution	3 days	Mon 11/6/17
8	Understanding some heuristic solutions type	5 days	Mon 11/20/17
9	Understanding Genetic Algorithm solution	6 days	Mon 11/27/17
10	Evalution the factors that affect the solution	6 days	Tue 12/5/17
11	**System Analysis**	**0 days**	**ed 12/13/17**
12	Requirement Elicitation	2 days	Thu 12/14/17
13	Function and Non-Functional Requirement	2 days	Mon 12/18/17
14	State Use Case	2 days	Wed 12/20
15	Develop ERD and Domain Class Diagram	3 days	Fri 12/22/17
16	**System Design**	**0 days**	**Thu 1/18/18**
17	Design Class Diagram	1 day	Fri 1/19/18
18	Design Sequence Diagram	1 day	Mon 1/22/18
19	Design Prototype	2 days	Tue 1/23/18
20	Completing the Documantion	5 days	Thu 1/25/18
21	Prapare for the presentation	4 days	Thu 2/1/18
22	**Implementation**	**0 days**	**Wed 2/7/18**
23	Start Learning Android	15 days	Thu 2/8/18
24	Start Learning Laravel	7 days	Thu 3/1/18
25	Define Required services	2 days	Mon 3/12/18
26	building & implementing the services using larave	7 days	Wed 3/14/18
27	implementing the GUI (android App)	10 days	Fri 3/23/18
28	Link the GUI with services	2 days	Fri 4/6/18
29	Developing & completing the Android App	7 days	Tue 4/10/18
30	Develop front end (web app)	6 days	Thu 4/19/18
31	Link with services & completing the wep site	6 days	Fri 4/27/18
32	Unit Testing	5 days	Mon 5/7/18
33	Integration Testing	5 days	Mon 5/14/18
34	Prapare for final documentation	7 days	Mon 6/11/18
35	finalize any unfinish work	3 days	Wed 6/20/18



	Task Name	Duration	Start
20	Completing the Documantion	5 days	Thu 1/25/18
21	Prapare for the presentation	4 days	Thu 2/1/18
22	**Implementation**	**0 days**	**Wed 2/7/18**
23	Start Learning Android	15 days	Thu 2/8/18
24	Start Learning Laravel	7 days	Thu 3/1/18
25	Define Required services	2 days	Mon 3/12/18
26	building & implementing the services using larave	7 days	Wed 3/14/18
27	implementing the GUI (android App)	10 days	Fri 3/23/18
28	Link the GUI with services	2 days	Fri 4/6/18
29	Developing & completing the Android App	7 days	Tue 4/10/18
30	Develop front end (web app)	6 days	Thu 4/19/18
31	Link with services & completing the wep site	6 days	Fri 4/27/18
32	Unit Testing	5 days	Mon 5/7/18
33	Integration Testing	5 days	Mon 5/14/18
34	Prapare for final documentation	7 days	Mon 6/11/18
35	finalize any unfinish work	3 days	Wed 6/20/18
36			
37			

	Task Name	Duration	Start
15	Develop ERD and Domain Class Diagram	3 days	Fri 12/22/17
16	**System Design**	**0 days**	**Thu 1/18/18**
17	Design Class Diagram	1 day	Fri 1/19/18
18	Design Sequence Diagram	1 day	Mon 1/22/18
19	Design Prototype	2 days	Tue 1/23/18
20	Completing the Documantion	5 days	Thu 1/25/18
21	Prapare for the presentation	4 days	Thu 2/1/18
22	**Implementation**	**0 days**	**Wed 2/7/18**
23	Start Learning Android	15 days	Thu 2/8/18
24	Start Learning Laravel	7 days	Thu 3/1/18
25	Define Required services	2 days	Mon 3/12/18
26	building & implementing the services using larave	7 days	Wed 3/14/18
27	implementing the GUI (android App)	10 days	Fri 3/23/18
28	Link the GUI with services	2 days	Fri 4/6/18
29	Developing & completing the Android App	7 days	Tue 4/10/18
30	Develop front end (web app)	6 days	Thu 4/19/18
31	Link with services & completing the wep site	6 days	Fri 4/27/18
32	Unit Testing	5 days	Mon 5/7/18
33	Integration Testing	5 days	Mon 5/14/18
34	Prapare for final documentation	7 days	Mon 6/11/18
35	finalize any unfinish work	3 days	Wed 6/20/18
36			
37			
38			
39			
40			

4 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.

5 References

References

[1] TSP. en.wikipedia.org/wiki/Travelling_salesman_problem

[2] TSP. people.ku.edu/ lmartin/courses/math105-F11/Lectures/chapter6-part4.pdf

[3] Christofides Algorithm. personal.vu.nl/r.a.sitters/AdvancedAlgorithms/2016/SlidesChapter2-2016.pdf

[4] Christofides Algorithm. en.wikipedia.org/wiki/Christofides_algorithm

[5] Held Karp Algorithm. en.wikipedia.org/wiki/Held–Karp_algorithm

[6] Held Karp Algorithm. www.geeksforgeeks.org/travelling-salesman-problem-set-1/

[7] Held Karp Algorithm. www.youtube.com/watch?v=-JjA4BLQyqE

[8] Heuristic Algorithms. https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf