

**Faculty of Computers and Information**

**Cairo University**

**Intelligent**

**Transportation System**

**Graduation Project presented to: Computer Science Department**

**Prepared By:**

|  |  |
| --- | --- |
| **Name** | **ID** |
| **Omar Abdul-Aziz Saeed** | **20140377** |
| **Hamed Mohamed Abdullah** | **20140375** |
| **Ziad Aamer Abdualhai** | **20140341** |
| **Asseel Ahmed** | **20140367** |
| **Mahmoud Mossad** | **20140256** |

**Supervised By: Dr.Emad Nabil**

# Contents

[Contents 1](#_Toc518151097)

[**1** Introduction 1](#_Toc518151098)

[**1.1** Introduction to the main area of the project: 1](#_Toc518151099)

[**1.2** Motivation: 1](#_Toc518151100)

[**1.3** Problem definition: 1](#_Toc518151101)

[**1.4** The Problem variants 2](#_Toc518151102)

[**1.4.1** Symmetric (STSP): 2](#_Toc518151103)

[**1.4.2** Asymmetric (ATSP) 2](#_Toc518151104)

[**1.4.3** The online TSP 2](#_Toc518151105)

[**1.4.4** The price collecting TSP 2](#_Toc518151106)

[**1.4.5** Bus, truck, vehicle routing 2](#_Toc518151107)

[**1.4.6** Edge/arc and node routing with capacities 2](#_Toc518151108)

[**1.5** Project Objective (suggested solution) 3](#_Toc518151109)

[**1.5.1** Exact Approach 3](#_Toc518151110)

[**1.5.2** Heuristic Approach 5](#_Toc518151111)

[**1.5.3** Meta Heuristic Approach 13](#_Toc518151112)

[**1.5.4** TSP Solver Modules 14](#_Toc518151113)

[**1.6** Gantt chart of project time plan 16](#_Toc518151114)

[**1.7** Project development methodology 19](#_Toc518151115)

[**1.8** The used tools in the project (SW and HW) 20](#_Toc518151116)

[**1.9** Report Organization (summary of the rest of the report) 21](#_Toc518151117)

[**2** Related work 22](#_Toc518151118)

[**2.1** Examples of other Similar Projects 22](#_Toc518151119)

[**2.1.1** Opti-Map 22](#_Toc518151120)

[**2.1.2** Concorde TSP solver 22](#_Toc518151121)

[**2.1.3** Google Operations Research C++ API 23](#_Toc518151122)

[**2.1.4** Driving Route Planner 24](#_Toc518151123)

[**2.1.5** TSPSG 24](#_Toc518151124)

[**3** System Analysis 25](#_Toc518151125)

[**3.1** Project specifications 25](#_Toc518151126)

[**3.1.1** Stakeholders 25](#_Toc518151127)

[**3.1.2** Functional requirement 25](#_Toc518151128)

[**3.1.3** Non-functional requirement 26](#_Toc518151129)

[**3.2** Use case Diagrams 27](#_Toc518151130)

[**3.3** Use Case Tables 28](#_Toc518151131)

[**3.3.1** Customer 28](#_Toc518151132)

[**3.3.2** Salesman 31](#_Toc518151133)

[**3.3.3** Admin 33](#_Toc518151134)

[**4** System Design: 35](#_Toc518151135)

[**4.1** System Component Diagram: 35](#_Toc518151136)

[**4.2** System Architecture 36](#_Toc518151137)

[**4.3** System Class Diagrams 37](#_Toc518151138)

[**4.4** Sequence Diagrams 39](#_Toc518151139)

[**4.4.1** Add Order 39](#_Toc518151140)

[**4.4.2** Edit order 40](#_Toc518151141)

[**4.4.3** Reject Request 41](#_Toc518151142)

[**4.4.4** Deliver Order 42](#_Toc518151143)

[**4.4.5** 3.5 Run TSP Solver: 43](#_Toc518151144)

[**4.4.6** Check In Availability 44](#_Toc518151145)

[**4.5** Project ERD (Entity Relationship Diagram) 45](#_Toc518151146)

[**4.6** System GUI Design 46](#_Toc518151147)

[**4.6.1** Admin (Web App) 46](#_Toc518151148)

[**4.6.2** Customer (Android) 47](#_Toc518151149)

[**4.6.3** Salesman 49](#_Toc518151150)

[**5** Implementation and Testing 51](#_Toc518151151)

[**5.1** Unit Testing 51](#_Toc518151152)

[**5.1.1** Admin Controller 51](#_Toc518151153)

[**5.1.2** Customer Controller 51](#_Toc518151154)

[**5.1.3** Product Controller 52](#_Toc518151155)

[**5.1.4** Request Controller 53](#_Toc518151156)

[**5.1.5** 53](#_Toc518151157)

[**5.1.6** Route controller 54](#_Toc518151158)

[**5.1.7** Salesman controller 54](#_Toc518151159)

[**5.1.8** User Controller 55](#_Toc518151160)

[**5.1.9** TSP Solver 55](#_Toc518151161)

[**5.2** System testing 56](#_Toc518151162)

[**5.2.1** Admin 56](#_Toc518151163)

[**5.2.2** Customer 58](#_Toc518151164)

[**5.2.3** Salesman 59](#_Toc518151165)

[**6** Conclusion 60](#_Toc518151166)

[**7** References 61](#_Toc518151167)

**List of Figures:**

[Figure 1 Travelling Salesman Problem 1](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152601)

[Figure 2 - Brute Force Algorithm 3](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152602)

[Figure 3 - Dynamic Programming Algorithm 4](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152603)

[Figure 4 - Nearest Neighbor Algorithm 6](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152604)

[Figure 5 - Cheapest Link Algorithm 7](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152605)

[Figure 6 - Depth first tree tour 8](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152606)

[Figure 7 - Depth First tree tour example 1 9](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152607)

[Figure 8 - Depth First tree tour example 2 9](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152608)

[Figure 9 - Depth First tree tour example 3 10](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152609)

[Figure 10 - Depth First tree tour example 4 10](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152610)

[Figure 11 - OPT Impovement 11](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152611)

[Figure 12 - OPT Improvement example 1 12](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152612)

[Figure 13 - OPT Improvement example 2 12](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152613)

[Figure 14 - OPT Improvement example 3 12](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152614)

[Figure 15 - OPT Improvement example 4 13](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152615)

[Figure 16 - Genetic Algorithm 14](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152616)

[Figure 17 - Clustering Algorithm 15](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152617)

[Figure 18 - Waterfall System development methodology 19](#_Toc518152618)

[Figure 19 - Use case diagram 27](#_Toc518152619)

[Figure 20 - System Component diagram 35](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152620)

[Figure 21 - System Architecture diagram 36](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152621)

[Figure 22 - class diagram 1 37](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152622)

[Figure 23 - class diagram 2 38](#_Toc518152623)

**List of Tables:**

[Table 1 - Exponential Time scaling with # of cities 5](#_Toc518152517)

[Table 2 - Gantt chart 1 16](#_Toc518152518)

[Table 3 - Gantt chart 2 17](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152519)

[Table 4 - Gantt chart 3 18](#_Toc518152520)

[Table 5 - Customer add order 28](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152521)

[Table 6 - Customer submit order 29](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152522)

[Table 7 - Customer edit orders 30](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152523)

[Table 8 - Salesman Deliver request 31](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152524)

[Table 9 - Salesman Check in 32](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152525)

[Table 10 - Admin Reject request 33](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152526)

[Table 11 - Admin Run TSP Solver 34](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152527)

[Table 12 - Admin controller login 51](#_Toc518152528)

[Table 13 - Customer Signup 51](#_Toc518152529)

[Table 14 - Product controller 52](#_Toc518152530)

[Table 15 - Request Controller 53](#_Toc518152531)

[Table 16 - Route Controller 54](#_Toc518152532)

[Table 17 - Salesman controller 54](#_Toc518152533)

[Table 18 - User Contoller 55](#_Toc518152534)

[Table 19 - TSP solver 55](#_Toc518152535)

[Table 20 - Poster 62](file:///E:\UN\Graduation%20Project\slides\GP%20Final\Documentation.docx#_Toc518152536)

# Introduction

## Introduction to the main area of the project:

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 salesmen should deliver orders.

## 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 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.

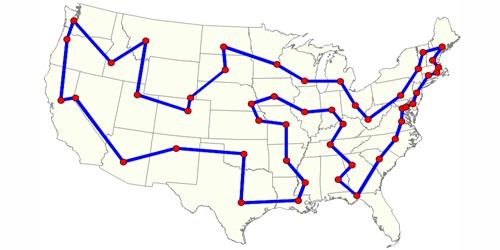


Figure 1 Travelling Salesman Problem

## The Problem variants

### 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.

### Asymmetric (ATSP)

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

### 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.

### 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.

### Bus, truck, vehicle routing

Asks "What is the optimal set of routes for vehicles to traverse in order to deliver to a given set of customers?". It generalizes the well-known travelling salesman problem (TSP).

### Edge/arc and node routing with capacities

The CARP aims to traverse 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.

## Project Objective (suggested solution)

Travel Salesman Problem has more than one approach to solve it.

e.g. Exact Approach, Greedy Approach, Heuristic Approach and Genetic Algorithm Approach.

### Exact Approach

The solution is optimal but very slow.

#### *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!).

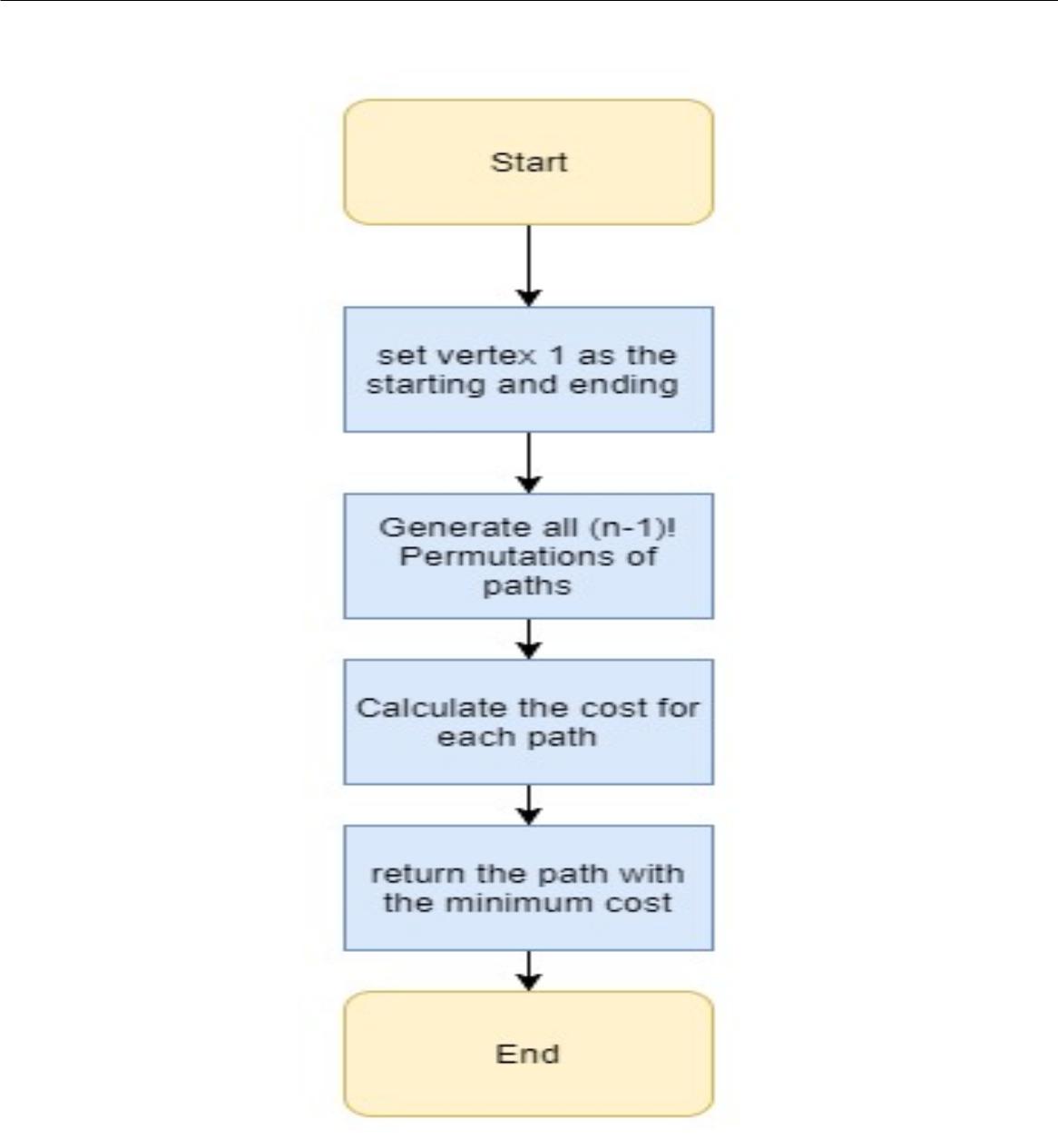


Figure 2 - Brute Force Algorithm

#### 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(2nn2).

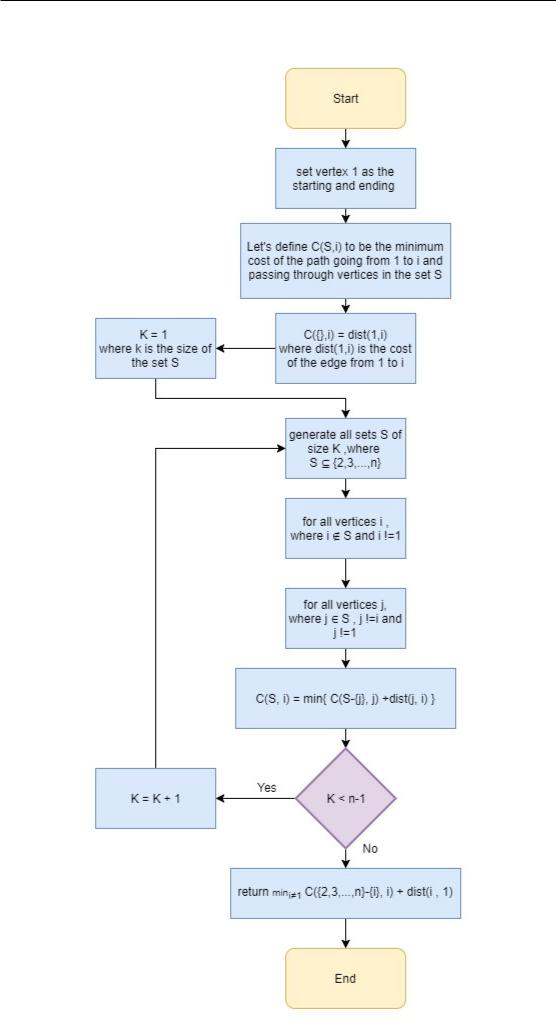
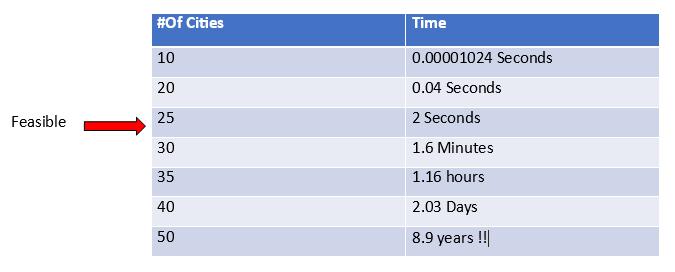


Figure 3 - Dynamic Programming Algorithm

Assuming we run the algorithm on 1010 instructions per seconds processor.

Table 1 - Exponential Time scaling with # of cities



### Heuristic Approach

Approximation: Solving the TSP optimally takes too long, in-stead uses approximation algorithms, or heuristics and can get good solutions but may not optimal.

Tour Construction: Tour construction algorithms have one thing in common, 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 performance is somewhat linked to the construction heuristic used.

#### Nearest Neighbor Algorithm (Construction)

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

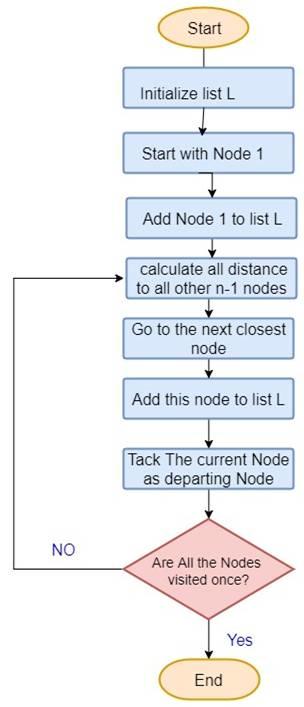


Figure 4 - Nearest Neighbor Algorithm

#### Cheapest Link (Construction)

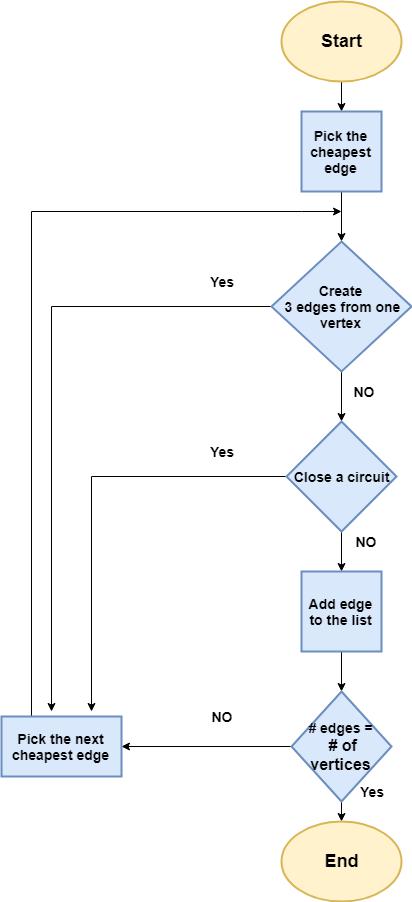


Figure 5 - Cheapest Link Algorithm

#### 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.

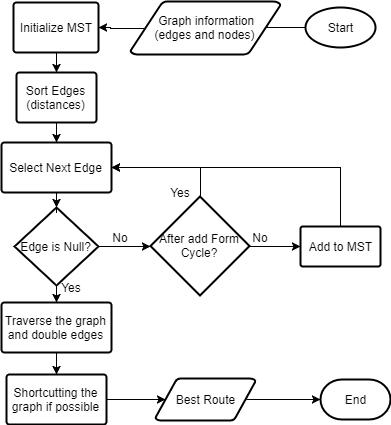


Figure 6 - Depth first tree tour

Graph Example:

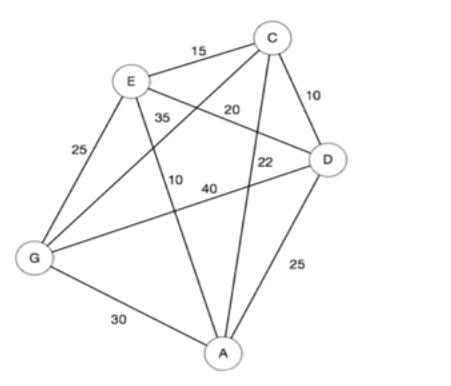


Figure 7 - Depth First tree tour example 1

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

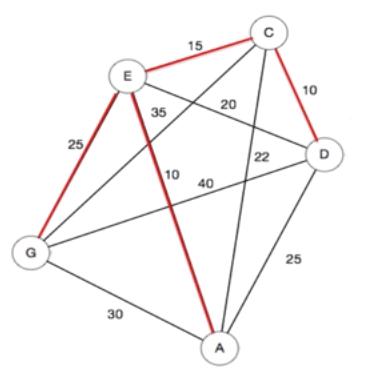


Figure 8 - Depth First tree tour example 2

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

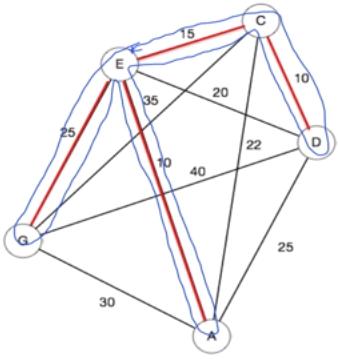


Figure 9 - Depth First tree tour example 3

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

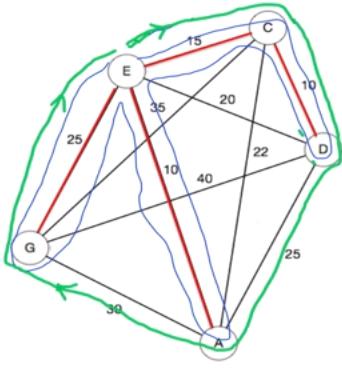
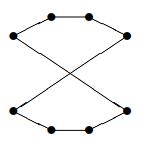


Figure 10 - Depth First tree tour example 4

#### 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 gure explain cross over 2-opt can solve it.



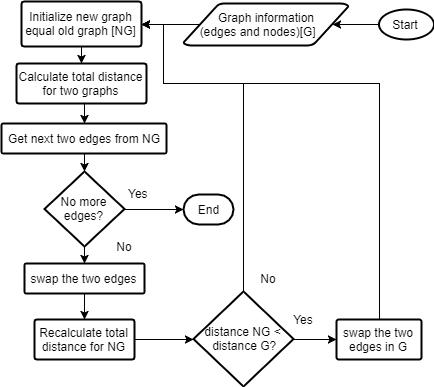
This Flowchart explain all the process.

Figure 11 - OPT Impovement

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

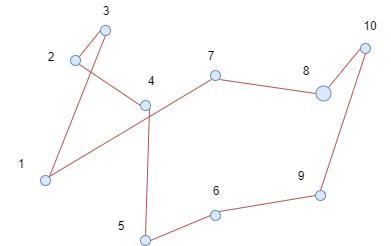


Figure 12 - OPT Improvement example 1

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

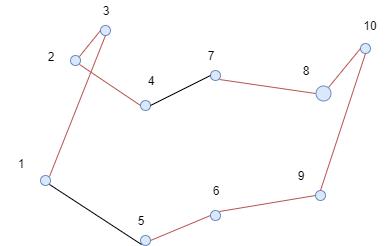


Figure 13 - OPT Improvement example 2

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

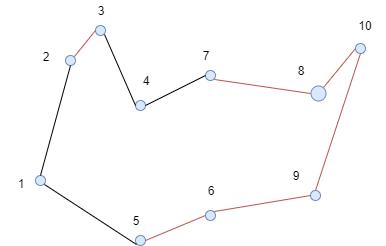


Figure 14 - OPT Improvement example 3

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

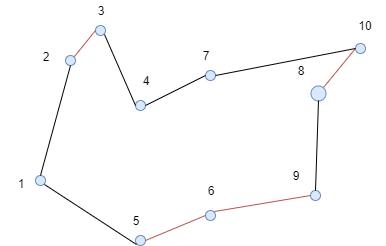


Figure 15 - OPT Improvement example 4

#### Christofide’s algorithm (Construction)

Solutions is guaranteed to be within a factor of 3/2 of the optimal solution O(n3). 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 solution, but unfortunately it is impractical because of the constraints it has.

### Meta Heuristic Approach

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

#### Genetic Algorithm

Fitness function:

n

Fi = 1= Distance (Gi; Gi+1)

i=1

Complexity: O(n3).

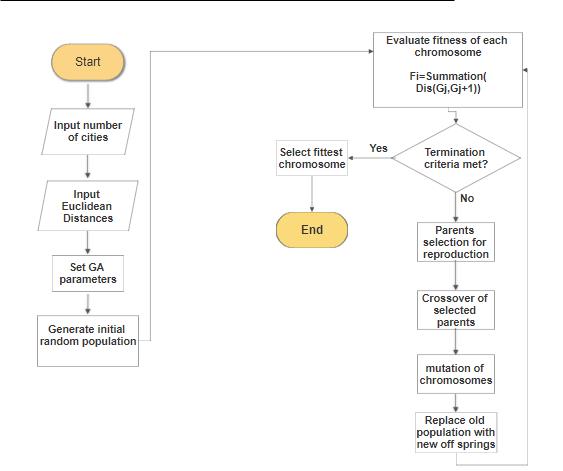


Figure 16 - Genetic Algorithm

### TSP Solver Modules

#### Clustering module:

We use Combining Clusters in the Agglomerative Approach for clustering. So, in the agglomerative hierarchical approach,

we define each data point as a cluster and combine existing clusters at each step.

we use Single Linkage approach: So, in single linkage, we define the distance between two clusters as the minimum distance between any single data point in the first cluster and any single data point in the second cluster.

On the basis of this definition of distance between clusters, at each stage of the process we combine the two clusters with the smallest single linkage distance.

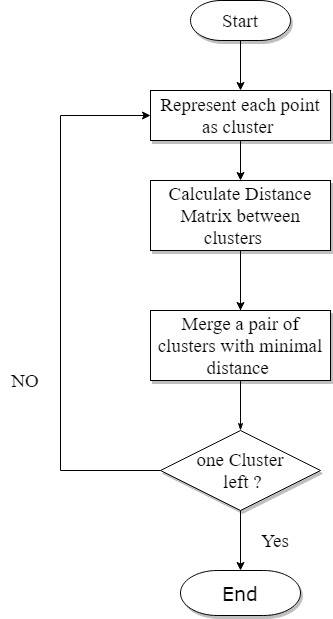


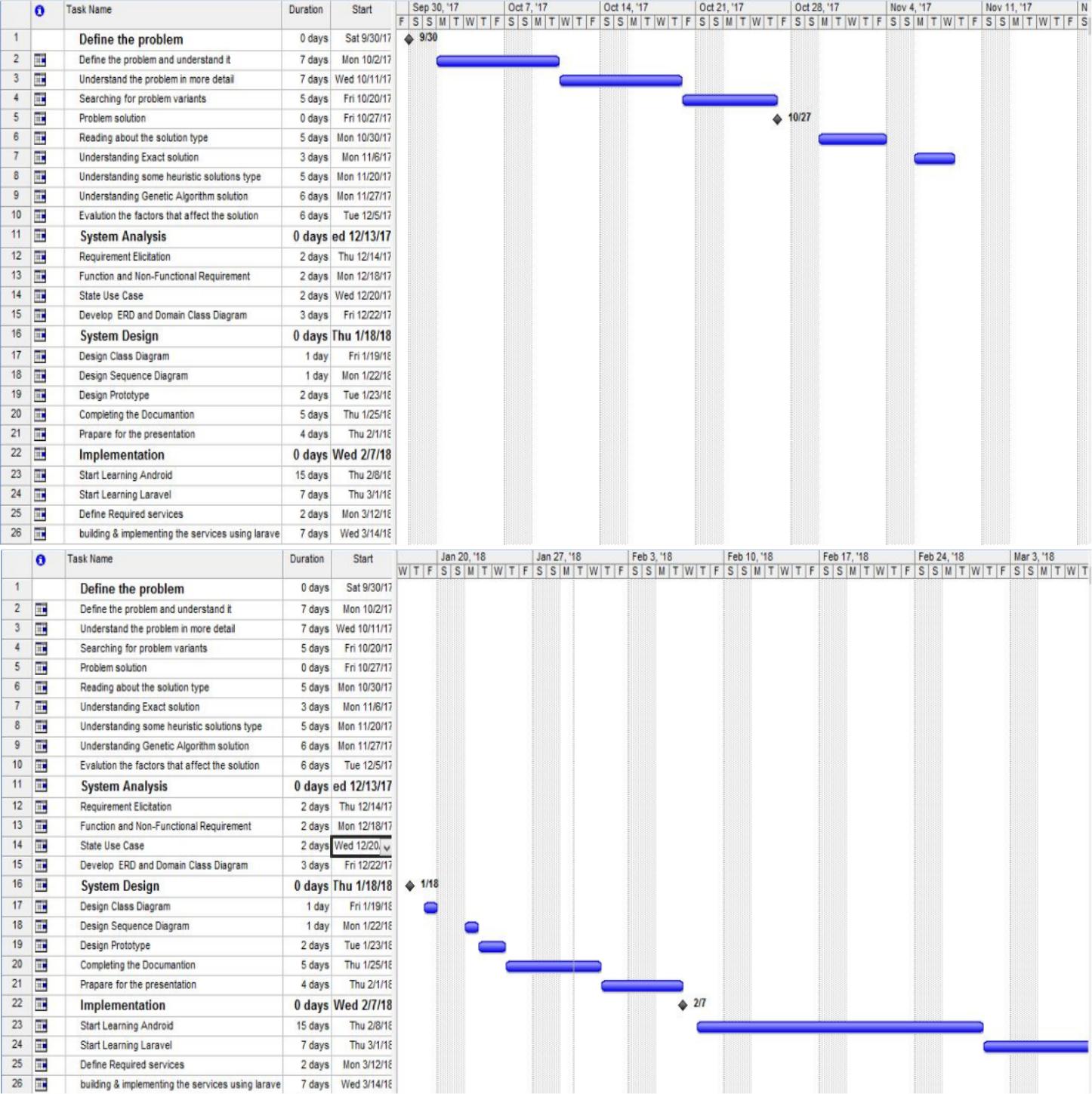
Figure 17 - Clustering Algorithm

#### 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

## Gantt chart of project time plan

Table 2 - Gantt chart 1



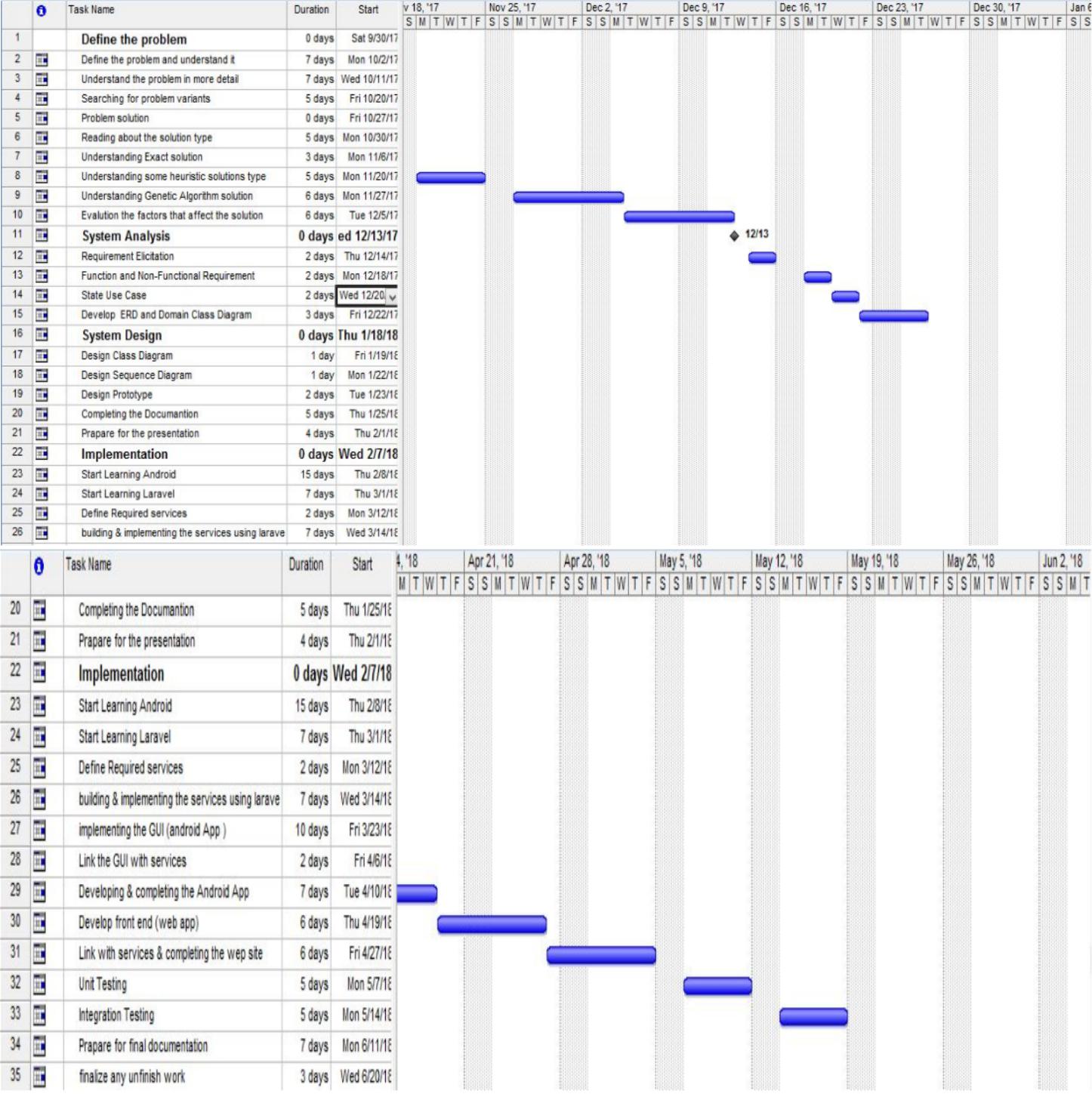
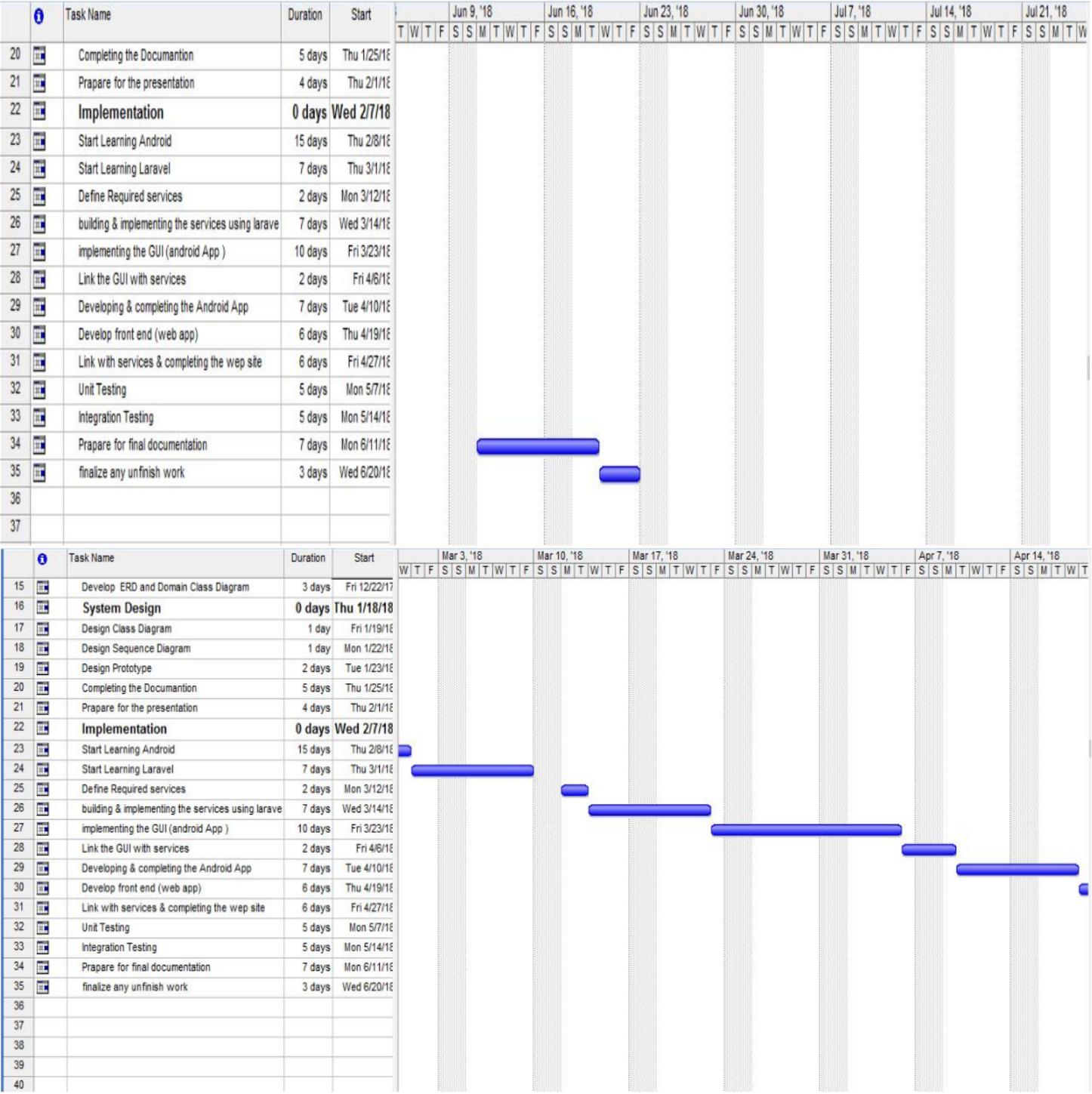


Table 3 - Gantt chart 2

Table 4 - Gantt chart 3



## Project development methodology

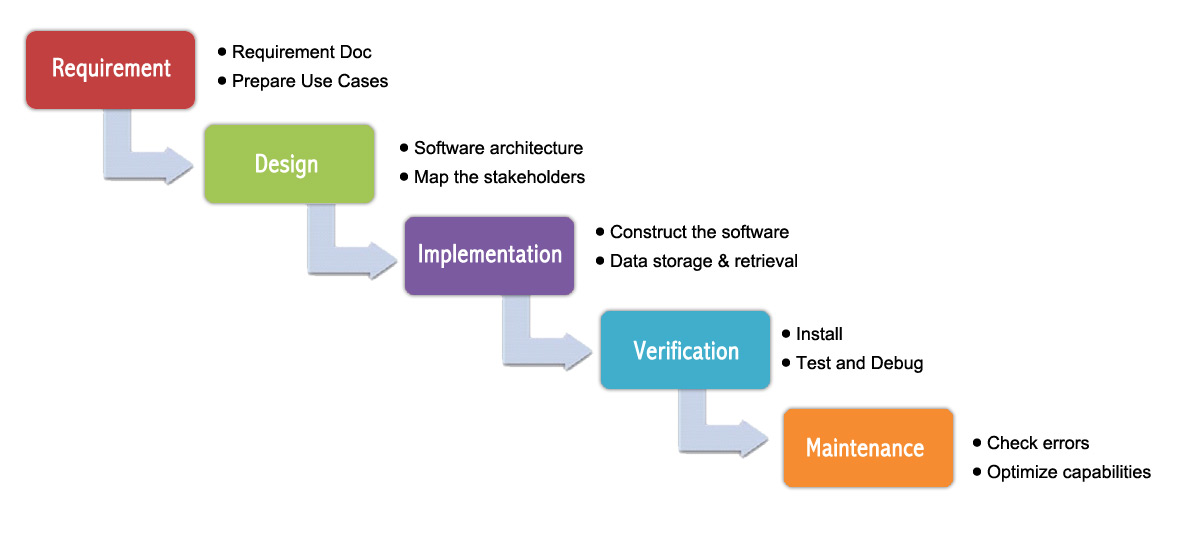


Figure 18 - Waterfall System development methodology

We’ve used Waterfall methodology the sequential phases in Waterfall model are:

* **Requirement Gathering**: in this phase we searched and understood the problem, and we chose the algorithms that we will apply in order to solve the problem.
* **System Design**: in this phase we’ve mentioned the system stakeholders and what’s the provided functionality for each stakeholder and at the end of this phase we came up with the use case diagram, detailed class diagram, sequence diagram.
* **Implementation**: in this phase we divided the system to modules upon the stakeholders then we broke each module to units to make the implementation process less complex then we started coding and at the end of this phase we integrated the units and the modules to be one integrated system.
* **Verification**: in this phase we applied unit testing for all the services, in addition we applied system testing.
* **Maintenance**: in this phase we’ve handled all the expected inputs for each service.

## The used tools in the project (SW and HW)

* **Software tools:**
  + *Backend*:

Eclipse

Eclipse Spring Tool Suite

Postman

Xampp

Google maps

Google maps API

* + *Frontend*:

Androird Studio

Microsoft Visual code

* + *Documentation:*

StarUML

Draw.io

Word

Notepad++

* **Hardware tools:**

laptops and the internet network

## Report Organization (summary of the rest of the report)

* Chapter 2:

in this chapter we mention some of the most known projects that are similar to ours.

we start by an introduction about the project and how it calculates the TSP, then

we mention the differences between this project and ours.

* Chapter 3:

here in this chapter we introduce our stakeholders, what each one will be able to do on our system (functional req.).

Also, how we make those functions more efficient and secure (non-functional req.) and the use case diagram and description of each use case through tables.

* Chapter 4:

The System designs, overall system architecture, the functional requirements workflow designs are mentioned in this chapter through sequence diagrams.

Also, class diagram, component diagram and the GUI design of our different users of the system: Customers, Salesman and Admin GUIs.

* Chapter 5:

The system implementation and running of different cases through screenshots. Unit testing and Integration testing cases are applied with their respective inputs and output samples to test the full functionality of the system.

# Related work

## Examples of other Similar Projects

### Opti-Map

This website takes a number of location on google maps, that are reachable through driving way and calculate the fastest way to traverse them (solve the Traveling Salesman problem).

It uses the brute force algorithm as an exact algorithm and this results in the max # of cities to be 10. Also, it uses heuristics for more than 10 cities: Greedy, Ant Colony optimization (ACO) and ACO k2-opt

***Main differences:***

1. We use Held-Karp Dynamic programming algorithm as our exact algorithm which faster than brute force to enable us to go almost 25 cities for optimal solution!
2. Our heuristics: Greedy, Genetic algorithms and Cheapest link algorithm

which have good near optimal solutions.

1. Also, we use machine learning to cluster our requests locations in case of

more than one salesman using Agglomerative clustering, based on min cost.

1. We have android app which makes it easier to access and use anywhere

***References:***

<https://gebweb.net/optimap/>

<https://gebweb.net/blogpost/2007/07/05/behind-the-scenes-of-optimap/>

### Concorde TSP solver

The Concorde is a computer code for the symmetric traveling salesman problem (TSP) and some related network optimization problems. The code is written in the ANSI C programming language and it is available for academic research use.

The Concorde graphical user interface can be used to apply the Concorde TSP Solver to a specified set of cities. The Concorde solver uses the cutting-plane method, iteratively solving linear programming relaxations of the TSP.

its source is available and there is an executable program that can be downloaded for many Operating systems.

***Main differences:***

1. It solves only the symmetric traveling salesman problem where distance is equal on going to and coming from point to point.
2. it has a wide library of callable functions
3. There is a GUI desktop application that can be downloaded
4. Concorde functions are thread-safe for programming in shared-memory parallel environments.

***References:***

<http://www.math.uwaterloo.ca/tsp/concorde/index.html>

### Google Operations Research C++ API

You can solve TSPs using the OR-Tools vehicle routing library, a collection of algorithms designed especially for TSPs, and more general problems with multiple vehicles.

This library contains a set of algorithms and functions that can be use for TSP and more general problems, The vehicle routing library lets one model and solve generic vehicle routing problems ranging from the Traveling Salesman Problem to more complex problems such as the Capacitated Vehicle Routing Problem with Time Windows.

***Main Differences:***

* Not a ready to use program that can be used to solve tsp, need to be configured and setup first
* Good for not making the tsp solver from the scratch, but using already made functions and code API.

***References:***

<https://developers.google.com/optimization/routing/tsp>

<https://developers.google.com/optimization/reference/>

### Driving Route Planner

provide names of location points or the latitude/ longitude addresses and then calculate the shortest route solving the TSP.as google applying charges for more cities and restrictions, this site will remove some features and will be paid instead of free in the future.

it does not say much about the algorithms it uses.

***Main differences:***

* We use machine learning to cluster our requests locations in case of more than one salesman using Agglomerative clustering, based on min cost.
* Again, it is a website and we have android app which makes it easier to access and use anywhere

***References:***

<https://www.drivingrouteplanner.com>

### TSPSG

This software is intended to generate and solve Travelling Salesman Problem (TSP) tasks. It uses Branch and Bound method for solving.

It is a desktop application

***References:***

<https://tspsg.info/>

# System Analysis

## Project specifications

## 

### Stakeholders

A stakeholder can be internal or external.

* **Customer**

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

* **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.

* **Salesman**

Is Internal-Operational stakeholder, receives the orders that they should de-liver and ask the system to get the optimal route to follow, this stakeholder will interact with the system via Android application.

### Functional requirement

* **Customer**
  + - Order any Product including quantity and location.
    - Cancel order before distributor takes it.
* **Admin**
  + - Reject or accept an order.
    - Run TSP solver.
    - Add salesman.
* **Salesman**

- view requests

- View path.

- Check availability.

### Non-functional requirement

* **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.

* 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.

* **Reliability**

-The system has to be 100% reliable.

* **Availability**

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

* **Usability**

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

## Use case Diagrams

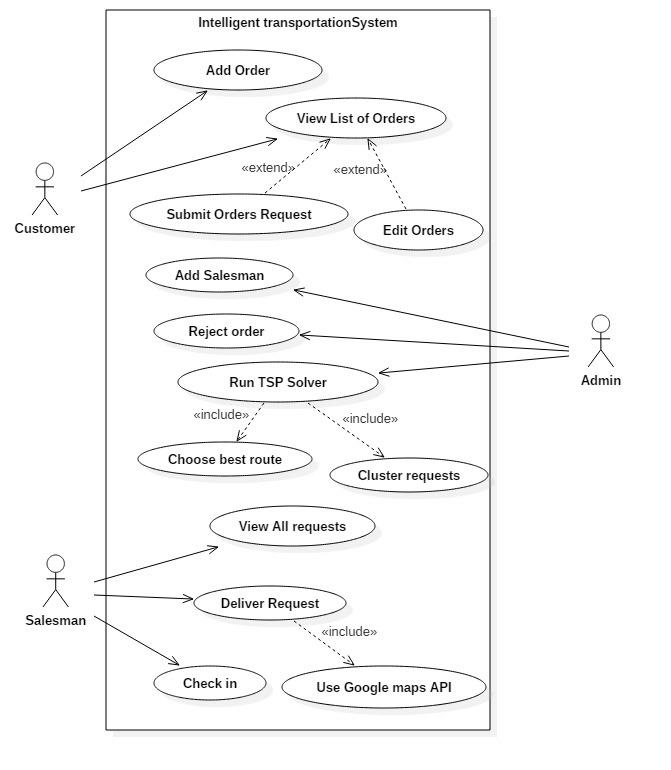


Figure 19 - Use case diagram

## Use Case Tables

### Customer

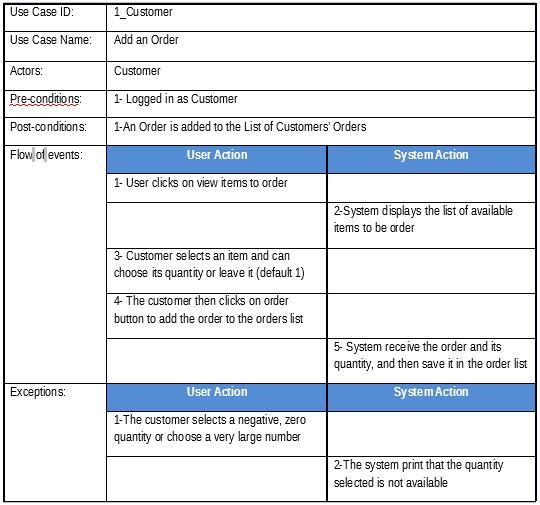


Table 5 - Customer add order

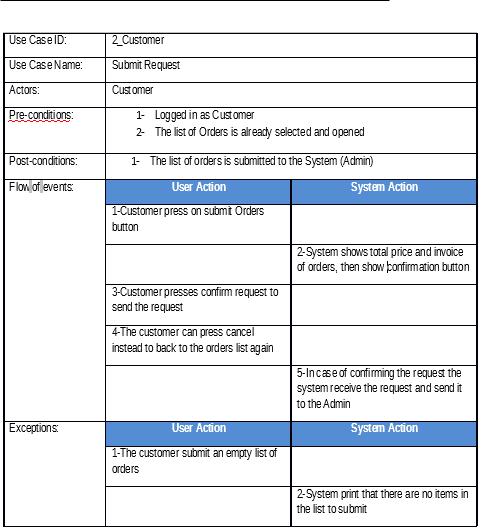


Table 6 - Customer submit order

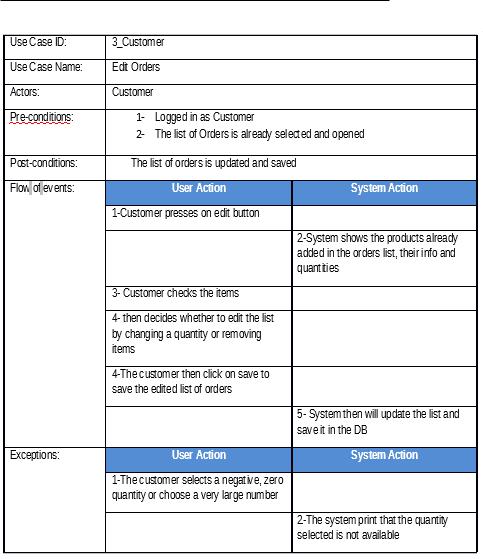


Table 7 - Customer edit orders

### Salesman

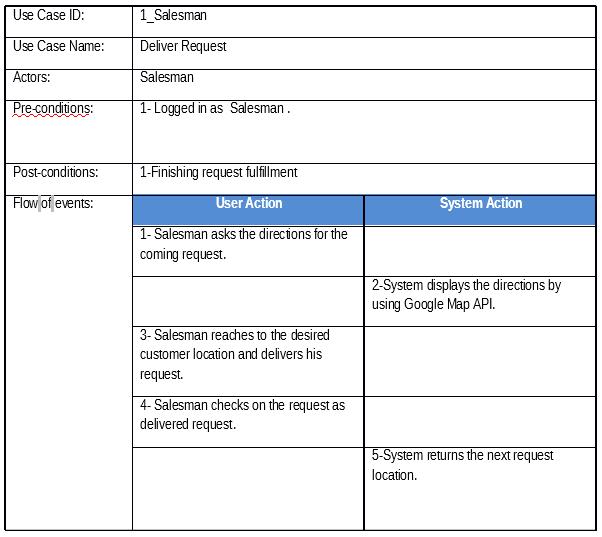


Table 8 - Salesman Deliver request

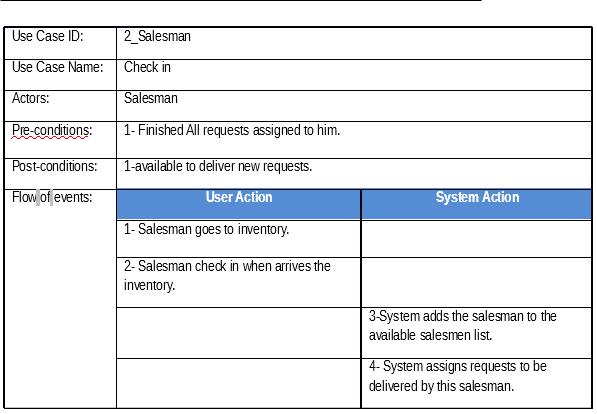


Table 9 - Salesman Check in

### Admin

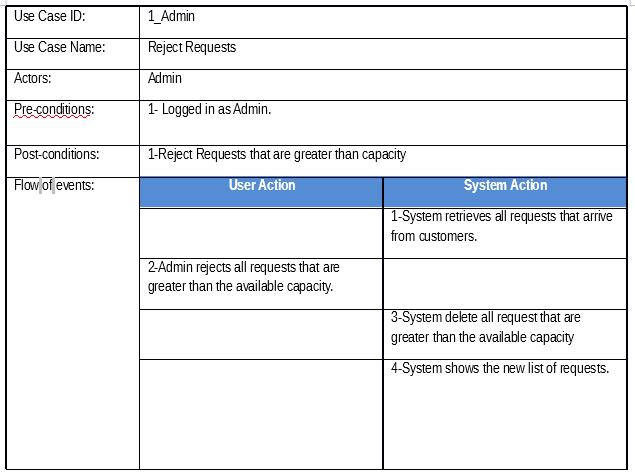


Table 10 - Admin Reject request

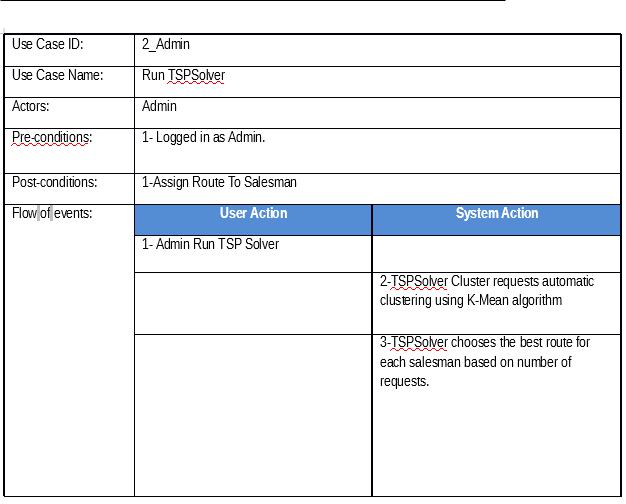


Table 11 - Admin Run TSP Solver

# System Design:

## System Component Diagram:

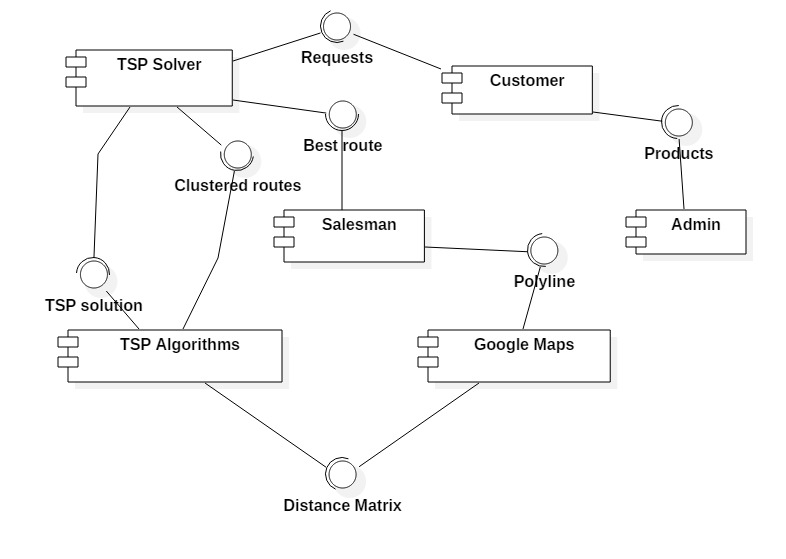


Figure 20 - System Component diagram

## System Architecture

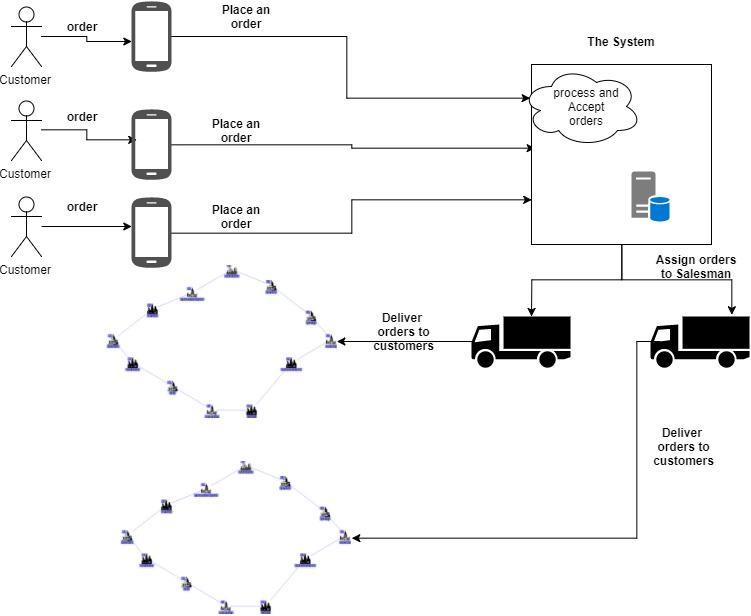


Figure 21 - System Architecture diagram

## System Class Diagrams

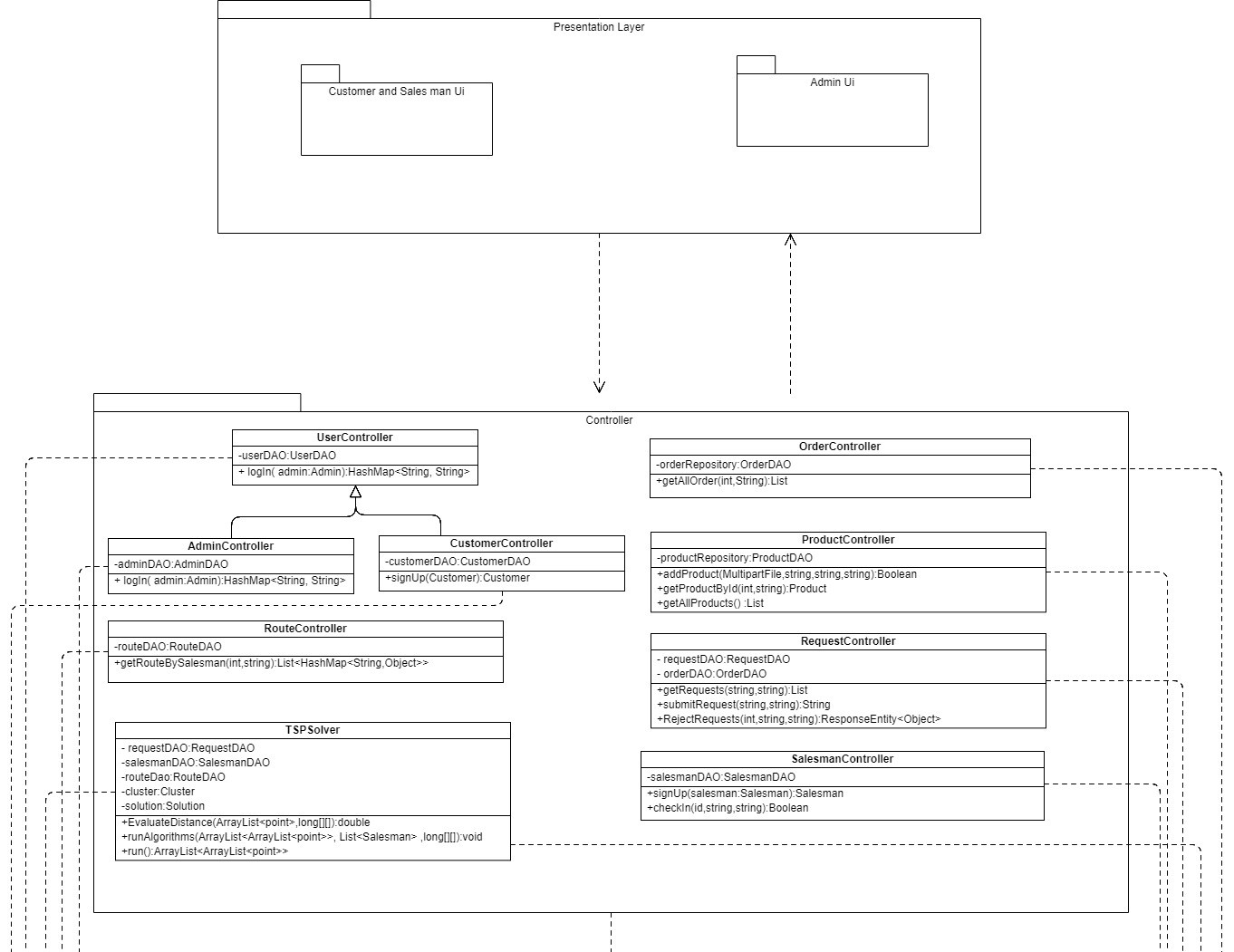


Figure 22 - class diagram 1

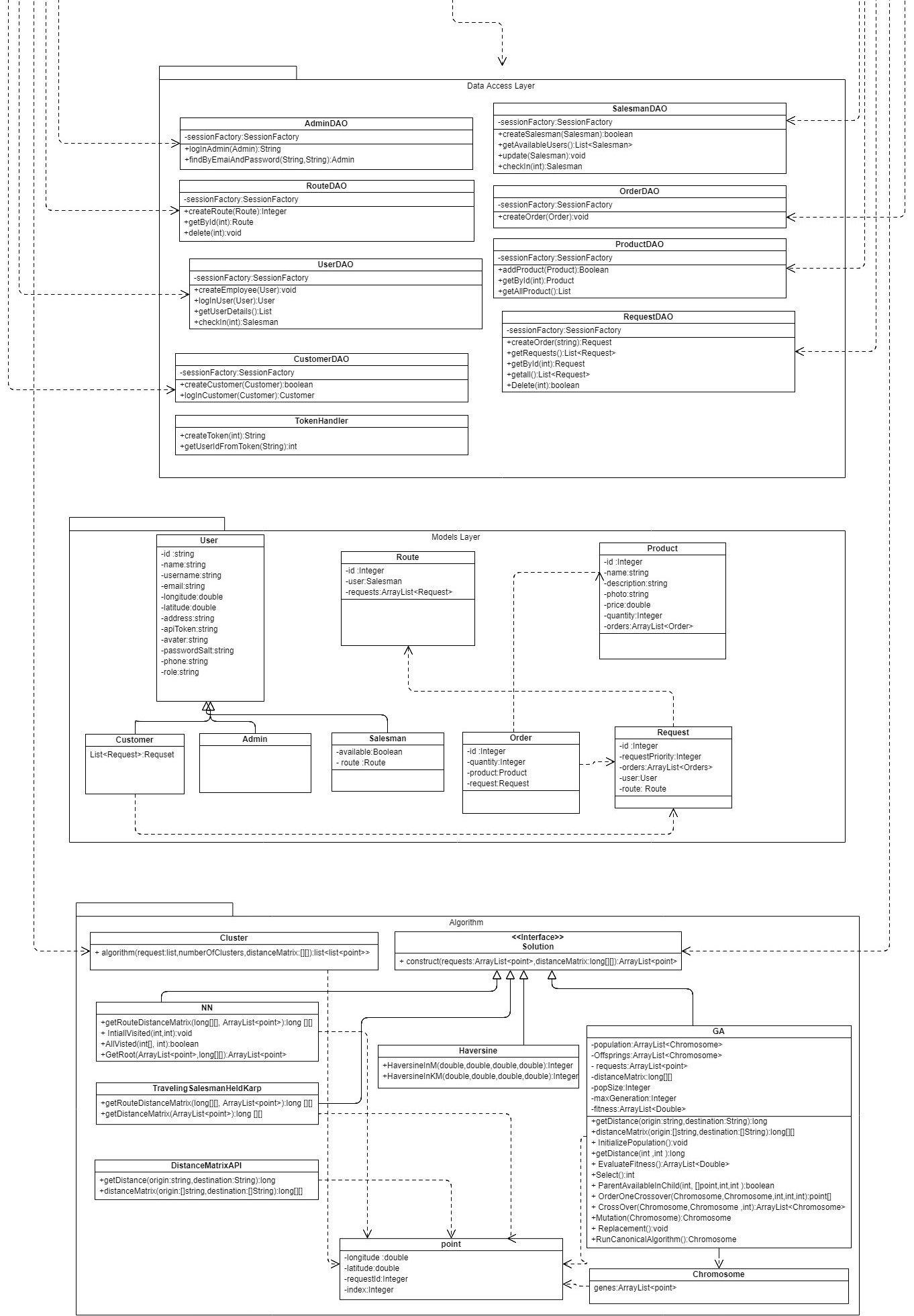


Figure 23 - class diagram 2

## Sequence Diagrams

### Add Order

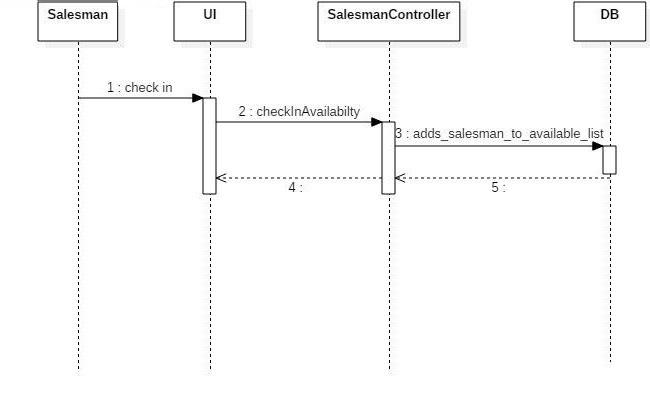
### Edit order

### Reject Request

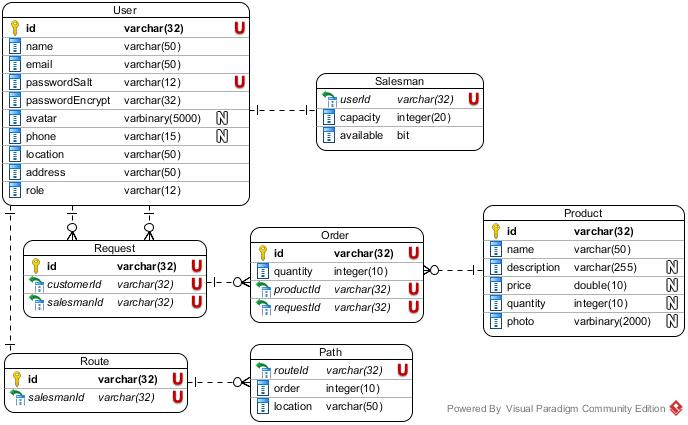
### Deliver Order

### 3.5 Run TSP Solver:

### Check In Availability

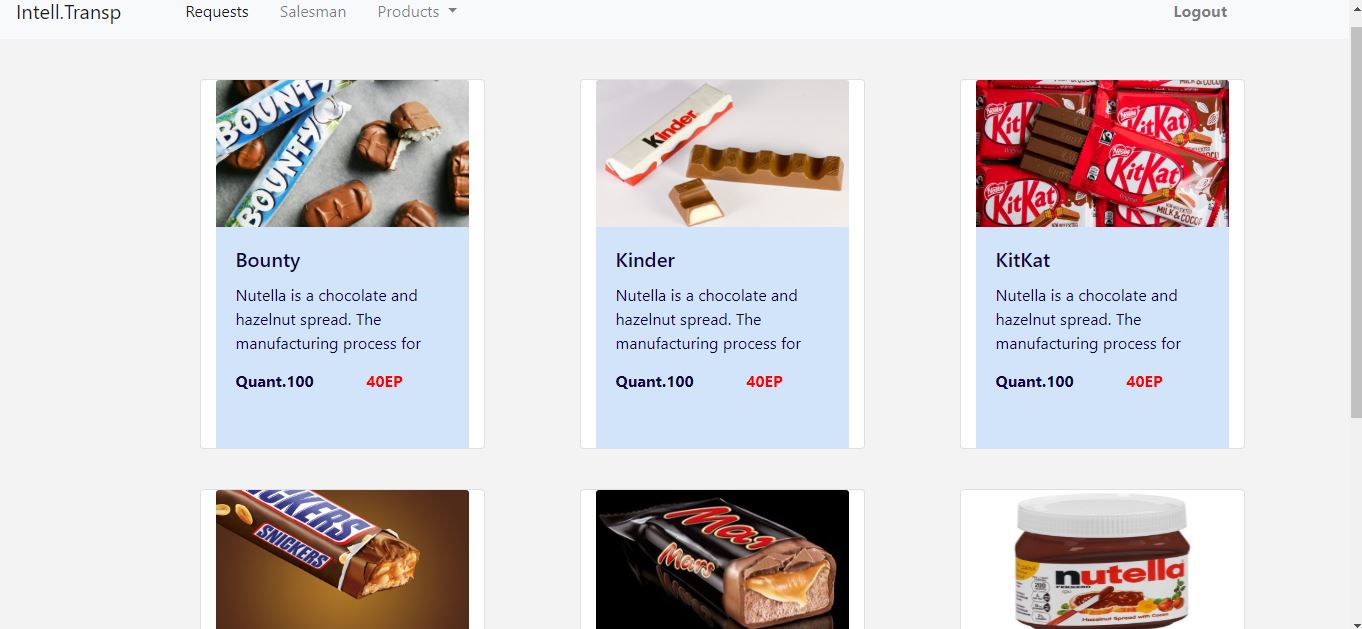


## Project ERD (Entity Relationship Diagram)

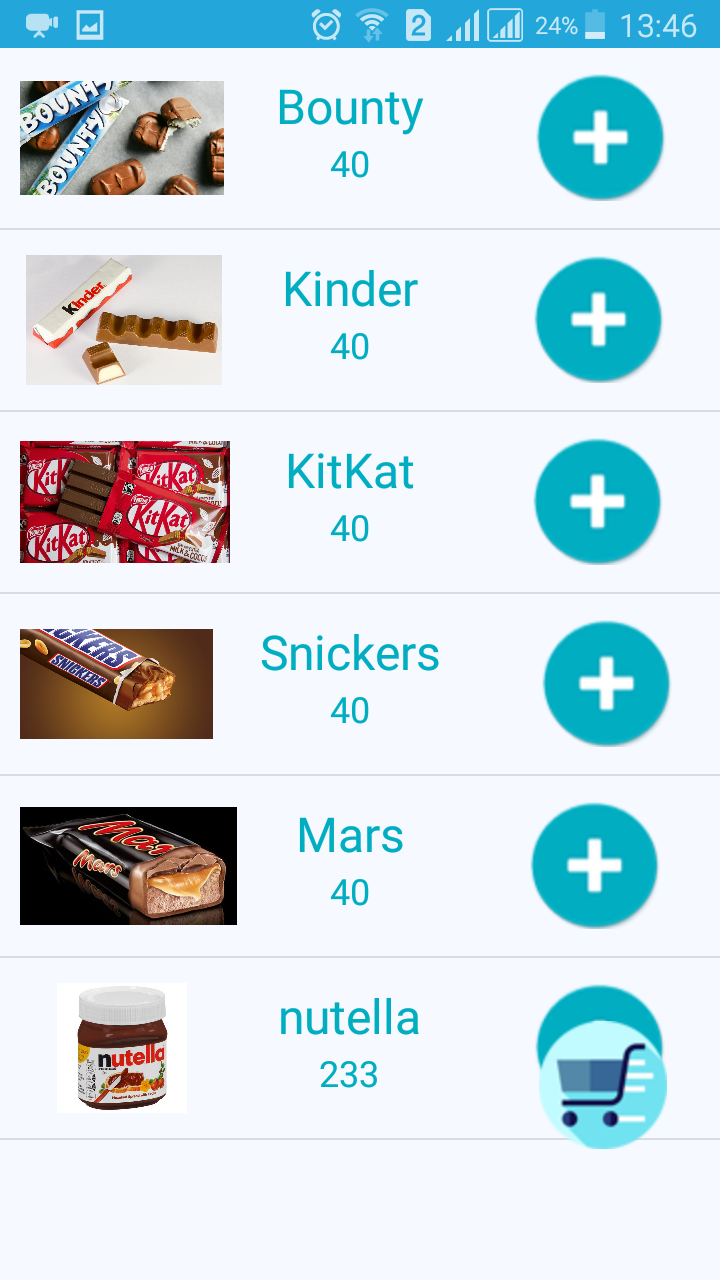
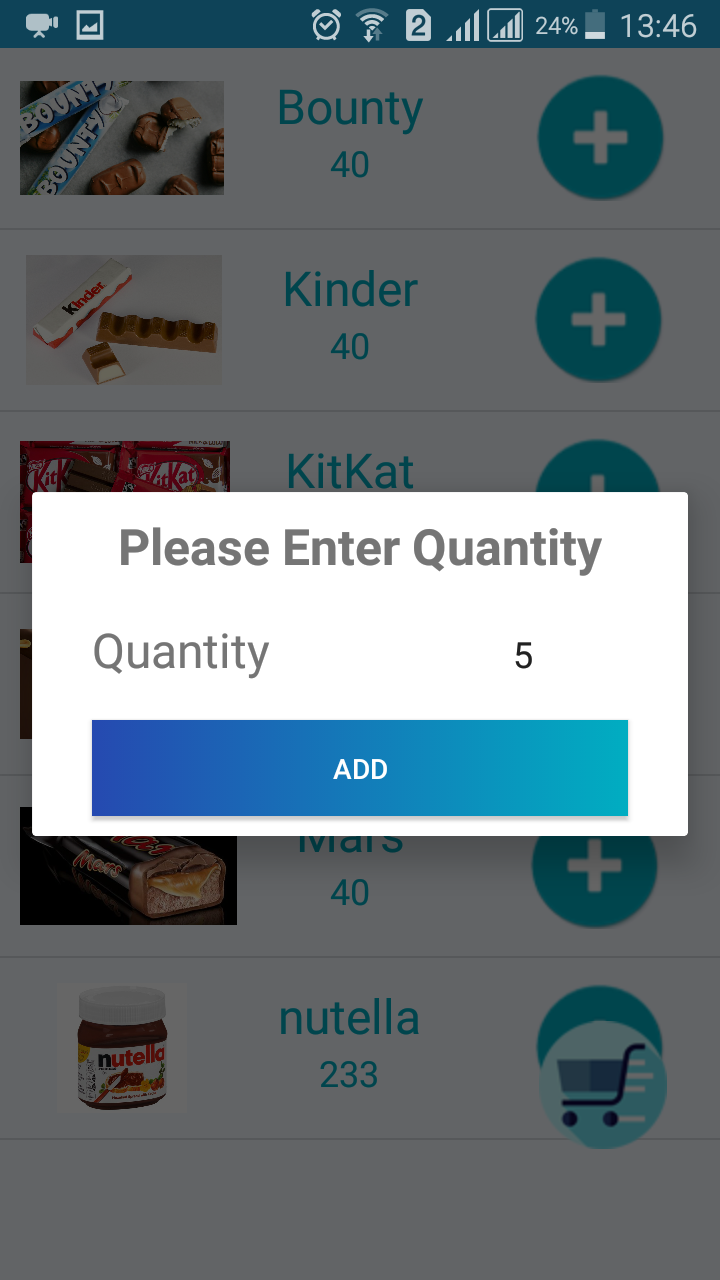


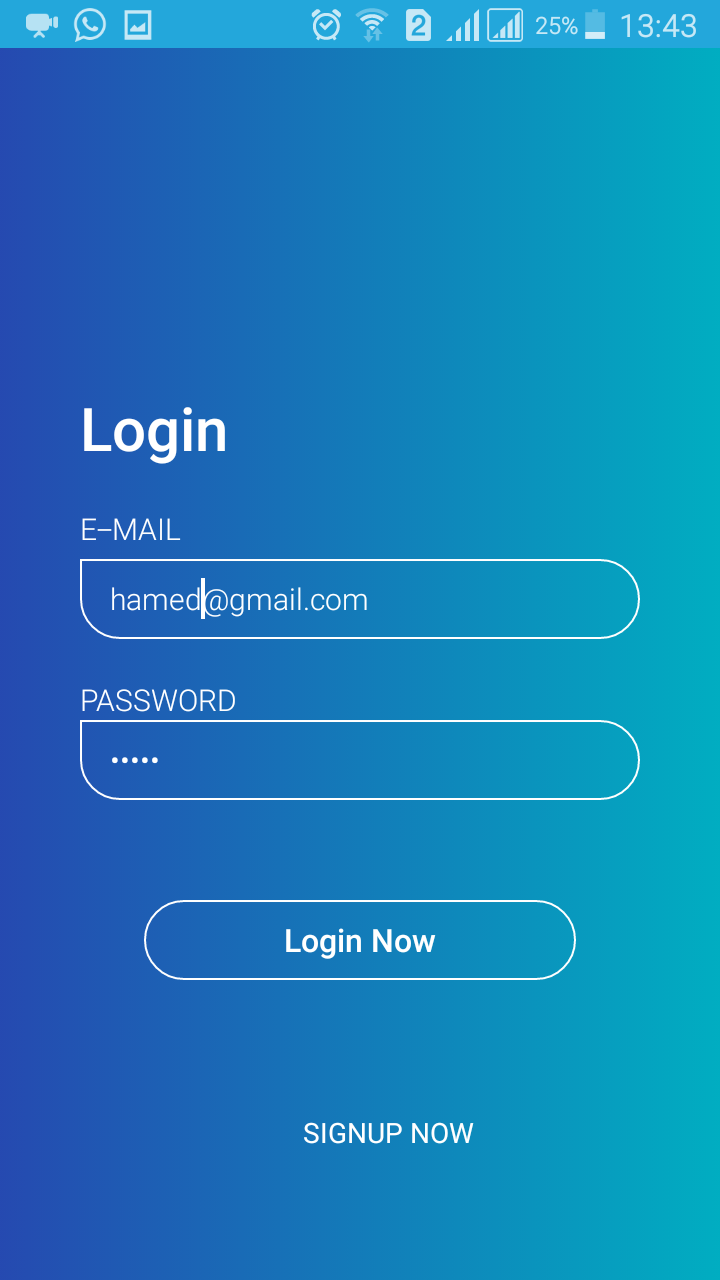
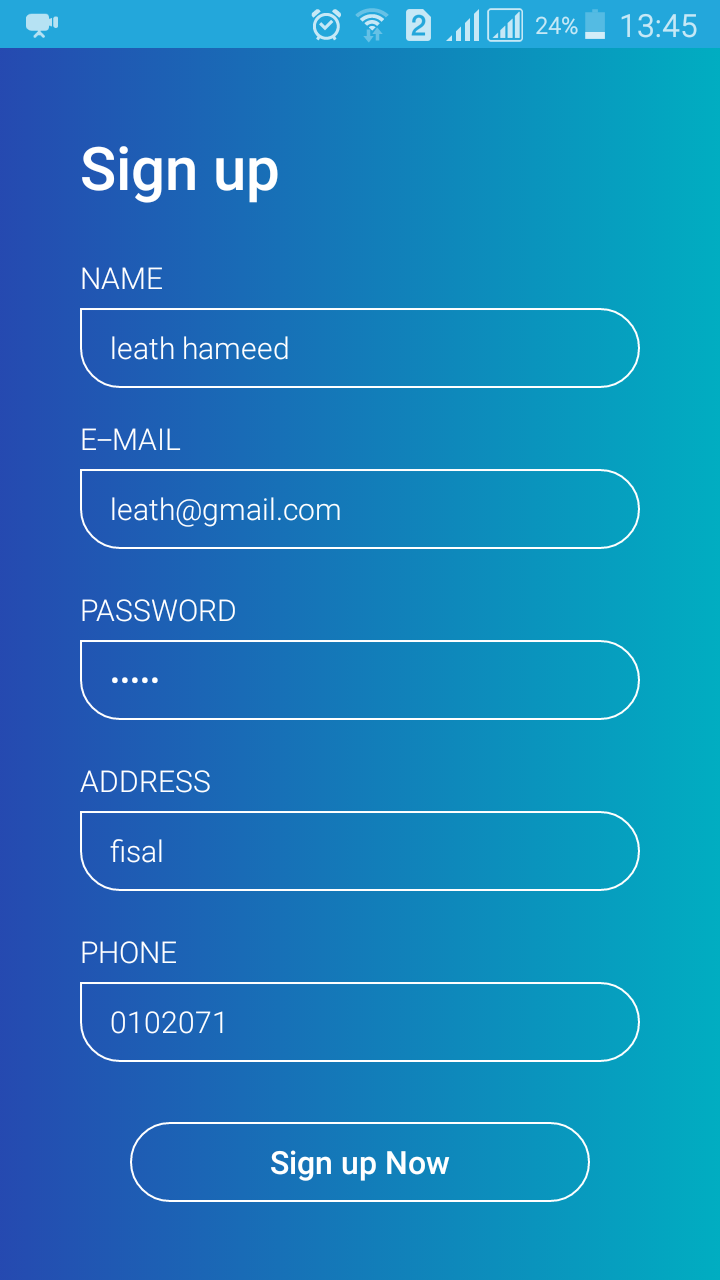
## System GUI Design

### Admin (Web App)



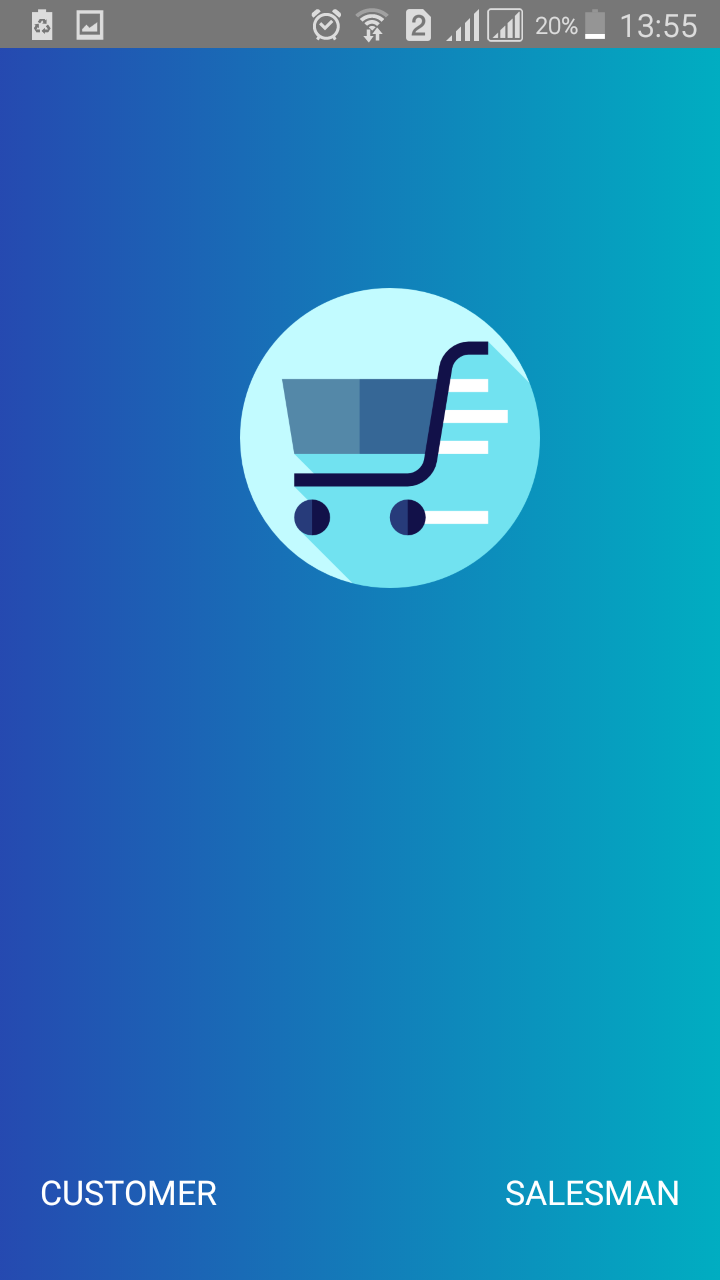
### Customer (Android)





### Salesman





# Implementation and Testing

## Unit Testing

### Admin Controller

Table 12 - Admin controller login

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /login | Normal case:  Correct password  Correct email | Return tokens for this user correctly. |
| Null case:  Empty password  Empty email | Return null + Bad request  Status 400 |
| Not exist case:  Wrong email  Wrong password | Return null + Bad request  Status 400 |

### Customer Controller

Table 13 - Customer Signup

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /Signup | Normal case:  Correct input | Successfully added to database |
| Null case:  Empty info | Bad request  Status 400 |
| Existing email | Bad request  Status 400 |

### Product Controller

Table 14 - Product controller

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /Add Product | Normal case:  Correct input | Successfully added to database,  Return true |
| Null case:  No pictures, no product parameter | Can’t send. |
| Missing value name=’ ‘or product Quantity=’ ‘or price =’ ‘ | Bad request  Status 400  And return false. |

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /get product by id | Normal case:  Correct input | Successfully return a product |
| Null case:  No id | Bad request. |
| Wrong id | Return null. |

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /get all product | Normal case: | Successfully return all products |
| Worst case | Empty array. |

### Request Controller

Table 15 - Request Controller

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /submit Request | Normal case:  Correct input | Successfully return all product |
| Wrong customer id | Bad request 400 |
| Empty array of products | Bad request 400 |
|  | Wrong product id | Will not added into the product order Data Base. |

### 

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| / get Requests | Normal case:  Correct input | Successfully return all requests |
| Worst case | Empty null. |

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /reject request | Normal case:  Correct ID input | Successfully delete request |
| Null case:  No id | Bad request. |
| Wrong id | Return null. |

### Route controller

Table 16 - Route Controller

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /get Route by Salesman Id | Normal case:  Correct input | Return array of requests |
| Wrong id | Empty array |

### Salesman controller

Table 17 - Salesman controller

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /Signup | Normal case:  Correct input | Successfully added to database |
| Null case:  Empty info | Bad request  Status 400 |
| Existing email | Bad request  Status 400 |

|  |  |  |
| --- | --- | --- |
| /check in/{id} | Normal case:  Correct role | Successfully  Return true |
| Wrong id /not exist id | Authorized error  403 |
| Wrong role | Authorized error  403 |

### User Controller

Table 18 - User Contoller

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /login | Normal case:  Correct password  Correct email | Return token, type and ID for this user correctly. |
| Null case:  Empty password  Empty email | Return null + No Content  Status 204 |
| Not exist case:  Wrong email  Wrong password | Return null + No Content  Status 204 |

### TSP Solver

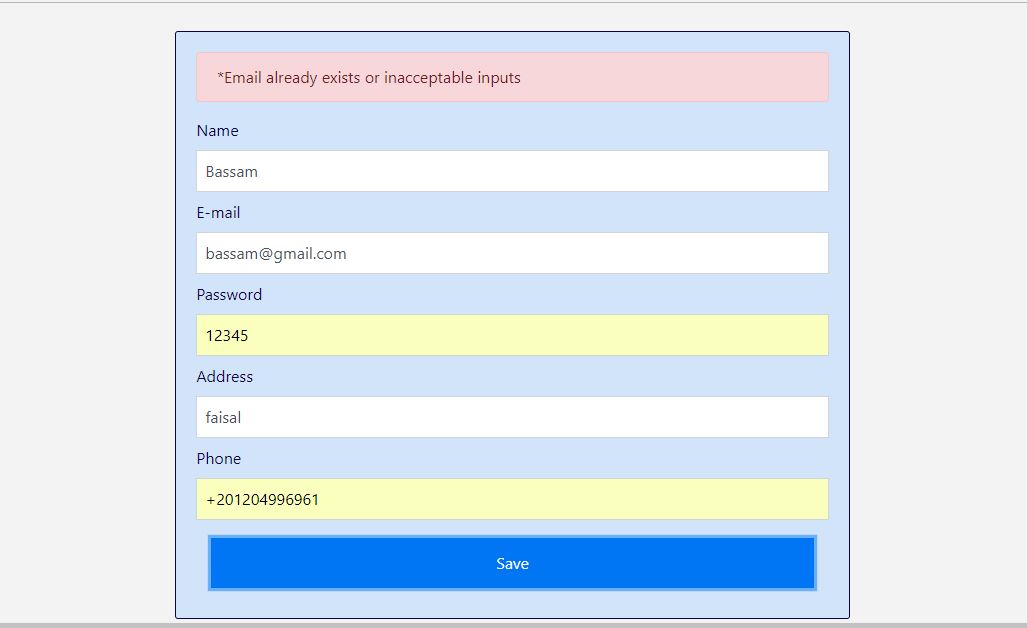
Table 19 - TSP solver

|  |  |  |
| --- | --- | --- |
| **Services** | **Input** | **output** |
| /run | Normal case:  # of Salesman < # of requests | List of routes for each |
| empty requests or Salesmen | No run and return null |
| large distances | use double types to handle overflows. |
| # of Salesman > # of requests | list of routes = # of requests each salesman takes one requests, other salesmen will remain free |
| # of requests <=2 | no need to run algorithm, returns points as it is. |

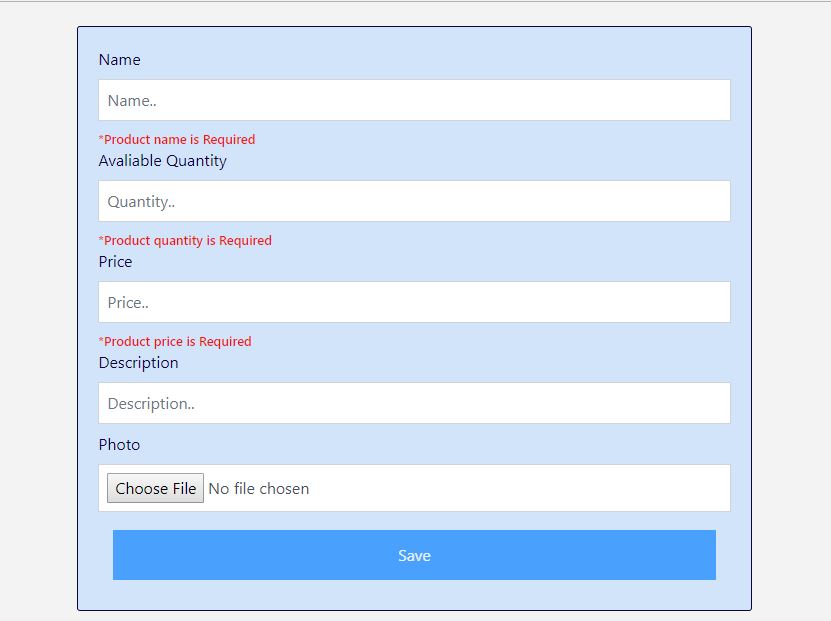
## System testing

### Admin

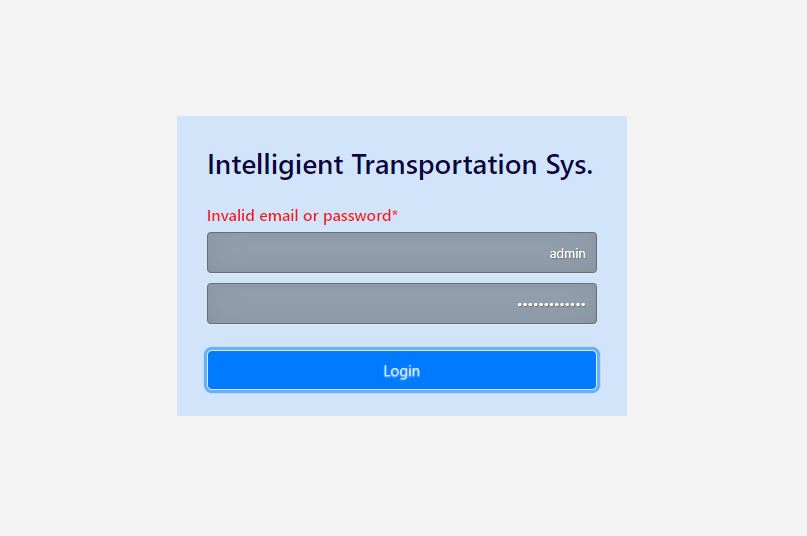
* adding new salesman with existing email



* Adding product Error



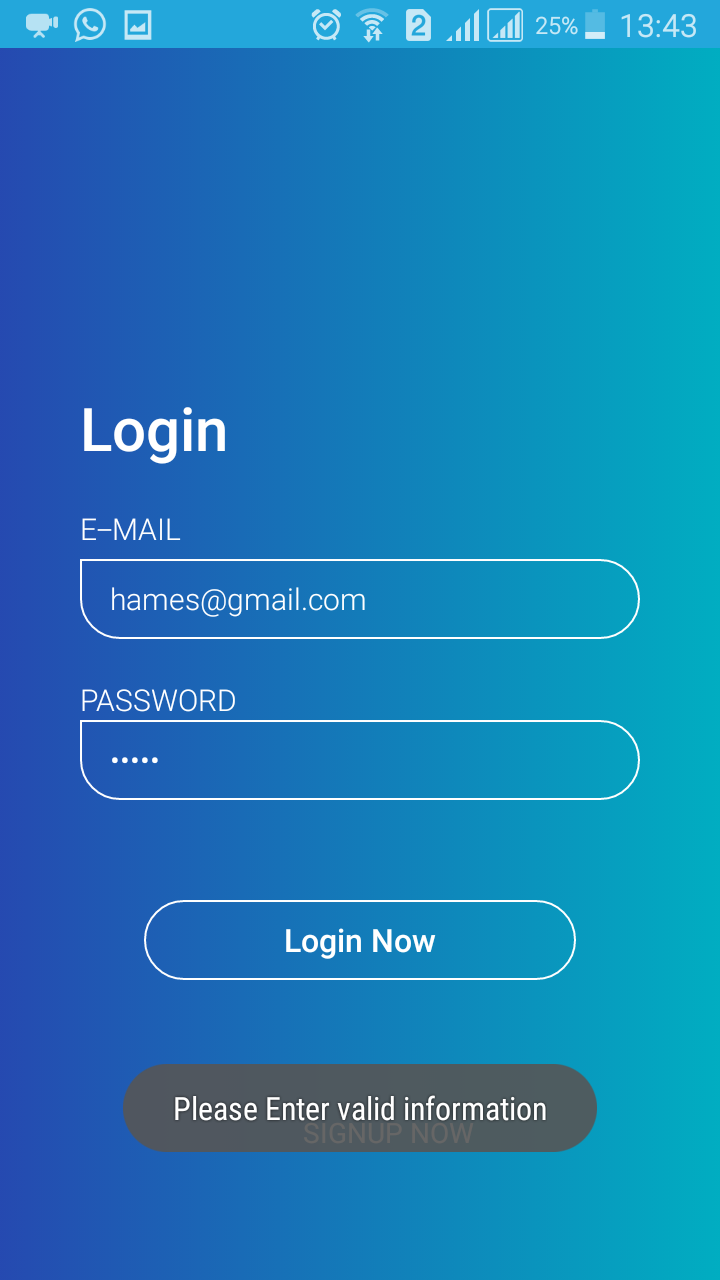
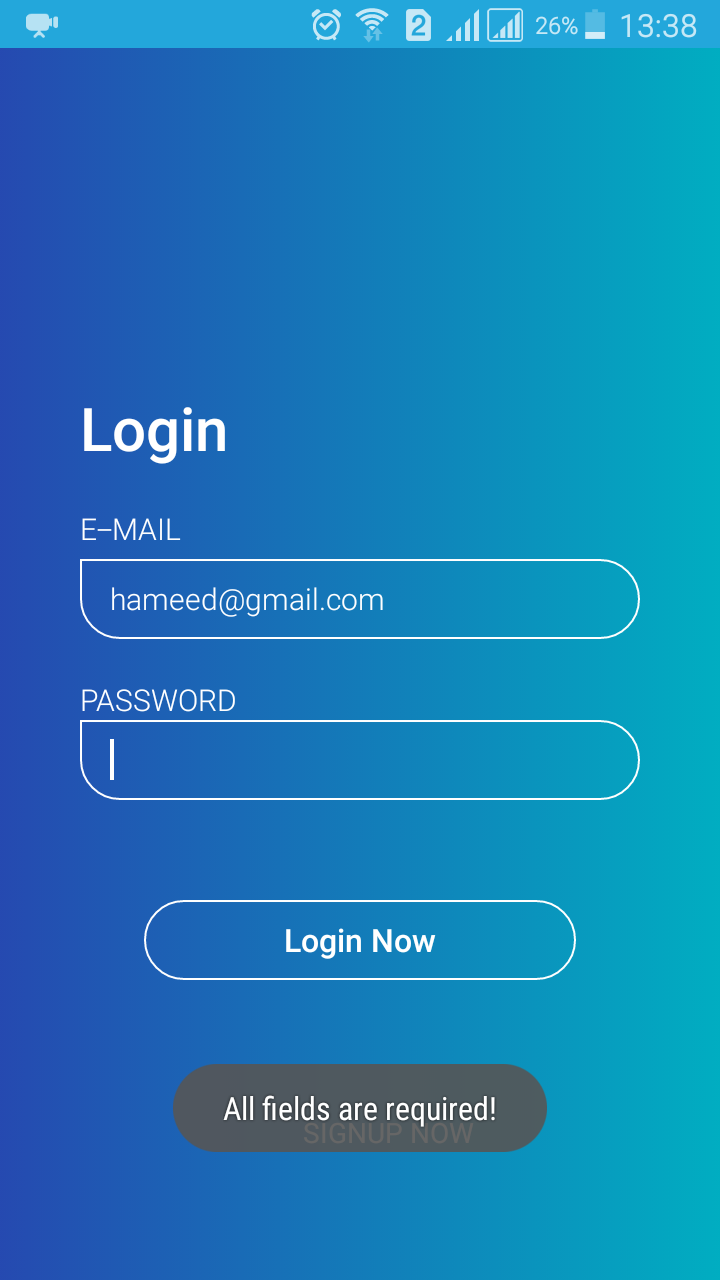
* Login errors



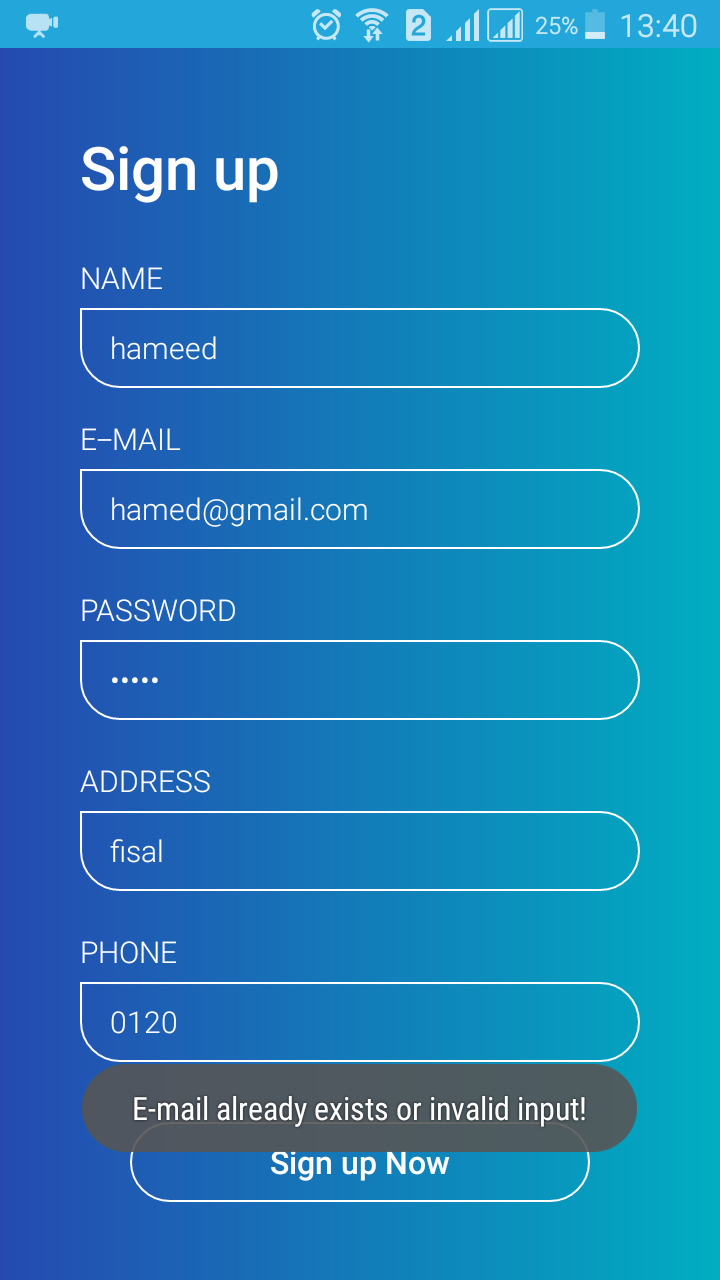
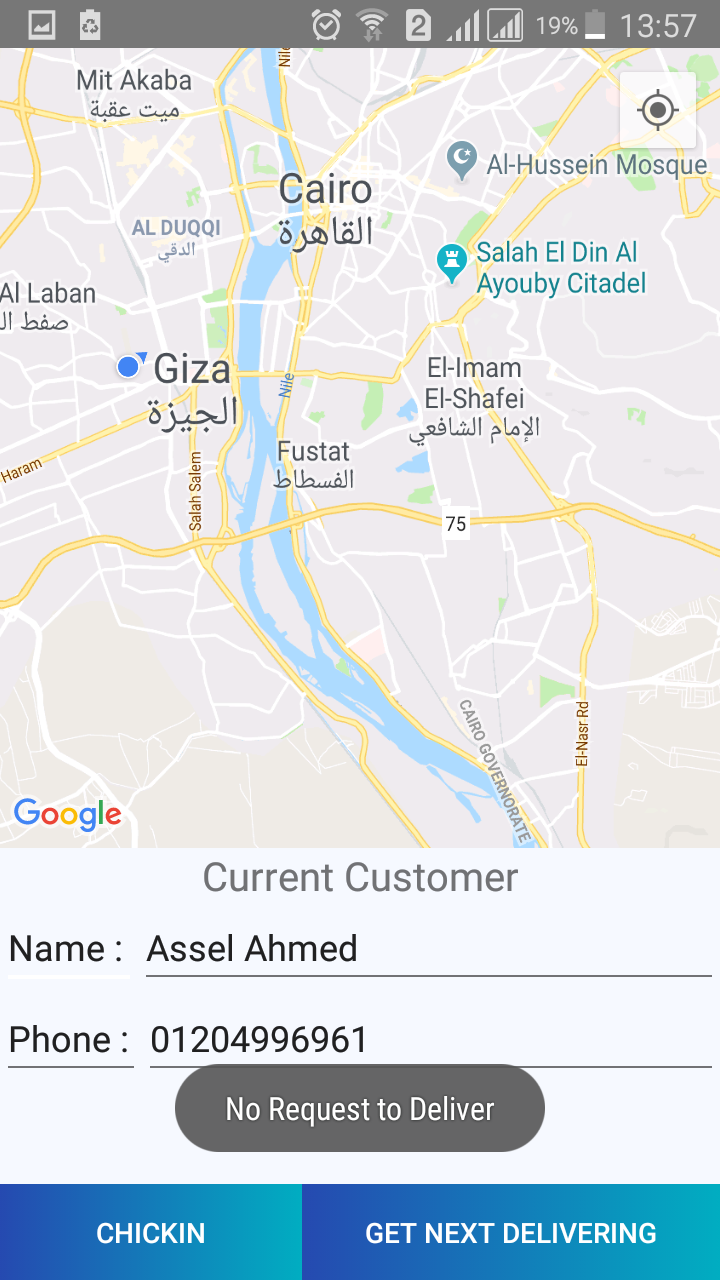


### Customer

* Log in



### Salesman

* Signup error

# Conclusion

We can see now that this project will help a lot by saving money, time and effort 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.

# References

1. TSP. [en.wikipedia.org/wiki/Travelling salesman problem](https://en.wikipedia.org/wiki/Travelling_salesman_problem)
2. TSP. [people.ku.edu/ lmartin/courses/math105-F11/Lectures/chapter6-](http://people.ku.edu/~/%7b~/%7djlmartin/courses/math105-F11/Lectures/chapter6-part4.pdf)[part4.pdf](http://people.ku.edu/~%7b~%7djlmartin/courses/math105-F11/Lectures/chapter6-part4.pdf)
3. Christo des Algorithm. [personal.vu.nl/r.a.sitters/AdvancedAlgorithms/2016/SlidesChapter2](http://personal.vu.nl/r.a.sitters/AdvancedAlgorithms/2016/SlidesChapter2-2016.pdf)-[2016.pdf](http://personal.vu.nl/r.a.sitters/AdvancedAlgorithms/2016/SlidesChapter2-2016.pdf)
4. Christo des Algorithm. [en.wikipedia.org/wiki/Christo des algorithm](https://en.wikipedia.org/wiki/Christofides_algorithm)
5. Held Karp Algorithm. [en.wikipedia.org/wiki/Held{Karp algorithm](https://en.wikipedia.org/wiki/Held/OT1/textendash%20Karp_algorithm)
6. Held Karp Algorithm. [www.geeksforgeeks.org/travelling-salesman-](http://www.geeksforgeeks.org/travelling-salesman-)[problem-set-1/](https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/)
7. Held Karp Algorithm. [www.youtube.com/watch?v=-JjA4BLQyqE](https://www.youtube.com/watch?v=-JjA4BLQyqE)
8. Heuristic Algorithms. <https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf>
9. Android Development: <https://developer.android.com/>
10. Genetic Algorithms <https://arxiv.org/ftp/arxiv/papers/1203/1203.3099.pdf>
11. Angular <https://angular.io/>
12. Spring https://spring.io/projects/spring-boot

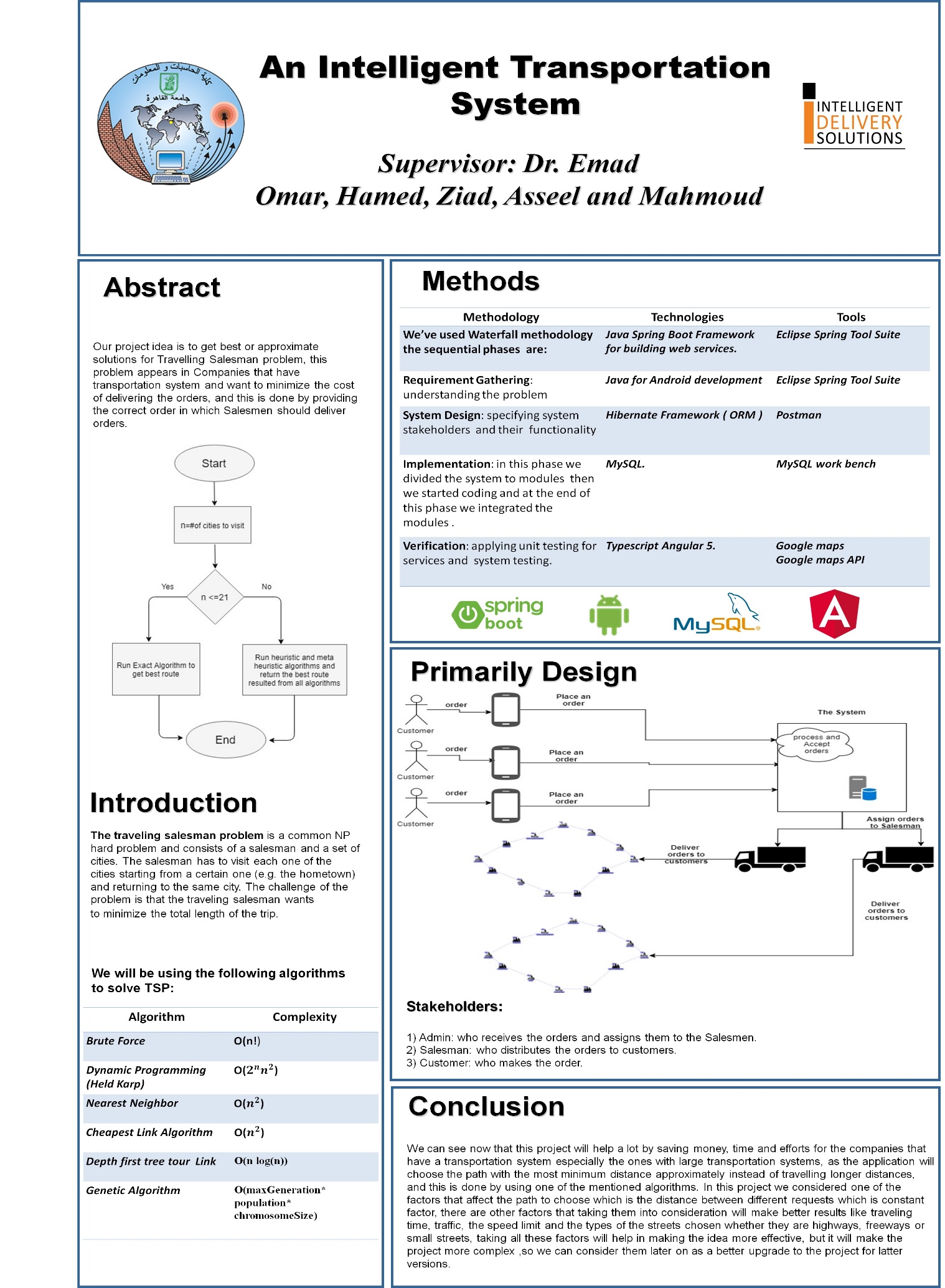
Poster

Table 20 - Poster