

Class description

Package com.CabBooking.Controller

Launcher

1. Main method to create main frame for application

SignInPanel

1. Allow user to login

SignUpPanel

1. Allow user to create a new account

UserPanel

1. Make Coordination between MenuPanel and ControlPanel

MenuPanel

1. Provides different operations to user

ProfilePanel

1. Shows user Info

EditProfilePanel

1. Allow user to change their details

CabBookingPanel

1. Allow user to book cab
2. Check if source and destination are valid
3. Find nearest driver for valid locations
4. Show driver and trip details - fare, time etc.
5. Update user and driver status on trip start

CabBookedPanel

1. Provide animation until trip ends
2. Allow user to give rating to driver
3. Asking user to make payment

WalletPanel

1. Shows current balance
2. Allow user to add amount

Package com.CabBooking.View

BackgroundPanel

1. Provides custom background design to all panels

MainFrame

1. creates main JFrame of the application

Button, TextField, TextLabel, PasswordField

1. Provides custom design to respective views

Package com.CabBooking.Model

CommonConstants

1. Contains constants frequently used in code

Customer

1. Contains all info of the user

eg - name, mobile, wallet balance, trip status etc.

DriverReshuffler

1. Find location with highest number of drivers and move half of them to current empty location

IntermediatePoint

1. Show intermediate locations during trip animation (CabBooked UI)

Package

com.CabBooking.Model.mapgraphutilities

Driver

1. Contains all info of the driver

eg - name, mobile, vehicle ID, rating.

MapBuilder

1. Build and store graph in database
2. Sending drivers to random locations
3. Store drivers in database

EdgeWeightedGraph

1. Provides an edge weighted graph

WeightedEdge

1. Represent a weighted edge between 2 locations

ShortestPath

1. Find shortest path between 2 locations using Dijkstra's Algorithm

Package com.CabBooking.Utils

Auth

1. Class to manage all authentication activities and CRUD operations from database.

EdgeWeightedGraph	
vertices	int
adjacencyList	List<ArrayList<WeightedEdge>>
EdgeWeightedGraph(int)	
addEdge(WeightedEdge)	void
adjacent(int)	ArrayList<WeightedEdge>
numberVertices()	int

ShortestPath	
edgeFrom	WeightedEdge[]
nodes	PriorityQueue<Integer>
from	int
ShortestPath(EdgeWeightedGraph, int)	
relaxEdge(WeightedEdge)	void
getPathTo(int)	Stack<Integer>
getDistanceTo(int)	double
distances	double[]

WeightedEdge	
e1	int
e2	int
weight	double
WeightedEdge(int, int, double)	
from()	int
other(int)	int
to()	int
getWeight()	double

WeightedEdgeComparator	
distTo	double[]
WeightedEdgeComparator(double[])	
compare(Integer, Integer)	int

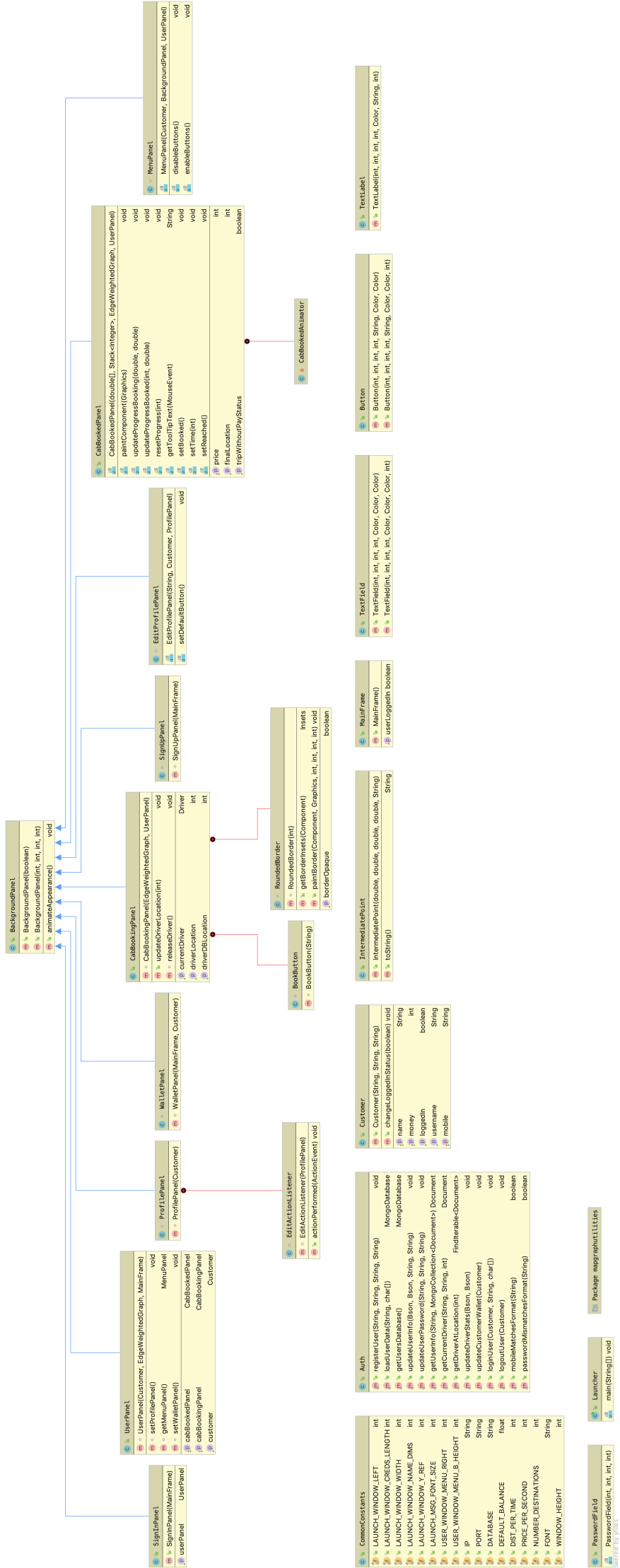
MapBuilder	
IP	String
PORT	String
DATABASE	String
main(String[])	void

Driver	
Driver(String, String, String, double)	
name	String
vehicleID	String
location	int
rating	double
ratedTrips	int
mobile	String

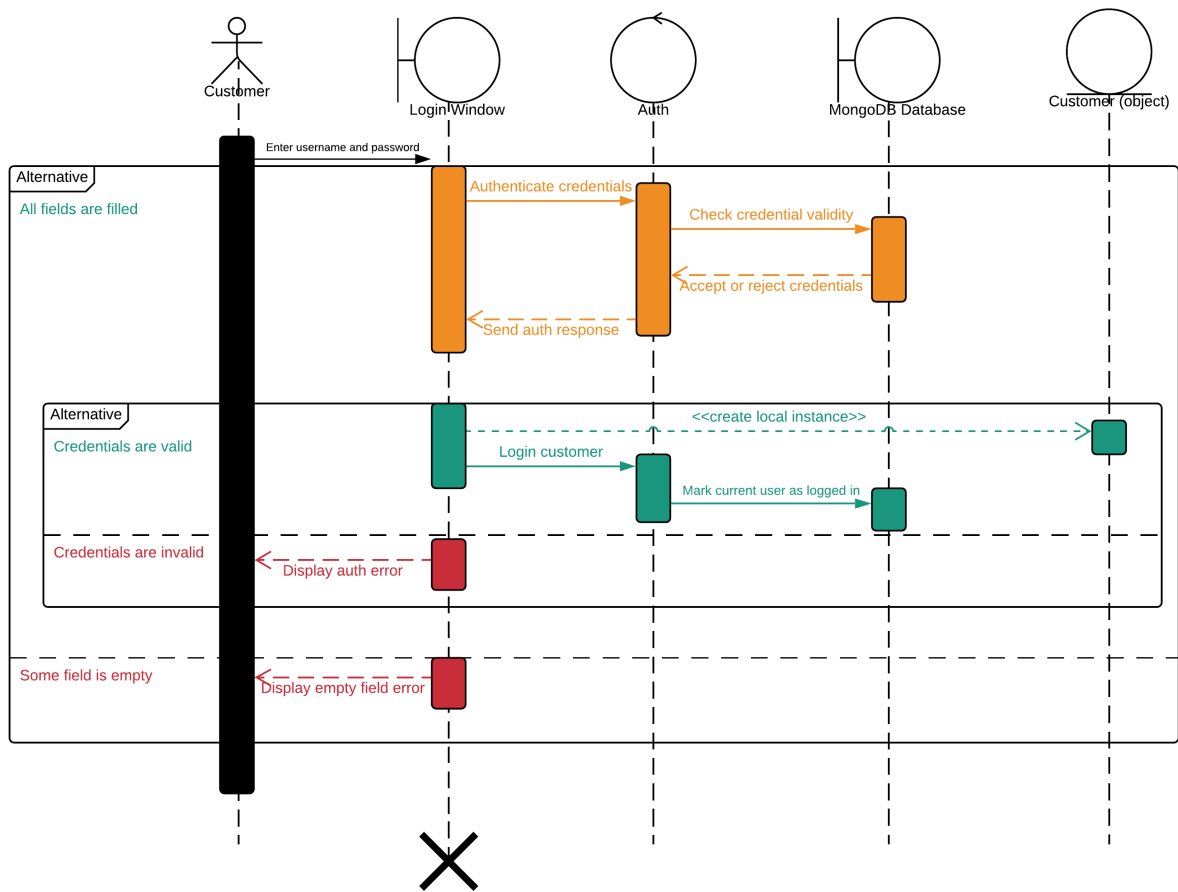
DriverReshuffler	
from	int
DriverReshuffler(int)	
run()	void

MapGraphUtilities class diagram

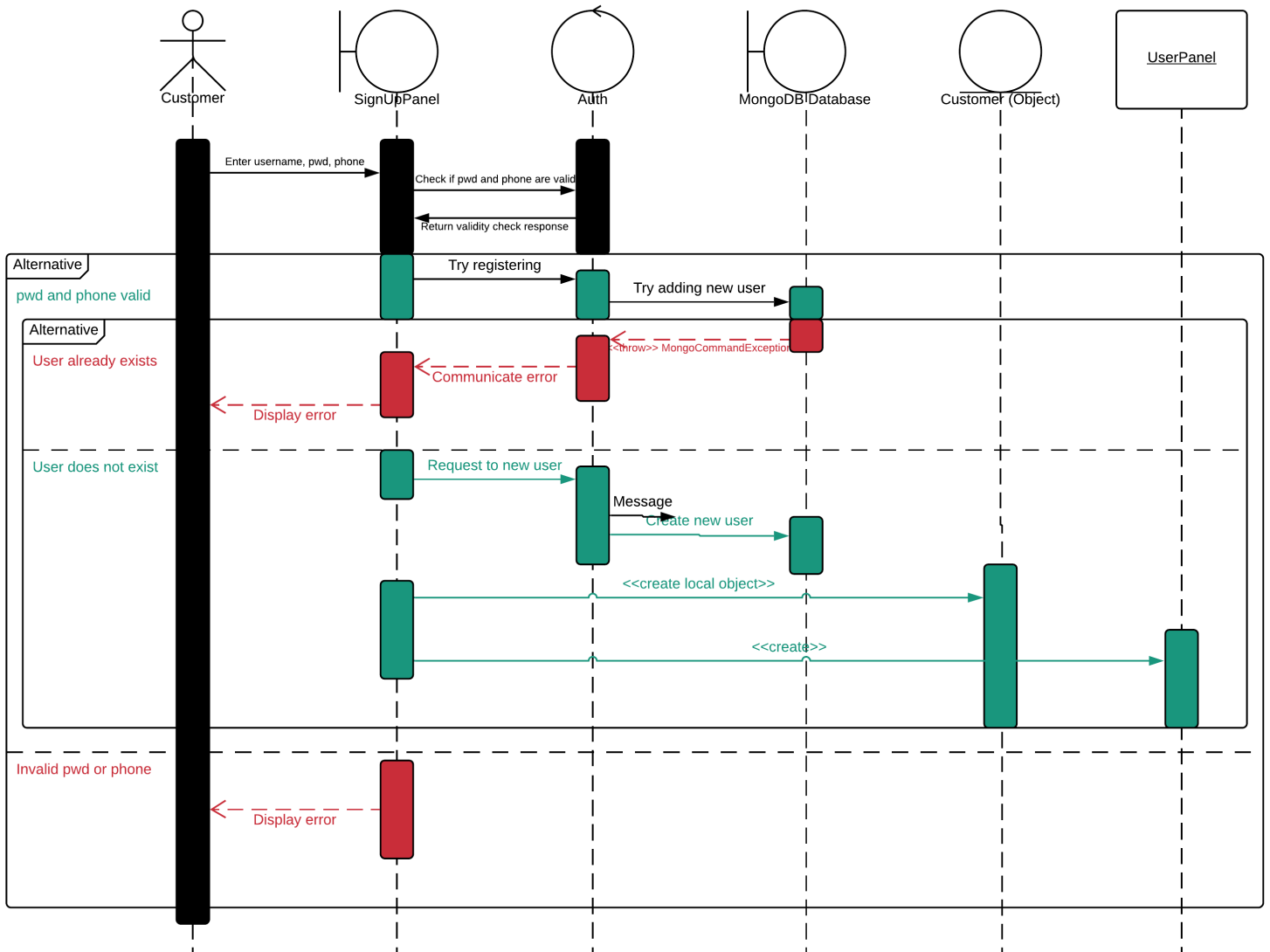
Class Diagram



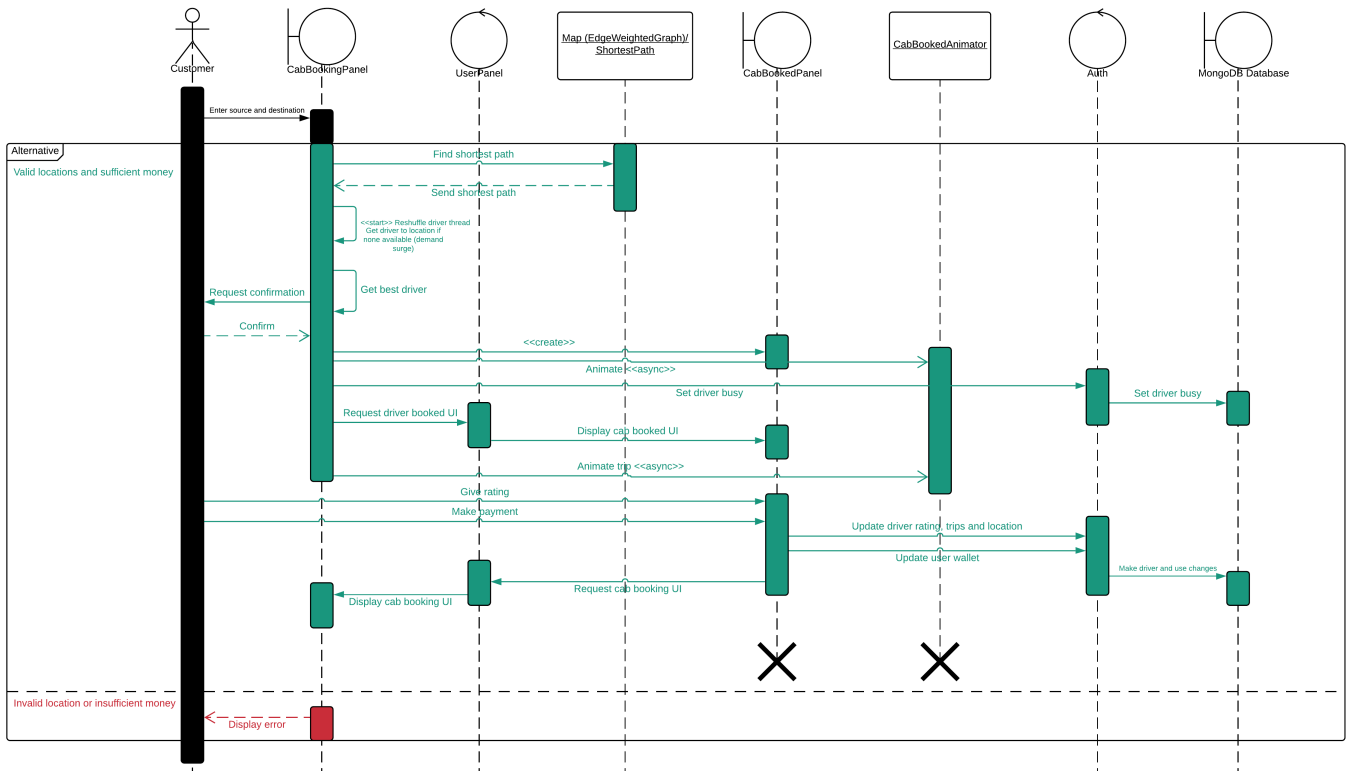
Login Sequence Diagram



Registration Sequence Diagram



Booking Sequence Diagram



CabBooking_usecase

