**Dance Studio**

By:-

Surjani Biswas (46001701)

**Table of Contents**

**1. Abstract**

**2. Technology Used**

**3.Scope**

3.1. Abstract Level Use Case Diagram

**4. Project Design**

**5.Sequence Diagrams**

5.1. Customer Sequence Diagram

5.2. Driver Sequence Diagram

**6.Activity Diagram**

6.1. Customer Activity Diagram

6.2. Driver Activity Diagram

6.3 Admin Activity Diagram

**7. Entity Relation Diagram**

**8. Use Case Diagrams**

7.1 Driver Use Case Diagram

7.2 Customer Use Case Diagram

7.3 Admin Use Case Diagram

**9. Class Description**

8.1 Attributes

8.2 Methods

**Abstract**

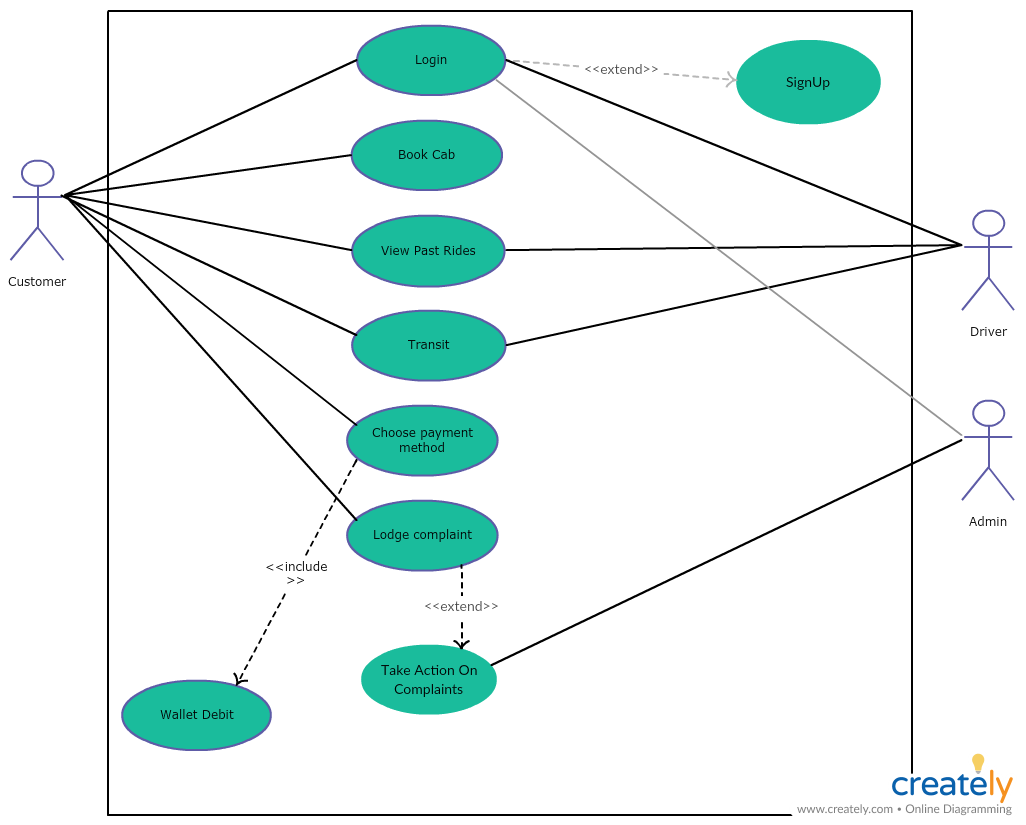
This project is aimed to develop a Dance Studio application. This is an integrated web-based application that provides a platform for all age groups to learn and excel in the most amazing and loved field called Dance. This contains the student component, choreographer component and the administrator component. This lets new students and choreographers to register or existing users to login and proceed with the bookings. The application ensures that the students can register for dance courses as per their choice of dance form, dance shifts and dance services by logging on to the website and choreographer will be added as per the availability. It allows students to view their details, their bookings and search for choreographers. The choreographer can view their details, their bookings. The admin can login to the system using his/her credentials and view booking details, verify student and choreographer by changing status.

**Technology Used**

* Java 8(base programming language)
* Jdk1.8.0\_65
* Jre1.8.0\_65
* Log4j-api-2.11.2.jar
* Junit 4.12
* Spring Web 5.1.10.RELEASE
* Spring Boot 2.1.9
* Spring Data 2.1.9
* My SQL 5.5.43(database storage)
* GIT 2.23.0(DevOps VCS tool)
* Maven 4.0.0(DevOps tool- Build & manage project)
* Postman 7.11.0(for running RESTful service)
* Node JS 10.11.0(Javascript Runtime Environment)
* Angular CLI 6.2.9(Framework to develop web apps)
* VsCode 1.39.1

**Scope**

There are three categories of people who would access the system viz. admin, students and choreographers. Each one of them would have some exclusive privileges.



**Admin:**

1. Login to the system using his/her credentials.
2. Verify choreographer details and student details at the time of registering in the application.

**Student:**

1. Login into the system using his/her credentials.
2. Sign up by using valid details.
3. Register for a course according to their choice.
4. Can view his/her details.
5. Search for choreographers.
6. Can view their bookings.

**Choreographer:**

1. Login into the system using his/her credentials.
2. Sign up by using valid personal details.
3. View his/her details.
4. View the bookings registered for them by the students.

**Project Design**

In this project, there are seven independent functions and the description about those functional components are as:

**Login/Sign Up Module**:The operations defined in this module are login and signup for both customers and drivers.

**Sign Up:**User details are added on signup page and these details are injected into the corresponding tables into the database. Only valid data are added to the tables else error messages are displayed. For Customer personal details like name, date-of-birth, gender, contact number, email-ID will be added to the database and for driver along with personal details, the vehicle details (vehicle number, vehicle type) owned by the driver will also be added.

**Login:** Driver and Customer can login into his or her account by giving proper email-ID and password. Only users with valid details can login into their account.

**Trip Selector Module:** In this module a logged in customer can choose his/her trip details viz. trip starting location, trip ending location, Cab modeletc. Cab model is selected from the given list of cabs (e.g. Mini, Micro, Sedan, SUV). After taking the inputs from the customer an estimated price will be shown to the customer.

**Cab Allocation Module(For Customer):**The functionality of this module is to select a cab based on the requirements of the customer. It takes the current location of customer and cab into consideration. After selecting the source and destination and the cab model, all the vehicles that are within the 3 km radius of the source location are filtered. Allocated cab details will be shown to the customer after the cab allocation is done.

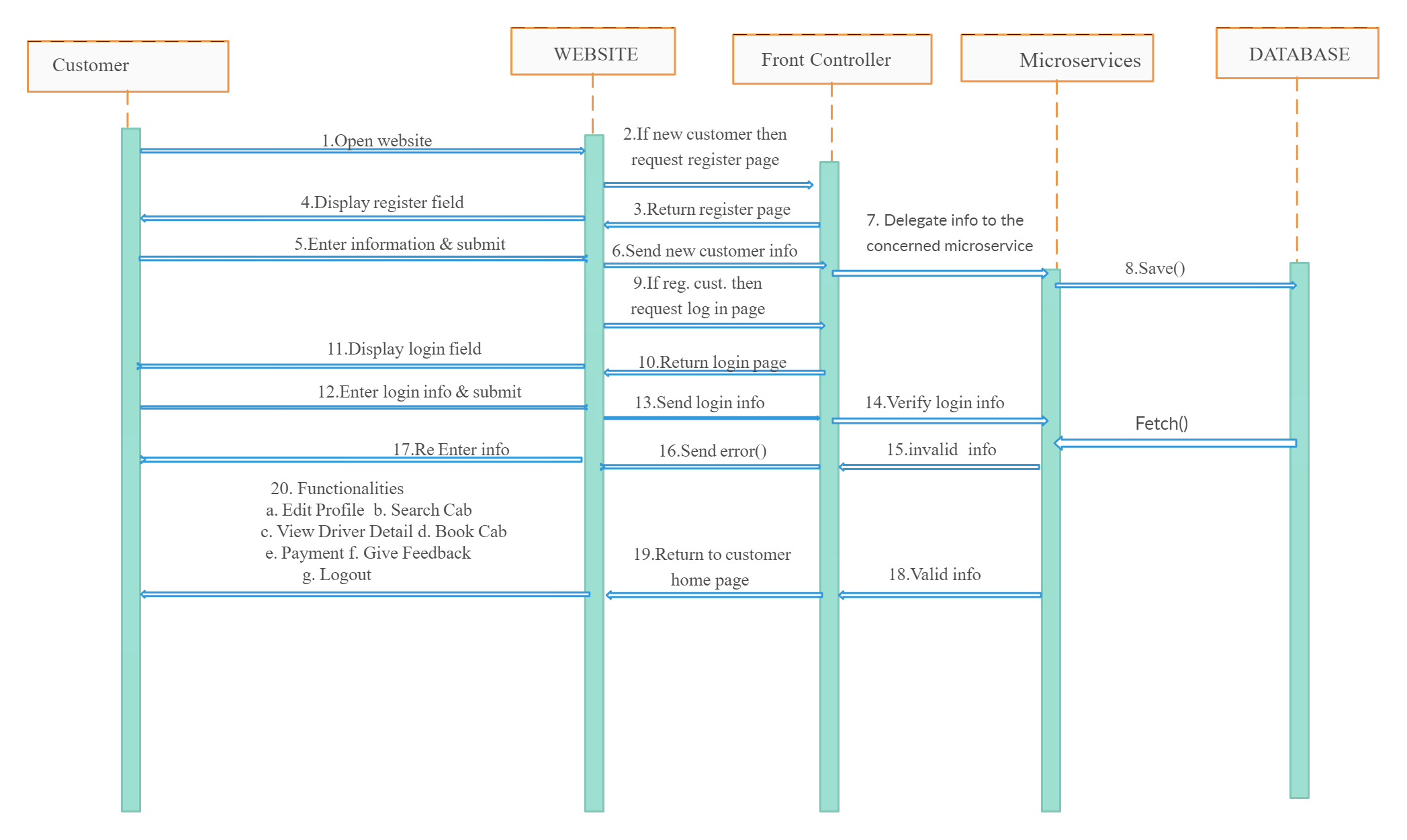
**Cab Allocation Module(For Driver):**In this module the list of vehicles available nearby the customer is accepted. After that a filtering operation is performed and the suitable driver is allotted to the coming customer request. After cab allocation is done customer details are shown to the driver for informing the driver about the customer.

**Transit Module:**The functionality of this module is to start and end a trip based on driver and changes in the database will be done accordingly. While transit a timer will be running and it will store end time of the trip. After the trip completion the status of the cab will be modified and the details related to the trip will be updated in the database. Rating for the trip is also requested from the customer in this module and this rating will be stored in the database. Finally, the fare will be generated considering the type of vehicle, rate per km and waiting per minute and then displayed. After the completion of this module it will be navigated to the payment page.

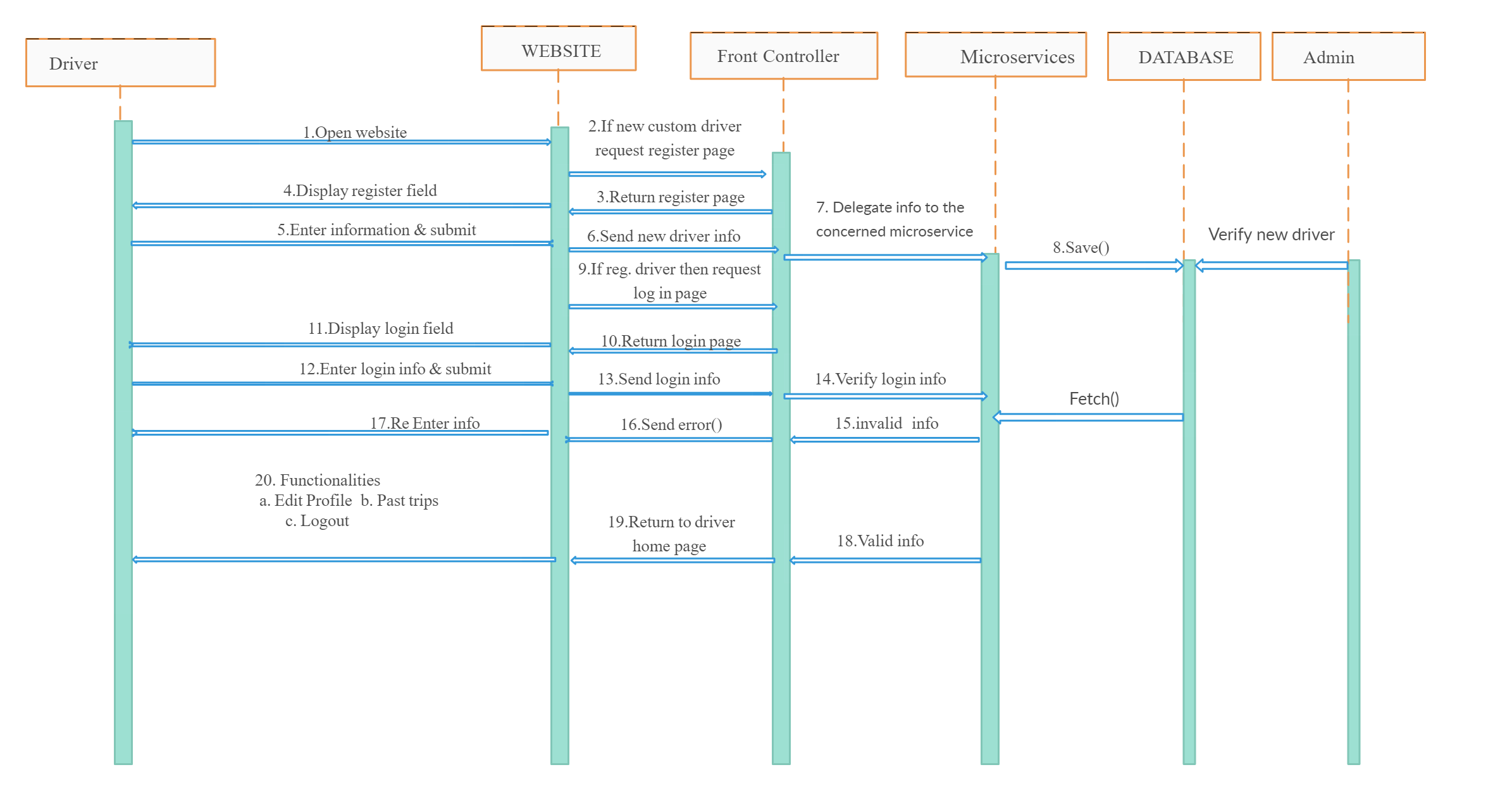
**Paymentsand Feedback Module:**The operations defined in this module are payment and feedback for customers. At first checking for sufficient wallet balance will be done for the customer. And if sufficient balance is present in the wallet, the fare will be paid directly from the wallet and if not then an add and pay option will come. Finally, after payment is done a feedback for the trip is requested from the customer in this module. The submission of feedback will lead to successful end of the trip.

**Admin Module:**The functionality of this module is to manage all the activities of the driver and customer. Admin can login to his/her account using valid email id and password and see all the booking details of customers. Feedbacks and reviews will also be managed by him. Admin also verifies driver details at the time of registering in the application.

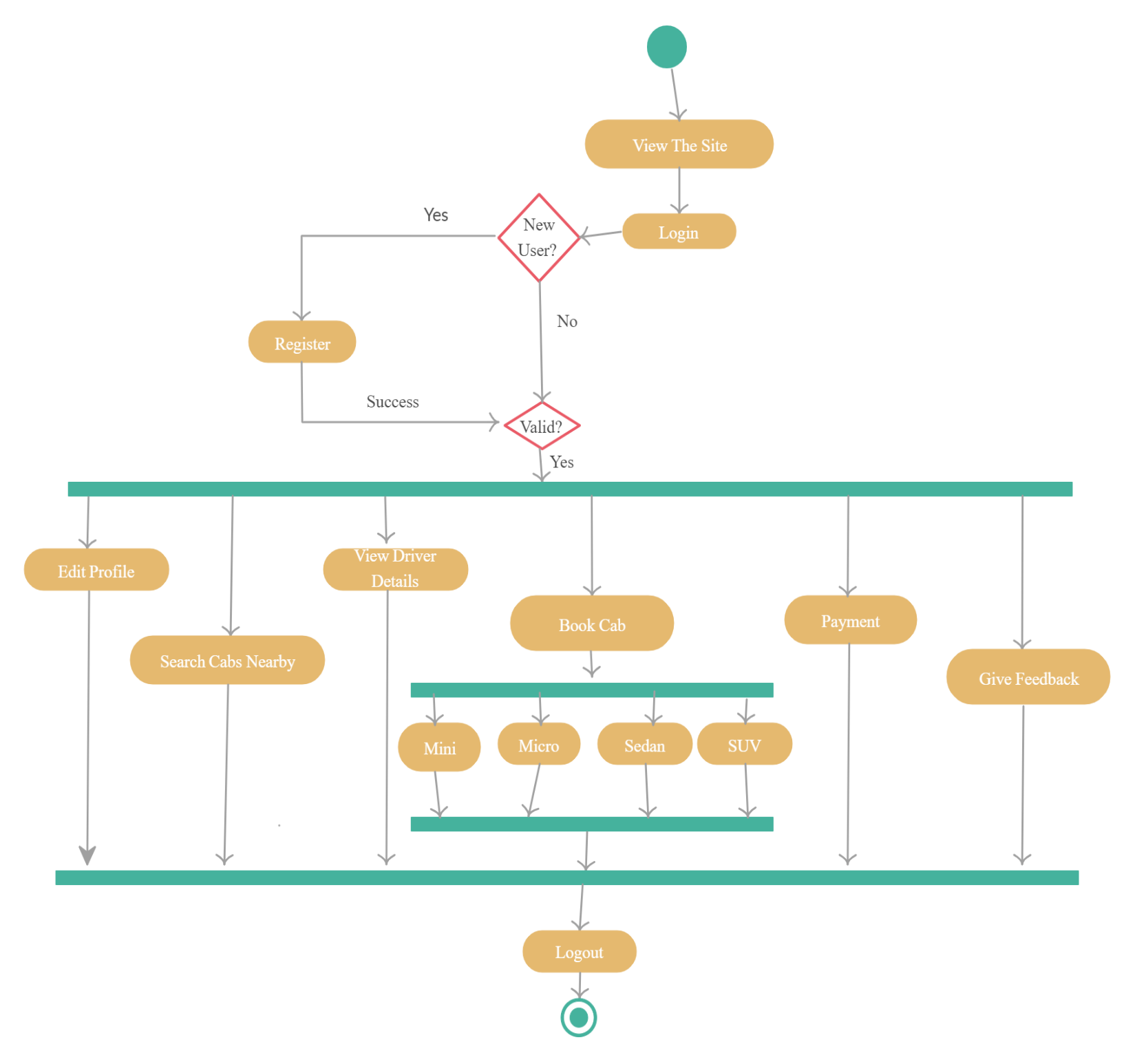
**Sequence Diagram for Customer**



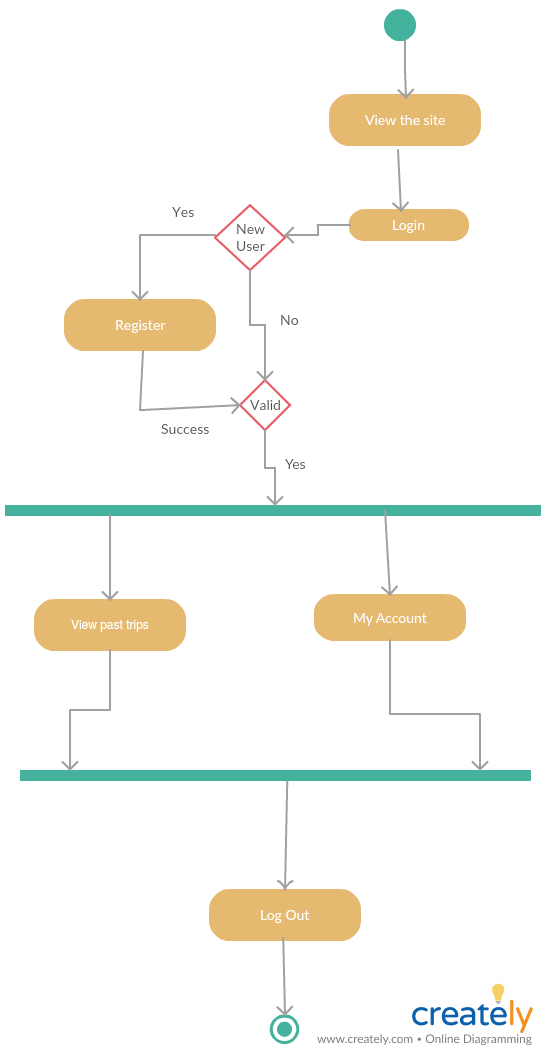
Sequence Diagram for DRIVER



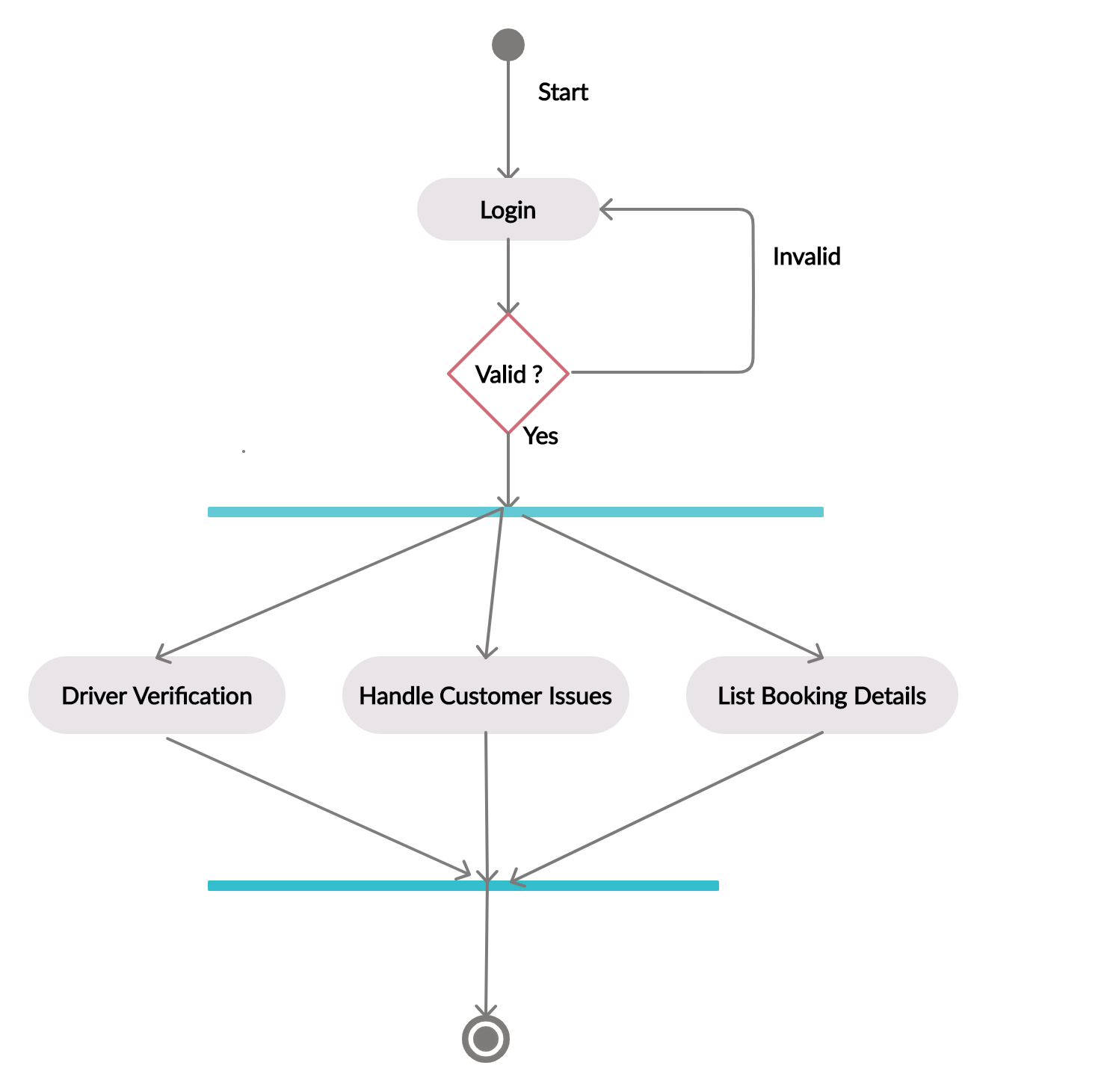
ACTIVITY DIAGRAM(For CUSTOMER)



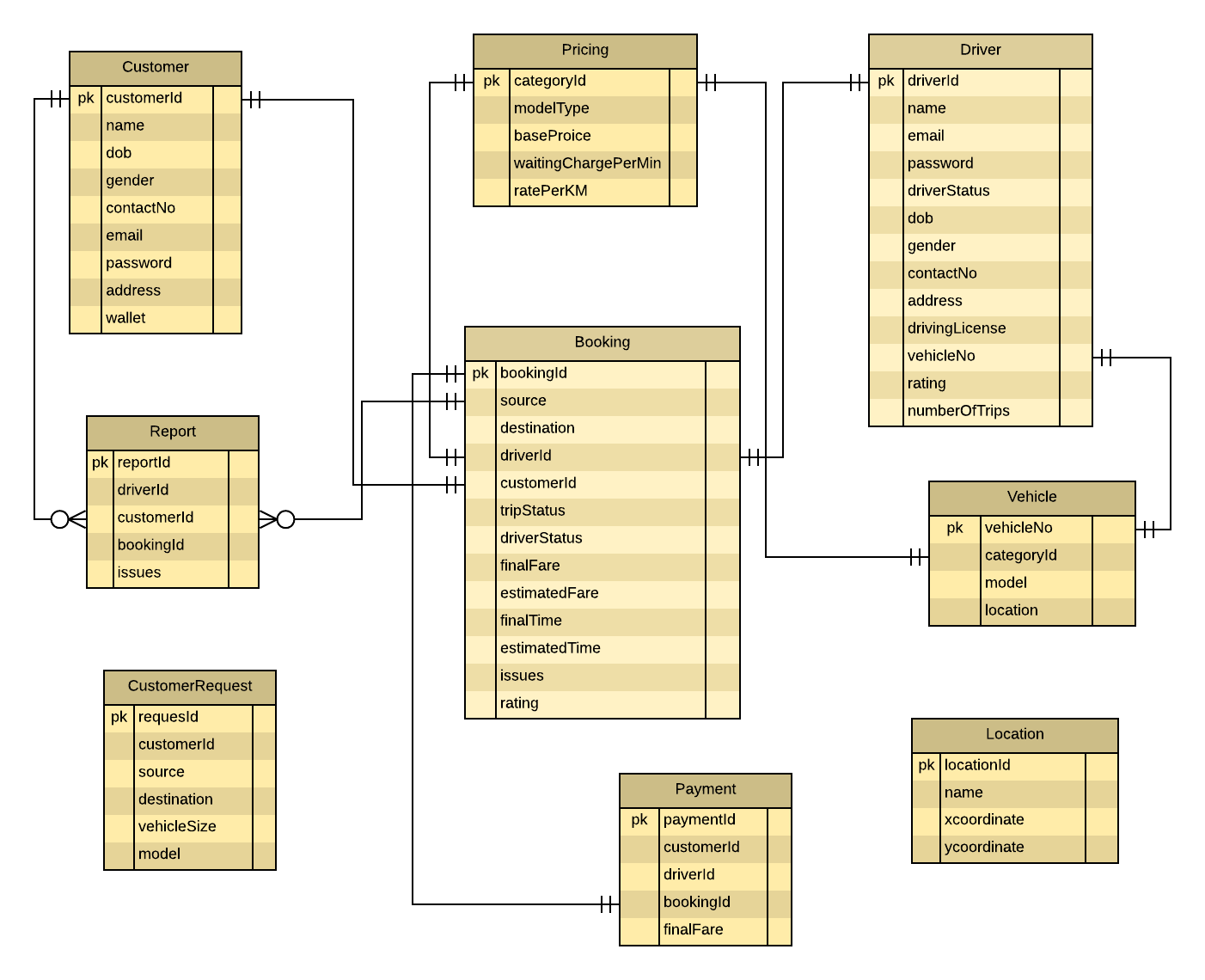
Activity Diagram (For Driver)



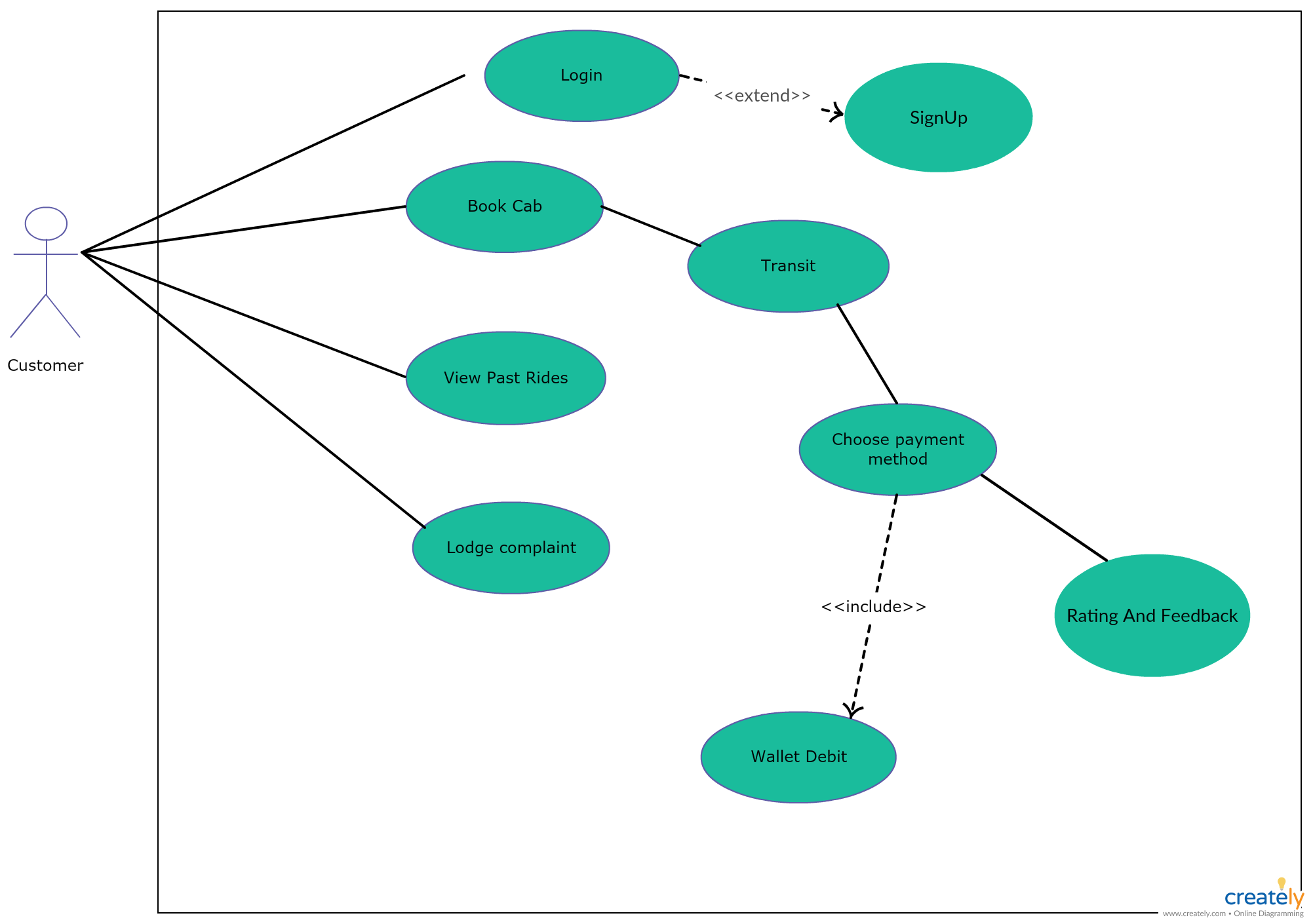
Activity diagram for admin



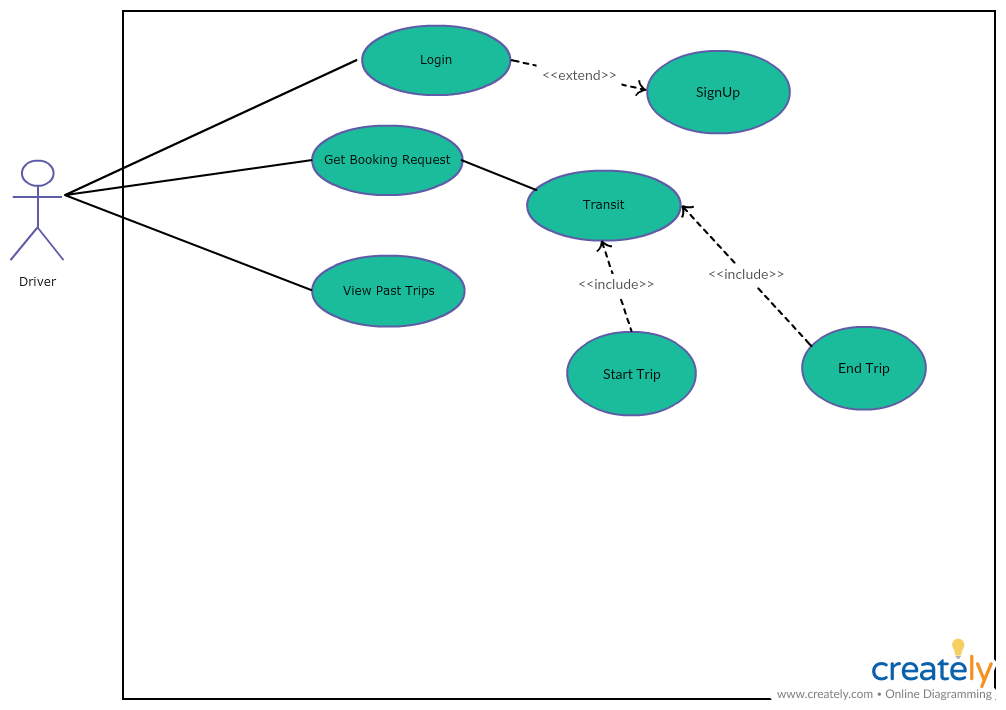
Entity relationship diagram



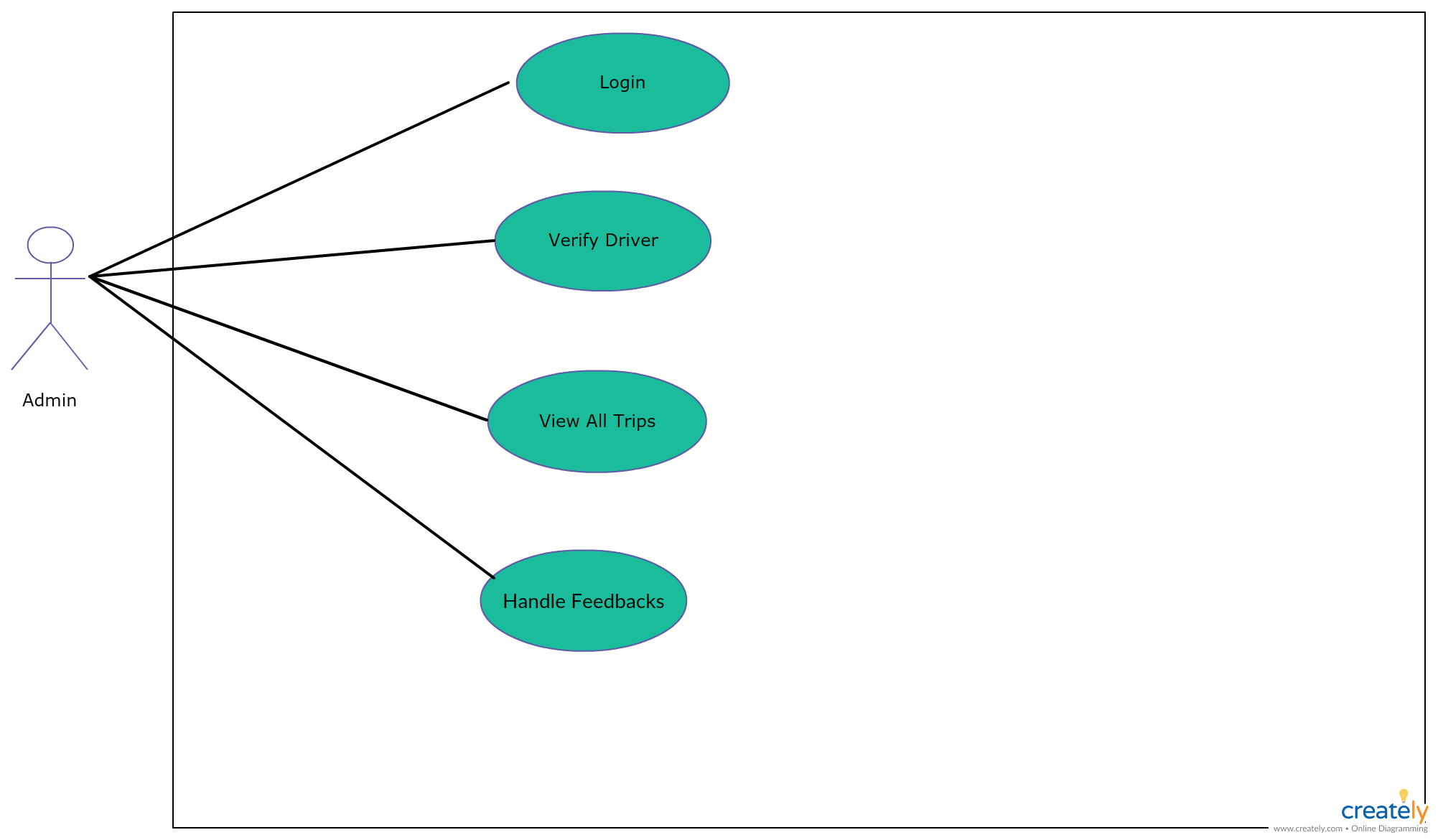
Use Case Diagram (Customer)



Use Case Diagram (Driver)



Use Case Diagram (Admin)



Class Description

Attributes:

1. User:
2. email: String
3. password: String
4. role: String
5. Driver:
6. driverId: Integer
7. name: String
8. password: String
9. dob: Date
10. gender: String
11. contactNo: String
12. email: String
13. address: String
14. driverStatus: String
15. vehicleNo: String
16. drivingLicense: String
17. rating: Integer
18. numberOfTrips: Integer
19. Customer:
20. customerId: Integer
21. name: String
22. password: String
23. dob: Date
24. gender: String
25. contactNo: String
26. email: String
27. address: String
28. wallet: Double
29. Vehicle
30. vehicleNo: Integer
31. categoryId: Integer
32. model: String
33. status: String
34. location: String
35. Location
36. locationId: Integer
37. xCoordinate
38. yCoordinate
39. Pricing
40. categoryId: Integer
41. modelType: String
42. basePrice: Double
43. ratePerKm: Double
44. waitingChargePerMin
45. CustomerRequest
46. requestId: Integer
47. customerId: Integer
48. source: String
49. destination: String
50. vehicleSize: String
51. modelString
52. BookingId
53. bookingId: Integer
54. source: String
55. destination: String
56. tripStatus: String
57. estimatedFare: Double
58. finalFare: Double
59. estimatedTime: Integer
60. finalTime: Integer
61. rating: Integer
62. issues: String
63. customerId: Integer
64. driverStatus: String
65. driverId: Integer

Methods:

1. Login/SignUp Module
   1. addCustomer(Customer customer): Customer – Add a new customer to the database.
   2. addDriver(Driver driver): Driver – Add a new driver to the database.
   3. findCustomer(String email, String password): Customer – Find a customer from the database for the given email and password.
   4. findDriver(String email, String password): Driver – For the given email and password this method finds the driver from the database.
2. Trip Selector Module:

2.1) saveTripDetails(TripDetails details): TripDetails – Add the details of the trip (viz. source, destination and cab model) to the database.

2.2)getLocations(): List<String> - Fetch all the locations.

2.3)estimatePrice(TripDetails details): double –Fetch the location details of source and destination and calculate the distance among them. According to the distance, calculates the estimated fare and returns the fare.

1. Cab Allocation Module(Customer):

3.1)getNearByVehicles(CustomerRequest req): List<Vehicle> - Find the vehicles that are within the 3KM radius of the customer pickup location.

3.2) bookACab(Booking booking): Booking – Save the booking details to the database.

3.3) pastTrips(Customer customer): List<Booking> - Fetches the list of past trips for a particular customer.

4)Cab Allocation Module(Driver):

4.1)getListOfDrivers(List<Vehicle> vehicles): Driver - List of vehicles coming from customerside which contains information of the vehicles that are nearby and free. Returns the selected driver who is assigned for the coming customer request.

4.2)getAllTripsOfADriver(int userId): List<Booking> - Shows a list of past rides of the logged in driver.

4.3) fetchByEmail(String email):Driver – Gets a driver by email id.

4.4) searchForBooking(int driverId):Customer - Searches for a current Booking for the driver who is logged in. Returns the customer details who is allotted to the logged in driver.

4.5)getBookingDetailsForADriver(int driverId):Booking – Get the booking details for a particular driver.

5)Transit Module:

5.1)startTrip(Booking booking): Booking – Start the trip by setting the status of the trip to ‘started’ and update it to the database.

5.2)endTrip(Booking booking): Booking – End the trip by setting the status of the trip to ‘completed’ and update it to the database.

5.3) rateTrip(Booking booking): Booking – Rate for the trip. It will call updateDriverRating(Booking booking), updateVehicleStatus(Booking booking, Driver driver) and finalFareGeneration(Booking booking, Vehicle vehicle) methods.

5.4)updateDriverRating(Booking booking): Driver- Update the rating of the driver.

5.5)updateVehicleStatus(Booking booking, Driver driver): Vehicle – When the trip ends, update the vehicle status to ‘free’.

5.6) finalFareGeneration(Booking booking, Vehicle vehicle): double- Final price is calculated.

5.7) getBooking():Booking – Find the details for a particular booking.

6)Payments and Feedback Module

6.1)addPayment(Booking booking): Payment – Add payment details to the database.

6.2)updateReport(Booking booking): Report –this will update feedback issues (if any)to the reports table.

6.3) updateCustomerWallet(Booking booking): Customer – this will update the customer’s wallet balance after each ride’s payment completion.

6.4)findCustomer(Booking booking): Customer – it will fetch the corresponding customer from the booking details passed.

7)Admin Module

7.1)saveDriver (Driver driver): add a new driver to database.

7.2)deleteDriver (int id) to delete a driver details for the given id;

7.3)updateDriver (Driver driver): it will update driver details for a given id.

7.4)findDriver (int id): to get a driver details for the given id.

7.5)getAllReport (): get the list of all reports stored in database.

7.6)getAllDriver (): get the list of all driver details stored in database

7.7)saveReport (Report report): save issues raised by customers in database.

7.8)getBookingDetails (): get the list of booking details stored in database

7.9)saveBooking (Booking booking): add the booking details in the database.