Project CSC343

NASIR ALSAEED : [redacted]
NASSER HAQSHA : [redacted]
MUTEB AlSAQOUB : [redacted]
NAWAF ALDHOWAILIA : [redacted]

Functional Requirements:

1-User Management:

-Login and access control for administrators
-Login and access control for drivers
-Login and access control for students
-Login and access control for staff members

2- Entity Management:

-Ability to add, update, delete and view information for stations
-Ability to add, update, delete and view information for passengers
-Ability to add, update, delete and view information for school

2-**School Management:**

- the admin has the ability to add information to school
- the admin has the ability to update school information
- the admin has the ability to delete information from school
- the admin has the ability to view school information

3-**Vehicle Management:**

- the admin has the ability to add vehicles
- the admin has the ability to update vehicle information
- the admin has the ability to delete vehicles
-the admin has the ability to view vehicles information

4-**Route Management**:

- the admin has the ability to add information of routes
- the admin has the ability to update routes information
- the admin has the ability to delete routes information
- the admin has the ability to view routes information
- the admin has the ability to create and assign routes
- the admin has the ability to assign stations to the routes

5-**Driver Management**:

- the admin has the ability to add drivers
- the driver has the ability to update his own information
- the admin has the ability to delete drivers
- the admin has the ability to view drivers information
- the admin has the ability to drivers to vehicles

6-**Station Management:**

- the admin has the ability to add stations
- the admin has the ability to update stations information
- the admin has the ability to delete stations
- the admin has the ability to view stations information

7-**Passenger Management**:

- the admin has the ability to add passengers’ information to the system
- the passenger has the ability to update their own info
- the admin has the ability to add delete passengers from the system
- the admin has the ability to view passengers’ information

Non-Functional Requirements:

Program developing(Organizational): the program should be developed using java so that it becomes easier to make the GUI
Governmental caution: before registering any driver or customer the application should search the persons criminal record
Payment VAT(External): the payment should have a 15% VAT as required by the government
Mobile Compatibility: Ability to access the application from mobile devices.

Actor Goal List:

Administrator:

Manage user accounts and access controls
Add information for vehicles, drivers, routes, stations, and passengers
update information for vehicles, drivers, routes, stations, and passengers
delete information for vehicles, drivers, routes, stations, and passengers
Generate reports and view detailed information
Monitor and track the performance of drivers and vehicles
Manage user accounts and access controls

Driver:

View assigned routes and vehicles
Update their availability and attendance status
View passenger information
View assigned routes and vehicles
Provide feedback and raise any concerns

Customer:

View assigned vehicle and route information
View payment history and receipt Provide feedback and raise any concerns
Pay monthly transport fee online
Provide feedback and raise any concerns

Description of use cases

First use case

Use Case: Add vehicle

Actor: system admin.

Precondition*:* vehicle does not exist

Post condition:vehicle is added.

User Actions	**System response**
i. The admin enters the vehicle's license plate to check it in the database.	i. System responses with the validation message. **Exp1**
ii. The admin enters the vehicle's information. License plate, type, model	ii. System responses with the validation message. **Exp2**
iii. The admin directs the system to add the vehicle. **Alt1**	iv. The system responses with the adding message. Use case ends.

Exception 1: vehicle exists. No vehicle is added. Use case ends.

Exception 2: model is too old. Use case ends.

Alternative 1: Admin directs the system to cancel the transaction. Use case ends.

Second use case

Use case: Update school information

Actor: Customer.

Precondition*:* School has old registration info

Post condition: School gets the new info

User Actions	**System response**
The admin searches for school registry	System responses with the validation message. **Exp1**
The admin enters the wanted information	.System responses with the validation message. **Exp2**
The admin direct the system to update the information **Alt1**	.The system responses with the Update message. Use case ends

Exception 1: School does not exist. Use case ends

Exception 2: Some information is not valid. Use caee ends

Alternative 1: the admin cancels the update process. Use case ends

Third use case

Use Case:

Actor: The customer

Precondition: The customer has no vehicle

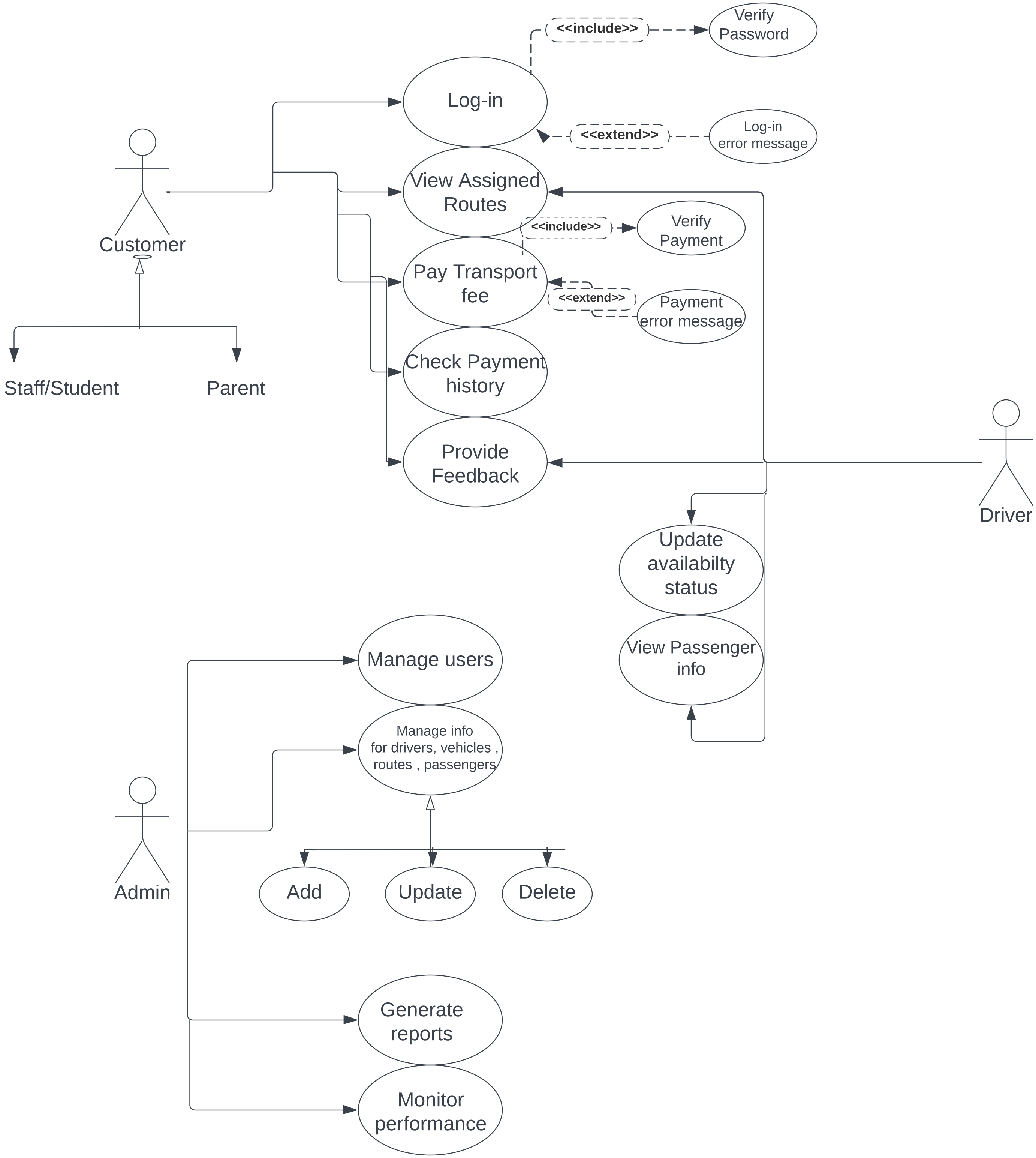
Post condition: The customer has a vehicle

User Actions	System response
The customer tries to find a car	The system finds a car for the customer. Exp1
The customer checks what type of car they got. Alt2	System responses with the validation message. .
View the routes and fixed fees of each one.	Return with the fee of each route.
The customer pay the required fee. Alt3	The system responses with the payment validation message. Use case ends.

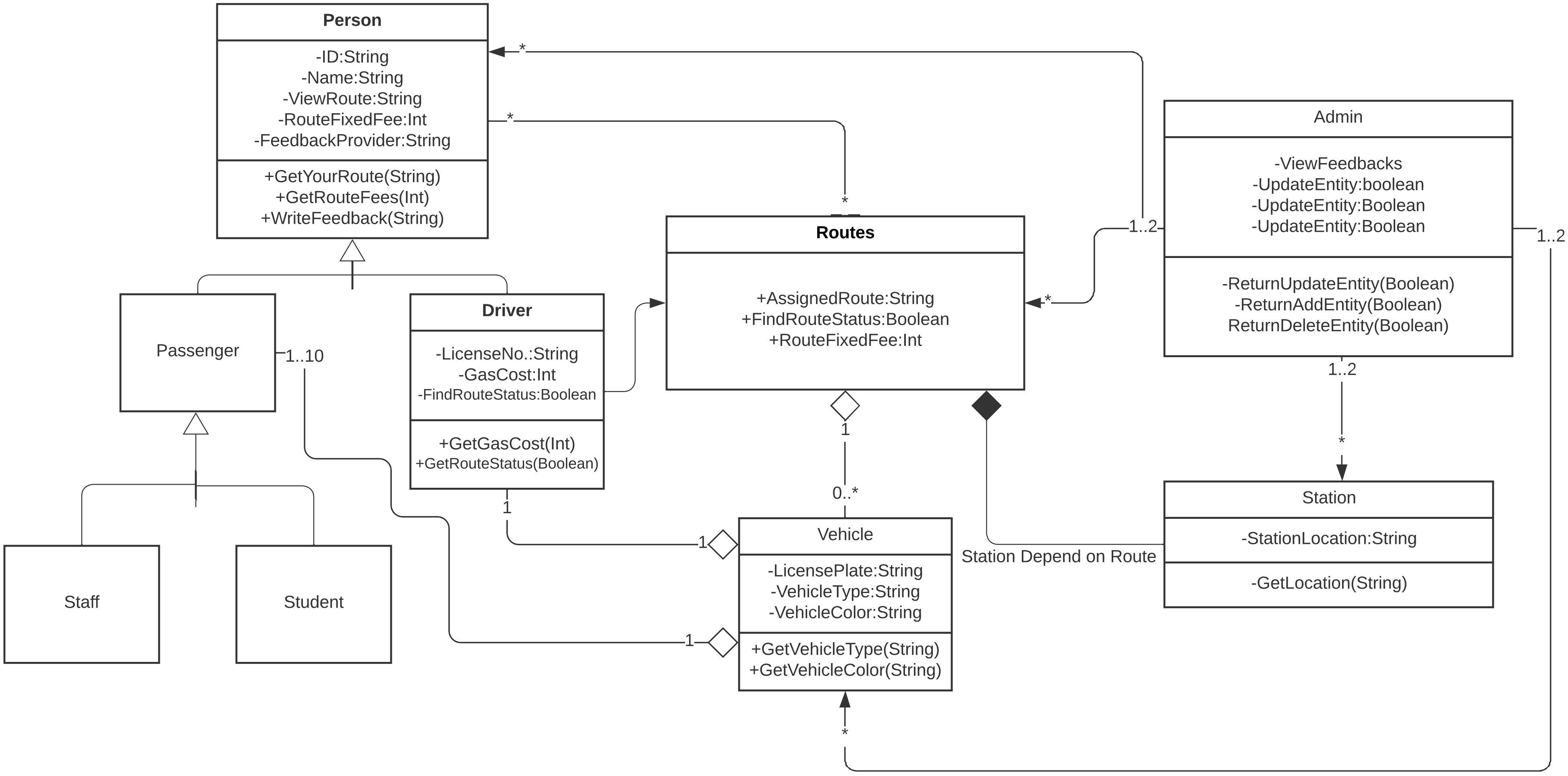
Exception 1:No car is available at the moment.. Use case ends.

Alternative 1: The customer doesn’t like the car. Use case ends

Alternative 2:: Customer has a problem with the required fee and doesn’t like it.. Use case ends.



<<include>>
Verify Password
Log-in
<<extend>>
Log-in error message
Customer
View Assigned Routes
<<include>>
Verify Payment
Pay Transport fee
<<extend>>
Payment error message
Check Payment history
Provide Feedback
Staff/Student
Parent
Driver
Update availabilty status
View Passenger info
Manage users
Manage info for drivers, vehicles , routes , passengers
Admin
Add
Update
Delete
Generate reports
Monitor performance


Person
-ID:String
-Name:String
-ViewRoute:String
-RouteFixedFee:Int
-FeedbackProvider:String
+GetYourRoute(String)
+GetRouteFees(Int)
+WriteFeedback(String)
*
*
*
Routes
+AssignedRoute:String
+FindRouteStatus:Boolean
+RouteFixedFee:Int
*
1..2
Admin
-ViewFeedbacks
-UpdateEntity:boolean
-UpdateEntity:Boolean
-UpdateEntity:Boolean
-ReturnUpdateEntity(Boolean)
-ReturnAddEntity(Boolean)
ReturnDeleteEntity(Boolean)
1..2
1..2
*
Passenger
1..10
Driver
-LicenseNo.:String
-GasCost:Int
-FindRouteStatus:Boolean
+GetGasCost(Int)
+GetRouteStatus(Boolean)
1
1
0..*
1
Station
-StationLocation:String
-GetLocation(String)
Station Depend on Route
Vehicle
-LicensePlate:String
-VehicleType:String
-VehicleColor:String
+GetVehicleType(String)
+GetVehicleColor(String)
1
1
*
Staff
Student

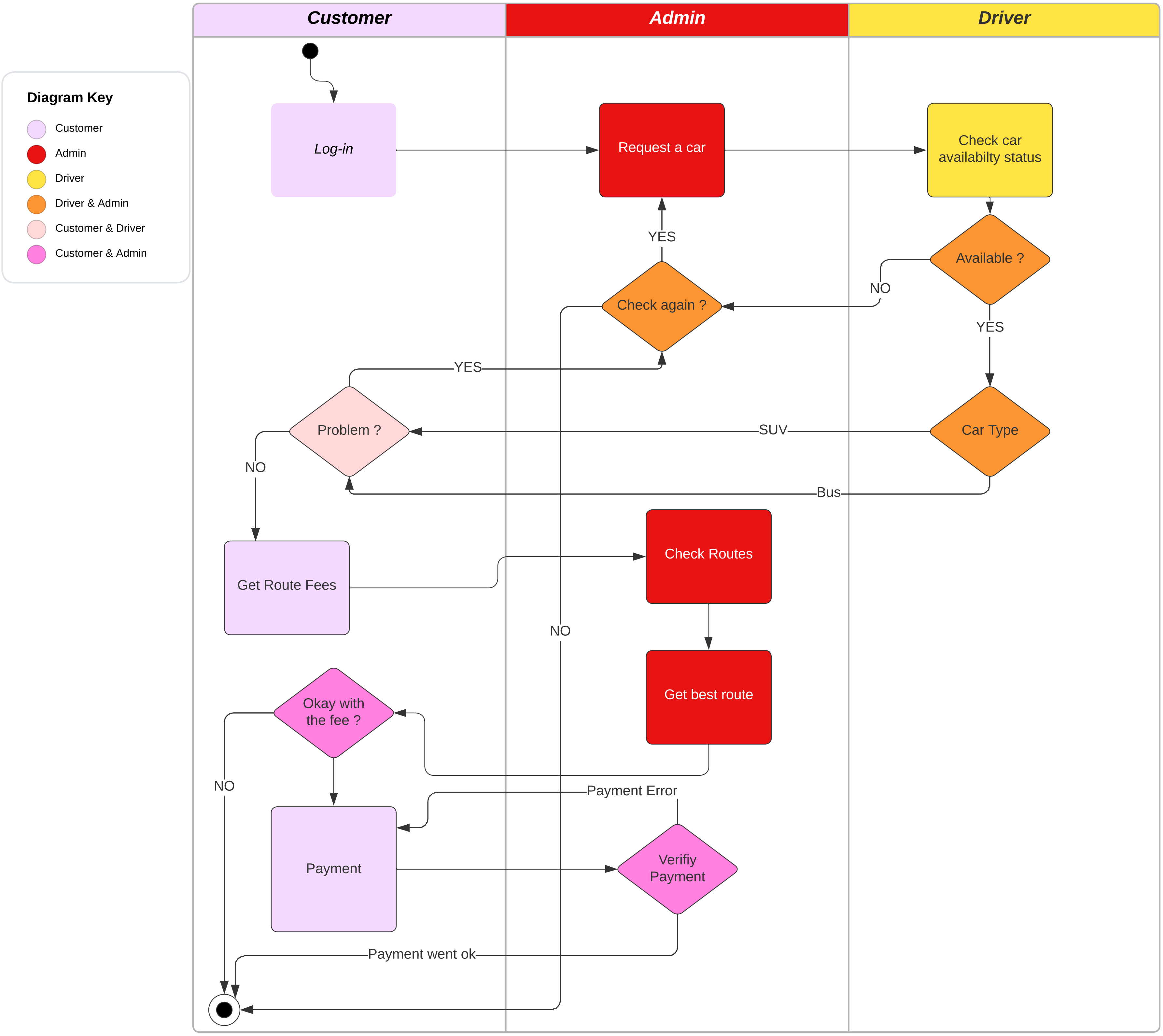


Diagram Key
Customer
Admin
Driver
Driver & Admin
Customer & Driver
Customer & Admin
Customer
Admin
Driver
Log-in
Request a car
Check car availabilty status
Available ?
NO
YES
Check again ?
YES
YES
Car Type
SUV
Bus
Problem ?
NO
Get Route Fees
Check Routes
Get best route
NO
Okay with the fee ?
NO
Payment
Payment Error
Verifiy Payment
Payment went ok