Software Engineering -2

Cover Sheet

Project Title:

Bus Reservation System “swvl”

Project Description:

A Bus Booking System is an online platform that allows users to book bus tickets conveniently from anywhere. It serves as a bridge between passengers and bus operators, providing a seamless booking experience. The system offers various features to streamline the booking process, manage reservations, and ensure a smooth journey for the passengers.

Main Functions:

Convenient way to order a bus and allows customers to browse the Station menu, select items to order customize their reservation if needed, and submit their reservations for **processing, Registration() – Login() –Reservation() Feedback()**
And for the Admin site they can : **AddBus() – UpdateBus() – DeleteBus()**

Project Title: Bus Reservation System

Row Number (in PDF): 12 Time: 11:45

#	Member Name (printed in arabic)	Member Id (printed)	Handwritten signature
1	ابانوب ابراهيم سعد	20210001	
2	يوسف رفاعى عبدالنبى	20211074	
3	تقى محمود ابراهيم	20210244	
4	رحمه هشام هلال	20210329	
5	حسام محمد الدمرداش	20210285	
6	بسمه علاء ابراهيم	20210225	
7			

Project Requirements(grades)

SRS	2	
SDD	2	
Validation	3	
OCL	4	
AOP	4	
Microservices	1	
Cloud	4	

Functional Requirements:

User Registration and Authentication:

Users should be able to register with the system using their email or phone number.

The system should authenticate users securely before allowing access to reservation features.

Bus Management:

Admin should be able to add, update, and delete bus routes.

Each bus should have details such as route, departure time, arrival time, capacity, and fare.

Reservation Management:

Users should be able to search for available buses based on route, date, and time.

Users should be able to reserve seats on a selected bus.

The system should confirm reservations and assign seat numbers to users.

Feedback Management:

Users should be able to provide feedback on their journey experience.

Admin should be able to view and respond to user feedback.

Admin Management:

Admin should have access to a dashboard to manage buses, reservations, and feedback.

Admin should be able to view, add, update, and delete users.

User Profile Management:

Users should be able to view and edit their profile information.

Users should be able to view their reservation history.

Non-Functional Requirements:

Performance:

The system should be responsive even under peak loads.

Bus search and reservation processes should be completed within seconds.

Security:

User authentication and data transmission should be encrypted using industry-standard protocols.

Admin actions should be protected with appropriate access control measures.

Reliability:

The system should minimize downtime and data loss.

It should have mechanisms for data backup and recovery.

Usability:

The system should have an intuitive user interface for easy navigation.

It should be accessible across multiple devices and screen sizes.

Scalability:

The system should be able to handle an increasing number of users and buses without significant performance degradation.

It should support horizontal scaling by adding more servers or instances.

Compatibility:

The system should be compatible with popular web browsers and operating systems.

It should adhere to web standards for compatibility with various devices.

Maintainability:

The system should be designed with modularity and well-documented code for ease of maintenance.

Updates and patches should be applied without disrupting service availability.

Legal and Regulatory Compliance:

The system should comply with data protection laws and regulations.

It should maintain logs of user actions for audit purposes.

OCL CONSTRAINS

(1) context User

inv: self.password.size() >= 8

This constraint will ensure that the password attribute of the
user object has a length
greater than or equal 8.

(2) context Admin

inv: self.name -> notEmpty()

The admin must have a name.

(3) context Bus

inv: self.seats -> notEmpty()

The seats must have a value.

(4) context Feedback

**inv: self.overallRating >= 1 and self.
overallRating <= 5**

The overallRating must be greater than or equal to one and less than or equal to five.

(5) context Bus

inv: self.fare -> forAll (fare | fare.quantity > 0)

A quantity of each fare in the bus must be greater than zero.

(6) context Bus

inv: self.availableSeats() = self.seats - self.reservations->select(status = 'Confirmed')->size()

The number of available seats on a bus must be equal to its capacity minus the number of confirmed reservations.

(7) context User

inv: self.mobile.matches (‘[0-9]{11}’)

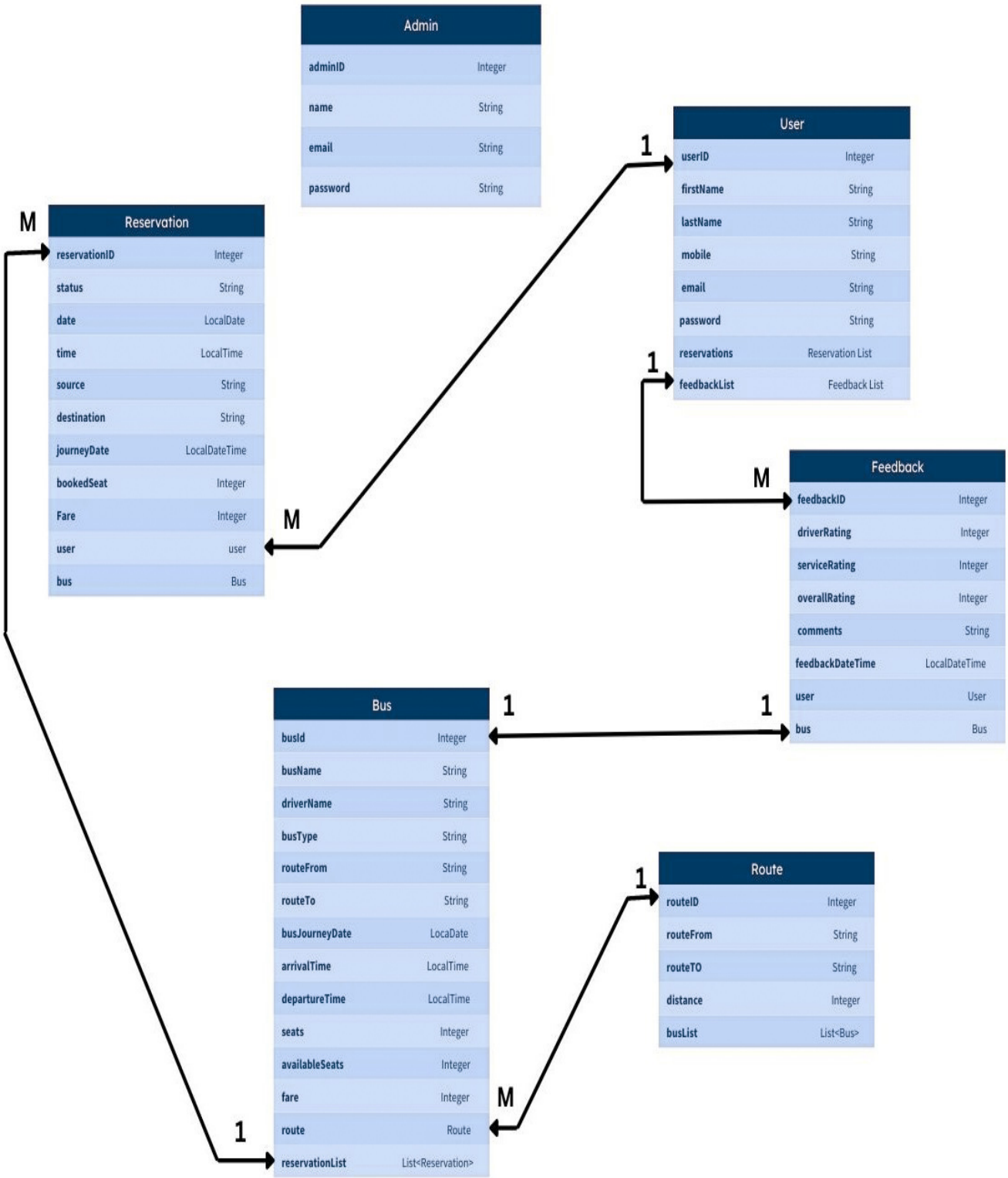
The mobile number must consist of 11 digits.

(8) context User::makeReservation()

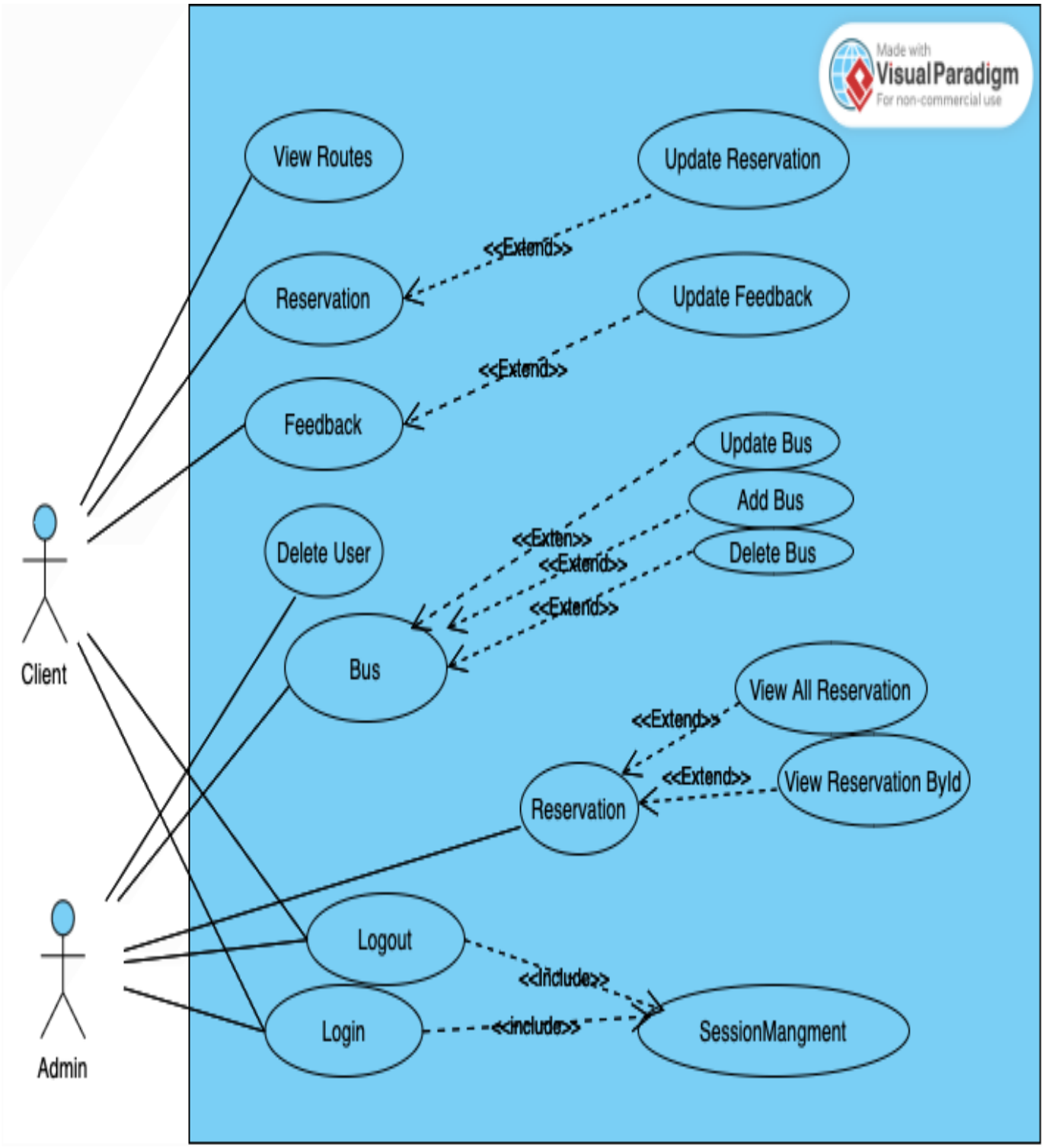
pre: self.loggedin = true

Each user must be logged in before making any reservations.

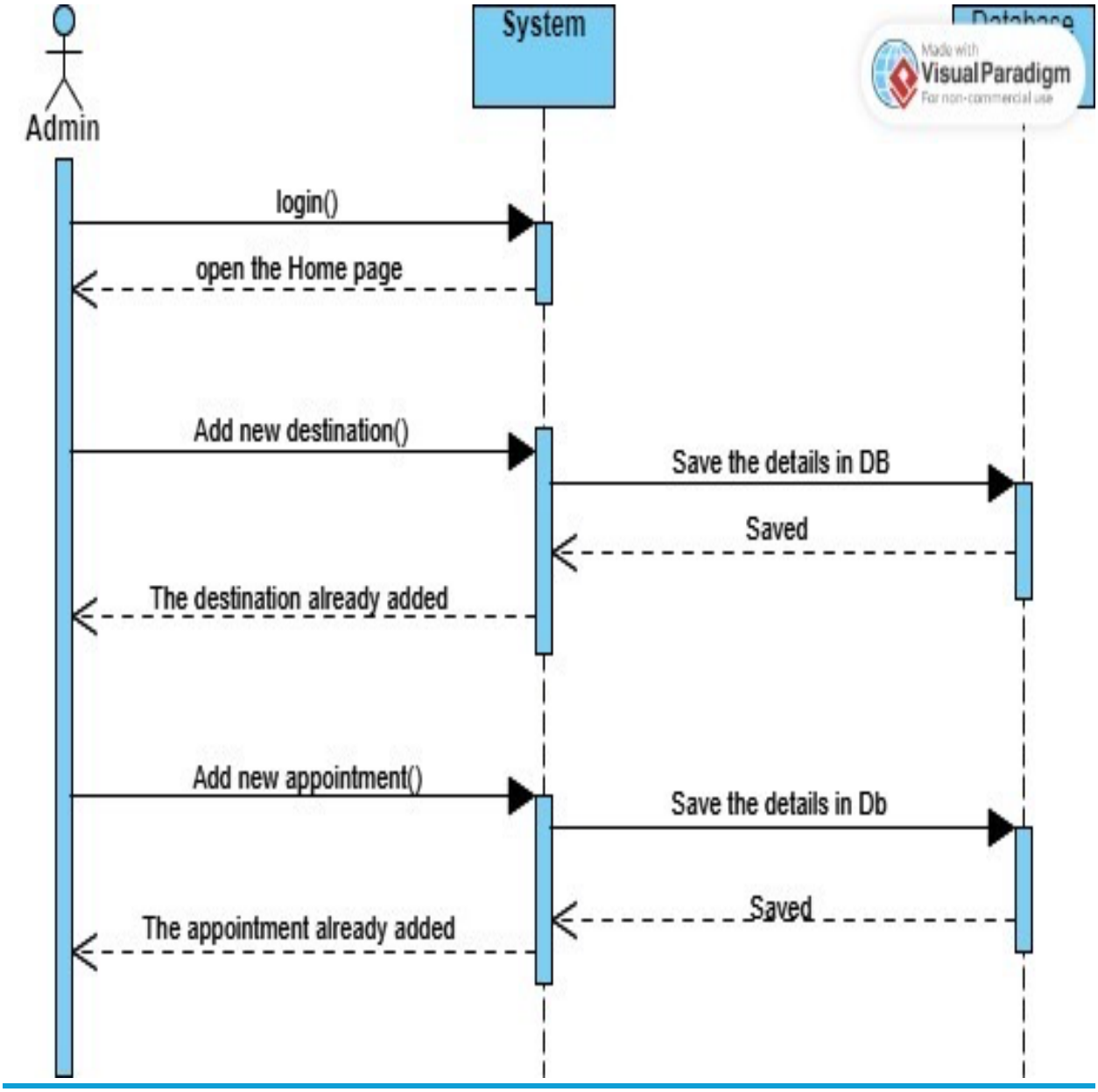
ERD

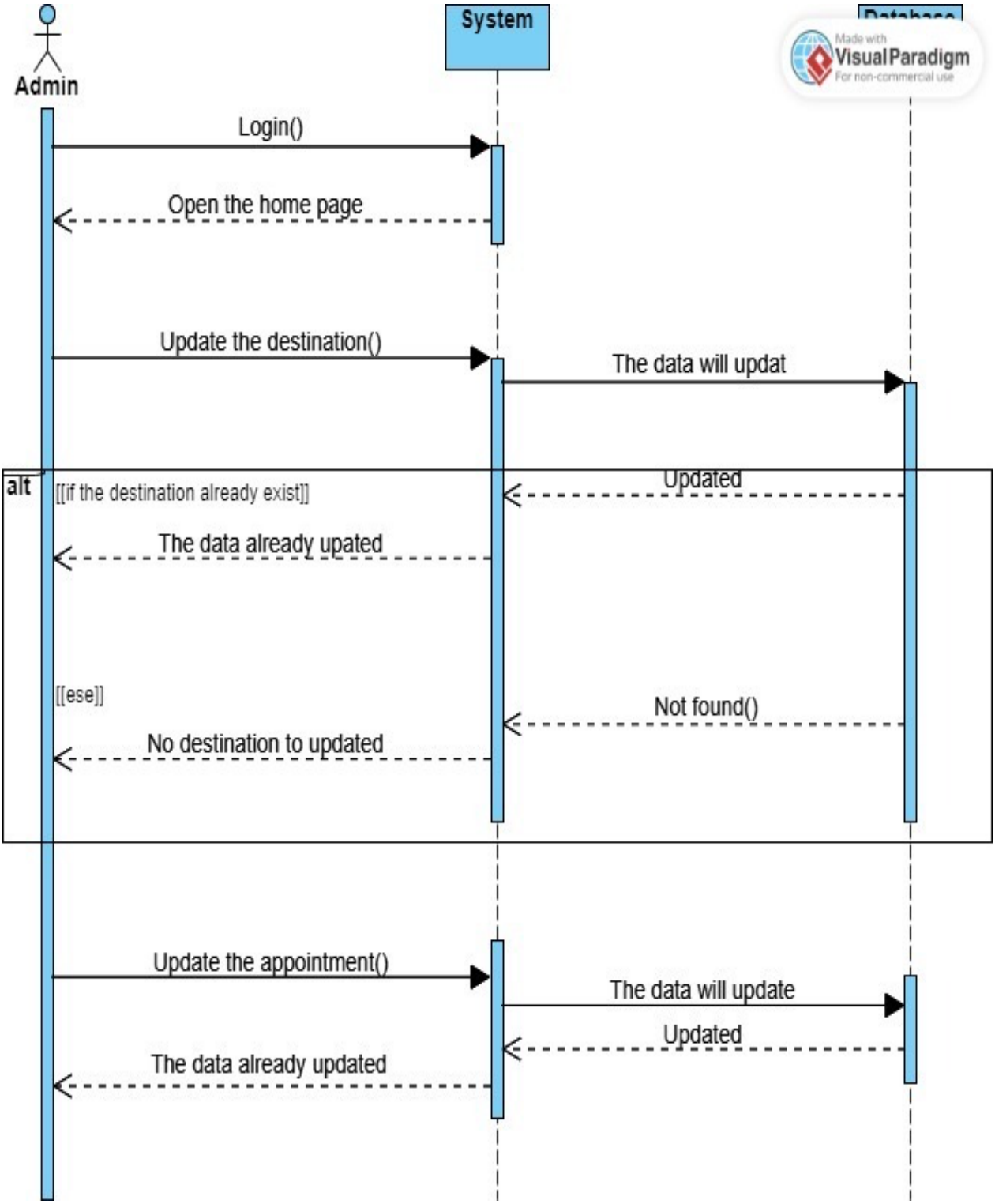

Admin
adminID Integer
name String
email String
password String
User
userID Integer
firstName String
lastName String
mobile String
email String
password String
reservations Reservation List
feedbackList Feedback List
Reservation
reservationID Integer
status String
date LocalDate
time LocalTime
source String
destination String
journeyDate LocalDateTime
bookedSeat Integer
Fare Integer
user user
bus Bus
Feedback
feedbackID Integer
driverRating Integer
serviceRating Integer
overallRating Integer
comments String
feedbackDateTime LocalDateTime
user User
bus Bus
Bus
busId Integer
busName String
driverName String
busType String
routeFrom String
routeTo String
busJourneyDate LocaDate
arrivalTime LocalTime
departureTime LocalTime
seats Integer
availableSeats Integer
fare Integer
route Route
reservationList List<Reservation>
Route
routeID Integer
routeFrom String
routeTO String
distance Integer
busList List<Bus>
M
1
M
1
1
1
1
M
1
M

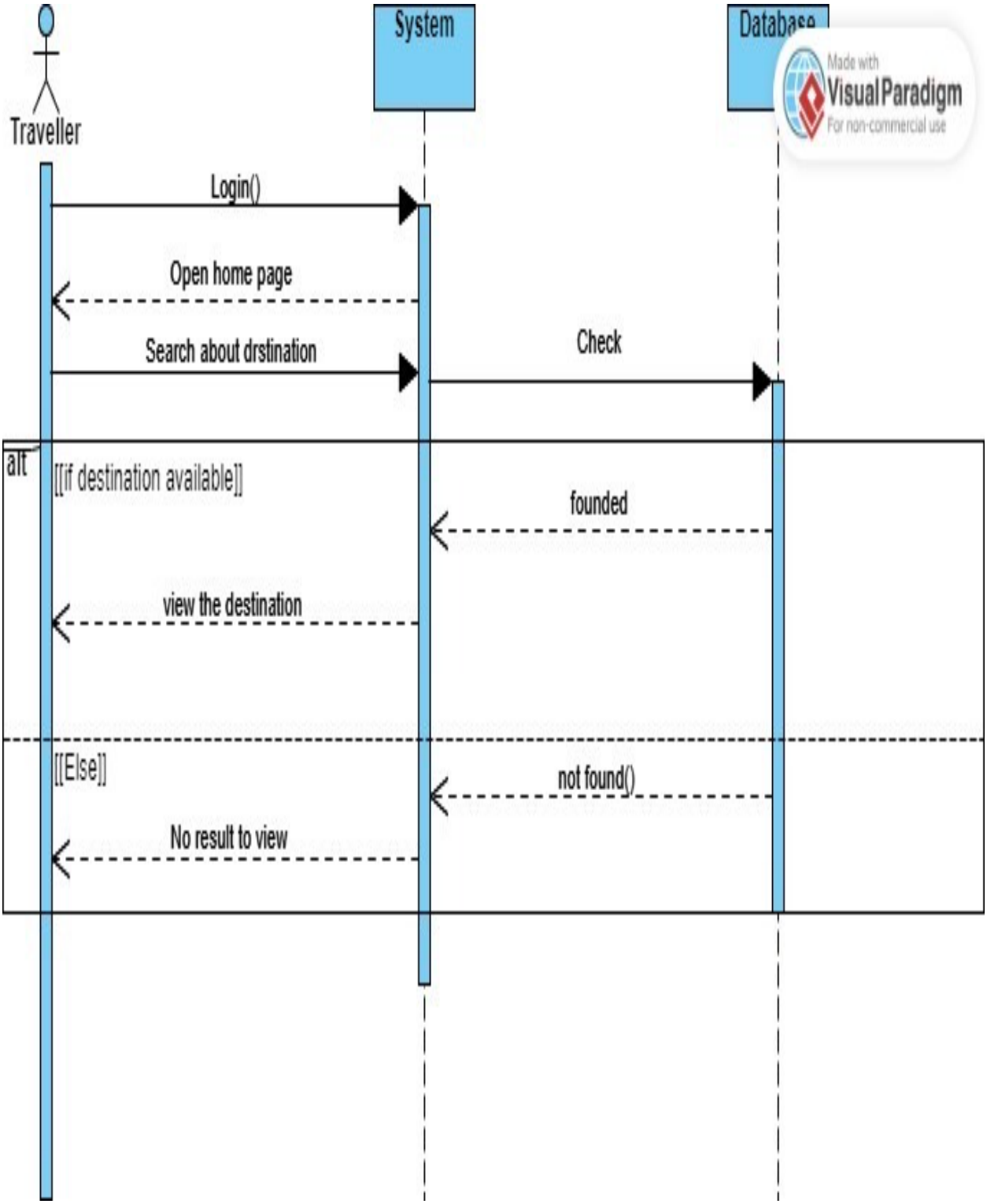
Use Case Diagram

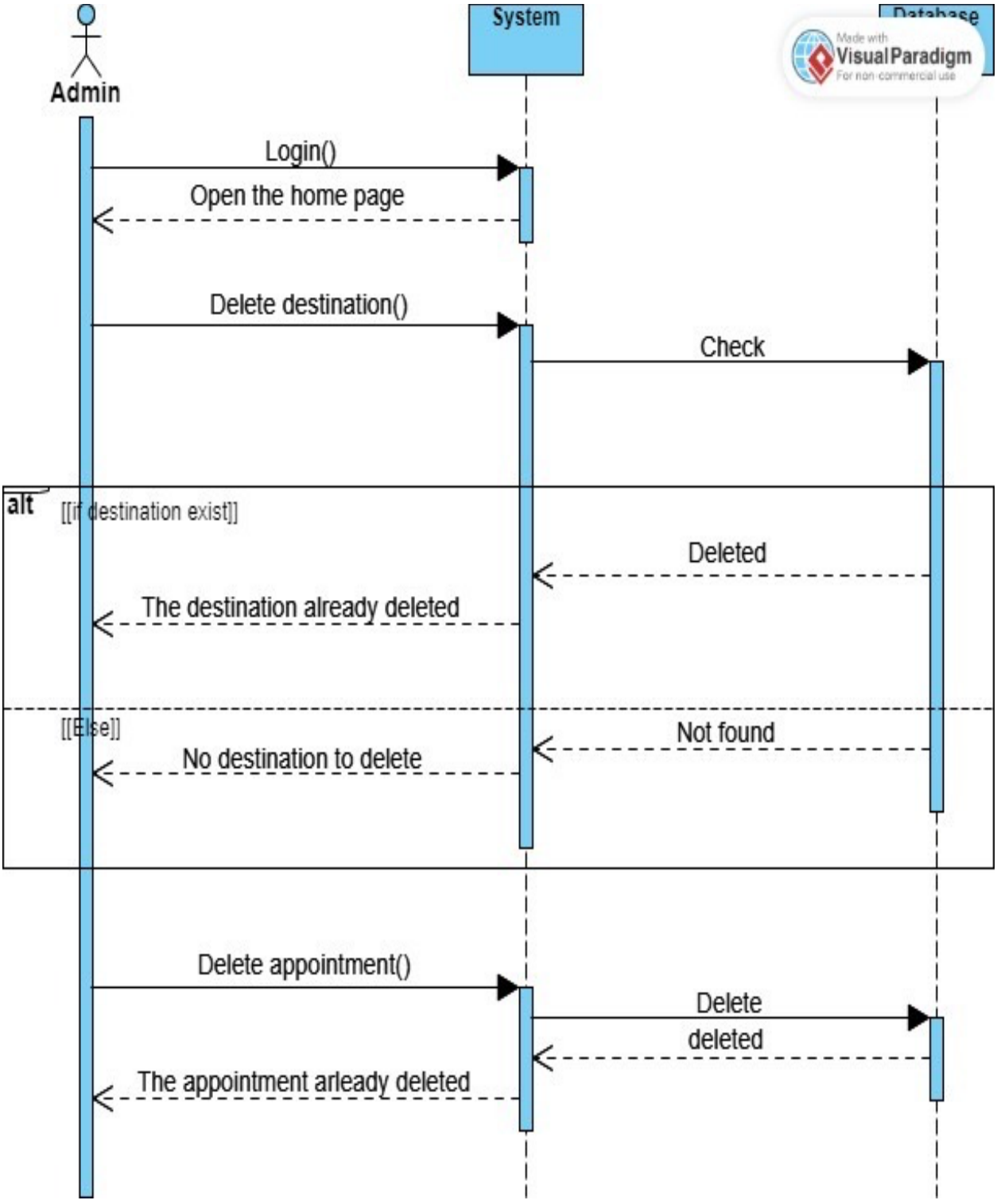


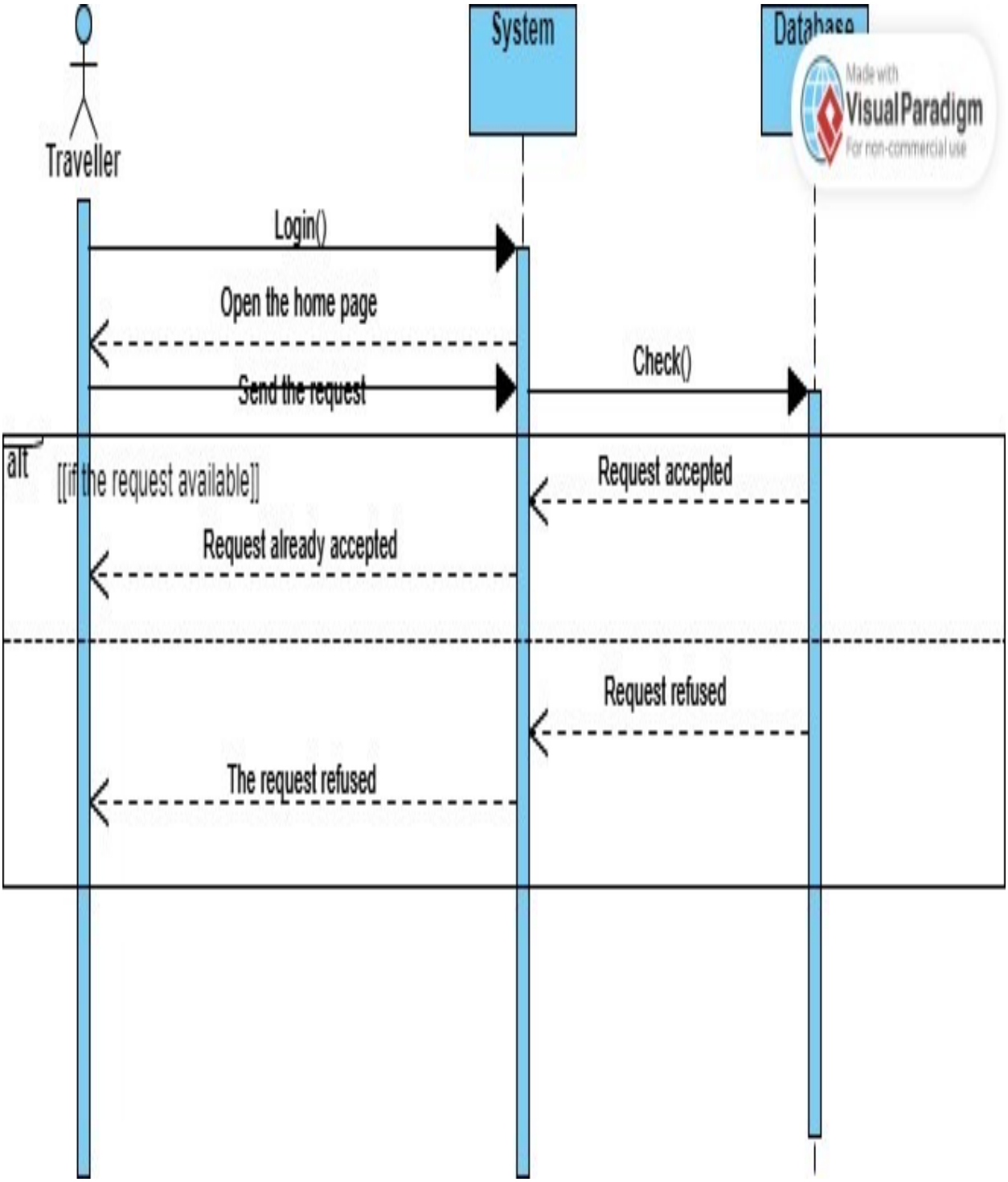
Sequence Class Diagram

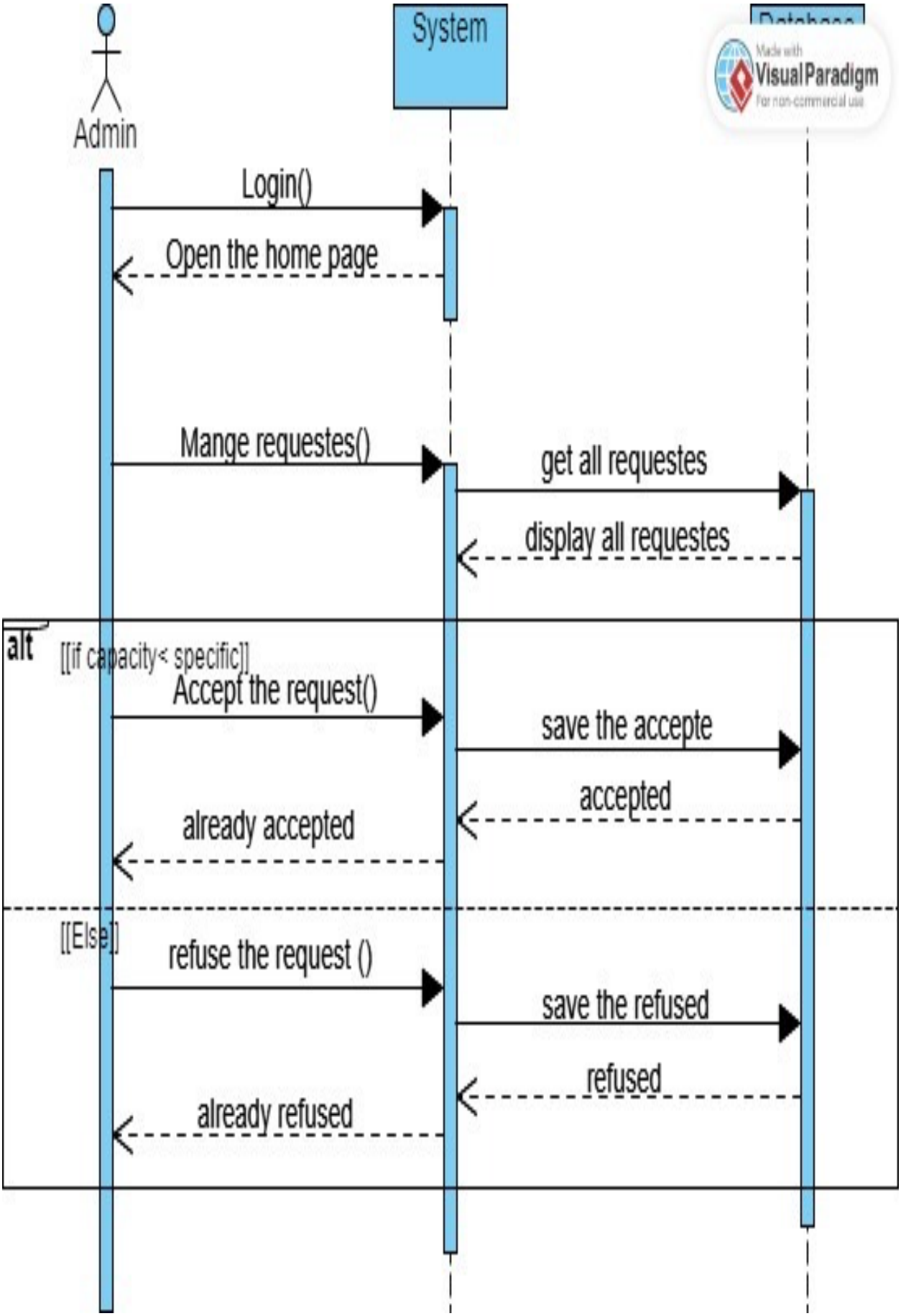



Admin
System
Made with Visual Paradigm For non-commercial use
Login()
Open the home page
Update the destination()
The data will updat
alt
[[if the destination already exist]]
Updated
The data already upated
[[ese]]
Not found()
No destination to updated
Update the appointment()
The data will update
Updated
The data already updated

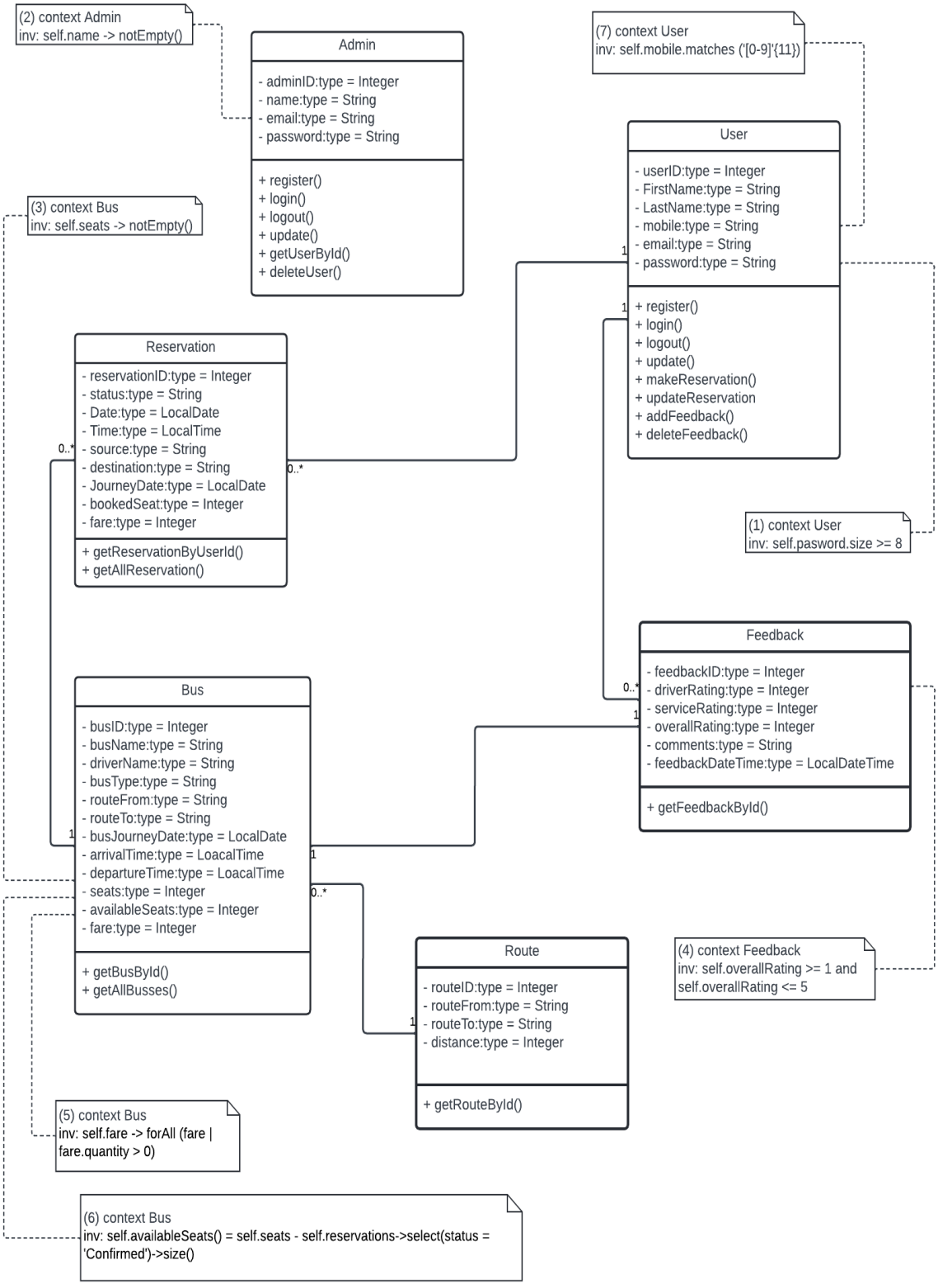

Traveller
System
Database
Made with VisualParadigm For non-commercial use
Login()
Open home page
Search about drstination
Check
alt
[[if destination available]]
founded
view the destination
[[Else]]
not found()
No result to view


Admin
System
Database
Made with VisualParadigm For non-commercial use
Login()
Open the home page
Delete destination()
Check
alt
[[if destination exist]]
Deleted
The destination already deleted
[[Else]]
Not found
No destination to delete
Delete appointment()
Delete
deleted
The appointment arleady deleted

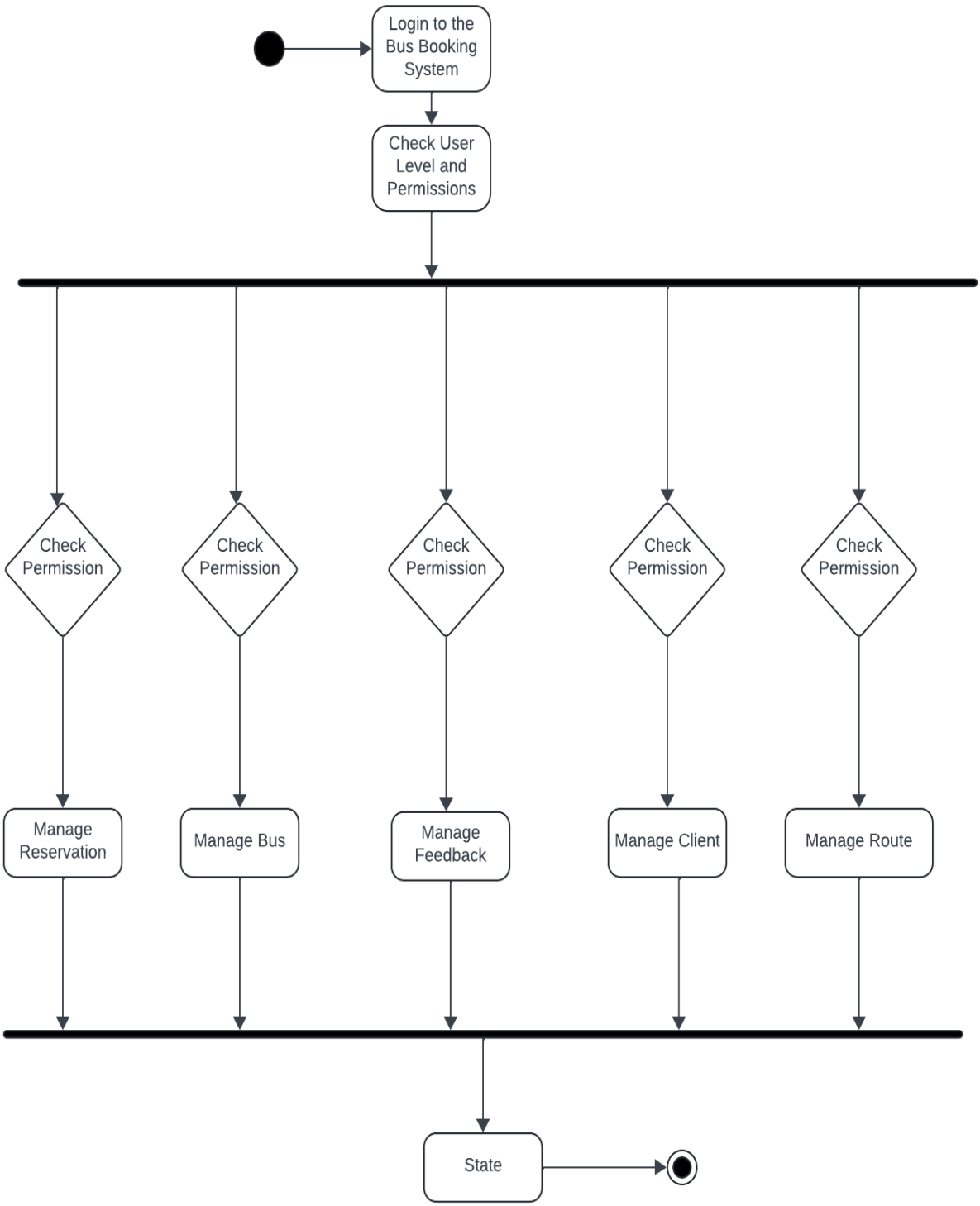

Traveller
System
Database
Made with VisualParadigm
For non-commercial use
Login()
Open the home page
Send the request
Check()
alt
[[if the request available]]
Request accepted
Request already accepted
Request refused
The request refused

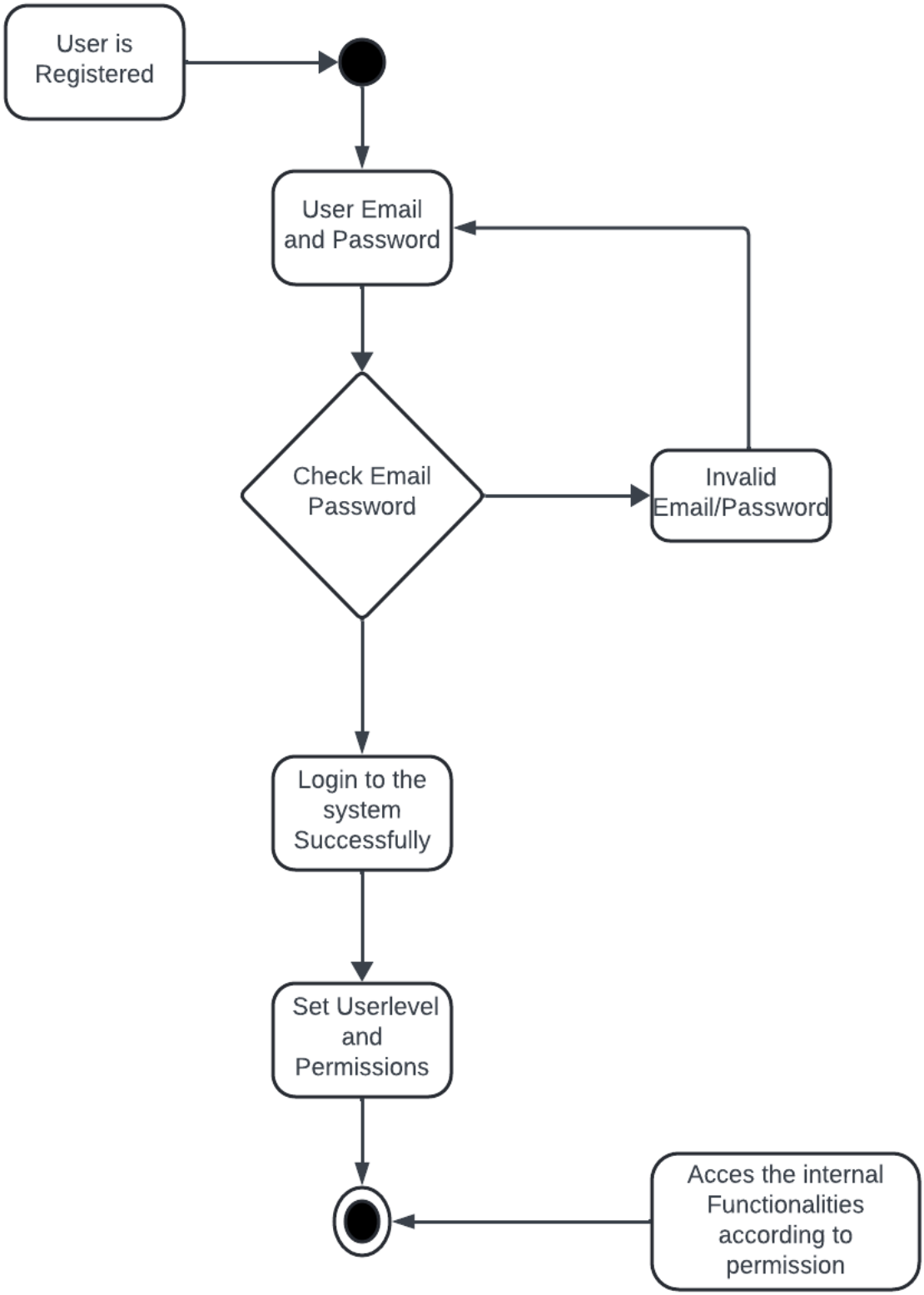

Admin
System
Made with VisualParadigm
For non-commercial use
Login()
Open the home page
Mange requestes()
get all requestes
display all requestes
alt
[[if capacity< specific]]
Accept the request()
save the accepte
accepted
already accepted
[[Else]]
refuse the request ()
save the refused
refused
already refused

Class Diagram



Activity Diagram




User is Registered
User Email and Password
Check Email Password
Invalid Email/Password
Login to the system Successfully
Set Userlevel and Permissions
Acces the internal Functionalities according to permission