

Documentation on

BUS RESERVATION SYSTEM

for JEE Cloud Project Work

By:

Mayank Raj

Tejaswini Bandaru

Aditya Gautam Mishra

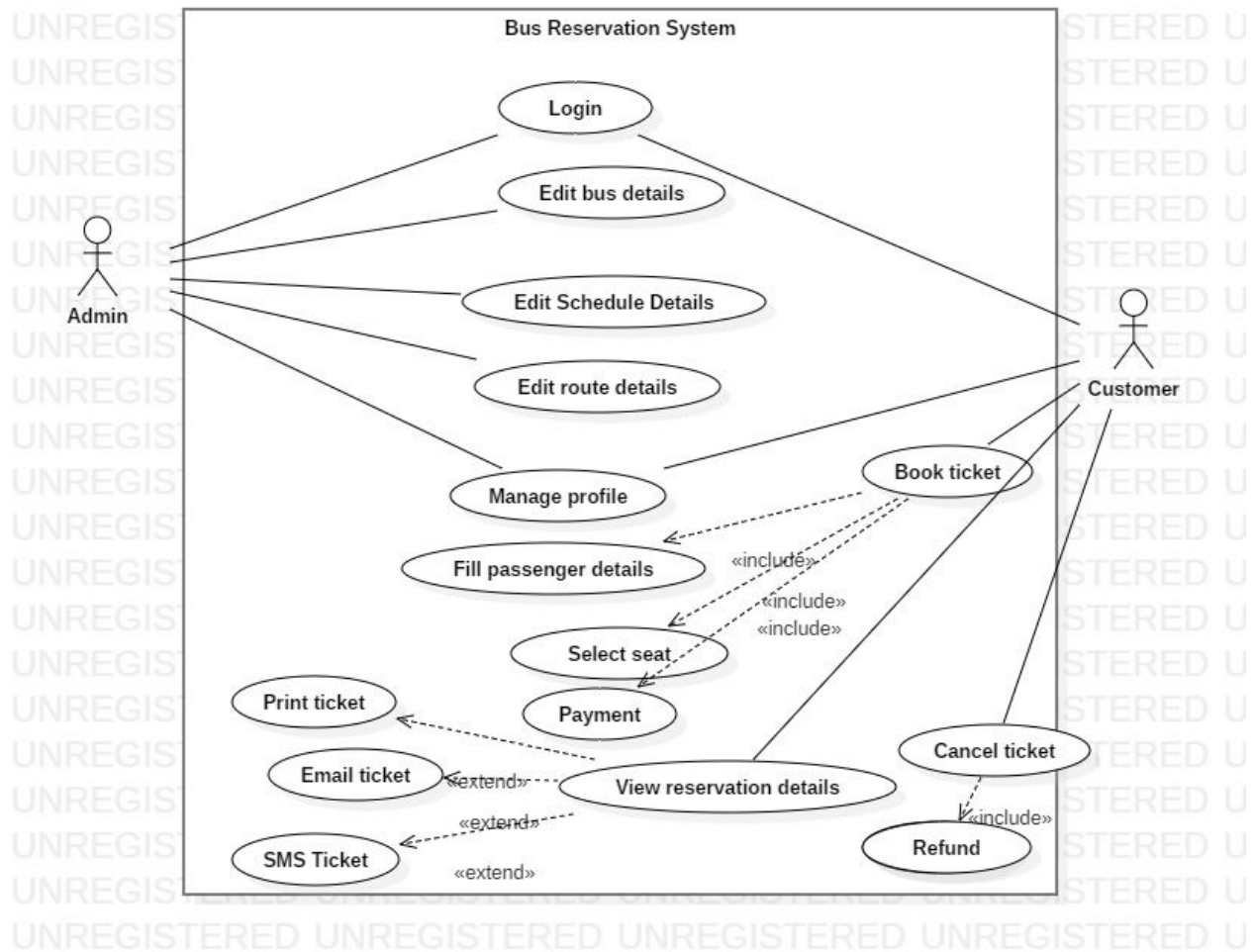
Table of Contents

1. Abstract
2. Use case diagram
3. Classes used:-
 - a. Class Diagram
 - b. Class Description
4. Scope

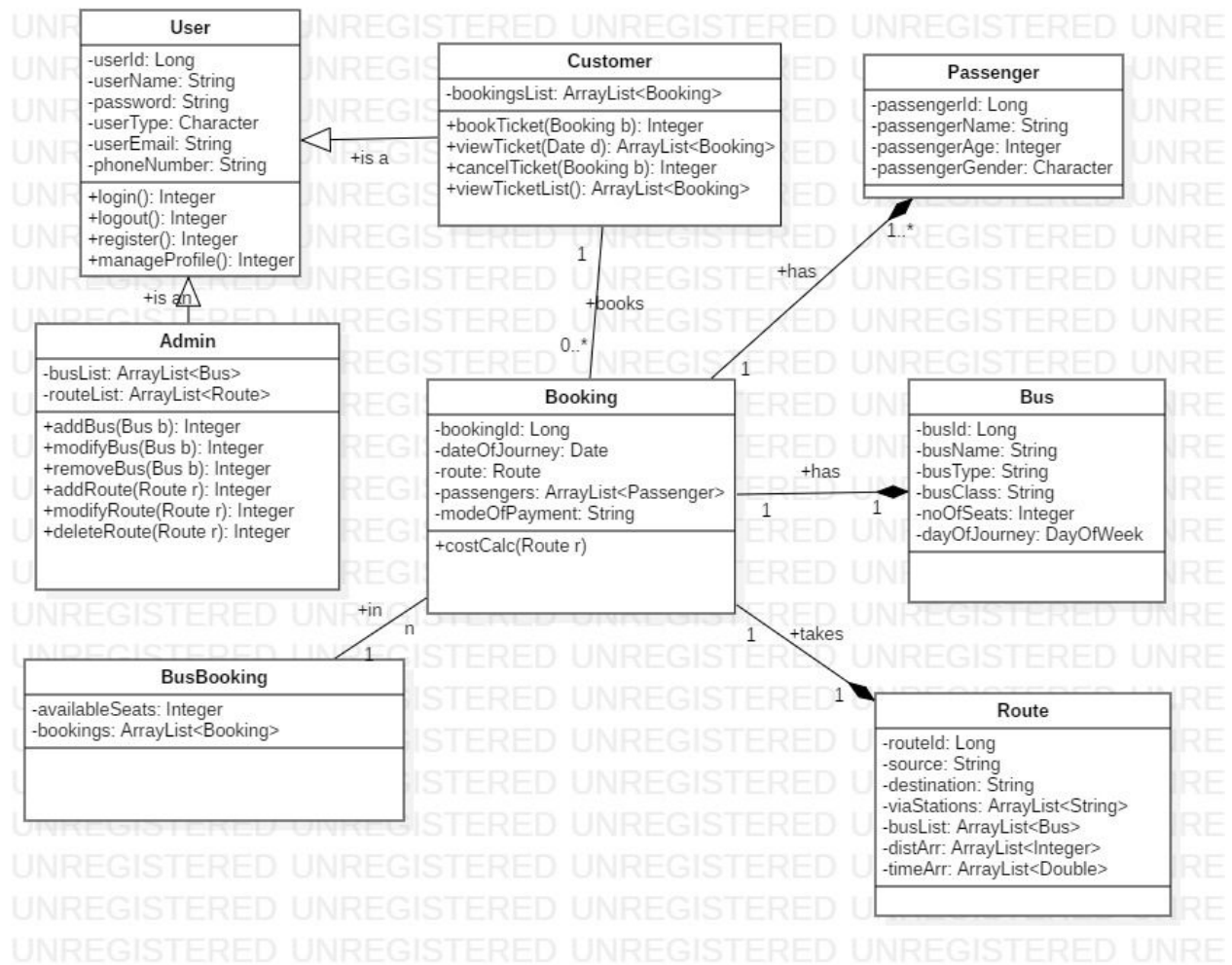
1. Abstract

Bus Reservation System is designed to automate the online ticket purchasing through an easy-to-use online bus booking system. This project is aimed at developing Bus Reservation System for customers, a web based application that can be accessed throughout the web. This system enables customers to login into BRS Application, search Bus details, book tickets for various routes and destinations, view booked tickets, their booking history and cancel any booking. This is an integrated system that contains both the user and the admin. The admin can login to the system using his/her credentials, edit and view bus details; edit and view schedule details; edit and view route details.

2. Use Case Diagram



3a. Class Diagram



3b. Class Description

The following classes are being used in our Bus Reservation System :

1. **User**: General class containing basic properties of user

Attributes:

1. *userId:Long*
2. *Username:String*
3. *Password:String*
4. *userType:Character-'A'* for admin and 'C' for 'Customer'.
5. *userEmail:String*
6. *phoneNumber: String*

Methods:

1. *login():Integer*- Logs the user onto the website. Returns 1 if the login credentials of the user are correct, else
2. *logout():Integer* - Logs out of the website. Returns 1 showing that the user has logged out successfully.
3. *register() :Integer*- Creates an account for the new user. Asks user to fill in the details. Prompts the user to add password of at least 8 characters, minimum 1 uppercase character, 1 lowercase character, 1 digit and 1 special character; phone number should have 10 digits.
4. *manageProfile():Integer* - Lets the user to update their details such as changing their password, updating their phone number.

2. **Admin**: Contains admin object, would perform all admin functions

Attributes:

1. *busList:ArrayList<Bus>*
2. *routeList:ArrayList<Route>*

Methods :

1. *addBus(Bus b) : Integer* :- Adds the details for a new bus. Checks whether all the details have been added properly like busType should be sleeper or non-sleeper, busClass should be AC or non-AC. Returns 1 if all the details have been updated successfully, else 0.
2. *modifyBus(Bus b):Integer*:-Update details of the bus if required. Returns 1 if details are updated successfully, else 0;
3. *removeBus(Bus b):Integer*:- Search for the bus using busId, if the busId is present in the list, removes the corresponding bus details. Removes successfully and returns 1. If busId does not exist in the list, it returns 0;
4. *addRoute(Bus b):Integer*:- Add route details. Checks whether the source and destination are not same.Returns 1, if added successfully, else 0.

5. *modifyRoute(Bus b):Integer*:- Updates route details, such as adding new bus station between source and destination, adding new bus to that route, updating distance and arrival time. Returns 1 if updated successfully, else 0.
6. *deleteRoute(Bus b):Integer*:- Searches the route using routeld, if it is present in the list, deletes successfully and returns 1, else 0;

3. **Customer**: client / customer object

Attributes: *bookingsList:ArrayList<Booking>*

Methods :

1. *bookTicket(Booking b):Integer* - Get the route details, bus details and passenger details from the user. The count of passengers per ticket should not cross more than 5. Date of journey should be either today's date or the date in the future.
2. *viewTicket(Date d):ArrayList<Booking>* - Lists the tickets based on their travel date.
3. *cancelTicket(Booking b):Integer* - Searches the ticket from the booking history using ticketID and cancels it, and updates the booking history.
4. *viewTicketList():ArrayList<Booking>* - Lists all the bookings made by the customer.

4. **Passenger**: details of individual passenger undertaking the journey

Attributes:

1. *passengerId:Long*
2. *passengerName:String*
3. *passengerAge:Integer*
4. *passengerGender:Character*-'M' for male, 'F' for female

5. **Booking** : object to store booking made by customers

Attributes:

1. *bookingId:Long*
2. *dateOfJourney:Date*
3. *route:Route*
4. *passengers:ArrayList<Passenger>*
5. *modeOfPayment:String*

Methods :

1. *costCalc(Route r):Double*-Updates the count of availableSeats by reducing the noOfSeats with the passenger count and calculates the final cost, including tax.

6. **Bus**: contains attributes of bus type

Attributes:

1. *busId:Long*
2. *busName:String*
3. *busType:String*-Sleeper or Semi-Sleeper
4. *busClass:String*-AC or non-AC
5. *noOfSeats:Integer*
6. *dayOfJourney:DayOfWeek*-On which days of the week, the bus will run.

7. **BusBooking** : contains object for booking of bus on that particular day

Attributes:

1. *bookings*: *ArrayList<Booking>*
2. *availableSeats*: *Integer*

8. **Route** : contains information about routes

Attributes:

1. *routeId*:Long
2. *source*:String
3. *destination*:String
4. *viaStations*:ArrayList<String>
5. *busList*:ArrayList<Bus>
6. *distArr*:ArrayList<Integer>:-Distance from source to destination are stored.
7. *timeArr*:ArrayList<Double>:Time taken from source to destination are stored.

4. Scope

1. Payments are not being covered at present.
2. Reservation quota (Women, Handicapped or Senior Citizen) are currently not implemented.
3. Limited routes are being used in this model.