

# ***Bus Reservation and Locator System Software Design Specification***

BY

LAKESH KUMAR, CMS. ID: 053-24-0036

MUNESH KUMAR, CMS. ID: 053-24-0037

YOUGESH KUMAR, CMS. ID: 053-24-0026



DEPARTMENT OF COMPUTER SCIENCE  
FACULTY OF SCIENCE AND INFORMATION TECHNOLOGY  
SUKKUR IBA UNIVERSITY

DECEMBER 2025

## *Software Design Specification Document*

1. INTRODUCTION .....	1
1.1. Purpose of Document.....	1
1.2. Definitions, acronyms, and abbreviations.....	1
2. OVERALL SYSTEM DESCRIPTION .....	2
2.1. Project Background.....	2
2.2. Project Objectives .....	2
3. SYSTEM ARCHITECTURE.....	3
4. DOMAIN MODEL .....	5
5. CLASS DIAGRAM .....	7
6. DATABASE DIAGRAM .....	9
7. ENTITY RELATIONSHIP DIAGRAM (ERD) .....	10
8. SEQUENCE DIAGRAM.....	12
9. SYSTEM INTERFACE DESIGN.....	16
10. TEST CASES.....	20

# 1. Introduction

## 1.1. Purpose of Document

The purpose of this Software Design Specification (SDS) is to describe the detailed design of the **Bus Reservation and Locator System**.

- Overall architecture of the system
- Main components and their responsibilities
- Data model and database design
- Interaction between classes and modules
- Sequence of events for major features
- User interface structure
- Test case outline for system verification

This document will be used by:

- **Developers** – to implement frontend, backend, and database.
- **Testers** – to design and execute test cases.
- **Project Supervisor / Stakeholders** – to review and verify that design matches requirements.

## 1.2. Definitions, acronyms, and abbreviations

- **BRLS** – Bus Reservation and Locator System
- **SRS** – Software Requirements Specification
- **SDS** – Software Design Specification
- **GPS** – Global Positioning System
- **API** – Application Programming Interface
- **UI** – User Interface
- **UX** – User Experience
- **DB** – Database
- **SQL** – Structured Query Language
- **CRUD** – Create, Read, Update, Delete
- **User** – Passenger using the system to search, book, and track buses
- **Operator** – Bus company staff managing buses, routes, schedules
- **Admin** – System administrator managing users, operators, and overall system settings

## 2. Overall System Description

### 2.1. Project Background

In many cities, bus systems are still managed manually.

- No clear bus schedule
- No information about available seats
- No live updates about delays
- Long waiting time at bus stops

Bus operators find it difficult to:

- Manage routes and schedules efficiently
- Track bus status in real time
- Monitor seat utilization and bookings

To solve these issues, a **Bus Reservation and Locator System** is being developed as a **web and Android-based application**. It will allow passengers to book seats online and track buses using GPS, while operators can manage buses, routes, and bookings from a single platform.

### 2.2. Project Objectives

The main objectives of the system are:

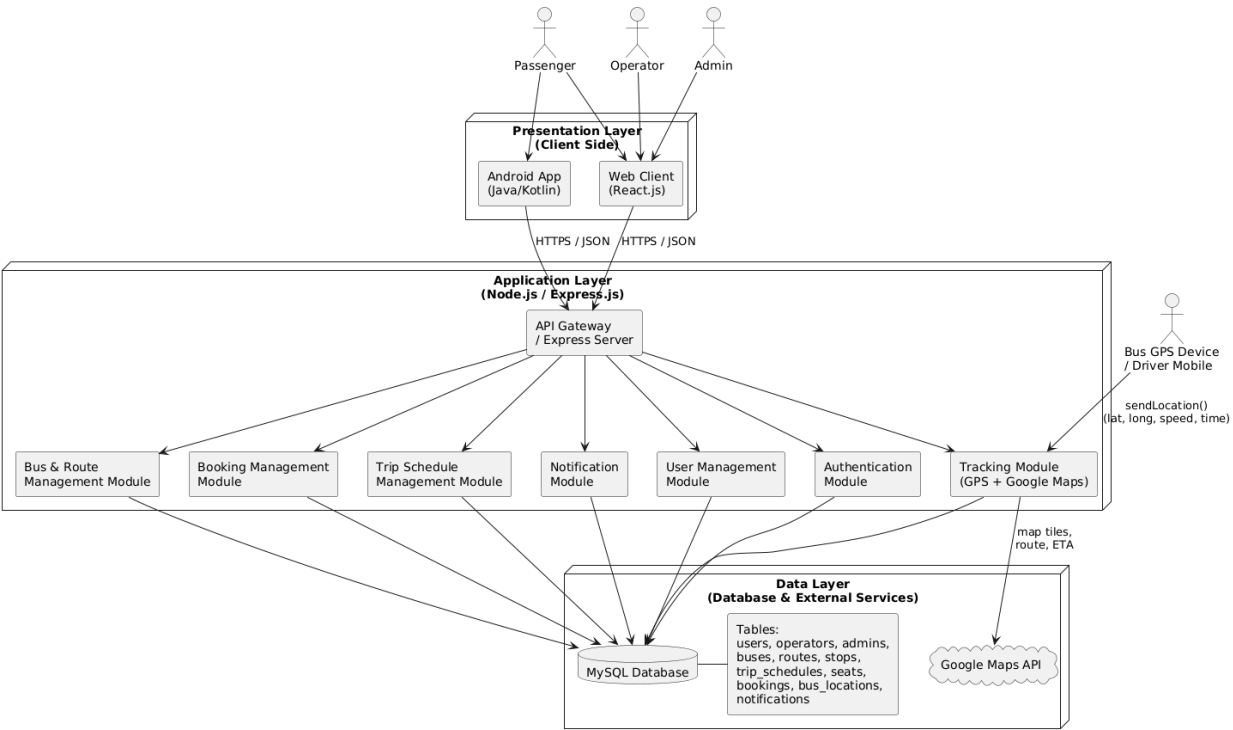
- To provide an easy and user-friendly platform for **online bus seat reservation**.
- To show **real-time bus location** using GPS and Google Maps API.
- To allow passengers to view **bus routes, timings, available seats, and expected arrival time**.
- To help operators manage **routes, schedules, bus status, and seat availability**.
- To reduce waiting time, improve convenience, and increase transparency in bus services.

### 3. System Architecture

- The system follows a **3-tier architecture**:
- **Presentation Layer (Client Side)**
  - Technologies:
    - **Frontend (Web):** React.js
    - **Data Fetching:** AJAX / Fetch API for asynchronous communication with backend APIs
    - **Mobile Application:** Android (Java/Kotlin)
    - **Maps & GPS Services:** Google Maps API + Bus GPS Device / Mobile GPS
    - **Authentication:** JWT-based login system
  - Responsible for:
    - Displaying UI to the user
    - Taking user input (login, search, booking, tracking)
    - Calling backend APIs over HTTP/HTTPS
- **Application Layer (Server Side / Backend)**
  - Technology: **Node.js** (Express.js)
  - Responsible for:
    - Implementing business logic (searching buses, checking seat availability, creating bookings)
    - Validating user inputs
    - Managing authentication and authorization using **JWT**
    - Communicating with database and third-party APIs (Google Maps)
- **Data Layer (Database & External Services)**
  - Technology: **MySQL** (SQL Database)
  - Stores:
    - Users, Operators, Admins
    - Buses, Routes, Stops
    - Schedules / Trips
    - Bus Locations (GPS coordinates, speed, timestamp)
  - External Service:
    - **Google Maps API** for map display and GPS tracking
    - GPS device or mobile app on bus to send location data
    - **Motion Sensor** Detects whether the bus is moving or stationary (optional enhancement for Android GPS mode)
- **High-Level Components**
- **Frontend Screens (React / Android)**
  - Login / Sign Up
  - Search Bus Screen
  - Seat Selection Screen
  - Booking Confirmation Screen
  - Bus Tracking Map Screen
  - Operator Dashboard
  - Admin Panel
- **Backend Modules (Node JS)**
  - Authentication Module
  - User Management Module
  - Bus & Route Management Module
  - Booking Management Module
  - Trip Schedule Management Module
  - Tracking Module (GPS data processing + Google Maps API integration)
  - Notification Module
- **Database**
  - Tables for: users, operators, buses, routes, bus stops, trip schedules, seats, bookings, bus locations, notifications.

Software Design Specification Document

3-Tier System Architecture - Bus Reservation and Locator System



## 4. Domain Model

The domain model represents main **real-world entities** and their relationships.

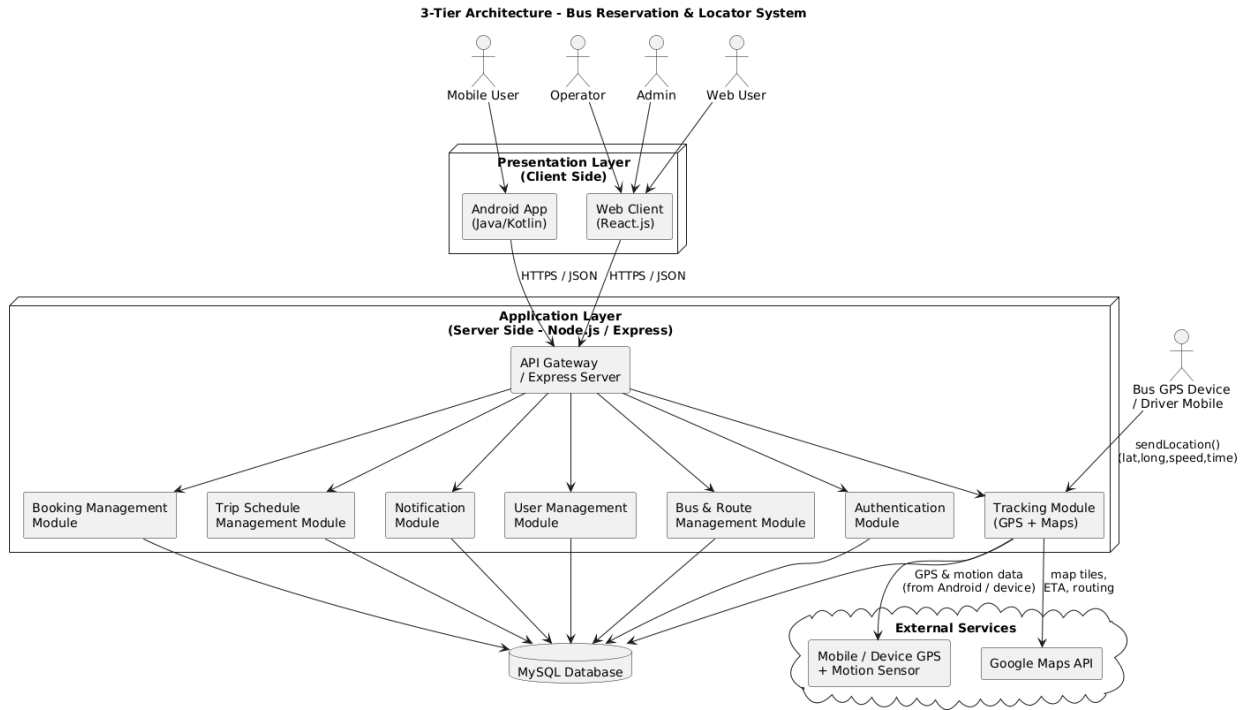
### Main Entities:

- **User (Passenger)**
  - Attributes: userId, name, email, phone, passwordHash, role, createdAt
- **Operator**
  - Attributes: operatorId, name, contactInfo, companyName, status
- **Admin**
  - Attributes: adminId, name, email, passwordHash
- **Bus**
  - Attributes: busId, busNumber, registrationNumber, capacity, operatorId, status
- **Route**
  - Attributes: routeId, routeName, source, destination, totalDistance, approxDuration
- **BusStop**
  - Attributes: stopId, stopName, latitude, longitude, sequenceNumber, routeId
- **TripSchedule (or Trip / Journey)**
  - Attributes: tripId, busId, routeId, departureTime, arrivalTime, date, status
- **Seat**
  - Attributes: seatId, busId, seatNumber
- **Booking**
  - Attributes: bookingId, userId, tripId, seatId, bookingTime, status (Booked/Cancelled)
- **BusLocation**
  - Attributes: locationId, busId, latitude, longitude, speed, timestamp
- **Notification**
  - Attributes: notificationId, userId, message, type, createdAt, isRead

### Relationships (Conceptually):

- A **User** can make **many Bookings**.
- A **Bus** belongs to one **Operator** but can have many **TripSchedules**.
- A **Route** has many **BusStops**.
- A **TripSchedule** uses one **Bus** and one **Route**.
- A **TripSchedule** has many **Bookings**.
- A **Bus** has many **Seats**.
- A **Bus** has many **BusLocation** entries over time.

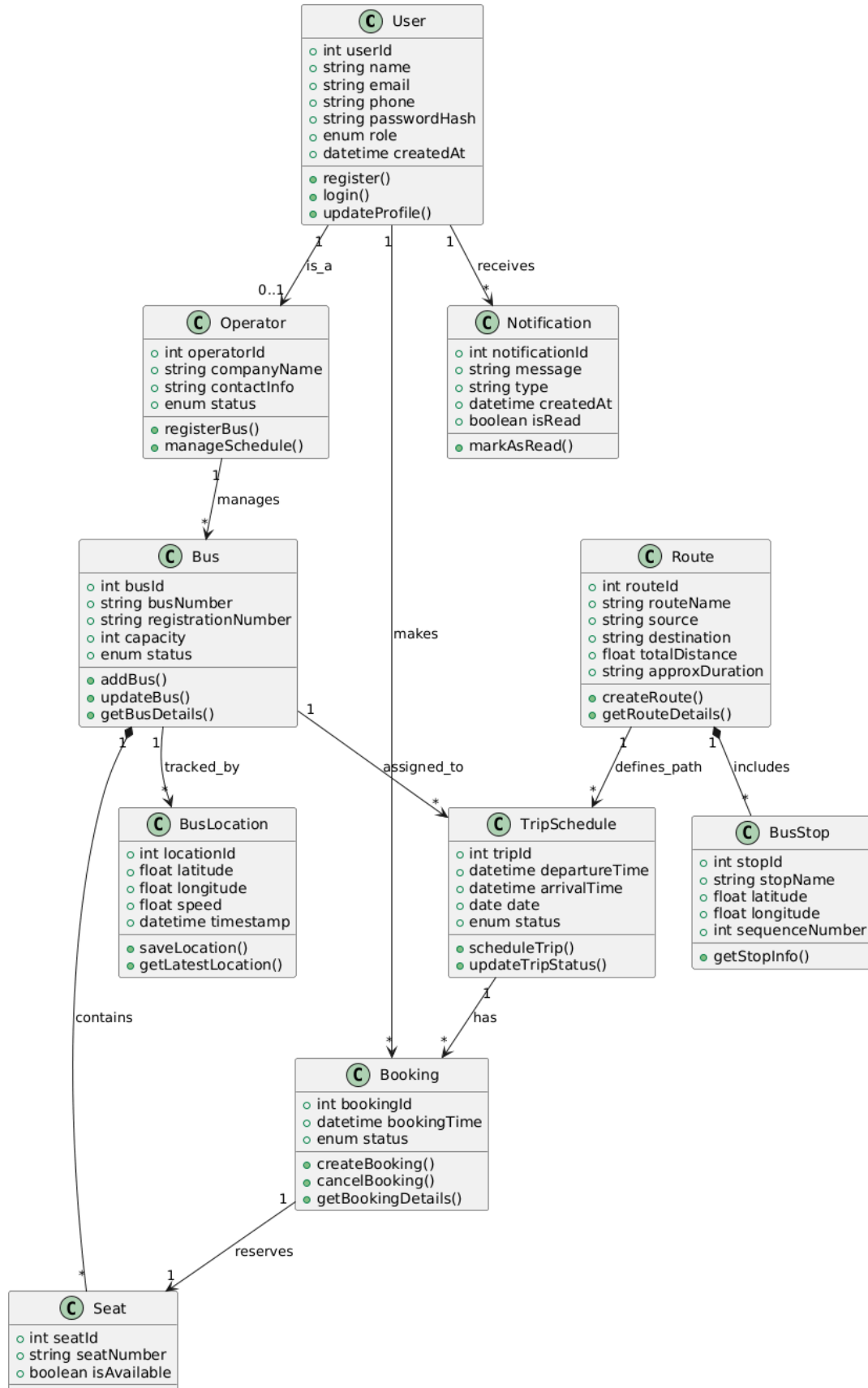
Software Design Specification Document





## 5. Class Diagram

- **User**  
Represents all system users (Passenger, Operator, Admin).  
Stores personal and login information and provides basic operations such as registration and login.
- **Operator (inherits User)**  
Extends User with operator-specific details such as company information and status.  
Responsible for managing buses and trip schedules.
- **Notification**  
Stores system messages sent to users (e.g., delay, cancellation).  
Each user can receive multiple notifications.
- **Bus**  
Represents a physical bus with capacity and operational status.  
Managed by an operator and linked with trip schedules.
- **BusLocation**  
Stores the latest GPS coordinates of a bus (latitude, longitude, speed, timestamp).  
Used for real-time tracking.
- **Route**  
Defines the path from a source to a destination along with distance and duration.  
Consists of multiple bus stops.
- **BusStop**  
Represents a stop along a route, including its sequence number and coordinates.
- **TripSchedule**  
Describes a scheduled trip for a specific bus on a given route and date.  
Contains departure/arrival times and trip status.
- **Seat**  
Represents an individual seat in a bus.  
Each bus contains multiple seats that can be reserved.
- **Booking**  
Links the user, trip schedule, and seat together to form a reservation.  
Stores booking time and status (Booked/Cancelled).



## **6. Database Diagram**

Database Diagram will illustrate table relationships aligned with the ERD

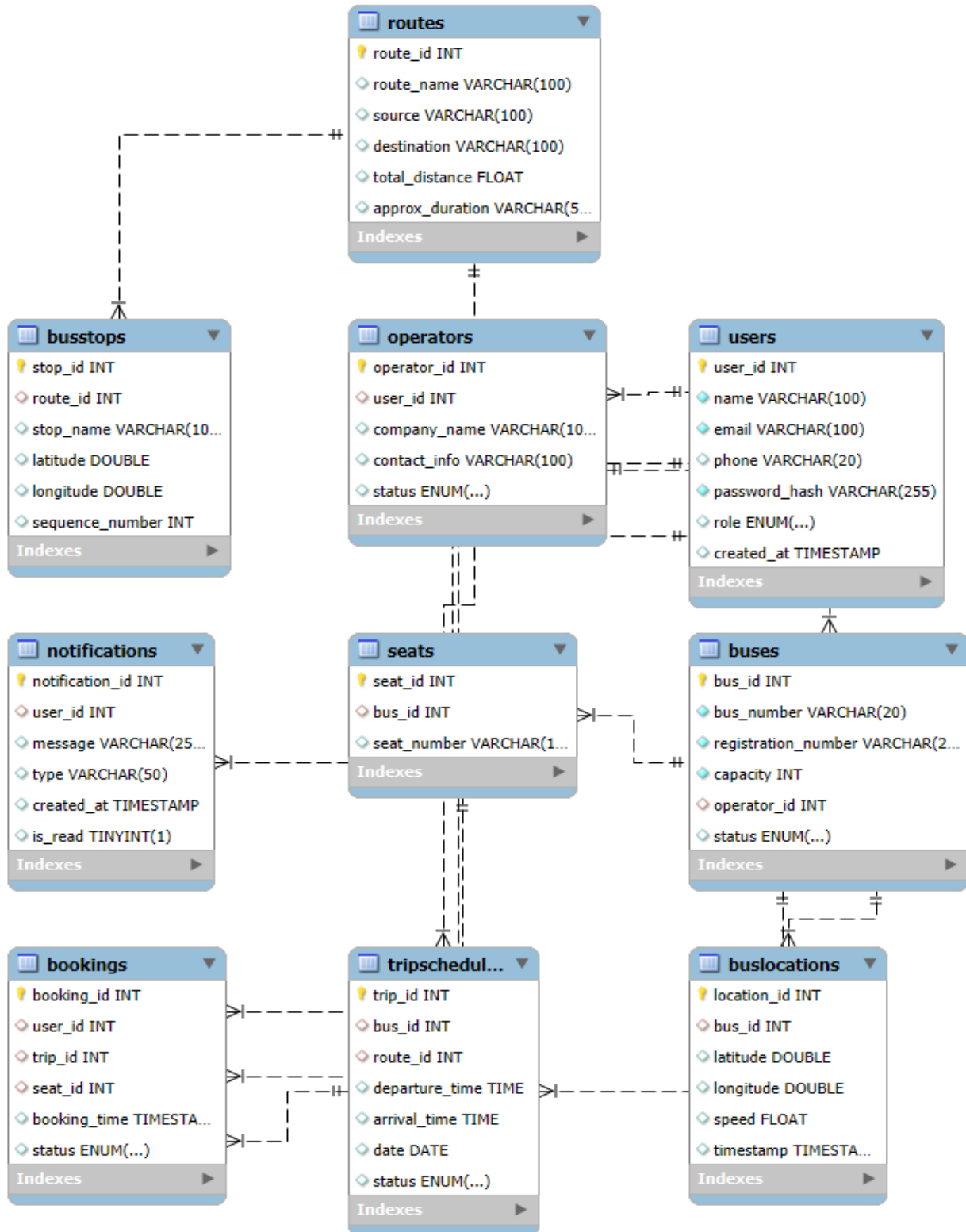
## 7. Entity Relationship Diagram (ERD)

The main entities are:

- **User** (userId PK) – passenger who searches, books, and tracks buses.
- **Operator** (operatorId PK) – manages buses, routes, and schedules.
- **Bus** (busId PK, operatorId FK → Operator) – physical bus with a specific capacity.
- **Route** (routeId PK) – defines source, destination, and overall path.
- **Bus Stop** (stopId PK, routeId FK → Route) – individual stops on a route.
- **TripSchedule** (tripId PK, busId FK → Bus, routeId FK → Route) – one scheduled trip of a bus on a route at a particular date/time.
- **Seat** (seatId PK, busId FK → Bus) – individual seat on a bus.
- **Booking** (bookingId PK, userId FK → User, tripId FK → TripSchedule, seatId FK → Seat) – reservation of a specific seat on a specific trip for a user.
- **BusLocation** (locationId PK, busId FK → Bus) – GPS positions of a bus over time.
- **Notification** (notificationId PK, userId FK → User) – messages sent to users about bookings, delays, etc.

Main relationships:

- **User – Booking:** one User (1) can have many Bookings (N).
- **Bus – TripSchedule:** one Bus (1) can be assigned to many TripSchedules (N).
- **Route – TripSchedule:** one Route (1) can have many TripSchedules (N).
- **Route – Bus Stop:** one Route (1) contains many BusStops (N).
- **TripSchedule – Booking:** one TripSchedule (1) can have many Bookings (N).
- **Bus – Seat:** one Bus (1) has many Seats (N).
- **Bus – BusLocation:** one Bus (1) has many BusLocation records (N) over time.



## 8. Sequence Diagram

### 8.1 Sequence Diagram – Search and Book Bus

**Description:**

This sequence diagram shows how a passenger searches for available buses and completes a seat booking.

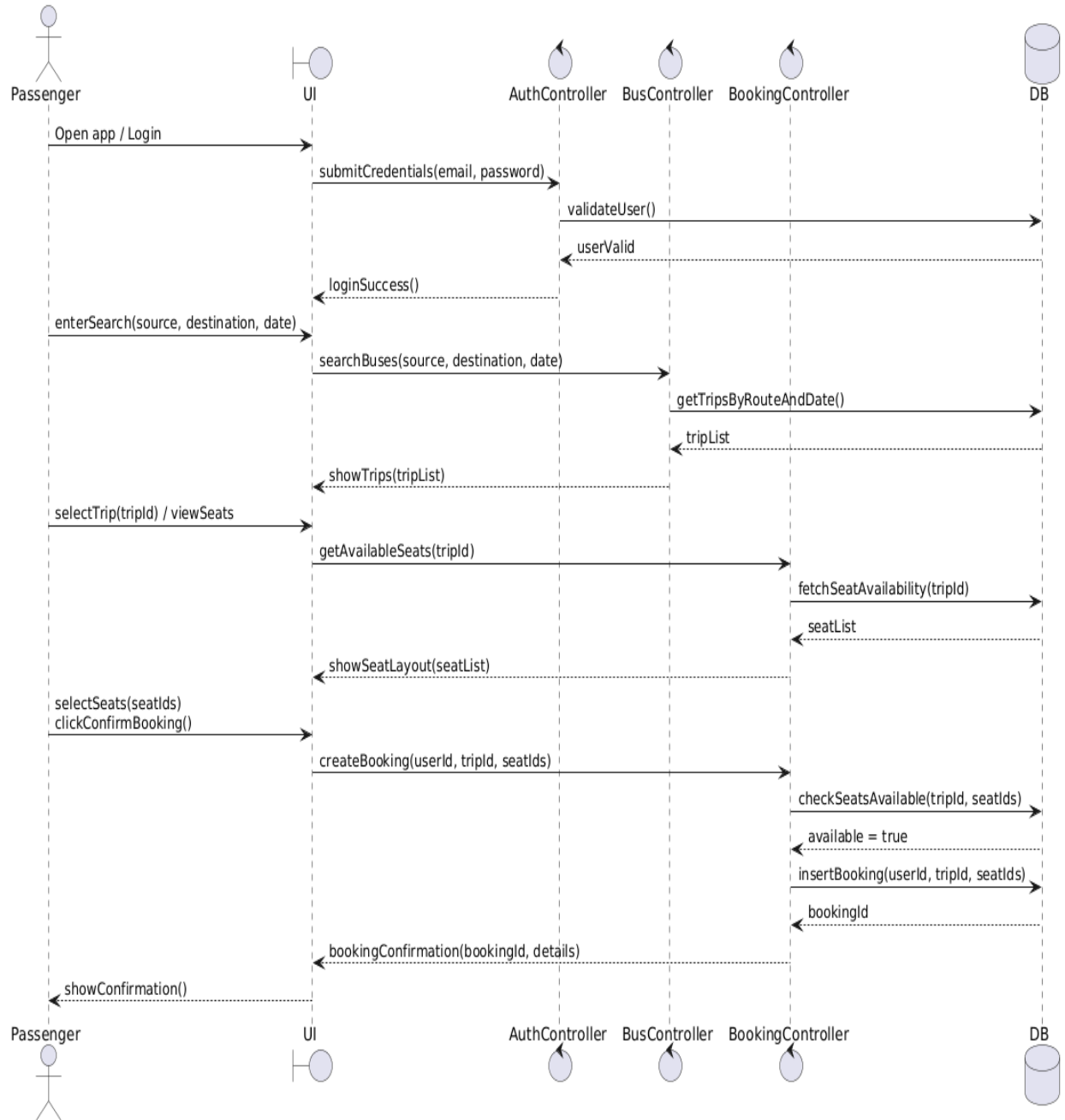
**Participants (lifelines):**

- Passenger
- UI (Web / Mobile)
- AuthController
- BusController
- BookingController
- Database

**Main message flow (summary):**

1. Passenger selects **Login** → UI sends credentials to **AuthController** → AuthController validates with **Database** and returns success.
2. Passenger enters **source, destination, date** and clicks **Search** → UI sends request to **BusController**.
3. BusController queries **Database** for matching TripSchedule records and returns list to UI.
4. Passenger selects a trip and chooses **View Seats / Book** → UI requests seat availability from **BookingController**.
5. BookingController reads seat status for that trip from **Database** and returns seat layout to UI.
6. Passenger selects seat(s) and presses **Confirm Booking** → UI sends booking details to **BookingController**.
7. BookingController re-checks availability in **Database**, creates a Booking record and updates seat status.
8. BookingController returns **booking confirmation** (Booking ID, details) to UI, which is shown to the passenger.

### Search and Book Bus



## 8.2 Sequence Diagram – Real-Time Bus Tracking

### Description:

This sequence diagram shows how the system provides live bus location to a passenger.

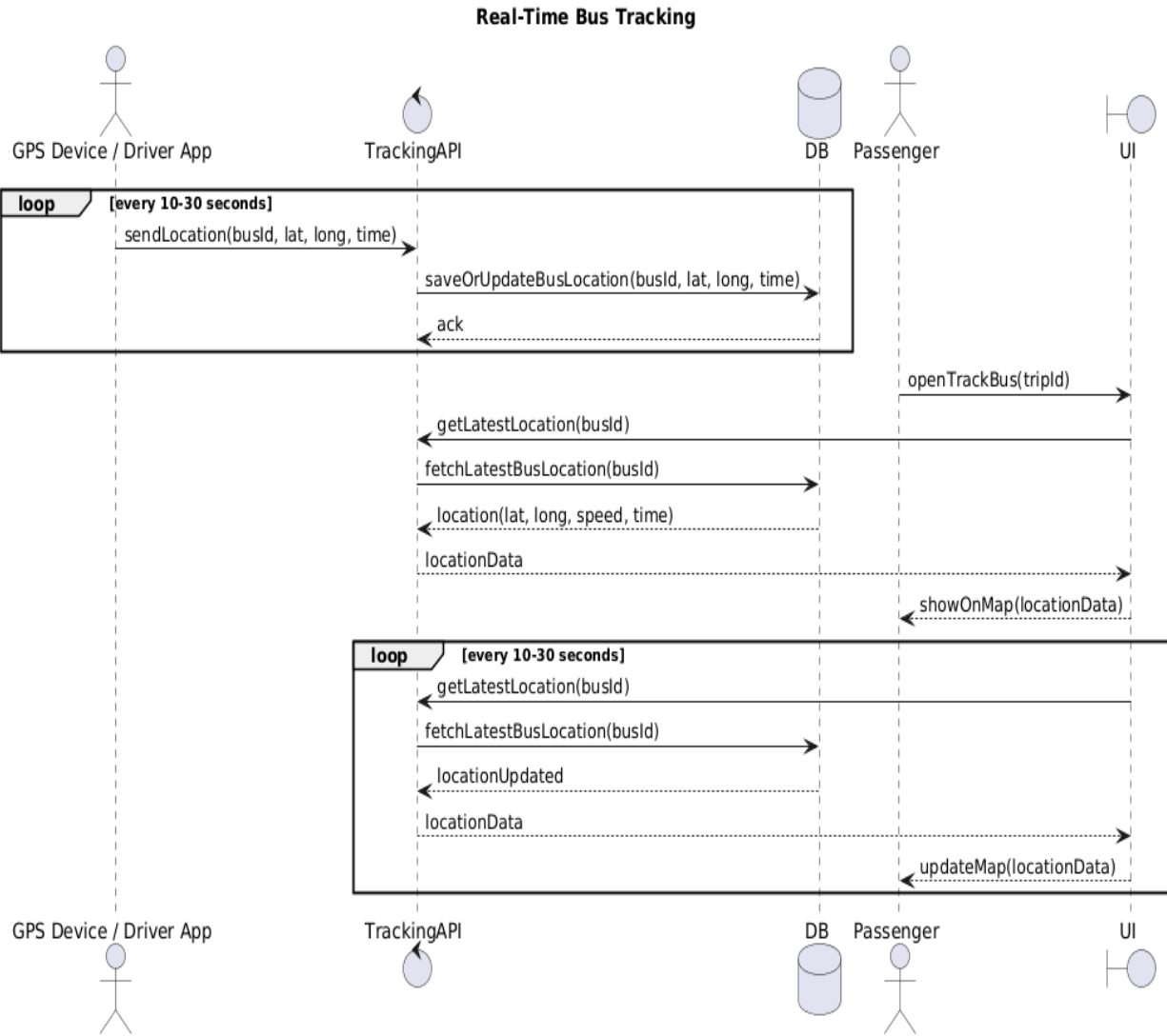
### Participants (lifelines):

- GPS Device / Driver App
- TrackingAPI (Server)
- Database
- Passenger
- UI (Tracking Screen)

### Main message flow (summary):

1. GPS Device/Driver App periodically sends **busId + latitude + longitude + timestamp** to **TrackingAPI**.
2. TrackingAPI stores each update as a new BusLocation record in the **Database** (or updates latest record).
3. Passenger opens **Track Bus** from the app → UI requests latest location from **TrackingAPI** for a specific bus/trip.
4. TrackingAPI queries **Database** for the most recent BusLocation of that bus and returns it to the UI.
5. UI displays the position on the **map view** (bus marker, route line, status, ETA).
6. UI repeats the location request after a fixed interval (e.g., every 10–30 seconds) to refresh the bus position.





## 9. System Interface Design

### 9.1 Login / Sign Up Screen

- Fields: Email, Password
- Buttons: Login, Sign Up, Forgot Password
- Purpose: Authenticate users (passenger / operator / admin)

The screenshot shows a web application window titled "Bus Reservation & Locator System". The main heading is "Bus Reservation & Locator System". Below the heading, there are input fields for "Email:" (containing "lakeshkumarkhatri7@gmail.com") and "Password:" (containing "\*\*\*\*\*"). There are three buttons: "Login", "Sign Up", and "Forgot Password". Below these buttons is a "Login as:" dropdown menu with "Passenger" selected.

### 9.2 Passenger Home Screen

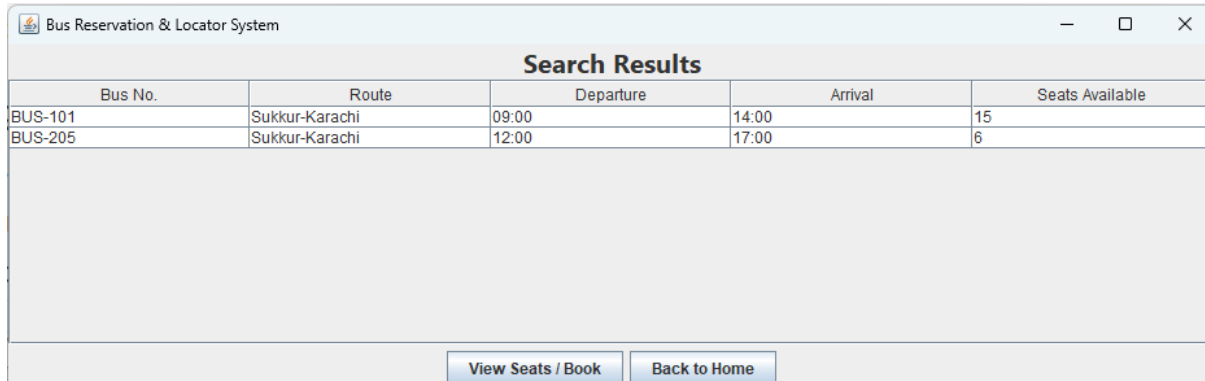
- Sections:
  - Search Bus (source, destination, date, time)
  - Upcoming Bookings
  - Quick link to "Track Bus"

The screenshot shows a web application window titled "Bus Reservation & Locator System". The main heading is "Search Bus". Below the heading, there are input fields for "Source:", "Destination:", "Date (YYYY-MM-DD):" (containing "2025-11-30"), and "Time (HH:MM):" (containing "12:00"). There is a "Search Buses" button. Below the search fields, there is a section titled "Upcoming Bookings" with a table showing booking details. At the bottom, there is a "Track Selected Bus" button and a "Logout" button.

Booking ID	Route	Date	Time	Seat	Status
B001	Sukkur-Karachi	2025-12-01	09:00	5A	Booked

### 9.3 Search Results Screen

- List of available buses / trips:
  - Bus Number, Route, Departure Time, Arrival Time
  - Seat Availability
  - “View Seats / Book” button



Bus No.	Route	Departure	Arrival	Seats Available
BUS-101	Sukkur-Karachi	09:00	14:00	15
BUS-205	Sukkur-Karachi	12:00	17:00	6

### 9.4 Seat Selection Screen

- Visual seat layout (green = available, blue = booked)
- Shows fare, selected seat numbers
- “Confirm Booking” button



Select Seats (Green = Available)

Seat 1	Seat 2	Seat 3	Seat 4
Seat 5	Seat 6	Seat 7	Seat 8
Seat 9	Seat 10	Seat 11	Seat 12
Seat 13	Seat 14	Seat 15	Seat 16
Seat 17	Seat 18	Seat 19	Seat 20

Fare: 1500 PKR      Selected Seats: (demo only)

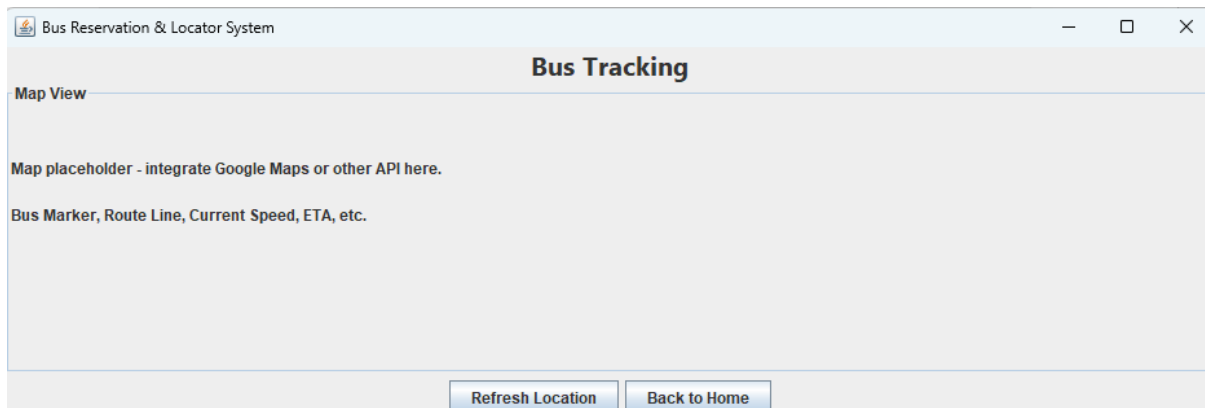
## 9.5 Booking Confirmation Screen

- Displays: Booking ID, bus details, seat numbers, date/time
- Option: “Track This Bus” (if available)



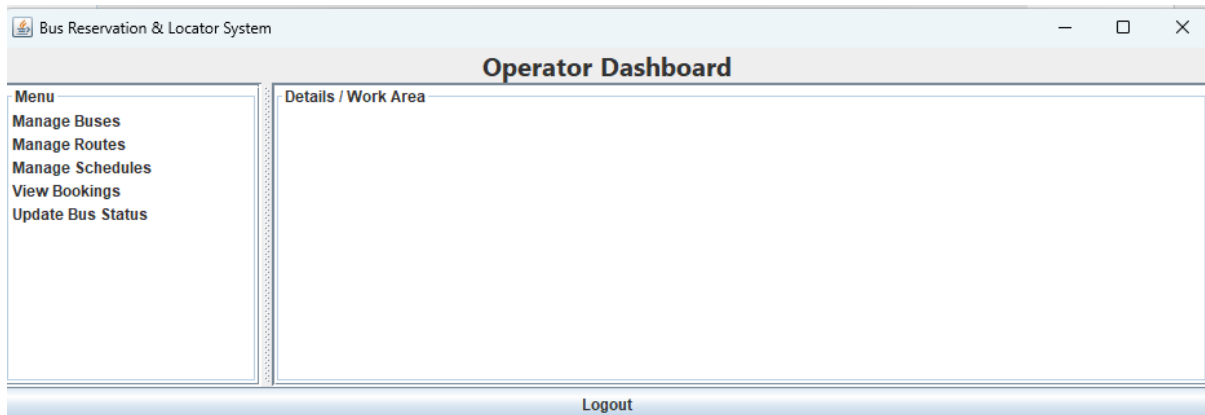
## 9.6 Bus Tracking Screen

- Map view with:
  - Bus marker
  - Route line
  - Current speed / status
  - Expected arrival time



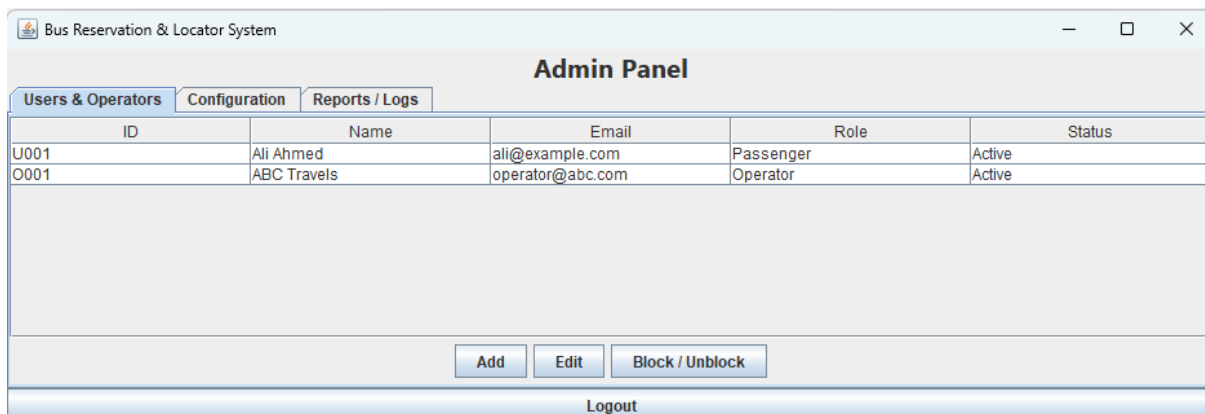
## 9.7 Operator Dashboard (Operator Role)

- Menu:
  - Manage Buses
  - Manage Routes
  - Manage Schedules (TripSchedules)
  - View Bookings
  - Update Bus Status



## 9.8 Admin Panel (Admin Role)

- Manage Users & Operators
- System configuration
- View logs / reports (basic)



## 10. Test Cases

### TC-01: User Registration

- Pre-condition: User not registered
- Steps: Open sign up, fill valid data, submit
- Expected: Account created, success message displayed

### TC-02: User Login (Valid)

- Pre-condition: User exists
- Steps: Enter correct email & password
- Expected: Login successful, redirected to home page

### TC-03: User Login (Invalid)

- Steps: Enter wrong password
- Expected: Error message shown, user not logged in

### TC-04: Search Bus with Valid Route

- Steps: Enter existing source, destination, and valid date
- Expected: List of available buses/trips displayed

### TC-05: Search Bus with No Trips

- Steps: Enter route/date where no trip is scheduled
- Expected: "No buses found" message

### TC-06: Seat Booking (Normal Flow)

- Pre-condition: Trip with available seats
- Steps: Search bus → select trip → select seat → confirm booking
- Expected Result: System records booking, updates seat status to 'Booked', and displays confirmation.

### TC-07: Prevent Double Booking

- Pre-condition: Seat already booked
- Steps: Try to book same seat again
- Expected: System prevents booking and shows error

### TC-08: View Booking History

- Pre-condition: User has past bookings
- Steps: Open "My Bookings"
- Expected: List of bookings shown with correct details

### TC-09: Real-Time Tracking

- Pre-condition: Bus has active GPS updates
- Steps: Open tracking screen for that bus
- Expected: Current location appears on map and updates periodically

### TC-10: Operator Adds New Bus

- Steps: Operator logs in → opens “Manage Buses” → adds bus details
- Expected: Bus stored in database and visible in bus list