BR Documentation – Group 11

Team Member-1: Anjana A P (285571)

Team Member-2: Annu K Prince (285578)

Problem Statement:

Scenario: Urban residents often face significant challenges when relying on public transportation. In a bustling city, commuters regularly deal with issues such as unpredictable bus schedules, overcrowded trains, and inadequate service coverage. For instance, imagine a daily commuter named Raj, who relies on the bus to get to work. Raj frequently experiences frustration because his bus arrives late or sometimes not at all, forcing him to either wait indefinitely or find alternative transportation. On days when the bus does arrive on time, Raj often finds it so overcrowded that he has to stand for the entire journey, which can be uncomfortable and stressful, especially during rush hour. Additionally, unexpected delays or detours often disrupt his commute, leading to late arrivals at work. These challenges are not just inconveniences—they discourage the use of public transport, prompting many to use personal vehicles instead, which exacerbates traffic congestion and contributes to environmental pollution.

SCOPE:

1. Functional Requirements:

1.1. User Interface (UI)

- Dashboard for Users:
    - Overview: Display user profile details, current bus schedules, and real-time bus statuses.
    - Navigation: Provide easy access to bus locations, upcoming schedules, and passenger occupancy statistics.
- Bus Tracking Interface:
    - Real-Time Map: Show bus locations, routes, and statuses on a map.
    - Bus Details: Clickable bus entries for detailed information, including route history and estimated arrival times.
- Passenger Management Interface:
    - Boarding and Exiting Records: View passenger boarding and exiting logs with timestamps and seat assignments.

    - Occupancy Statistics: Display real-time occupancy levels for each bus.

- Notification Settings Interface:
    - Configuration: Set up and manage notification preferences for bus arrivals, delays, and occupancy changes.
    - Alerts History: Access a history of notifications received.

- Admin Control Panel:
    - Bus Management: Manage bus records, update schedules, and view operational statistics.
    - User Management: View and manage user accounts and roles.
    - System Monitoring: Monitor system performance, bus thread health, and notification statuses.

1.2. Bus Management Service

- CRUD Operations:
    - Create: Add new bus records with details such as bus ID, route ID, capacity, and operational status
    - Read: Retrieve and view detailed information about each bus, including current location and status.
    - Update: Modify existing bus records to reflect changes such as schedule adjustments or route modifications.
    - Delete: Remove bus records for decommissioned or retired buses to maintain an accurate database.

- Real-Time Tracking:
    - Monitor and update bus statuses in real-time, providing current location and route information to users.

- Integration:
    - Share data with the Passenger Management Service to synchronize occupancy
    - information and update bus records accordingly.

1.3. Passenger Management Service

- Boarding and Exiting:
    - Boarding: Record when passengers board a bus, capturing details such as seat assignments and boarding times

- Exiting: Track when passengers exit the bus, updating occupancy data

- Real-Time Updates:
  - Maintain and share current passenger information with the Bus Management Service to ensure accurate data synchronization and support operational decisions.

1.4. User Management Service

- Registration and Authentication:
  - Registration: Allow users to create accounts with necessary details such as username and password.
  - Login: Implement authentication using JWT for secure access to the system. Users must log in to access their dashboards and manage their interactions with the system.

- Role Management:
  - Assign and manage user roles such as admin, operator, and passenger, ensuring appropriate access and permissions based on user roles.

1.5. Bus Thread Service

- Multithreading:

  Use separate threads to simulate bus operations, allowing for real-time processing of multiple buses concurrently. Each bus operates as an independent thread, handling its own route and schedule.

- Heartbeat Mechanism:

  Monitor the health of each bus thread, ensuring continuous operation and detecting anomalies such as delays or route deviations. This mechanism will trigger alerts and corrective actions when needed.

1.6. Notification Service

- Notification Dispatching:

  Send real-time updates about bus statuses, delays, and overcrowding.

- Integration:

Integrate with external services for notification delivery, allowing users to configure their notification preferences and receive timely alerts.

2. Non-Functional Requirements

2.1. Performance

The software should be able to handle many buses concurrently, simultaneously take note of and update the routes of each of the buses. It must also be able to update the passengers' details whenever requested. It must be able to handle all the load especially during the peak times, without any performance degradation.

2.2. Scalability

The system should be able to accommodate and handle an increasing number of buses and passengers, in the sense that it must be able to increase the scaling of the number of threads to simulate additional buses while maintaining the performance.

2.3. Security

All the buses and the user data must be stored and transmitted securely. Authentication using JWT token must be provided to ensure that only authorized users gain access to information.

2.4. Reliability and Maintainability

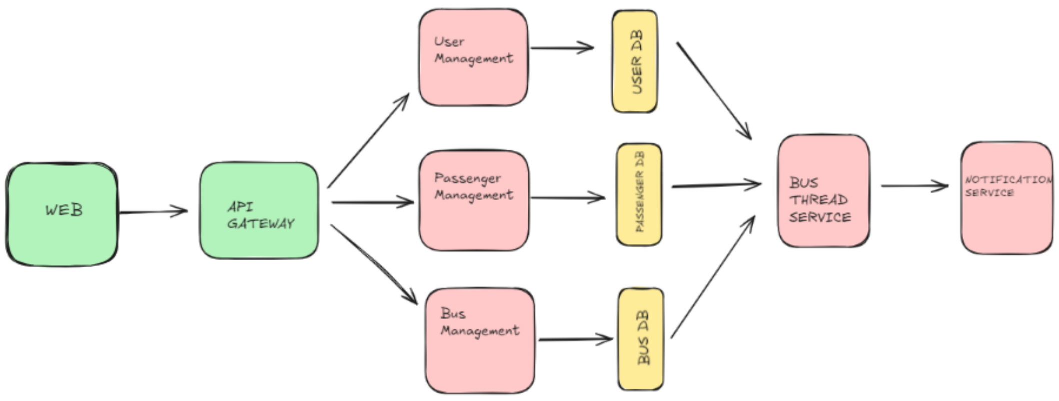
The system should be reliable, and using heartbeat mechanism, must be able to identify if any kind of problem or issue exists with the bus. If the bus thread is unresponsive, the system must be able to immediately identify the issue and corrective actions should be taken to solve it.

2.5. Usability

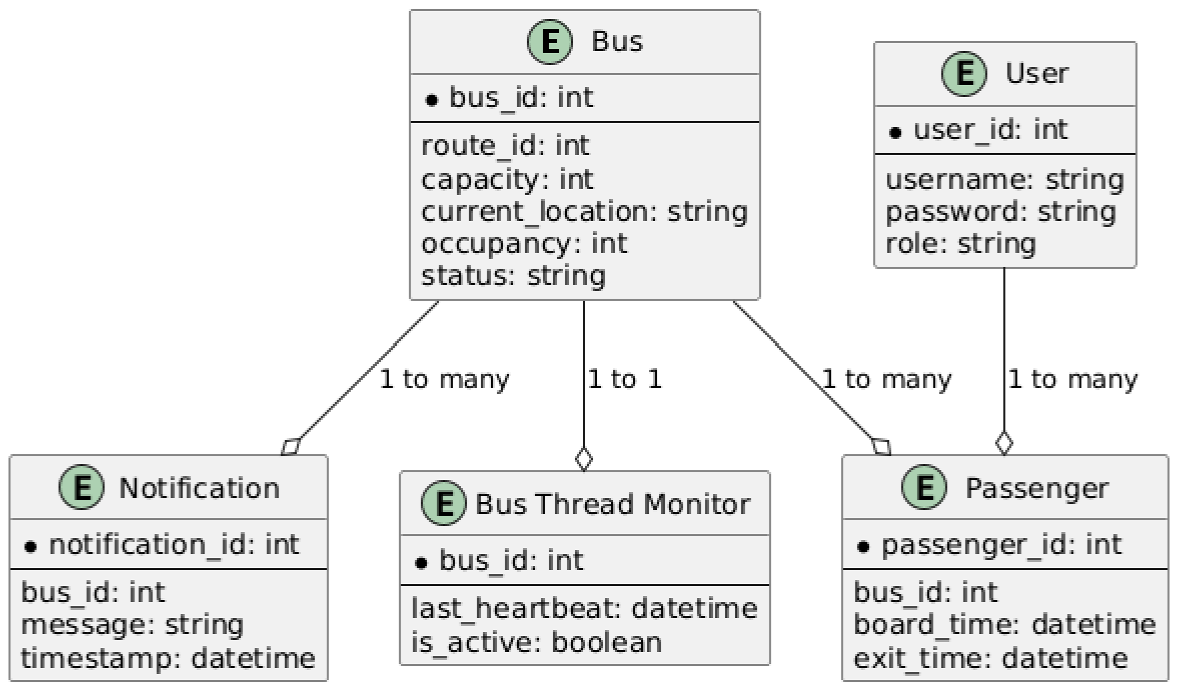
The system should be intuitive and easy to use, both for passengers booking tickets and for the admins managing buses and routes. A simple and responsive user interface should be provided to ensure a seamless experience on both desktop and mobile devices.

3. High Level Design:

- **Web:** The localhost website from which endpoints will be accessed.
- **API Gateway:** The port from which each microservices, which are in turn in different port numbers, are routed to.
- **User Management:** The microservice which manages the User sign up and login.
- **Passenger Management:** The microservice which manages the passenger CRUD operations.
- **Bus Management:** The microservice which manages the CRUD operations of the buses.
- **Bus, Passenger and User DB:** The MySQL repositories for each of the Bus, Passenger and User microservices, respectively.
- **Bus Thread Service:** Stimulates the buses as multithreads and implements heartbeat mechanism on each bus to ensure each bus arrives on their respective scheduled times.



4. Class Diagram:



4.1. Bus

- **Attributes**:
  - bus_id: Unique identifier for each bus.
  - route_id: Links the bus to a specific route for tracking.
  - capacity: Defines the total seating capacity of the bus.
  - occupancy: Tracks the current number of passengers on board.
  - current_location: Tracks the real-time location of the bus.
  - status: Indicates whether the bus is operational, inactive, or in maintenance.
- **Functionality**:
  - Critical for tracking the movement, status, and capacity of buses in real-time.
  - Ensures efficient use of routes, capacity management, and operational status reporting.
- **Relationships**:

    - **One-to-many** with Passenger: Multiple passengers can be assigned to one bus.
    - **One-to-many** with Notification: Multiple notifications can be associated with a bus.
    - **One-to-one** with Bus Thread Monitor: Each bus has its own dedicated monitor to check its activity.

4.2. User

- **Attributes**:
    - user_id: Unique identifier for each user.
    - username: User's login identifier.
    - password: User's authentication credentials.
    - role: Specifies the user type, such as passenger, admin, or driver.
- **Functionality**:
    - Manages user credentials and access rights within the system.
    - Supports role-based access control to restrict or allow different functionalities.
- **Relationships**:
    - **One-to-many** with Notification: Each user can receive multiple notifications related to bus services, updates, or alerts.

4.3. Notification

- **Attributes**:
    - notification_id: Unique identifier for each notification.
    - bus_id: Links the notification to a specific bus.
    - message: Contains the content of the notification (e.g., delays, route changes).
    - timestamp: Records the time the notification was sent.
- **Functionality**:

    - Facilitates real-time communication between the system and users regarding bus status.
    - Sends notifications for important updates like delays, overcrowding, or breakdowns.
- **Relationships**:
    - **Many-to-one** with Bus: Multiple notifications can be associated with a single bus.
    - **Many-to-one** with User: Each user can receive multiple notifications.

4.4. Bus Thread Monitor

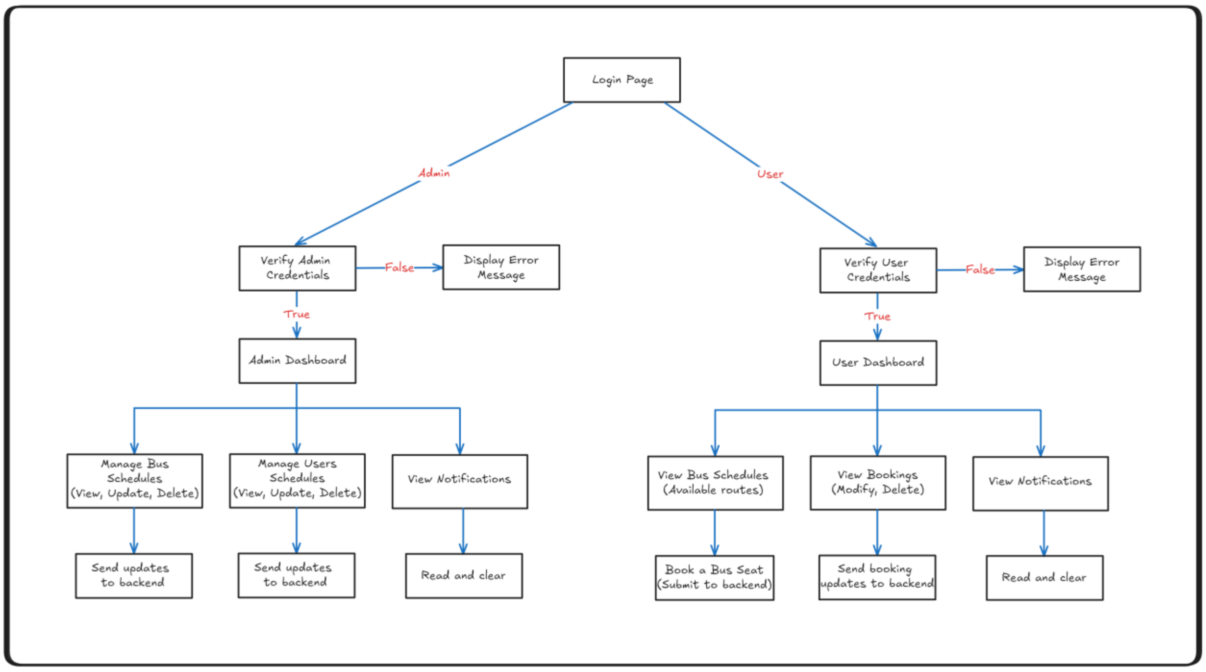
- **Attributes**:
    - bus_id: Identifies the bus being monitored.
    - last_heartbeat: Timestamp of the last system update from the bus.
    - is_active: Boolean value indicating whether the bus is currently operational.
- **Functionality**:
    - Monitors the bus's real-time status and health.
    - Detects issues such as inactive or unresponsive buses and triggers alerts for intervention.
- **Relationships**:
    - **One-to-one** with Bus: Each bus has a dedicated thread monitoring its activity and status.

4.5. Passenger

- **Attributes**:
    - passenger_id: Unique identifier for each passenger.
    - bus_id: Links the passenger to the bus they are on.
    - board_time: Timestamp of when the passenger boarded the bus.
    - exit_time: Timestamp of when the passenger exited the bus.

- **Functionality**:
  - Tracks individual passengers' boarding and exit times.
  - Helps calculate occupancy and provides data for scheduling and capacity management.
- **Relationships**:
  - **Many-to-one** with Bus: Multiple passengers can be associated with a single bus, helping monitor bus occupancy in real-time.

5. Flowchart



5.1. Login Page

- The starting point of the entire flow.
- A universal login page where both **Admins** and **Users** enter their credentials.
- Depending on the role of the person logging in (admin or user), the flow will split into different paths.
- This serves as the common entry point to the system.
-

5.2. Admin Flow

Verify Admin Credentials:

- This block verifies if the entered credentials belong to an **Admin**.

- The credentials are checked against stored admin records in the system's backend.
- If credentials are correct, the system proceeds to the **Admin Dashboard**.
- If credentials are invalid, the system will redirect the admin to an **Error Message**.

Display Error Message (Admin):

- If the admin enters incorrect credentials, an error message is shown on the screen.
- The admin is notified that the credentials are wrong, or the account doesn't exist.
- This allows the admin to retry login or check credentials for any mistakes.

Admin Dashboard:

- This is the main hub for all admin-related operations.
- From here, the admin can manage various aspects of the system, including bus schedules, user accounts, and notifications.
- The dashboard serves as a control centre for administrative tasks.

Manage Bus Schedules (View, Update, Delete):

- Admins can view all the current bus schedules in the system.
- They can modify these schedules by updating routes, timings, and bus details.
- Admins also have the ability to delete outdated or invalid bus schedules.
- After making changes, these updates are sent back to the system's backend for processing and storage.

Manage Users (View, Update, Delete):

- Admins are responsible for user management.
- This feature allows them to view all registered users and their details.
- Admins can modify user information (e.g., username, role) and also delete user accounts if needed.
- Once updated, changes are sent to the backend for updating the database.

View Notifications:

- Admins can view all system-generated notifications.
- Notifications could include system alerts, updates, maintenance messages, or errors.
- Admins have the option to clear out notifications that have already been addressed or read.
- This helps keep the system clean and ensures admins are aware of critical updates.

Send Updates to Backend:

- For both **Bus Schedule** and **User Management** tasks, after the admin makes any changes (view, update, or delete), the updates are sent to the backend.
- The backend then processes these updates and stores them in the database.
- This ensures that all the system data is up-to-date and consistent across the application.

5.3. User Flow:

5.3.1. Verify User Credentials:

- This block verifies the credentials entered by the **User**.
- The system checks if the credentials match any stored user accounts.
- If credentials are correct, the system proceeds to the **User Dashboard**.
- If credentials are invalid, the user will be directed to an **Error Message**.

5.3.2. Display Error Message (User):

- If the user enters incorrect credentials, an error message is shown.
- The message informs the user that the login attempt has failed.
- This allows the user to either try again or check for errors in the credentials.

5.3.3. User Dashboard:

- This is the main interface for users, where they can access various features.
- Users can view bus schedules, manage bookings, and see notifications relevant to them.
- It acts as the user's personal area to interact with the bus system.

5.3.4. View Bus Schedules (Available Routes):

- Users can browse through available bus schedules and routes.
- This feature lists all buses, their timings, and the routes they cover.
- Users can select and book seats for a specific bus.
- Once a seat is booked, the system sends the details to the backend for storage and processing.

5.3.5. Book a Bus Seat (Submit to Backend):

- After viewing the available bus schedules, the user has the option to book a seat on a specific bus.

- Once a seat is selected and confirmed, this information is sent to the backend.
- The backend processes the booking and reserves the seat for the user in the database.

5.3.6. View Bookings (Modify, Delete):

- Users can view all of their current and past bookings.
- This feature allows users to modify an existing booking, such as changing a seat or route, or delete a booking altogether.
- After modifications, the updated booking details are submitted to the backend for processing.
- This keeps the user's booking information up-to-date.

5.3.7. View Notifications:

- Similar to the admin's flow, users also receive system notifications.
- These could be related to booking confirmations, cancellations, system updates, or important alerts.
- Users can read and clear notifications to stay informed about any critical updates.

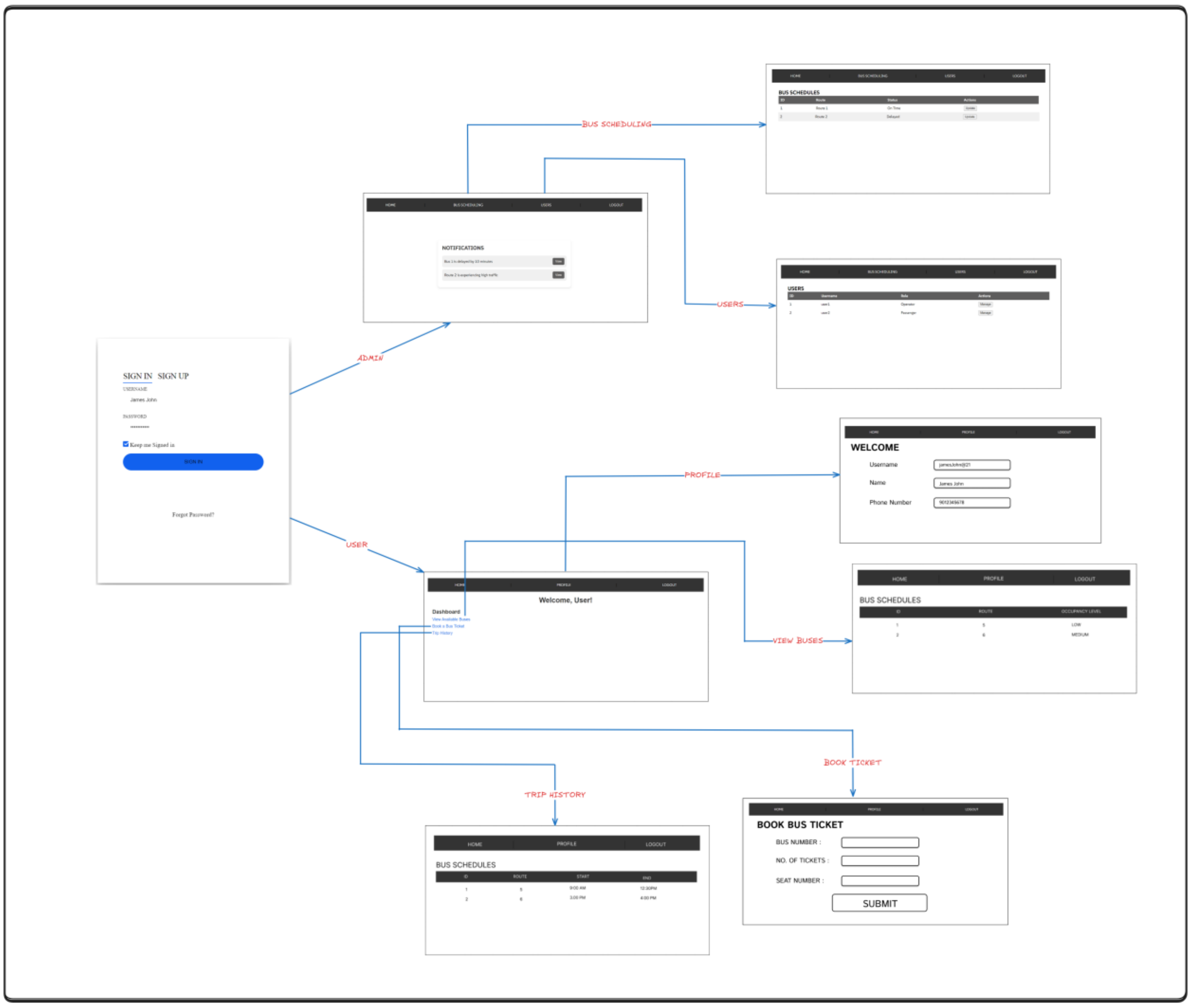
5.3.8. Send Booking Updates to Backend:

- When users modify or delete their bookings, the changes are sent to the backend.
- The backend then processes the updates and reflects the changes in the database.
- This ensures that the user's booking history is accurate and up-to-date.

5.4. Key Flow Connections:

- **Admin and User Flows**: Both flows start at the **Login Page** and split based on the credentials entered (admin or user).
- **Error Messages**: Both admin and user have error handling, displaying appropriate messages for failed login attempts.
- **Dashboards**: Both the **Admin** and **User** have their own dashboards, offering different functionalities tailored to their role in the system.
- **Backend Communication**: Both flows interact heavily with the backend to update schedules, manage users, book seats, and store notifications.

6. Wireframe



6.1. Sign In / Sign Up Page

- Purpose: Allows users to sign in or sign up for an account.
- Features: Username, password input fields, and a sign-in/sign-up button. Option to keep signed in and a "Forgot Password?" link.

6.2. Notification Page

- Purpose: Displays notifications to the user.
- Functionality: Lists any updates or alerts relevant to the user.

6.3. Dashboard Page

- Purpose: Acts as the main landing page after sign-in.

- Features: Welcomes the user and provides navigation options.

6.4. User Profile Page

- Purpose: Displays user-specific information.
- Features: Editable fields for username, name, and phone number.

6.5. Bus Schedules Page

- Purpose: Shows a list of available bus schedules.
- Features: Columns for bus numbers, destinations, and departure details.

6.6. Users Page

- Purpose: Lists all users.
- Functionality: Admin feature for managing user accounts.

6.7. View Buses Page

- Purpose: Displays detailed information about available buses.
- Features: Lists each bus with the ability to view more details.

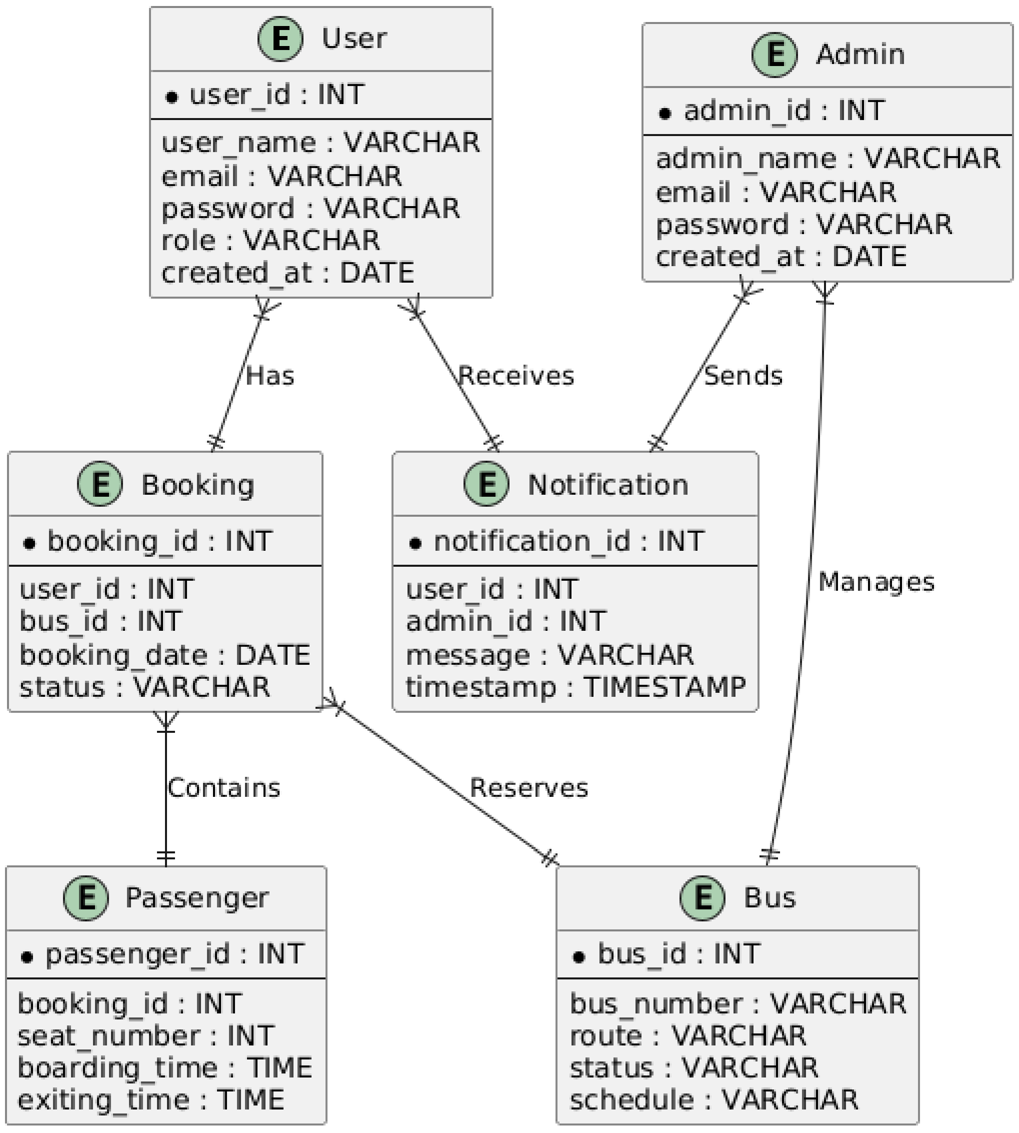
6.8. Trip History Page

- Purpose: Shows the user's past trip bookings.
- Functionality: Lists previous trips with relevant details.

6.9. Book Bus Ticket Page

- Purpose: Allows users to book a bus ticket.
- Features: Input fields for bus number, number of tickets, and seat number with a submit button.

7. Database Schemas



E User
● user_id : INT
user_name : VARCHAR
email : VARCHAR
password : VARCHAR
role : VARCHAR
created_at : DATE
E Admin
● admin_id : INT
admin_name : VARCHAR
email : VARCHAR
password : VARCHAR
created_at : DATE
Has
Receives
Sends
Manages
E Booking
● booking_id : INT
user_id : INT
bus_id : INT
booking_date : DATE
status : VARCHAR
E Notification
● notification_id : INT
user_id : INT
admin_id : INT
message : VARCHAR
timestamp : TIMESTAMP
Contains
Reserves
E Passenger
● passenger_id : INT
booking_id : INT
seat_number : INT
boarding_time : TIME
exiting_time : TIME
E Bus
● bus_id : INT
bus_number : VARCHAR
route : VARCHAR
status : VARCHAR
schedule : VARCHAR