**Bus Management System**

**Business Requirements Document – Group 15**

**Problem Statement**

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
* **Real-Time Tracking:**
  + Monitor and update bus statuses in real-time
  + Provide current location and route information
* **Integration:**
  + Share data with the Passenger Management Service
  + Synchronize occupancy information and update bus records

**1.3. Passenger Management Service**

* **Boarding and Exiting:**
  + Boarding: Record passenger boarding with seat assignments and timestamps
  + Exiting: Track passenger exits and update occupancy data
* **Real-Time Updates:**
  + Maintain current passenger information
  + Share data with Bus Management Service
  + Support operational decisions

**1.4. User Management Service**

* **Registration and Authentication:**
  + Registration: Allow account creation with username and password
  + Login: Implement JWT authentication for secure access
* **Role Management:**
  + Assign and manage user roles (admin, operator, passenger)
  + Ensure appropriate access permissions

**1.5. Bus Thread Service**

 **Multithreading:**

* Simulate bus operations using separate threads for each active bus in the system
* Enable real-time processing of multiple buses concurrently
* Implement thread pooling to manage system resources efficiently
* Handle thread creation and destruction based on bus schedule activation/deactivation
* Process concurrent passenger boarding/deboarding events across multiple buses

 **Location Tracking:**

* Update bus location coordinates every 30 seconds using GPS integration
* Calculate and store route progress percentage
* Track deviation from planned route
* Maintain location history for analytics and delay prediction
* Implement geofencing to detect when buses enter/exit designated stops
* Store timestamp with each location update for accurate arrival predictions
* Update expected arrival times at upcoming stops based on current position

 **Occupancy Management:**

* Track real-time passenger count through automated boarding/deboarding detection
* Update occupancy status every time a passenger boards or exits:
  + Increment counter when boarding pass is scanned
  + Decrement counter when exit is recorded
  + Validate against maximum bus capacity
* Monitor seat availability in real-time
* Track standing passenger count during peak hours
* Generate alerts when approaching maximum capacity
* Calculate occupancy percentage for each bus segment

 **Performance Optimization:**

* Implement lazy loading for inactive bus threads
* Use thread priority scheduling for buses during peak hours
* Buffer location updates to prevent database overflow
* Implement connection pooling for database operations
* Cache frequently accessed route and schedule data
* Optimize passenger count updates during high-frequency stops

 **Heartbeat Mechanism:**

* Send heartbeat signals every 10 seconds from each bus thread
* Monitor thread health and responsiveness
* Detect anomalies such as:
  + Missing location updates
  + Delayed responses
  + Route deviations
  + Unexpected stops
  + Unusual passenger count changes
* Trigger alerts when issues are detected
* Verify consistency between passenger counts and bus capacity

 **Data Synchronization:**

* Coordinate location updates with the central database
* Sync passenger count data with occupancy tracking
* Update estimated arrival times based on current location and traffic conditions
* Maintain consistency between thread states and database records
* Handle concurrent updates from multiple buses
* Reconcile passenger boarding/deboarding data with ticket validations

 **Integration Features:**

* Push location updates to the Notification Service for passenger alerts
* Share real-time data with the Bus Management Service
* Provide occupancy updates to the Passenger Management Service
* Interface with external traffic monitoring systems
* Support integration with third-party mapping services
* Connect with automated passenger counting systems

 **Passenger Flow Analytics:**

* Track boarding/deboarding patterns at each stop
* Calculate average occupancy rates per route segment
* Identify peak occupancy times and locations
* Monitor dwell time at stops based on passenger flow
* Generate real-time capacity utilization metrics
* Predict crowding at upcoming stops

**1.6. Notification Service**

* **Notification Dispatching:**
  + Send real-time updates about bus status
  + Alert users about delays and overcrowding
* **Integration:**
  + Connect with external notification services
  + Allow user preference configuration

**1.7. CarPool Service**

* **CRUD Operations:**
  + Create: Add new carpool offers with route and driver details
  + Read: View carpool listings and availability
  + Update: Modify existing offers and schedules
  + Delete: Remove expired listings
* **Driver Management:**
  + Profile verification and credentials
  + Schedule management
  + Rating system
  + Earnings tracking
* **Passenger Matching:**
  + Route matching algorithm
  + Seat allocation
  + Booking system
* **Real-Time Features:**
  + Live tracking
  + Instant notifications
* **Integration Points:**
  + User management integration
  + Payment processing
  + Map services
  + Notification system

**2. Non-Functional Requirements**

**2.1. Performance**

* Handle multiple concurrent buses
* Update routes and passenger details in real-time
* Maintain performance during peak times

**2.2. Scalability**

* Accommodate increasing buses and passengers
* Scale thread count for additional buses
* Maintain consistent performance

**2.3. Security**

* Secure data storage and transmission
* JWT authentication
* Authorized access control

**2.4. Reliability and Maintainability**

* Implement heartbeat mechanism
* Monitor bus thread health
* Quick issue identification and resolution

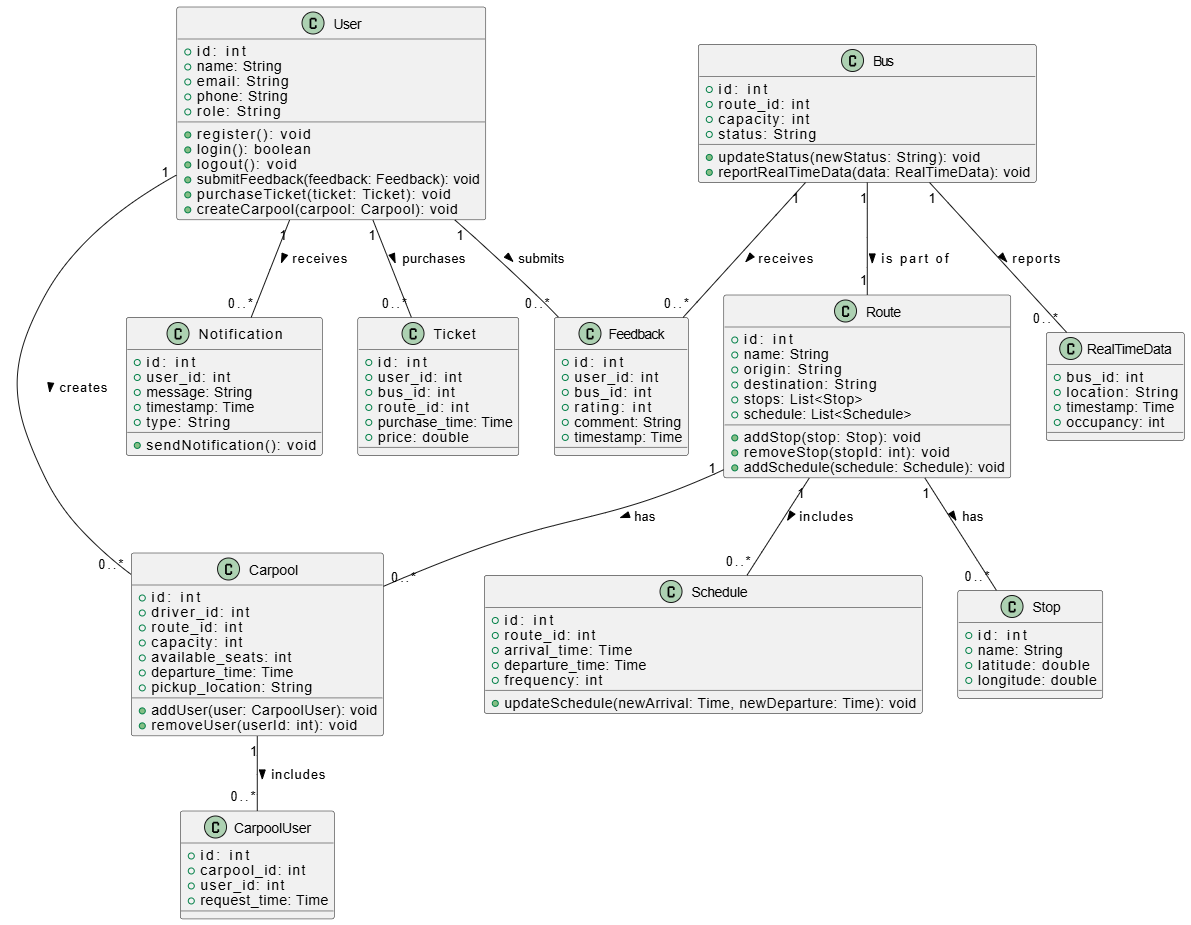
**2.5. Usability**

* Intuitive interface
* Responsive design
* Mobile and desktop compatibility

**3. High Level Design**

* Web Interface
* API Gateway
* Microservices Architecture
  + User Management
  + Passenger Management
  + Bus Management
  + CarPool Management
* Database Systems
* Bus Thread Service
* Notification System

**4. Class Diagram**

****

**4.1. Bus**

* **Attributes:**
  + bus\_id
  + route\_id
  + capacity
  + occupancy
  + status
* **Relationships:**
  + One-to-many with Passenger
  + One-to-many with Notification
  + One-to-one with Bus Thread Monitor

**4.2. User**

* **Attributes:**
  + user\_id
  + username
  + password
  + role
* **Relationships:**
  + One-to-many with Notification
  + One-to-many with CarPool (as driver or passenger)

**4.3. Notification**

* **Attributes:**
  + notification\_id
  + bus\_id
  + message
  + timestamp
* **Relationships:**
  + Many-to-one with Bus
  + Many-to-one with User

**4.4. Bus Thread Monitor**

* **Attributes:**
  + bus\_id
  + last\_heartbeat
  + is\_active
* **Relationships:**
  + One-to-one with Bus

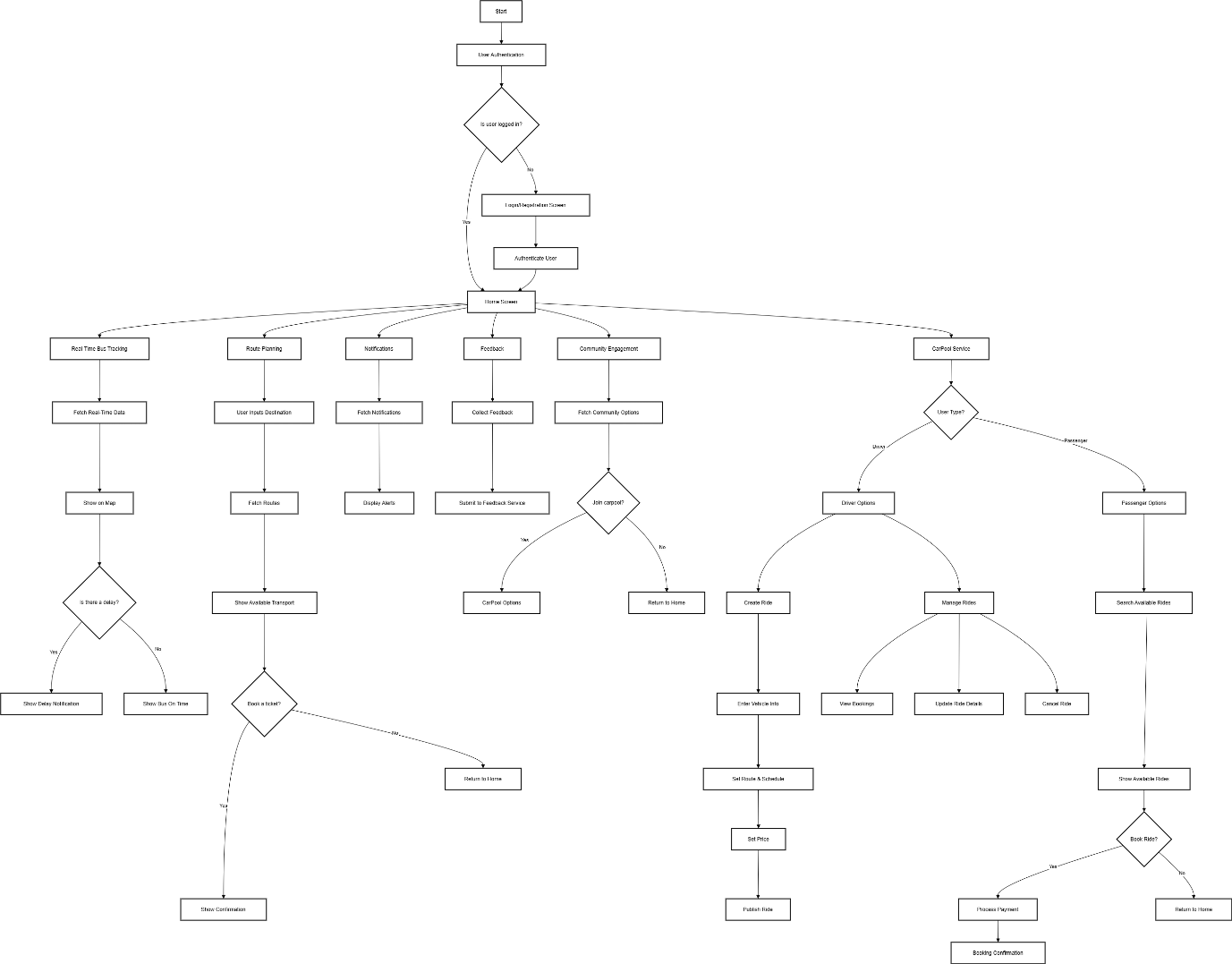
**4.5. Passenger**

* **Attributes:**
  + passenger\_id
  + bus\_id
  + board\_time
  + exit\_time
* **Relationships:**
  + Many-to-one with Bus

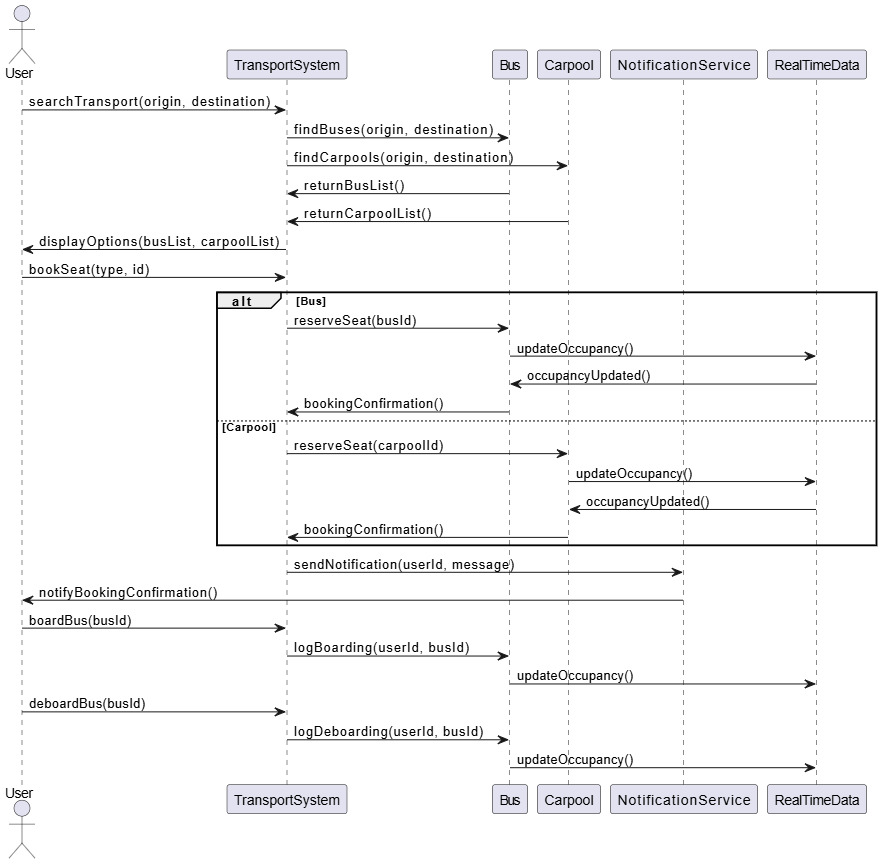
**4.6. CarPool**

* **Attributes:**
  + carpool\_id
  + driver\_id
  + route\_details
  + available\_seats
  + schedule
  + price
* **Relationships:**
  + Many-to-one with User (driver)
  + Many-to-many with User (passengers)

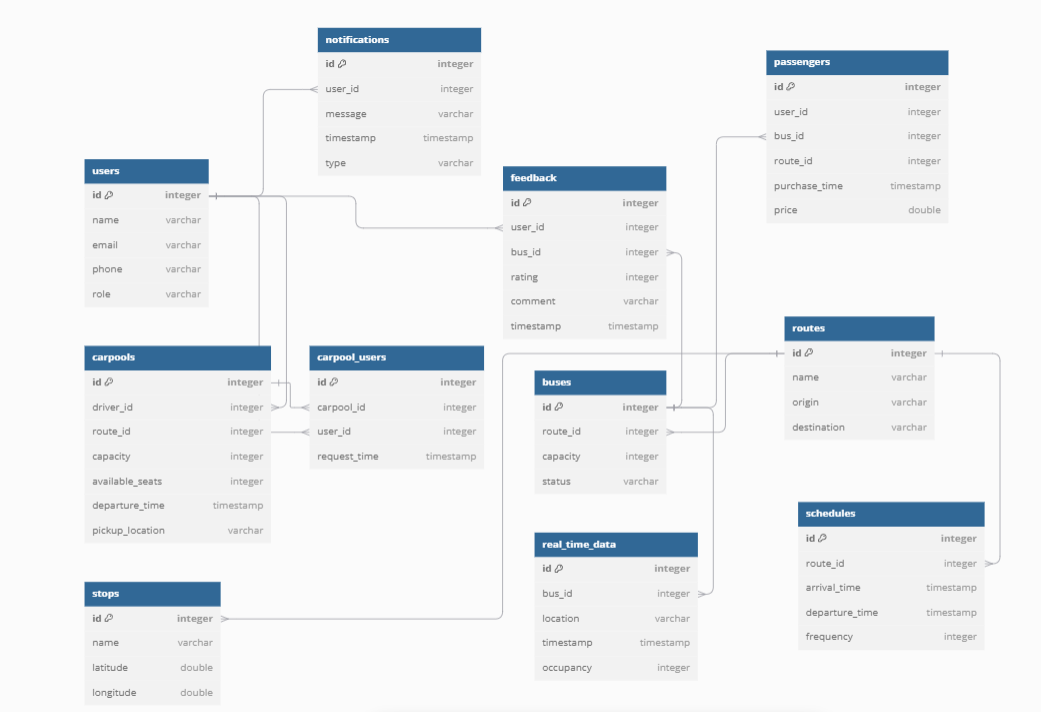
**5. Flowchart**

****

**6. Sequence Diagram**

****

**7. Database Schema**

****