MODULE 2: CORE API SERVICES

WHERE IS MY BUS? - Backend API Documentation

Project: Where Is My Bus? - Smart Bus Tracking System
Developer: Arpit Anand (23BCS12710)
Module: 2 of 6 - Core API Services
Technology Stack: Spring Boot 3.2, Java 17, REST APIs, Spring Security
Status: Planning & Documentation Phase

1. OVERVIEW

Module 2 implements the **RESTful API services** that power the Where Is My Bus? platform. This module provides:

- **User Management APIs** - Registration, authentication, profile management
- **Bus Management APIs** - CRUD operations for buses and fleet
- **Route Management APIs** - Create, read, update routes and stops
- **Report Management APIs** - Handle community bus sightings
- **Trip Management APIs** - Record and track bus trips
- **Messaging APIs** - Passenger-driver communication

2. TECHNOLOGY STACK

2.1 Core Technologies

Technology	Version	Purpose
Spring Boot	3.2.0	Backend framework
Java	17 (LTS)	Programming language
Spring Data MongoDB	3.2.0	Database access
Spring Security	6.2.0	Authentication & authorization
Spring Validation	3.2.0	Request validation
Lombok	1.18.30	Reduce boilerplate code
MapStruct	1.5.5	DTO mapping
Swagger/OpenAPI	2.2.0	API documentation

2.2 Development Tools

- **Maven** - Dependency management
- **Postman** - API testing
- **JUnit 5** - Unit testing
- **Mockito** - Mocking framework

3. PROJECT STRUCTURE

```
where-is-my-bus-backend/
│
├── src/main/java/com/whereismybus/
│   ├── WheresMyBusApplication.java
│   │
│   ├── config/                    # Configuration classes
│   │   ├── SecurityConfig.java
│   │   ├── MongoConfig.java
│   │   ├── CorsConfig.java
│   │   └── SwaggerConfig.java
│   │
│   ├── controller/                # REST Controllers
│   │   ├── AuthController.java
│   │   ├── UserController.java
│   │   ├── BusController.java
│   │   ├── RouteController.java
│   │   ├── ReportController.java
│   │   ├── TripController.java
│   │   └── MessageController.java
│   │
│   ├── service/                   # Business Logic
│   │   ├── AuthService.java
│   │   ├── UserService.java
│   │   ├── BusService.java
│   │   ├── RouteService.java
│   │   ├── ReportService.java
│   │   ├── TripService.java
│   │   ├── MessageService.java
│   │   └── JwtService.java
│   │
│   ├── repository/                # MongoDB Repositories
│   │   ├── UserRepository.java
│   │   ├── BusRepository.java
│   │   ├── RouteRepository.java
│   │   ├── ReportRepository.java
│   │   ├── TripRepository.java
│   │   └── MessageRepository.java
│   │
│   ├── model/                     # Domain Models (Entities)
│   │   ├── User.java
│   │   ├── Bus.java
│   │   ├── Route.java
│   │   ├── Report.java
│   │   ├── Trip.java
│   │   └── Message.java
│   │
│   ├── dto/                     # Data Transfer Objects
│   │   ├── request/
│   │   │   ├── LoginRequest.java
│   │   │   ├── RegisterRequest.java
│   │   │   ├── BusRequest.java
│   │   │   ├── RouteRequest.java
│   │   │   └── ReportRequest.java
│   │   └── response/
│   │       ├── AuthResponse.java
│   │       ├── UserResponse.java
│   │       ├── BusResponse.java
│   │       └── ApiResponse.java
│   │
│   ├── mapper/                  # DTO Mappers
│   │   ├── UserMapper.java
│   │   ├── BusMapper.java
│   │   └── RouteMapper.java
│   │
│   ├── exception/               # Exception Handling
│   │   ├── GlobalExceptionHandler.java
│   │   ├── ResourceNotFoundException.java
│   │   ├── UnauthorizedException.java
│   │   └── ValidationException.java
│   │
│   ├── security/                # Security Components
│   │   ├── JwtAuthenticationFilter.java
│   │   ├── JwtAuthenticationEntryPoint.java
│   │   └── CustomUserDetailsService.java
│   │
│   ├── util/                    # Utility Classes
│   │   ├── DistanceCalculator.java
│   │   ├── ETACalculator.java
│   │   └── ValidationUtils.java
│   │
│   └── constant/                # Constants
│       ├── Role.java
│       ├── BusStatus.java
│       └── ReportStatus.java
│
├── src/main/resources/
│   ├── application.properties
│   ├── application-dev.properties
│   └── application-prod.properties
│
├── src/test/java/
│   └── (Test classes mirror main structure)
│
├── pom.xml
└── README.md
```

4. API ENDPOINTS

4.1 BASE URL

```
Development: http://localhost:8080/api
Production: https://api.whereismybus.com/api
```

4.2 AUTHENTICATION ENDPOINTS

POST /auth/register

Register a new user (passenger, driver, or admin).

Request Body:

```
{
  "name": "Arjun Kumar",
  "email": "arjun@example.com",
  "phone": "+919876543210",
  "password": "SecurePass123",
  "role": "PASSENGER"
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "User registered successfully",
  "data": {
    "userId": "507f1f77bcf86cd799439011",
    "name": "Arjun Kumar",
    "email": "arjun@example.com",
    "role": "PASSENGER",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "tokenType": "Bearer",
    "expiresIn": 900
  }
}
```

Validations:

- Email must be valid and unique
- Phone must be Indian format and unique
- Password min 8 chars, 1 uppercase, 1 number
- Role must be PASSENGER, DRIVER, or ADMIN

POST /auth/login

Authenticate user and receive JWT tokens.

Request Body:

```json
{
  "email": "arjun@example.com",
  "password": "SecurePass123"
}
```

Response (200 OK):

```json
{
  "success": true,
  "message": "Login successful",
  "data": {
    "userId": "507f1f77bcf86cd799439011",
    "name": "Arjun Kumar",
    "email": "arjun@example.com",
    "role": "PASSENGER",
    "trustScore": 850,
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "tokenType": "Bearer",
    "expiresIn": 900
  }
}
```

Error Response (401 Unauthorized):

```json
{
  "success": false,
  "message": "Invalid email or password",
  "error": "INVALID_CREDENTIALS"
}
```

POST /auth/refresh

Refresh access token using refresh token.

Request Body:

```json
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "tokenType": "Bearer",
    "expiresIn": 900
  }
}
```

POST /auth/logout

Logout user and invalidate refresh token.

Headers:

```
Authorization: Bearer &lt;ACCESS_TOKEN&gt;
```

Response (200 OK):

```
{
  "success": true,
  "message": "Logout successful"
}
```

4.3 USER MANAGEMENT ENDPOINTS

GET /users/me

Get current authenticated user profile.

Headers:

```
Authorization: Bearer &lt;ACCESS_TOKEN&gt;
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "userId": "507f1f77bcf86cd799439011",
    "name": "Arjun Kumar",
    "email": "arjun@example.com",
    "phone": "+919876543210",
    "role": "PASSENGER",
    "profilePicture": "https://s3.amazonaws.com/users/profile_xyz.jpg",
    "trustScore": 850,
    "points": 1250,
    "level": "EAGLE_EYE",
    "badges": [
      {
        "badgeId": "badge_scout",
        "name": "SCOUT",
        "unlockedAt": "2025-09-01T10:00:00Z"
      }
    ],
    "reportCount": {
      "submitted": 125,
      "approved": 118,
      "rejected": 7
    },
    "createdAt": "2025-08-10T12:00:00Z"
  }
}
```

PUT /users/me

Update current user profile.

Headers:

```
Authorization: Bearer &lt;ACCESS_TOKEN&gt;
```

Request Body:

```
{
  "name": "Arjun Kumar Updated",
  "phone": "+919876543211",
  "profilePicture": "https://s3.amazonaws.com/users/new_profile.jpg"
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Profile updated successfully",
  "data": {
    "userId": "507f1f77bcf86cd799439011",
    "name": "Arjun Kumar Updated",
    "phone": "+919876543211"
  }
}
```

GET /users (ADMIN ONLY)

Get all users with pagination and filtering.

Headers:

```
Authorization: Bearer &lt;ADMIN_ACCESS_TOKEN&gt;
```

Query Parameters:

- `page` (default: 0)
- `size` (default: 20)
- `role` (optional: PASSENGER, DRIVER, ADMIN)
- `search` (optional: search by name/email)
- `sortBy` (default: createdAt)
- `sortOrder` (default: DESC)

Example Request:

```
GET /users?page=0&amp;size=20&amp;role=PASSENGER&amp;search=arjun&amp;sortBy=trustScore&a
```

Response (200 OK):

```
{
  "success": true,
  "data": {
    "users": [
      {
        "userId": "507f1f77bcf86cd799439011",
        "name": "Arjun Kumar",
        "email": "arjun@example.com",
        "role": "PASSENGER",
        "trustScore": 850,
        "accountStatus": "ACTIVE",
        "createdAt": "2025-08-10T12:00:00Z"
      }
    ],
    "pagination": {
      "currentPage": 0,
      "totalPages": 5,
      "totalItems": 100,
      "itemsPerPage": 20
    }
  }
}
```

GET /users/{userId} (ADMIN ONLY)

Get specific user details.

Response (200 OK):

```
{
  "success": true,
  "data": {
    "userId": "507f1f77bcf86cd799439011",
    "name": "Arjun Kumar",
    "email": "arjun@example.com",
    "phone": "+919876543210",
    "role": "PASSENGER",
    "accountStatus": "ACTIVE",
    "trustScore": 850,
    "points": 1250,
    "reportCount": {
      "submitted": 125,
      "approved": 118,
      "rejected": 7
    },
    "lastLoginDate": "2025-11-14T10:15:00Z",
    "createdAt": "2025-08-10T12:00:00Z"
  }
}
```

PUT /users/{userId}/suspend (ADMIN ONLY)

Suspend a user account.

Request Body:

```
{
  "reason": "Repeated spam reports",
  "suspensionDays": 7
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "User suspended for 7 days",
  "data": {
    "userId": "507f1f77bcf86cd799439011",
    "accountStatus": "SUSPENDED",
    "suspensionEndDate": "2025-11-21T00:00:00Z"
  }
}
```

DELETE /users/{userId} (ADMIN ONLY)

Delete a user account (soft delete).

Response (200 OK):

```
{
  "success": true,
  "message": "User deleted successfully"
}
```

4.4 BUS MANAGEMENT ENDPOINTS

GET /buses

Get all buses with filtering.

Query Parameters:

- `page` (default: 0)
- `size` (default: 20)
- `status` (optional: ACTIVE, MAINTENANCE, RETIRED)
- `routeId` (optional: filter by route)

Response (200 OK):

```
{
  "success": true,
  "data": {
    "buses": [
      {
        "busId": "bus_789",
        "busNumber": "HR26Q4321",
        "busType": "AC_DELUXE",
        "capacity": 65,
        "status": "ACTIVE",
        "assignedRoute": {
          "routeId": "route_42",
          "routeNumber": "42",
          "routeName": "Chandigarh - Panchkula Express"
        },
        "currentLocation": {
          "latitude": 28.7041,
          "longitude": 77.1025
        },
        "lastUpdated": "2025-11-14T13:45:00Z"
      }
    ],
    "pagination": {
      "currentPage": 0,
      "totalPages": 3,
      "totalItems": 50,
      "itemsPerPage": 20
    }
  }
}
```

GET /buses/{busId}

Get specific bus details.

Response (200 OK):

```json
{
  "success": true,
  "data": {
    "busId": "bus_789",
    "busNumber": "HR26Q4321",
    "registrationNumber": "HR-26-Q-4321",
    "busType": "AC_DELUXE",
    "capacity": {
      "seating": 45,
      "standing": 20,
      "total": 65
    },
    "manufacturer": "Ashok Leyland",
    "model": "Viking",
    "yearOfManufacture": 2022,
    "status": "ACTIVE",
    "operationalStatus": "IN_SERVICE",
    "assignedRoute": {
      "routeId": "route_42",
      "routeNumber": "42",
      "routeName": "Chandigarh - Panchkula Express"
    },
    "assignedDriver": {
      "driverId": "driver_123",
      "name": "Rajesh Sharma"
    },
    "currentLocation": {
      "latitude": 28.7041,
      "longitude": 77.1025,
      "speed": 35,
      "heading": 180,
      "lastUpdated": "2025-11-14T13:45:00Z"
    },
    "features": ["GPS", "AC", "WiFi", "USB_CHARGING"],
    "lastMaintenanceDate": "2025-10-01T00:00:00Z",
    "nextMaintenanceDate": "2025-12-01T00:00:00Z"
  }
}
```

POST /buses (ADMIN ONLY)

Create a new bus.

Request Body:

```
{
  "busNumber": "HR26Q4321",
  "registrationNumber": "HR-26-Q-4321",
  "busType": "AC_DELUXE",
  "capacity": {
    "seating": 45,
    "standing": 20
  },
  "manufacturer": "Ashok Leyland",
  "model": "Viking",
  "yearOfManufacture": 2022,
  "features": ["GPS", "AC", "WiFi", "USB_CHARGING"]
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Bus created successfully",
  "data": {
    "busId": "bus_789",
    "busNumber": "HR26Q4321",
    "status": "ACTIVE"
  }
}
```

PUT /buses/{busId} (ADMIN ONLY)

Update bus details.

Request Body:

```
{
  "busType": "AC_PREMIUM",
  "status": "MAINTENANCE"
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Bus updated successfully"
}
```

DELETE /buses/{busId} (ADMIN ONLY)

Delete a bus (soft delete - set status to RETIRED).

Response (200 OK):

```
{
  "success": true,
  "message": "Bus deleted successfully"
}
```

PUT /buses/{busId}/location (DRIVER ONLY)

Update bus location in real-time.

Request Body:

```
{
  "latitude": 28.7041,
  "longitude": 77.1025,
  "speed": 35,
  "heading": 180,
  "accuracy": 15
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Location updated successfully"
}
```

4.5 ROUTE MANAGEMENT ENDPOINTS

GET /routes

Get all routes.

Query Parameters:

- `page` (default: 0)
- `size` (default: 20)
- `status` (optional: ACTIVE, SUSPENDED)

Response (200 OK):

```
{
  "success": true,
  "data": {
    "routes": [
      {
        "routeId": "route_42",
        "routeNumber": "42",
        "routeName": "Chandigarh - Panchkula Express",
        "origin": "ISBT Sector 43, Chandigarh",
        "destination": "Panchkula Bus Stand",
        "distance": 18.5,
        "estimatedDuration": 45,
        "totalStops": 15,
        "status": "ACTIVE",
        "firstBus": "05:30",
        "lastBus": "23:00"
      }
    ],
    "pagination": {
      "currentPage": 0,
      "totalPages": 2,
      "totalItems": 25,
      "itemsPerPage": 20
    }
  }
}
```

GET /routes/{routeId}

Get specific route with all stops.

Response (200 OK):

```
{
  "success": true,
  "data": {
    "routeId": "route_42",
    "routeNumber": "42",
    "routeName": "Chandigarh - Panchkula Express",
    "origin": "ISBT Sector 43, Chandigarh",
    "destination": "Panchkula Bus Stand",
    "distance": 18.5,
    "estimatedDuration": 45,
    "stops": [
      {
        "stopId": "stop_isbt",
        "stopName": "ISBT Sector 43",
        "location": {
          "latitude": 30.7333,
          "longitude": 76.7933
        },
        "sequenceNumber": 1,
        "arrivalTime": "06:00",
        "departureTime": "06:05",
        "distanceFromOrigin": 0
      },
      {
        "stopId": "stop_sector17",
        "stopName": "Sector 17 Plaza",
        "location": {
          "latitude": 30.7409,
          "longitude": 76.7794
        },
        "sequenceNumber": 2,
        "arrivalTime": "06:15",
        "departureTime": "06:18",
        "distanceFromOrigin": 3.2
      }
    ],
    "operatingDays": ["MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY", "SATURDAY"]
    "frequency": 15,
    "firstBus": "05:30",
    "lastBus": "23:00",
    "fare": {
      "baseFare": 10,
      "perKmCharge": 2,
      "maxFare": 50
    },
    "status": "ACTIVE"
  }
}
```

POST /routes (ADMIN ONLY)

Create a new route.

Request Body:

```
{
  "routeNumber": "42",
  "routeName": "Chandigarh - Panchkula Express",
  "origin": "ISBT Sector 43, Chandigarh",
  "destination": "Panchkula Bus Stand",
  "distance": 18.5,
  "estimatedDuration": 45,
  "stops": [
    {
      "stopName": "ISBT Sector 43",
      "location": {
        "latitude": 30.7333,
        "longitude": 76.7933
      },
      "sequenceNumber": 1,
      "arrivalTime": "06:00",
      "departureTime": "06:05"
    }
  ],
  "operatingDays": ["MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY", "SATURDAY"],
  "frequency": 15,
  "firstBus": "05:30",
  "lastBus": "23:00",
  "fare": {
    "baseFare": 10,
    "perKmCharge": 2,
    "maxFare": 50
  }
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Route created successfully",
  "data": {
    "routeId": "route_42",
    "routeNumber": "42"
  }
}
```

PUT /routes/{routeId} (ADMIN ONLY)

Update route details.

Response (200 OK):

```
{
  "success": true,
  "message": "Route updated successfully"
}
```

DELETE /routes/{routeId} (ADMIN ONLY)

Delete a route (set status to SUSPENDED).

Response (200 OK):

```
{
  "success": true,
  "message": "Route deleted successfully"
}
```

4.6 REPORT MANAGEMENT ENDPOINTS

GET /reports

Get all reports (Admin sees all, Passenger sees own).

Query Parameters:

- `page` (default: 0)
- `size` (default: 20)
- `status` (optional: PENDING, APPROVED, REJECTED)
- `busNumber` (optional)
- `startDate` (optional)
- `endDate` (optional)

Response (200 OK):

```
{
  "success": true,
  "data": {
    "reports": [
      {
        "reportId": "RPT_20251106_001",
        "busNumber": "HR26Q4321",
        "reportType": "BUS_SIGHTING",
        "location": {
          "latitude": 28.7041,
          "longitude": 77.1025,
          "stopName": "Main Street Market"
        },
        "photo": {
          "url": "https://s3.amazonaws.com/reports/photo_abc123.jpg",
          "thumbnailUrl": "https://s3.amazonaws.com/reports/thumb_abc123.jpg"
        },
        "description": "Bus arrived on time",
        "passenger": {
          "passengerId": "passenger_123",
          "name": "Arjun Kumar",
          "trustScore": 850
        },
        "status": "APPROVED",
        "pointsAwarded": 15,
        "submittedAt": "2025-11-06T10:30:10Z",
        "validatedAt": "2025-11-06T10:45:30Z"
      }
    ],
    "pagination": {
      "currentPage": 0,
      "totalPages": 10,
      "totalItems": 200,
      "itemsPerPage": 20
    }
  }
}
```

GET /reports/{reportId}

Get specific report details.

Response (200 OK):

```
{
  "success": true,
  "data": {
    "reportId": "RPT_20251106_001",
    "reportType": "BUS_SIGHTING",
    "busNumber": "HR26Q4321",
    "location": {
      "latitude": 28.7041,
      "longitude": 77.1025,
      "accuracy": 15.5,
      "stopName": "Main Street Market"
    },
    "photo": {
      "url": "https://s3.amazonaws.com/reports/photo_abc123.jpg",
      "size": 2048576,
      "format": "image/jpeg"
    },
    "description": "Bus arrived at Main Street Market stop, on schedule",
    "passenger": {
      "passengerId": "passenger_123",
      "name": "Arjun Kumar",
      "trustScore": 850,
      "badge": "EAGLE_EYE"
    },
    "status": "APPROVED",
    "adminNotes": "",
    "pointsAwarded": 15,
    "trustAdjustment": 1.5,
    "submittedAt": "2025-11-06T10:30:10Z",
    "validatedAt": "2025-11-06T10:45:30Z"
  }
}
```

POST /reports (PASSENGER ONLY)

Submit a new bus report.

Request Body (multipart/form-data):

```
{
  "busNumber": "HR26Q4321",
  "reportType": "BUS_SIGHTING",
  "location": {
    "latitude": 28.7041,
    "longitude": 77.1025,
    "accuracy": 15.5
  },
  "description": "Bus at Main Street stop",
  "photo": "&lt;FILE_UPLOAD&gt;"
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Report submitted successfully. Admin will review.",
  "data": {
    "reportId": "RPT_20251106_001",
    "status": "PENDING"
  }
}
```

PUT /reports/{reportId}/validate (ADMIN ONLY)

Approve or reject a report.

Request Body:

```
{
  "decision": "APPROVED",
  "notes": ""
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Report approved successfully",
  "data": {
    "reportId": "RPT_20251106_001",
    "status": "APPROVED",
    "pointsAwarded": 15,
    "trustAdjustment": 1.5
  }
}
```

4.7 TRIP MANAGEMENT ENDPOINTS

GET /trips (DRIVER/ADMIN)

Get all trips.

Query Parameters:

- `page` (default: 0)
- `size` (default: 20)
- `busId` (optional)
- `driverId` (optional)
- `status` (optional)
- `startDate` (optional)
- `endDate` (optional)

Response (200 OK):

```
{
  "success": true,
  "data": {
    "trips": [
      {
        "tripId": "TRIP_20251114_001",
        "busNumber": "HR26Q4321",
        "routeNumber": "42",
        "driver": {
          "driverId": "driver_123",
          "name": "Rajesh Sharma"
        },
        "status": "COMPLETED",
        "scheduledStartTime": "2025-11-14T06:00:00Z",
        "actualStartTime": "2025-11-14T06:02:00Z",
        "scheduledEndTime": "2025-11-14T07:00:00Z",
        "actualEndTime": "2025-11-14T07:05:00Z",
        "totalDuration": 63,
        "distanceCovered": 18.7,
        "onTimePercentage": 80
      }
    ],
    "pagination": {
      "currentPage": 0,
      "totalPages": 5,
      "totalItems": 100,
      "itemsPerPage": 20
    }
  }
}
```

GET /trips/{tripId}

Get specific trip with all stop details.

Response (200 OK):

```json
{
  "success": true,
  "data": {
    "tripId": "TRIP_20251114_001",
    "busNumber": "HR26Q4321",
    "routeNumber": "42",
    "driver": {
      "driverId": "driver_123",
      "name": "Rajesh Sharma"
    },
    "status": "COMPLETED",
    "scheduledStartTime": "2025-11-14T06:00:00Z",
    "actualStartTime": "2025-11-14T06:02:00Z",
    "scheduledEndTime": "2025-11-14T07:00:00Z",
    "actualEndTime": "2025-11-14T07:05:00Z",
    "totalDuration": 63,
    "stops": [
      {
        "stopName": "ISBT Sector 43",
        "scheduledArrival": "2025-11-14T06:00:00Z",
        "actualArrival": "2025-11-14T06:02:00Z",
        "delay": 2,
        "status": "COMPLETED"
      }
    ],
    "metrics": {
      "totalStops": 15,
      "stopsCompleted": 15,
      "onTimeStops": 12,
      "delayedStops": 3,
      "avgDelayPerStop": 2.5
    },
    "distanceCovered": 18.7
  }
}
```

POST /trips (DRIVER ONLY)

Start a new trip.

Request Body:

```json
{
  "busId": "bus_789",
  "routeId": "route_42"
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Trip started successfully",
  "data": {
    "tripId": "TRIP_20251114_001",
    "status": "IN_PROGRESS"
  }
}
```

PUT /trips/{tripId}/complete (DRIVER ONLY)

Complete a trip.

Response (200 OK):

```
{
  "success": true,
  "message": "Trip completed successfully",
  "data": {
    "tripId": "TRIP_20251114_001",
    "status": "COMPLETED",
    "totalDuration": 63,
    "distanceCovered": 18.7
  }
}
```

PUT /trips/{tripId}/stops/{stopId}/checkin (DRIVER ONLY)

Check-in at a stop.

Request Body:

```
{
  "location": {
    "latitude": 30.7333,
    "longitude": 76.7933,
    "accuracy": 12
  }
}
```

Response (200 OK):

```
{
  "success": true,
  "message": "Checked in at ISBT Sector 43",
  "data": {
    "stopName": "ISBT Sector 43",
    "checkedInAt": "2025-11-14T06:02:00Z",
    "delay": 2
  }
}
```

4.8 MESSAGING ENDPOINTS

GET /messages (DRIVER/PASSENGER)

Get messages for authenticated user.

Query Parameters:

- `page` (default: 0)
- `size` (default: 50)
- `busId` (optional)

Response (200 OK):

```json
{
  "success": true,
  "data": {
    "messages": [
      {
        "messageId": "MSG_20251114_001",
        "sender": {
          "userId": "passenger_123",
          "name": "Arjun Kumar",
          "role": "PASSENGER"
        },
        "receiver": {
          "userId": "driver_456",
          "name": "Rajesh Sharma",
          "role": "DRIVER"
        },
        "content": "Will the bus reach Sector 17 in 5 minutes?",
        "status": "READ",
        "sentAt": "2025-11-14T10:30:00Z",
        "readAt": "2025-11-14T10:32:00Z"
      }
    ],
    "pagination": {
      "currentPage": 0,
      "totalPages": 2,
      "totalItems": 75,
      "itemsPerPage": 50
    }
  }
}
```

POST /messages (DRIVER/PASSENGER)

Send a message.

Request Body:

```
{
  "receiverId": "driver_456",
  "content": "Will the bus reach Sector 17 in 5 minutes?",
  "busId": "bus_789"
}
```

Response (201 Created):

```
{
  "success": true,
  "message": "Message sent successfully",
  "data": {
    "messageId": "MSG_20251114_001",
    "sentAt": "2025-11-14T10:30:00Z"
  }
}
```

5. AUTHENTICATION & AUTHORIZATION

5.1 JWT Token Flow

```
1. User logs in with email/password
2. Backend validates credentials
3. Backend generates:
   - Access Token (15 min expiry)
   - Refresh Token (7 days expiry)
4. Frontend stores tokens
5. Frontend includes Access Token in all requests
6. Backend validates token on each request
7. When Access Token expires, use Refresh Token to get new one
```

5.2 Authorization Headers

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

5.3 Role-Based Endpoints

Endpoint Pattern	Allowed Roles
`/auth/*`	Public
`/users/me`	Authenticated
`/users/{userId}`	ADMIN
`/buses` (GET)	Authenticated
`/buses` (POST/PUT/DELETE)	ADMIN
`/routes` (GET)	Authenticated
`/routes` (POST/PUT/DELETE)	ADMIN
`/reports` (POST)	PASSENGER
`/reports/validate`	ADMIN
`/trips` (START/COMPLETE)	DRIVER
`/messages`	DRIVER, PASSENGER

6. ERROR HANDLING

6.1 Standard Error Response

```
{
  "success": false,
  "message": "Resource not found",
  "error": "RESOURCE_NOT_FOUND",
  "details": {
    "resource": "Bus",
    "id": "bus_123"
  },
  "timestamp": "2025-11-14T10:30:00Z",
  "path": "/api/buses/bus_123"
}
```

6.2 HTTP Status Codes

Code	Meaning	Example
200	OK	Successful GET/PUT/DELETE
201	Created	Successful POST
400	Bad Request	Validation error
401	Unauthorized	Invalid/missing token
403	Forbidden	Insufficient permissions

Code	Meaning	Example
404	Not Found	Resource doesn't exist
409	Conflict	Duplicate resource
500	Internal Server Error	Server error

7. VALIDATION RULES

7.1 Request Validation

All requests are validated using Spring Validation annotations:

```
@NotNull
@NotEmpty
@NotBlank
@Email
@Size(min = 8, max = 100)
@Pattern(regexp = "...")
@Min(0)
@Max(100)
```

7.2 Custom Validators

- **Phone Number Validator** - Indian format +91XXXXXXXXXX
- **Bus Number Validator** - Format: HR26Q4321
- **GPS Accuracy Validator** - < 50 meters
- **File Upload Validator** - Max 5MB, JPEG/PNG/WebP only

8. PAGINATION & SORTING

All list endpoints support:

```
?page=0&amp;size=20&amp;sortBy=createdAt&amp;sortOrder=DESC
```

Response includes pagination metadata:

```
{
  "pagination": {
    "currentPage": 0,
    "totalPages": 5,
    "totalItems": 100,
    "itemsPerPage": 20,
    "hasNext": true,
    "hasPrevious": false
  }
}
```

9. FILE UPLOAD

9.1 Photo Upload (Reports)

- **Max Size:** 5 MB
- **Allowed Formats:** JPEG, PNG, WebP
- **Storage:** AWS S3
- **Compression:** Auto-compress to 80% quality
- **Thumbnail:** Auto-generate 200x200 thumbnail

9.2 Upload Flow

```
1. User selects photo
2. Frontend compresses image
3. Frontend uploads to /reports endpoint (multipart/form-data)
4. Backend validates (size, format)
5. Backend uploads to S3
6. Backend saves S3 URL in database
7. Backend returns URL to frontend
```

10. RATE LIMITING

To prevent abuse:

Endpoint	Rate Limit
`/auth/login`	5 requests/minute/IP
`/auth/register`	3 requests/hour/IP
`/reports` (POST)	10 requests/hour/user
`/messages` (POST)	30 requests/minute/user
All other endpoints	100 requests/minute/user

11. API DOCUMENTATION

11.1 Swagger UI

Access interactive API documentation at:

```
http://localhost:8080/swagger-ui.html
```

11.2 OpenAPI Spec

Download OpenAPI 3.0 specification at:

```
http://localhost:8080/v3/api-docs
```

12. TESTING STRATEGY

12.1 Unit Tests (JUnit 5 + Mockito)

- Service layer business logic
- Validation rules
- Utility functions
- JWT generation/validation

12.2 Integration Tests

- Controller endpoints
- Repository operations
- Authentication flow
- Complete workflows

12.3 API Testing (Postman)

- All endpoints tested
- Positive and negative scenarios
- Error handling
- Edge cases

13. DEPLOYMENT CONFIGURATION

13.1 Development Environment

```
server.port=8080
spring.data.mongodb.uri=mongodb://localhost:27017/where_is_my_bus_dev
jwt.secret=dev_secret_key_minimum_32_characters
```

13.2 Production Environment

```
server.port=8080
spring.data.mongodb.uri=${MONGODB_URI}
jwt.secret=${JWT_SECRET}
aws.s3.bucket=${S3_BUCKET}
```

14. SUCCESS CRITERIA

✅ All CRUD endpoints working
✅ Authentication & authorization implemented
✅ Role-based access control working
✅ Request validation implemented
✅ Error handling consistent
✅ File upload working
✅ Pagination & sorting functional
✅ Rate limiting configured
✅ API documentation generated
✅ Unit tests passing (>80% coverage)
✅ Integration tests passing

Next Steps: Proceed to **Module 3: Real-Time Services** documentation.