# Task Document: Ticket Management System

## Objective

Develop a backend system for managing bus tickets with the ability for users to purchase tickets for specific buses at specified time periods. The system must include authentication, role-based management (admin and user), and ticket purchasing functionality.

The project is to be developed using Node.js, Express.js, and Mongoose, following a modular design pattern. TypeScript is a plus.

## Requirements

### Core Functionalities

User Authentication:

- User registration, login, and logout.

- Password hashing and JWT-based authentication.

- Role-based authorization (Admin, User).

Admin Functionalities:

- Add, update, delete bus information.

- Upload, update, and delete tickets for buses with specific prices and time slots.

User Functionalities:

- View available buses and tickets.

- Purchase tickets for a specific bus at a specified time.

Additional Requirements:

- Ensure proper validation and error handling for all endpoints.

- Use a modular pattern for scalability and maintainability.

## Deliverables

1. ER Diagram:

- Create an entity-relationship diagram that outlines the relationships between users, buses, and tickets.

2. API Documentation:

- Submit Postman API Documentation with all endpoints, their request/response structures, and sample payloads.

3. Codebase:

- Upload the complete codebase to a GitHub repository and share the link.

- Ensure the repository is well-documented with a clear README.md.

4. Technology Stack:

- Backend: Node.js, Express.js.

- Database: MongoDB with Mongoose.

- Language: TypeScript (preferred but optional).

## Required APIs

1. Authentication APIs

- POST /auth/register: User registration.

- POST /auth/login: User login.

- POST /auth/logout: User logout.

2. Admin APIs

- POST /admin/bus: Add a new bus.

- PUT /admin/bus/:id: Update bus information.

- DELETE /admin/bus/:id: Delete a bus.

- POST /admin/ticket: Upload a new ticket for a specific bus and time.

- PUT /admin/ticket/:id: Update ticket information.

- DELETE /admin/ticket/:id: Delete a ticket.

3. User APIs

- GET /buses: View all available buses.

- GET /tickets: View available tickets for specific buses and time periods.

- POST /tickets/purchase: Purchase a ticket for a specific bus and time.

## Expectations

1. Deadline: Submission by 13 December 2024, 11:59 PM.

2. Database Structure:

- Use Mongoose for database modeling.

- Define schemas for User, Bus, and Ticket.

3. Code Quality:

- Follow modular patterns for scalability.

- Use TypeScript if possible.

- Maintain proper file structure and clean code.

4. Testing:

- Ensure the APIs are tested and functional.

- Include sample test cases for critical flows.

## Evaluation Criteria

1. API Functionality:

- All APIs must be functional and meet the outlined requirements.

2. Postman Documentation:

- Proper documentation of all endpoints, payloads, and responses.

3. Code Quality:

- Maintain clean, modular, and scalable code.

- Proper use of TypeScript (if implemented).

4. Database Design:

- Clear ER diagram with appropriate relationships.

5. GitHub Repository:

- Codebase must be well-documented with a clear README.md.

Good luck!