**PRD for Online Bus E-Ticketing System**

**Introduction**

The Online Bus E-Ticketing System is designed to automate the process of booking bus tickets over the internet. This system aims to provide a user-friendly interface for passengers to book, cancel, and manage their bus tickets online. It also facilitates bus operators in managing their fleets, schedules, and passenger details efficiently.

**Objective**

To design and implement an efficient online platform that allows users to book bus tickets, manage trips, and make payments, ensuring a smooth and streamlined experience for both users and bus operators.

**1. System Overview**

**1.1 Features**

* User registration and authentication
* Bus operator management
* Bus and route management
* Trip scheduling
* Online ticket booking and seat selection
* Payment processing
* Booking history and management
* Administrative dashboard for operators and system admins

**1.2 Stakeholders**

* Passengers
* Bus Operators
* System Administrators

**2. Database Design**

**2.1 Database Tables**

**2.1.1 Users Table**

**Table Name:** users

**Fields:**

* `id` (Primary Key)
* `name` : User's name
* `email`: User’s email (unique)
* `mobile`: User’s mobile (unique)
* `password` : Hashed password
* `role` (e.g., admin, customer)

**2.1.2Cutomers Table**

**Table Name:**customers

**Fields:**

* `id` (Primary Key)
* `user\_id` (Foreign Key)
* `fathers\_name` :
* `mothers\_name`:
* `present\_address`:
* `nid\_number` :
* `picture`

**2.1.3 Operators Table**

**Table Name:** operators

**Fields:**

* `id` (Primary Key)
* `owner\_name`: Name of the owner
* `company\_name`: Name of the bus operator or company
* `address`: Contact information of the operator (phone number, email, etc.)

**2.1.4 Buses Table**

**Table Name:** buses

**Fields:**

* `id` (Primary Key)
* `operator\_id` (Foreign Key referencing operators)
* `bus\_name`: Name or identifier of the bus
* `registration\_number` : Bus registration number
* `total\_seats`: Total number of seats in the bus

**2.1.5 Routes Table**

**Table Name:** routes

**Fields:**

* `id` (Primary Key)
* `route\_name`: Name or identifier of the route
* `origin`: Starting point of the route
* `destination`: Destination of the route
* `distance`: Distance covered by the route
* `duration`: Duration of the journey

**2.1.6 Trips Table**

**Table Name:** trips

**Fields:**

* `id` (Primary Key)
* `bus\_id` (Foreign Key referencing buses)
* `route\_id` (Foreign Key referencing routes)
* `departure\_time`: Date and time of departure
* `arrival\_time`: Estimated date and time of arrival
* `fare`: Fare for the trip
* `available\_seats`: Number of available seats for booking

**2.1.7 Bookings Table**

**Table Name:** bookings

**Fields:**

* `id` (Primary Key)
* `user\_id` (Foreign Key referencing users)
* `bus\_id` (Foreign Key referencing buses)
* `trip\_id` (Foreign Key referencing trips)
* `seat\_number`: Seat number(s) booked by the user
* `status` (e.g., confirmed, pending, cancelled)

**2.1.8 Payments Table**

**Table Name:** payments

**Fields:**

* `id` (Primary Key)
* `booking\_id` (Foreign Key referencing bookings)
* `amount`: Payment amount
* `payment\_method`: Payment method used (Nagad, bkash, etc.)
* `transaction\_id`: Unique identifier for the transaction
* `status` (e.g., successful, pending, failed)

**3. Functional Requirements**

**3.1 User Registration and Login**

* Users must be able to register with their name, email, mobile number, and password.
* Users must be able to log in using their email or mobile number and password.
* Passwords must be stored in a hashed format for security.

**3.2 Bus Operators Management**

* Operators must be able to register and manage their details.
* Operators can add, update, and delete buses associated with their company.

**3.3 Buses and Routes Management**

* Operators can add buses with details such as name, registration number, and total seats.
* Routes can be defined with origin, destination, distance, and duration.

**3.4 Trip Scheduling**

* Operators can schedule trips by associating buses with routes and defining departure and arrival times.
* The fare for each trip and the available seats must be specified.

**3.5 Online Ticket Booking**

* Users can search for available trips based on origin, destination, and travel dates.
* Users can select trips, choose seats, and make bookings.
* Bookings can have different statuses such as confirmed, pending, or cancelled.

**3.6 Payment Processing**

* Users can make payments for their bookings using various payment methods.
* Payment details, including transaction ID and status, must be recorded.

**3.7 Booking Management**

* Users can view their booking history and manage bookings (e.g., cancel or change seats).
* Operators can manage bookings for their trips and update the status as necessary.

**3.8 Administrative Dashboard**

* Admins can manage users, operators, buses, routes, trips, and bookings.
* Admins can view system reports and statistics.

**4. System Architecture**

**4.1 Frontend**

* The frontend will be developed using HTML, CSS, JavaScript, Ajax/Axios and bootstrap.
* The user interface will include forms for **registration, login, booking searches, seat selection, and payment processing.**

**4.2 Backend**

* The backend will be built using a server-side language PHP (**Laravel-10**).
* RESTful APIs will be created to handle requests from the frontend.

**4.3 Database**

* A relational database **MySQL** will be used to store all system data
* The database schema will be designed based on the tables described in the database design section.

**4.4 Security**

* User authentication will be managed using **breeze**.
* Passwords will be hashed using algorithms like bcrypt.
* Data validation and sanitization will be implemented to prevent SQL injection and other common attacks.

**4.5 Deployment**

* The application will be deployed on a cloud platform like AWS, Azure, or Google Cloud.
* Continuous Integration/Continuous Deployment (CI/CD) pipelines will be set up for automated testing and deployment.

**5. Non-Functional Requirements**

**5.1 Performance**

* The system should be able to handle a high number of concurrent users without performance degradation.
* Response times for user actions should be minimized.

**5.2 Scalability**

* The system should be scalable to accommodate future growth in users and data.
* Both horizontal and vertical scaling techniques should be considered.

**5.3 Security**

* Sensitive data must be encrypted both in transit and at rest.
* Regular security audits and updates must be performed.

**5.4 Usability**

* The user interface must be intuitive and easy to navigate.
* Accessibility standards should be adhered to, ensuring the system is usable by people with disabilities.

**5.5 Availability**

* The system should have high availability with minimal downtime.
* Backup and disaster recovery plans should be in place.

**6. Testing**

**6.1 Unit Testing**

* Each module and function will be tested individually to ensure they work as expected.

**6.2 Integration Testing**

* Different modules will be tested together to ensure they interact correctly.

**6.3 System Testing**

* The complete system will be tested to verify that it meets the requirements.

**6.4 User Acceptance Testing (UAT)**

* The system will be tested by a group of end-users to ensure it meets their needs and is user-friendly.

**7. Maintenance**

* Regular maintenance tasks will include updating the software, fixing bugs, and adding new features.
* A support system will be set up to handle user queries and issues.

**Conclusion**

This document outlines the detailed plan for developing an online bus e-ticketing system, including database design, functional and non-functional requirements, system architecture, testing, and maintenance. By following this plan, the system will provide a robust and user-friendly platform for booking bus tickets online, enhancing the travel experience for passengers and operational efficiency for bus operators.