**Project Title: Restaurant Reservation Web Application**

**Objective:**

The goal of this project is to design and develop a fully functional web application where users can book tables at a restaurant. The application will handle user authentication, restaurant table management, and reservation scheduling.

**Project Milestones:**

**1. Requirements Gathering**

Before starting development, outline the requirements:

• **User Requirements**:

• Users should be able to create an account and log in.

• Users can browse restaurant availability.

• Users can book a table for a specific time and date.

• **Admin Requirements**:

• Restaurant admin can manage table availability.

• Admin can view, edit, or cancel reservations.

**2. System Design**

• **Database Design**:

• Define entities like Users, Reservations, Tables, Restaurants.

• Create an ER (Entity-Relationship) diagram that models relationships among these entities.

• **Frontend Design**:

• Plan UI/UX using wireframes. Use tools like Figma or Balsamiq to design the flow for pages like login, reservation, and confirmation.

• The app should be mobile-responsive.

• **Backend Design**:

• Choose a technology stack (e.g., MERN Stack: MongoDB, Express.js, React, Node.js).

• Define RESTful API endpoints, e.g., /reservations, /users, /tables.

**3. Development**

• **Frontend (Client-side)**:

• Implement a user-friendly interface using React (or your preferred front-end framework).

• Build forms for login, signup, reservation search, and confirmation.

• Handle form validations (e.g., date and time selection).

• **Backend (Server-side)**:

• Implement a REST API using Express.js (or your preferred backend framework).

• Connect the API with a database like MongoDB or MySQL for storing reservation data.

• Create authentication (JWT-based) for user and admin roles.

• **Database**:

• Use a database management system (DBMS) like MongoDB or PostgreSQL.

• Implement schema and database models for tables, users, reservations, etc.

**4. Core Features to Implement:**

• **User Authentication**:

• Registration, login, and password management.

• Role-based access control (e.g., Admin vs. regular users).

• **Reservation System**:

• **Table Availability**: Show available time slots for specific dates.

• **Booking**: Allow users to book a table and receive confirmation.

• **View/Cancel Reservations**: Users should be able to see or cancel upcoming reservations.

• **Admin Panel**:

• Admin can manage the number of tables and time slots.

• Admin can view all reservations in a calendar or list view.

**5. Testing:**

• Write unit tests for your backend API using Jest or Mocha.

• Perform frontend testing using tools like Cypress for end-to-end tests.

• Ensure responsiveness and cross-browser compatibility (Chrome, Firefox, Safari).

**6. Deployment:**

• Deploy the web app on platforms like Heroku, Vercel, or Netlify.

• Use CI/CD pipelines for smooth deployments.

• Setup a cloud database service (e.g., MongoDB Atlas) for remote access to the DB.

**Bonus Features (Optional):**

• **Email Notifications**: Automatically send booking confirmation and reminder emails.

• **Google Maps Integration**: Show the restaurant’s location on the map.

• **Reviews System**: Allow users to rate and review their reservation experience.

• **Payment Integration**: Add a feature for users to pay a deposit or full payment when making a reservation.

**Tech Stack (Suggestion):**

• **Frontend**: React.js, HTML5, CSS3, Bootstrap/Tailwind CSS

• **Backend**: Node.js, Express.js

• **Database**: MongoDB (or SQL-based DB like PostgreSQL)

• **Authentication**: JWT (JSON Web Tokens)

• **Deployment**: Heroku or Netlify (Frontend), MongoDB Atlas (Database)

**Deliverables:**

• Fully functioning web app with source code hosted on GitHub.

• Documentation including setup instructions, API documentation, and user manual.

• Demonstration video showing key functionalities.

**Grading Criteria:**

• **Functionality**: The app should meet all core features.

• **Code Quality**: Clean, well-structured, and documented code.

• **UI/UX**: User-friendly, responsive, and visually appealing interface.

• **Testing**: Coverage of key features with proper testing.

• **Deployment**: Successful deployment and accessibility online.