**System Design Document: Badminton Court Booking System**

**1. Introduction:**

The Badminton Court Booking System is designed to facilitate the management of court bookings for two badminton courts. Users can view court availability, book slots in advance, and make payments through their wallets. The system consists of frontend, backend, and database components that work together to provide a seamless user experience.

**2. Frontend Architecture:**

The frontend of the system is developed using React.js, a popular JavaScript library for building user interfaces. Key components and their functionalities include:

* **App Component (App.jsx):** Initializes the application and defines the routing configuration using React Router.
* **Login Component (LoginPage.jsx):** Handles user authentication, allowing users to log in using their email and password. It communicates with the backend server to verify user credentials.
* **SignUp Component (SignUpPage.jsx):** Manages user registration by collecting user details and sending them to the backend for processing.
* **Home Component (HomePage.jsx):** Displays court availability for the next 7 days and allows users to select dates and times for booking. It communicates with the backend to fetch court availability and make bookings.
* **ViewBookings Component (ViewBookings.jsx):** Retrieves and displays the user's booking history, including selected dates, times, and court details.

**2. Backend Architecture:**

The backend of the Badminton Court Booking System is implemented using Node.js and Express.js, providing RESTful APIs for communication with the frontend. It interacts with the MongoDB database to store user information, booking details, and court availability.

* **Express Server (index.js):** Initializes the Express application, configures middleware for parsing JSON and enabling CORS, and defines routes for handling various API endpoints.
* **Routes (routes.js):** Contains route handlers for user registration, login, booking creation, wallet management, and booking history retrieval. Each route handler interacts with the corresponding controller to process requests and return responses.
* **Controllers (routes.js):** Implement business logic for handling requests received from the frontend. Controllers validate input data, interact with the database using Mongoose models, and return appropriate responses to the client.
* **Mongoose Models (User.js, Plot.js):** Define MongoDB schemas for User and Plot collections. These models provide an abstraction layer for interacting with the database and perform CRUD operations on the corresponding collections.

**3. Interaction Flow:**

* **User Registration (/register):** When a user registers, the backend receives the user details in the request body. It checks if the email already exists in the database, creates a new user if not, and returns the user object as a response.
* **User Login (/login):** Upon login, the backend verifies the user's credentials by querying the database. If the email and password match, it returns the user's email, password, and wallet balance as a response.
* **Booking Creation (/booking):** When a user makes a booking, the frontend sends a request containing the booking details. The backend validates the request, checks for sufficient funds in the user's wallet, and creates a new entry in the Plot collection. It then deducts the booking amount from the user's wallet and updates the wallet balance in the User collection.
* **Wallet Retrieval (/fetchWallet):** To fetch the user's wallet balance, the frontend sends a request with the user's email. The backend retrieves the wallet balance from the User collection and returns it as a response.
* **Booking History Retrieval (/fetchBookingsByEmail):** When the frontend requests the user's booking history, the backend queries the Plot collection for bookings associated with the user's email. It retrieves the bookings and returns them as a response.

**4. Conclusion:**

The backend architecture of the Badminton Court Booking System ensures efficient handling of user requests, secure data management, and seamless interaction with the frontend. By following the RESTful principles and utilizing Mongoose for MongoDB interaction, the system provides a reliable and scalable solution for managing court bookings and user wallets.