Bus Booking System

Akash Kolli
2019B3A70426G

TechStack used:

1. Backend Technologies:

- **Node.js and Express.js**: Leveraged Node.js as the server-side runtime environment and Express.js as the web application framework to handle server-side operations and API integrations efficiently.
- **MongoDB**: Chose MongoDB as the document database to store and manage data, providing flexibility in data handling and scalability for the application.
- **JWT Authentication**: Implemented JSON Web Token (JWT) authentication to ensure secure user access and authorization within the system.
- **Axios**: Employed Axios, a promise-based HTTP client, for making API calls to interact with the server, enabling seamless data exchange between the client and server.

2. Frontend Technologies:

- **React.js**: Utilised React.js as the client-side JavaScript framework for building a dynamic and responsive user interface, facilitating a modern and interactive user experience.
- **Ant Design Library**: Integrated the Ant Design library for UI components to create consistent, visually appealing, and user-friendly interfaces, leveraging its extensive set of pre-designed components and styles.
- **Redux Toolkit**: Implemented Redux Toolkit to manage application state and streamline data flow within the application, ensuring efficient state management and data consistency across components.

OOPS

userSchema:

Used to store the information regarding users.

Property	Datatype	Description
name	String	Name of the User
email	String	Email Address
password	String	Password
isAdmin	Boolean	To determine whether user is an Admin or not
isBlocked	Boolean	To determine whether the user is Blocked or not

busSchema:

Used to store the information regarding buses.

Property	Datatype	Description
name	String	Name of the Bus
number	String	Bus Number
capacity	Number	Capacity of the bus
from	String	Source
to	String	Destination
journeyDate	String	Journey Date
departure	String	Departure time
arrival	String	Arrival time
type	String	Bus Type(AC or Non-AC)
fare	Number	Ticket Price
seatsBooked	Array	Booked seats
status	String	Status of the Bus(Yet to leave, etc)

bookingSchema:

Used to store information regarding bookings.

Property	Datatype	Description
bus	ObjectId	BusId of the current booking
user	ObjectId	UserId of the current booking
seats	Array	Seats Booked in this booking
transactionId	String	TransactionId of the booking

Authentication:

1. **Authentication Protocols:**
   - **JWT (JSON Web Token)**: Utilised JWT for secure user authentication. Upon successful login, JWT tokens are generated, allowing secure access to authorised endpoints. This token-based stateless authentication mechanism enhances security and simplifies subsequent authorization.

2. **Password Encryption:**
   - **bcryptjs Encryption**: All user passwords are encrypted using the bcryptjs library before storage in the system's database. This encryption methodology significantly strengthens security measures, ensuring that passwords are securely hashed and safeguarded.

3. **Token-based User Identification and Validation:**
   - Storage of UserId as a Token: UserId is stored as a token upon successful authentication. This token is validated whenever necessary, eliminating the need for additional API calls to authenticate and validate UserId for subsequent requests. This streamlined approach enhances authentication efficiency.

4. **User Interface Restrictions:**
   - Restricted Access to Login/Register Pages: Users who are already logged in (i.e., UserId stored as a token in localStorage) are prohibited from accessing the Login or Register pages. This ensures that authenticated users cannot inadvertently access these pages, maintaining security and user flow.
   - Access for New or Logged-Out Users: New or logged-out users have access to the Login and Register pages, enabling them to log in or register their accounts without hindrance.

5. **Route Handling for Authentication:**

Private and Public Routes: The system employs Private and Public Routes to manage authentication.

- Public Routes: These routes, including Login and Register pages, are accessible to all users, whether authenticated or not.
- Private Routes: Accessible only to authenticated users, Private Routes secure sensitive areas of the application, ensuring that only authenticated users can access these functionalities.

The integration of JWT-based authentication, bcryptjs password encryption, and the use of Private and Public Routes ensures a robust and secure authentication system within the Bus Booking System. By implementing these measures, the system maintains high-security standards while ensuring an optimal user experience for both new and authenticated users.

Error and Exception Handling:

Handled all the negative scenarios.
Example: User cannot register two accounts with same email address, Two buses cannot have same us number

1. **Preventing Duplicate Entries:**
   - **Examples:**
   - **Unique Email Validation:** During user registration, the system validates that a user cannot register two accounts with the same email address. If an email address already exists in the system, the registration process is halted, and the user is prompted to use a different email address.
   - **Unique Bus Numbers:** When adding or updating bus details, the system ensures that two buses cannot have the same bus number (unique identifier). This validation prevents the creation of duplicate bus entries, maintaining the integrity of bus information.

2. **Validation:**
Form Validation: Implemented client-side and server-side form validation mechanisms to prevent invalid or duplicate entries. This validation process occurs in real-time as users input information, ensuring that data integrity is maintained before submission.

3. **Data Integrity Measures:**
Database Constraints: Utilised database constraints (such as unique constraints on columns) to enforce data integrity at the database level. These constraints act as an additional layer of protection, preventing the insertion of duplicate or conflicting data.

The implementation of measures to handle negative scenarios within the Bus Booking System ensures robust data integrity and system stability. By preventing duplicate entries, providing clear error messages, and implementing comprehensive validation mechanisms, the system maintains the accuracy and reliability of user and bus information. These strategies contribute to an enhanced user experience and a more reliable system overall.

API Endpoints:

1. User API Endpoints:

API Endpoints	Request Type	Description
/register	POST	New User Registration
/login	POST	Login user
/get-user-by-id	POST	Get User data by UserId
/get-all-users	POST	Get All Users
/update-user-permissions	POST	Updates the User permissions(ex: Admin or Normal User)

2. Bus API Endpoints:

API Endpoints	Request Type	Description
/add-bus	POST	Add New Bus
/update-bus	POST	Update the existing bus
/delete-bus	POST	Delete bus
/get-all-buses	POST	Get All Buses
/get-bus-by-id	POST	Get bus data by id

3. Booking API Endpoints:

API Endpoints	Request Type	Description
/book-seat	POST	Book Seat
/make-payment	POST	Make payment
/get-bookings-by-user-id	POST	Get all bookings of a User
/get-all-bookings	POST	Get All Bookings

Features:

1. **Distinctive Seat Assignment:**
    - Distinctive Seat Numbers: Following a successful booking, each user is assigned a unique and distinctive seat number. This ensures that each user has an allocated seat for their journey, enhancing organisation and user experience.

2. **Colour-coded Seat Availability Indicator:**
    - Seat Availability Colour Codes: Utilised a visual indicator system employing colour codes to help users quickly assess the availability of bus seats.
    - **Gray (Selection Phase):** Seats selected by the user are displayed in gray during the seat selection phase.
    - **Green (60% Full or Less)**: Seats are displayed in green when the occupancy percentage is 60% or less. This colour code signals ample availability, allowing users to comfortably book their seats.
    - **Yellow (60% - 90% Full):** When the occupancy percentage falls between 60% and 90%, seats are shown in yellow. This serves as a cautionary indicator, informing users that seats are filling up, but moderate availability still exists.
    - **Red (90% - 100% Full)**: Seats are displayed in red if the occupancy percentage ranges from 90% to 100%. This colour code indicates limited availability, prompting users to act quickly to secure their seats due to high demand.

3. **Seat Availability Checking:**
    - Real-time Seat Availability: Implemented seat availability checking functionality, allowing users to view the current availability status of seats on selected buses. This feature assists users in making informed booking decisions.

4. **Admin Management of Tickets and Users:**
    - **Admin Control:** From the Admin Page, administrators can manage tickets and user activities efficiently.
    - Ticket and User Management: Admins can oversee and manage ticket details and user accounts, ensuring streamlined operations and facilitating necessary administrative actions.

5. **User Management by Admin:**
    - **User Blocking and Unblocking:** Admins possess the ability to block or unblock users based on various criteria, allowing for enhanced control over user access and permissions.

6. **User Role Distinction:**

Two User Types: The system distinguishes between two types of users: Admin and Normal Users.

- **Admin Role**: Admins have extended privileges, including bus management, user control, and permissions modification.
- **Normal User Role**: Normal users can view available buses and book tickets but have limited administrative functionalities.

7**. Admin Authority over Bus Operations:**
- Bus Operations Management: Admins have full control over bus operations, enabling them to add, edit, or delete buses as needed to maintain an updated database.

8. **User Permission Alteration by Admin:**
- User Permission Control: Admins have the authority to modify user permissions, allowing them to grant admin privileges or restrict user access as necessary.

9. **User Capabilities:**
- Normal User Functions: Normal users can browse available buses and proceed with ticket bookings, providing a streamlined and user-centric experience.

Additional Features:

1. Stripe Payment Integration:
- **Stripe Integration:** The Bus Booking System has successfully integrated Stripe, a robust and secure payment processing platform, to handle payment transactions seamlessly within the application.
- Payment Processing for Bookings: Users leverage the Stripe payment system to finalise transactions while making bus bookings through the application.
- Secure Payment Gateway: Stripe provides a secure payment gateway, ensuring the confidentiality of users' financial information and enabling secure and encrypted transactions.

2. Features and Benefits of Stripe Integration:
- Convenient Booking Process: The Stripe integration streamlines the bus booking process by enabling users to make payments directly within the application. This feature significantly enhances user convenience and facilitates quicker transactions.

3. Printing Tickets:
- **Ticket Printing Capability:** Implemented the **useReactToPrint** library to empower users with the ability to print tickets directly from the application.
- **Ticket Information:** The printed tickets contain comprehensive details, including Bus Name, Source, Destination, Journey Date, Journey Time, Seat Numbers, and the Total Amount paid for the booking.

4. Enhanced Bus Filtering:
- **Filtering Based on Source, Destination, and Date:** Implemented enhanced filtering functionality that allows users to filter available buses based on specific criteria, such as source location, destination, and journey date. This feature provides users with tailored bus options that match their preferences and travel plans.