

Brief for Admin Dashboard

Project Overview

The objective is to create an Admin Dashboard for an Airbnb clone application. The dashboard should allow administrators to create, view, update, and delete property listings. Additionally, the admin dashboard should have a user authentication system to manage login, logout, and user sessions.

Expected Skills and Technologies

- **Frontend:** React.js, CSS
- **Backend:** Node.js, Express.js, MongoDB (CRUD operations are excluded from this rubric but included for backend rubric)
- **Authentication:** JWT (JSON Web Tokens)
- **File Handling:** Image upload and display (**Optional**)

Instructions

Admin Dashboard Instructions

1. Top Header:

- Implement a top header that includes the Airbnb logo and navigation links.
- When a user is logged in, display a greeting with their username and provide a dropdown menu with options to view reservations and log out.
- When a user is logged out, display a "Become a host" link.

2. Login Page:

- Create a login page with a form that includes fields for email and password.
- Implement input validation and display appropriate error messages for invalid inputs.
- On successful login, redirect the user to the admin dashboard.

3. Create Listing Page:

- Develop a form for creating new property listings with all necessary fields (title, location, description, bedrooms, bathrooms, guests, type, price, amenities, images, weekly discount, cleaning fee, service fee, occupancy taxes).
- Ensure the form has robust input validation and displays clear error messages.
- Include functionality for image uploads.

4. View Listings Page:

- Display a list of all property listings with key details (title, location, price, main image).
- Provide options to update or delete each listing.

5. Update Listing Page:

- Similar to the create listing page, but pre-fill the form with the existing data of the listing being updated.
- Ensure that updates are saved and reflected correctly.

6. User Authentication:

- Implement user authentication using JWT.
- Ensure that user sessions are maintained and only logged-in users can access the admin dashboard.

- Add functionality to the profile icon in the header to display a dropdown menu with options to view reservations and log out.

Recommended user Data Structure

```
{  
    username: 'John Doe',  
    password: 'password123',  
    role: 'user',  
},  
{  
    username: 'Jane Doe',  
    password: 'password321',  
    role: 'host',  
},
```

7. Navigation and Routing:

- Ensure smooth navigation between different pages.
- Update the URL to reflect the current view.

8. Styling:

- Apply consistent styling across the application.

9. Error Handling and Feedback:

- Implement error handling throughout the application.
- Provide clear feedback to users in case of errors.

10. Code Quality and Documentation:

- Write clean, well-organized, and commented code.
- Ensure that functions are modular and reusable.

Rubric for Admin Dashboard

Instructions for Scoring

- **10 marks:** Exceeds expectations, fully functional, and polished implementation.
- **7-9 marks:** Meets expectations with minor issues or missing minor elements.
- **4-6 marks:** Needs improvement with noticeable issues or missing elements.
- **1-3 marks:** Unsatisfactory, major issues, or non-functional.

Total Marks: 100

Brief for Airbnb Frontend Clone

The goal of this project is to create a functional and visually appealing frontend for an Airbnb clone using React. The project will cover three main views: Home Page, Location Page, and Location Details Page. Each view will have specific components and functionalities to meet the project requirements. The Home Page will be static with a dynamic filter and login functionality. The Location Page will display a list of locations based on user selection, and the Location Details Page will provide detailed information about a selected location, including a cost calculator and reservation functionality.

Views and Components

Home Page

- Hero Banner
- Inspiration for your next trip (with location cards)
- Discover Airbnb Experiences
 - Things to do on your trip (with static button and background image)
 - Things to do at home (with static button and background image)
- ShopAirbnb Section
 - Two columns: title and button on one side, image of gift cards on the other
- Inspiration for future getaways
 - Static tabs with one displaying content in a list format
- Static Footer
 - List of links in 4 columns
- Copyright Footer
 - Copyright text, social links, language selector, currency selector

Location Page

- Location Filter
- Location Cards
 - Image on the left, details on the right (type of accommodation, name, amenities, average star ranking, total reviews, cost per night)
- Heading
 - Total accommodations for the selected location and location name

Location Details Page

- Accommodation Type and Location (Heading)
- Subheading
 - Average star review and location
- Image Gallery
 - Large image on the left, four smaller images stacked 2 over 2
- Two Columns Layout
 - Left: accommodation details
 - Right: cost calculator
- Cost Calculator
 - Cost per night x total nights
 - Weekly discount
 - Cleaning fee
 - Service fee
 - Occupancy taxes and fees

- Dynamic updates with date pickers and guest count
- Reservation button to create a reservation in MongoDB

Recommended Reservation Data Structure

```
{
  id: 1,
  images: [
    "/images/new-york-lady-of-liberty.jpg",
    "/images/new-york-lady-of-liberty.jpg",
    "/images/new-york-lady-of-liberty.jpg",
    "/images/new-york-lady-of-liberty.jpg",
    "/images/new-york-lady-of-liberty.jpg",
    "/images/new-york-lady-of-liberty.jpg",
  ],
  type: "Entire apartment",
  location: "New York",
  guests: 4,
  bedrooms: 2,
  bathrooms: 2,
  amenities: ["wifi", "kitchen", "free parking"],
  rating: 4.5,
  reviews: 320,
  price: 320,
  title: "Modern Apartment in New York",
  host: "Johann",
  host_id: "6676f16fdace0e26aed41e79",
  weeklyDiscount: 0,
  cleaningFee: 50,
  serviceFee: 50,
  occupancyTaxes: 30,
  enhancedCleaning: true,
  selfCheckIn: true,
  description: "Stay in the heart of New York City...",
  specificRatings: {
    cleanliness: 4.8,
    communication: 4.7,
    checkIn: 4.9,
    accuracy: 4.6,
    location: 4.9,
    value: 4.5,
  }
}
```

Static Information Sections (Left Column)

- Accommodation details
- Where you'll sleep
- What this place offers
- 7 nights in New York
- Reviews

- Host Details
- House Rules, Health & Safety, Cancellation Policy
- Static Footer and Copyright Footer

Top Header

- Logo
- Location Filter
- Profile Section
- Login page or view reservations in a table format

Rubric for Frontend

Instructions for Scoring

- **10 marks:** Exceeds expectations, fully functional, and polished implementation.
- **7-9 marks:** Meets expectations with minor issues or missing minor elements.
- **4-6 marks:** Needs improvement with noticeable issues or missing elements.
- **1-3 marks:** Unsatisfactory, major issues, or non-functional.

Total Marks: 140

Brief for Node.js Backend

Objective:

The objective of this project is to create a backend system for an Airbnb clone using Node.js, Express, and MongoDB. The backend will handle **CRUD** operations for accommodation listings and Reservations, and users will require **reading** operations for authentication and validation.

Technology Stack:

- **Node.js**: Runtime environment for executing JavaScript code server-side.
- **Express.js**: Web framework for Node.js to build the API.
- **MongoDB**: NoSQL database to store the data.
- **Mongoose**: ODM library for MongoDB.
- **JWT**: For authentication and authorization.
- **Multer**: Middleware for handling multipart/form-data (image uploads). (**Optional**)

Expected Project Structure:

- controllers
 - accommodationController.js
 - reservationController.js
 - userController.js
- models
 - Accommodation.js
 - Reservation.js
 - User.js
- routes
 - accommodationRoutes.js
 - reservationRoutes.js
 - userRoutes.js
- middleware
 - auth.js
- server.js

Functional Requirements:

1. Accommodation Management:

- Create accommodation listing (POST /api/accommodations)
- Read all accommodation listings (GET /api/accommodations)
- Delete an accommodation listing (DELETE /api/accommodations/:id)

2. User Authentication:

- User login (POST /api/users/login)

3. Reservation Management:

- Create a reservation (POST /api/reservations)
- Get reservations by host (GET /api/reservations/host)
- Get reservations by user (GET /api/reservations/user)
- Delete a reservation (DELETE /api/reservations/:id)

Non-Functional Requirements:

- Use of JWT for authentication and authorization.
- Proper error handling and status codes.
- Connection to MongoDB using Mongoose.
- Middleware for authentication (auth.js).
- Modular and clean code structure.

Instructions for Scoring

- **10 marks:** Exceeds expectations, fully functional, and polished implementation.
- **7-9 marks:** Meets expectations with minor issues or missing minor elements.
- **4-6 marks:** Needs improvement with noticeable issues or missing elements.
- **1-3 marks:** Unsatisfactory, major issues, or non-functional.

Total Marks: 150