Architecture Document for HouseHunt Project

1. Introduction

- **Project Name**: Househunt: Finding your perfect rental home
- **Date**: June 22, 2025

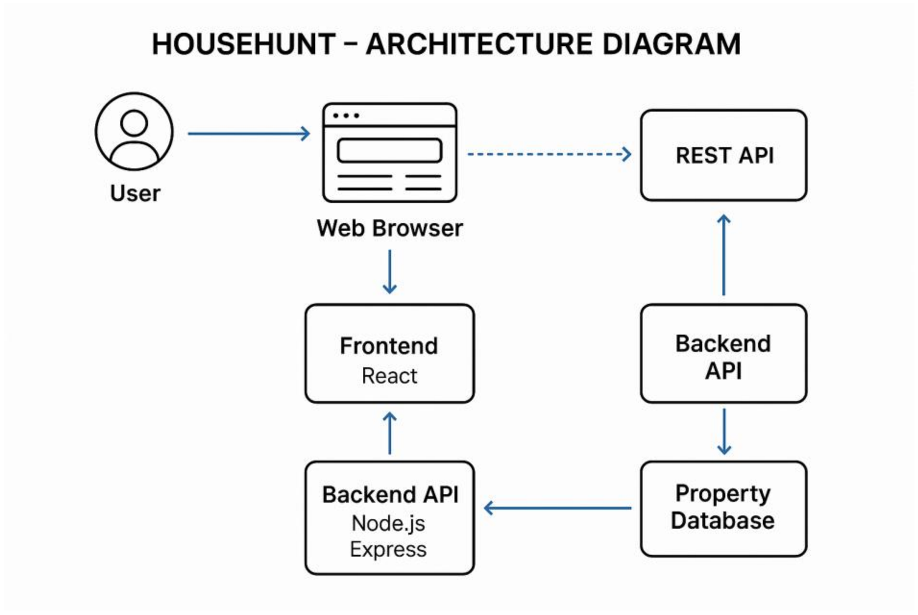
Team ID: LTVIP2025TMID57110

- **Version Control**: https://github.com/VenkataSandeep2/househunt-project.git

2. Project Overview

- **Purpose**: The HouseHunt project aims to provide a comprehensive platform for property rentals, connecting renters, owners, and administrators through a user-friendly interface.
- **Technology Stack**:
    - **Frontend**: React.js
    - **Backend**: Node.js, Express.js
    - **Database**: MongoDB
    - **Authentication**: JSON Web Tokens (JWT)

3. Architecture Diagram



Note: Replace the placeholder image with an actual architecture diagram that illustrates the interaction between the frontend, backend, and database.

4. Components Overview

4.1 Frontend

- **Framework**: React.js
- **Key Directories**:
    - **src/**: Contains the main source code.
        - **components/**: Reusable UI components.
        - **modules/**: Different modules for functionalities (Admin, User, etc.).
        - **images/**: Image assets used in the application.
    - **public/**: Static files like HTML and images.

4.2 Backend

- **Framework**: Node.js with Express.js
- **Key Directories**:
    - **controllers/**: Contains logic for handling requests and responses.
        - **adminController.js**: Logic for admin-related operations.
        - **ownerController.js**: Logic for owner-related operations.
        - **userController.js**: Logic for user-related operations.
    - **routes/**: API endpoint definitions.
        - **adminRoutes.js**: Routes for admin functionalities.
        - **ownerRoutes.js**: Routes for owner functionalities.
        - **userRoutes.js**: Routes for user functionalities.
    - **middlewares/**: Custom middleware functions for request processing.
    - **schemas/**: Database schemas for data modeling (Mongoose models).

4.3 Database

- **Database**: MongoDB

- **Key Models**:
    - **User Model**: Defines the structure for user data.
    - **Property Model**: Defines the structure for property listings.
    - **Booking Model**: Defines the structure for booking requests.

5. User Authentication

- **Method**: JSON Web Tokens (JWT)
- **Flow**:
    - Users register and log in to receive JWT tokens.
    - Protected routes utilize authentication middleware to verify user roles (Admin, Owner, User).

6. API Endpoints

6.1 User Authentication

- **POST /api/auth/register**: Register a new user.
- **POST /api/auth/login**: Log in a user and return a JWT.

6.2 Property Management

- **GET /api/properties**: Retrieve all properties.
- **POST /api/properties**: Add a new property (Admin/Owner).
- **PUT /api/properties/:id**: Update property details (Admin/Owner).
- **DELETE /api/properties/:id**: Delete a property (Admin).

6.3 Booking Management

- **POST /api/bookings**: Create a new booking request.
- **GET /api/bookings**: Retrieve all bookings (Admin).
- **PUT /api/bookings/:id**: Update booking status (Admin).

7. Deployment

- **Environment**: The application can be deployed on platforms like Heroku, AWS, or DigitalOcean.
- **Build Process**: Use npm scripts to build the frontend and start the backend server.

8. Future Enhancements

- **Mobile Application**: Develop a mobile version of the application.
- **Payment Integration**: Implement payment gateways for transactions.
- **Advanced Search Filters**: Enhance property search capabilities.
- **User Reviews and Ratings**: Allow users to review properties.

Conclusion

This Solution Architecture Document outlines the structure and components of the HouseHunt project, providing a clear understanding of its architecture and functionalities. For further details or contributions, please refer to the repository or contact the team members.