**Technical Design Document**

**Rooster Rentals**

**Project Overview**

**Rooster** Rentals is a MERN-stack (MongoDB, Express, React, Node.js) based full-stack web platform that allows users to list and rent apartments, manage rental information, process payments, and track rental revenue. The application supports authentication, CRUD operations, messaging, and real-time features.

**1.System Architecture**

**A diagram of a computer program

AI-generated content may be incorrect.**

**2.Tech Stack**

|  |  |
| --- | --- |
| **Type** | **Tech stack** |
| Frontend | React.js, Next.js, Tailwind css, Axios, CQRS |
| Backend | Node.js, Express.js, Mongoose |
| Database | MongoDB |
| Authentication | JWT + bcrypt |
| Image Uploads | Cloudinary |
| Real-time chat | Socket.IO |
| Payments | Stripe |
| Deployment | Aws/Netlify/Render |
| AI Integration | Open AI+Claude |
| testing | Jest for backend, React Testing Library for frontend |
| Google Maps |  |

**3.Key +Features**

**i.** Authentication

* LongIn/Signup(JWT)
* Forget/Reset password
* Login/SignUp(social media)

ii. Rental Posting Page

* Create/Edit/Delete rental listings
* Upload multiple images per listing
* Rental terms: price, availability, description, location

iii. Booking system

* Book rental dates (calendar UI)
* Checkout with Stripe
* Track payments and rental status

iv. Real-time Chat

* One-on-one messaging between renter and owner
* Socket.IO with WebSocket support

4. Database Schema(MongoDB)

|  |  |
| --- | --- |
| Users: | full\_Name, email, password, phone,dob |
| rentalInfo: | userId,picture location, room details-size, furnished, amenities- ac/wifi/heat/parking/laundry/  price, tenants requirements-first&last, pet friendly, security deposits etc  booking availability , comment and reviews, sharing |
| profile | Fullname, userphoto, validId,location, |
| paymentHistory | Rent paid/rent received/ |
| myRentals | Rental agreements, users rental score, inventory, |
| Chats | User1-user2 seen/unseen/chathistory |
|  |  |
|  |  |
|  |  |
|  |  |

* **Users**

{

\_id,

name,

email,

passwordHash,

profileImage,

createdAt

}

* **Rentals**

{

\_id,

title,

description,

location,

pricePerDays/months/year,

images: [URLs],

ownerId (ref to User),

availability: [Date],

createdAt

}

* **Bookings**

{

\_id,

userId,

rentalId,

startDate,

endDate,

totalPrice,

status: "pending" | "confirmed",

createdAt

}

* Messages

{

\_id,

senderId,

receiverId,

content,

chatId,

timestamp

}

**5. Stripe Payment Flow**

1. User selects rental dates → Clicks “Book Now”
2. Frontend sends request to /api/checkout/create
3. Backend creates Stripe Checkout Session → returns URL
4. User is redirected to Stripe to pay
5. Webhook listens for successful payment → creates booking

**7.Deployment**

**Frontend(Reatc)**

* Host on Netlify or vercel or aws
* .env for api based url

Backend(Node.js/Express)

* Host on Render/aws
* Store in env variables

Database

* MongoDB Atlas

Ci/CD

* Github

Testing

* Unit tests (Jest for backend, React Testing Library for frontend)

use roosterRentals