ANAS HAKEEM

Roll # 00061712

Name : ANAS HAKEEM

Day: 2 Activity

Roll No: 00061712

Market Place type: Rental E-commerce

Email: anashakeem05@gmail.com

RJ Rentique

1.Technical Documentation

Frontend Requirements:

- **User-Friendly Interface:**
  - A clean, intuitive design for browsing rental properties.
- **Responsive Design:**
  - Fully optimized for mobile and desktop users.
- **Essential Pages:**
  - **Home Page:** Overview of available rentals, featured listings, and user navigation.
  - **Property Listing Page:** Filters for location, price, and amenities.
  - **Property Details Page:** Detailed property information, including images, descriptions, and contact options.
  - **Booking Page:** A user-friendly form for booking requests.
  - **Confirmation Page:** Displays booking details and a confirmation message.

Sanity CMS as Backend:

- **Schema Design:**
  Focused on aligning the CMS structure with rental platform requirements:
  - **Property Schema:**
    - Fields: Property name, description, images, location, amenities, price, and booking availability.

    - **User Schema:**
        - Fields: Family name, contact details, and booking history.
    - **Booking Schema:**
        - Fields: User ID, property ID, booking date, and special requests.
- **Purpose of Sanity CMS:**
    - Store and manage property data, user information, and booking records effectively.

Third-Party APIs:

- **Shipment Tracking API:** *(If required for post-booking services, e.g., moving assistance)*
    - Implement integration for users to track moving services.
- **Payment Gateway API:** *(Optional for future scalability)*
    - Include options for secure online payments for property booking fees.
- **Map and Geolocation API:**
    - Display property locations and nearby amenities interactively.

Ensuring API Functionality:

- Ensure all integrated APIs deliver data required for the frontend, such as booking status and location details.
- **Design link Work Flow:**
- https://lucid.app/lucidchart/3e40a51b-a3b1-424b-95ff-59ef5ed9bf05/edit?viewport_loc=-604%2C-1093%2C3330%2C1407%2C0_0&invitationId=inv_d4da9833-99c8-48a4-925d-140bcaabcf5d

---

2. Design System Architecture

Homeowner:

- **List Property:** Starts a chat flow for creating a new property listing in Sanity CMS. This flow might involve:
    - Entering property details (address, description, amenities, photos).
    - Setting rental price and availability.
    - Uploading property photos.
- **Manage Listings:** Opens a chat flow for managing existing property listings. This could include:

    - Editing property details.
    - Updating availability.
    - Communicating with potential renters who have inquired about the property.
- **Booking Requests:** Receives notifications and manages chat interactions with renters who have requested to book a property. This might involve:
    - Approving or rejecting booking requests.
    - Answering questions from potential renters.
    - Exchanging contact information to finalize booking details (outside the app).

Renter:

- **Search Properties:** Uses the chat flow to interact with the search functionality. This could involve:
    - Specifying search criteria (location, price range, amenities).
    - Filtering search results.
    - Asking the system questions about available properties (potentially using a chatbot).
- **View Property Details:** Opens a chat flow to view details of a specific property. This might involve:
    - Seeing photos, descriptions, and amenities.
    - Asking the system questions about the property (potentially using a chatbot).
- **Book Property:** Initiates a chat flow to express interest in booking a property. This could involve:
    - Sending booking requests to the homeowner.
    - Asking questions about the property or booking process.
    - Providing additional information about themselves (if required).

General Chat:

- **Help & Support:** Opens a chat flow to access help documentation, FAQs, or contact customer support.
- **Report Issues:** Initiates a chat flow to report any issues with the application or property listings.

Modern Additional Considerations:

- The chat flow lines might differ depending on the specific functionalities you implement in your application.
- I can use a combination of chatbots and human agents to handle user interactions.
- Consider integrating sentiment analysis to understand user intent and provide better support.
- Design link architecture :
- https://lucid.app/lucidchart/6eff0be2-3b5f-483c-a882-fe95b07b30be/edit?view_items=bmrKS9WZWVuc&invitationId=inv_0c3a8eee-51f2-4414-8815-fb92f5db4baf

In this architecture a typical data flow could look like this:

- A user visits the rental marketplace frontend to browse properties.
- The frontend makes a request to the Product Data API (powered by Sanity CMS) to fetch property listings and details, which are displayed dynamically on the site.
- When the user requests to book a property, the booking request details (property ID, user details, dates, etc.) are sent to Sanity CMS via an API request, where the booking request is recorded.
- (Optional) If applicable, shipment tracking information for furniture delivery or other related services is fetched through a Third-Party API and displayed to the user in real-time.
- Payment details (security deposit, booking fee) are securely processed through the Payment Gateway, and a confirmation is sent back to the user and recorded in Sanity CMS.

3. Plan API Requirements

Data Structure (Simplified - Sanity Schema)

- **Property**
  - **_id:** (Unique ID)
  - **name:** (String)
  - **description:** (Text)
  - **address:** (Geopoint)
  - **city:** (String)
  - **images:** (Array of images)
  - **amenities:** (Array of strings - e.g., "Wi-Fi", "Parking", "Balcony")
  - **price:** (Number)
  - **availability:** (Boolean or Date range)
  - **owner:** (Reference to Owner document)
- **Owner**

    - **_id:** (Unique ID)
    - **name:** (String)
    - **email:** (String)
    - **phone:** (String)
- **Booking**
    - **_id:** (Unique ID)
    - **property:** (Reference to Property document)
    - **renter:** (Object or Reference to User document)
    - **startDate:** (Date)
    - **endDate:** (Date)
    - **totalPrice:** (Number)
    - **status:** (String - e.g., "Pending", "Confirmed", "Cancelled")
    - **paymentStatus:** (String - e.g., "Pending", "Success", "Failed")
    - **createdAt:** (Timestamp)
- **User** (Optional - if you have user accounts)
    - **_id:** (Unique ID)
    - **name:** (String)
    - **email:** (String)
    - **phone:** (String)

API Endpoints

- **/properties**
    - **Method:** GET
    - **Description:** Fetch all available properties from Sanity.
    - **Response:**
        - Array of objects, each containing:
            - _id
            - name
            - address (simplified - e.g., "City, State")
            - price
            - images (URLs or references)
            - amenities
- **/properties/:propertyId**
    - **Method:** GET
    - **Description:** Fetch details of a specific property.
    - **Response:** Full Property document (including description, owner details, etc.)
- **/properties/search**
    - **Method:** GET
    - **Description:** Search for properties based on filters (e.g., location, price range, amenities).
    - **Query Parameters:** (e.g., city, minPrice, maxPrice, amenities)
    - **Response:** Array of matching properties.
- **/bookings**
    - **Method:** POST

    - **Description:** Create a new booking request.
    - **Payload:**
        - propertyId
        - renter (user ID or basic user info)
        - startDate
        - endDate
    - **Response:**
        - bookingId
        - status (e.g., "Pending")
- **/bookings/:bookingId**
    - **Method:** GET
    - **Description:** Get details of a specific booking.
    - **Response:** Booking document.
- **/bookings/:bookingId/status**
    - **Method:** PUT
    - **Description:** Update the status of a booking (e.g., "Confirmed", "Cancelled").
    - **Payload:** status
- **/payments**
    - **Method:** POST
    - **Description:** Process a payment for a booking.
    - **Payload:** (Payment Gateway specific)
    - **Response:**
        - paymentId
        - status (e.g., "Success", "Failed")
- **/users** (Optional - if you have user accounts)
    - **Method:** POST
    - **Description:** Create a new user account.
    - **Payload:** User information.

Rental Duration Endpoint (Example)

- **Endpoint Name:** /bookings/:bookingId/duration
- **Method:** PUT
- **Description:** Update the rental duration of a booking.
- **Payload:** duration (e.g., "7 days", "14 days")
- **Response:**
    - bookingId
    - status (e.g., "Duration Updated")

4.Write Technical Documentation

Marketplace Technical Foundation - " RJ Rentique "

1. System Architecture Overview

Diagram:

```
flowchart LR
  A[Frontend (Next.js)] --> B(API Gateway)
  B --> C[Property Service]
  B --> D[Booking Service]
  B --> E[Payment Service]
  B --> F[Notification Service]
  C --> G[Sanity CMS]
  D --> G
  G --> C
  G --> D
  E --> H[Payment Gateway]
  H --> E
  F --> I[Email/SMS Provider]
  I --> F
```

Components:

- **Frontend (Next.js):** User interface for property browsing, booking, and user management.
- **API Gateway:** Handles all API requests, routing them to appropriate services.
- **Property Service:** Manages property data, including fetching, searching, and creating new listings.
- **Booking Service:** Handles booking requests, including creation, confirmation, cancellation, and payment integration.
- **Payment Service:** Processes payments through integrated payment gateways.
- **Notification Service:** Sends notifications (email, SMS) to users.
- **Sanity CMS:** Stores and manages all content, including property data, user information, and booking details.
- **Payment Gateway:** Processes online payments securely.
- **Email/SMS Provider:** Sends notifications to users via email or SMS.

2. Key Workflows

- **User Property Search:**
    1. User enters search criteria (location, price, amenities) on the frontend.
    2. Frontend sends the search request to the API Gateway.
    3. API Gateway forwards the request to the Property Service.
    4. Property Service queries Sanity CMS for matching properties.
    5. Property Service returns the results to the API Gateway.
    6. API Gateway sends the property listings to the frontend for display.
- **Booking a Property:**
    1. User selects a property and initiates a booking request on the frontend.
    2. Frontend sends booking details (property ID, dates, user information) to the API Gateway.
    3. API Gateway forwards the request to the Booking Service.
    4. Booking Service creates a new booking record in Sanity CMS.
    5. Booking Service sends a request to the Payment Service to process the payment.
    6. Payment Service processes the payment and updates the booking status in Sanity CMS.
    7. Notification Service sends a booking confirmation email/SMS to the user.
- **Homeowner Listing a Property:**
    1. Homeowner accesses the "List Property" section on the frontend.
    2. Frontend allows the homeowner to enter property details.
    3. Frontend sends the property data to the API Gateway.
    4. API Gateway forwards the data to the Property Service.
    5. Property Service creates a new property document in Sanity CMS.

3. Category-Specific Instructions

- **Rental eCommerce:**
  - **Workflows:**
    - **Rental Duration:**
      - Endpoint: `/bookings/{bookingId}/duration`
      - Method: PUT
      - Description: Update the rental duration of a booking.
      - Payload: { "duration": "7 days" }
    - **Condition Reports:**
      - Schema Field: `conditionStatus` (String) - e.g., "Good", "Fair", "Damaged"
      - Workflow: Allow homeowners to submit condition reports after each rental.
    - **Return Management:**
      - Workflow: Handle return requests from renters.
      - Functionality: Allow renters to initiate return requests, track return status, and manage potential disputes.
      -
- **Sanity Schema Example (Property)**

```
import { defineField, defineType } from 'sanity';


export default defineType({
  name: 'property',
  title: 'Property',
  type: 'document',
  fields: [
    defineField({
      name: 'name',
      title: 'Property Name',
```

```
    type: 'string',
  }),
  defineField({
    name: 'description',
    title: 'Description',
    type: 'text',
  }),
  defineField({
    name: 'address',
    title: 'Address',
    type: 'geopoint',
  }),
  defineField({
    name: 'city',
    title: 'City',
    type: 'string',
  }),
  defineField({
    name: 'images',
    title: 'Images',
    type: 'array',
    of: [{ type: 'image' }],
```

```
  }),
  defineField({
    name: 'amenities',
    title: 'Amenities',
    type: 'array',
    of: [{ type: 'string' }],
  }),
  defineField({
    name: 'pricePerNight',
    title: 'Price Per Night',
    type: 'number',
  }),
  defineField({
    name: 'availability',
    title: 'Availability',
    type: 'array',
    of: [{ type: 'date' }],
  }),
  defineField({
    name: 'owner',
    title: 'Owner',
    type: 'reference',
```

```
    to: [{ type: 'owner' }],
    }),
  ],
});
```

4. API Endpoints

Endpoint	Method	Purpose	Response Example
/properties	GET	Fetches all available properties	[{ "_id": "1", "name": "Cozy Apartment", "pricePerNight": 100 }, ...]
/properties/:propertyId	GET	Fetches details of a specific property	{ "_id": "1", "name": "Cozy Apartment", "description": "...", ... }
/properties/search	GET	Searches for properties based on filters	[{ "_id": "2", "name": "Modern House", ... }, ...]
/bookings	POST	Creates a new booking request	{ "bookingId": "123", "status": "Pending" }
/bookings/:bookingId	GET	Fetches details of a specific booking	{ "_id": "123", "property": { "_ref": "ref(property.1)" }, ... }
/bookings/:bookingId/status	PUT	Updates the status of a booking	{ "status": "Confirmed" }
/payments	POST	Processes a payment	{ "paymentId": "abc", "status": "Success" }