Full Stack Development with MERN

Project Documentation

1. Introduction

• **Project Title:** House Rent App

• **Team Members:**

Mohammed Ahsen – Project Manager & Testing: Responsible for testing, overseeing the entire project, ensuring timelines are met, and leading the team.

Jaweeth Akther – Database Administrator: Manages and optimizes the database, ensuring data integrity and efficient querying.

Nadeem Hussain – Backend Developer: In charge of designing and implementing the server-side logic, APIs, and database interactions.

Abubakker Siddiq – Frontend Developer: Develops and styles the user interface, ensuring a seamless and responsive user experience.

2. Project Overview

Purpose: The House Rent App aims to simplify the process of renting and managing rental properties by providing an intuitive platform for property owners and renters.

Features:

- **Property Listing & Management:** Users can add, edit, and delete property listings.

- **User Registration & Login:** Secure authentication for property owners, renters, and administrators.

- **Search & Filter Options:** Users can filter properties by location, price range, property type, etc.

- **Booking System:** Renters can book properties and view their booking status.

- **Admin Panel:** Admins can manage user accounts and oversee bookings and listings.

- **Responsive UI:** Optimized for use on various devices, including desktops, tablets, and smartphones.

3. Architecture

• Frontend:

- Developed using React, the app utilizes functional components and React Hooks for state management.

- The design is component-based, making it scalable and easy to maintain. React-Bootstrap is used for a sleek and responsive UI design.

• Backend:

- The server is built with Node.js and Express.js. It handles CRUD operations, user authentication, and integrates with a MongoDB database.

- RESTful API structure is implemented for clean and efficient communication between the frontend and backend.

• Database:

- MongoDB is used as the database. The schema is designed using Mongoose to handle data efficiently, with collections for users, properties, and bookings.

4. Setup Instructions

- **Prerequisites:**

  - Node.js: Ensure you have the latest version installed.

  - MongoDB: A running instance of MongoDB is required for database operations.

  - npm (Node Package Manager): Comes with Node.js.

- **Installation:**

1. Clone the repository:

```
git clone https://github.com/Ahsen2004/NM-Project---House-Rent-App.git
```

2. Navigate to the project directory:

```
cd RenderGO
```

3. Install dependencies for both frontend and backend:

   - For frontend:

```
npm install
```

   - For backend:

```
npm install
```

4. Set up environment variables: - Create a .env file in the server directory and define variables such as MongoDB URI, JWT Key & Port number.

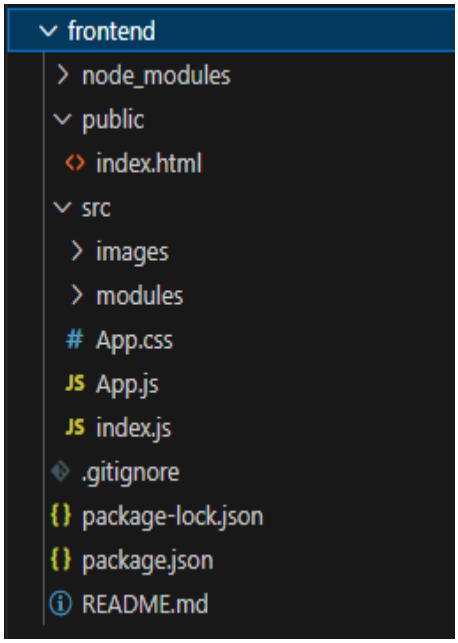
MONGODB_URI=mongodb+srv://nadeemnadeem:NadeemBot@nadeem1.pmoha.mongodb.net/HR-DB-Main?retryWrites=true&w=majority&appName=Nadeem1

JWT_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6InVzZXIxMjMiLCJuYW1lIjoiSm9obiBEb2UiLCJlbWFpbCI6ImplZmYuZG9lQGV4YW1wbGUuY29tIiwiaWF0IjoxNjA4MjkzMjAwLCJleHAiOjE2MDgyOTY4MDB9.ZrCtC1POj7u_YK6M7Lq6UO_i4SjGZmydhvAhI80A3g8

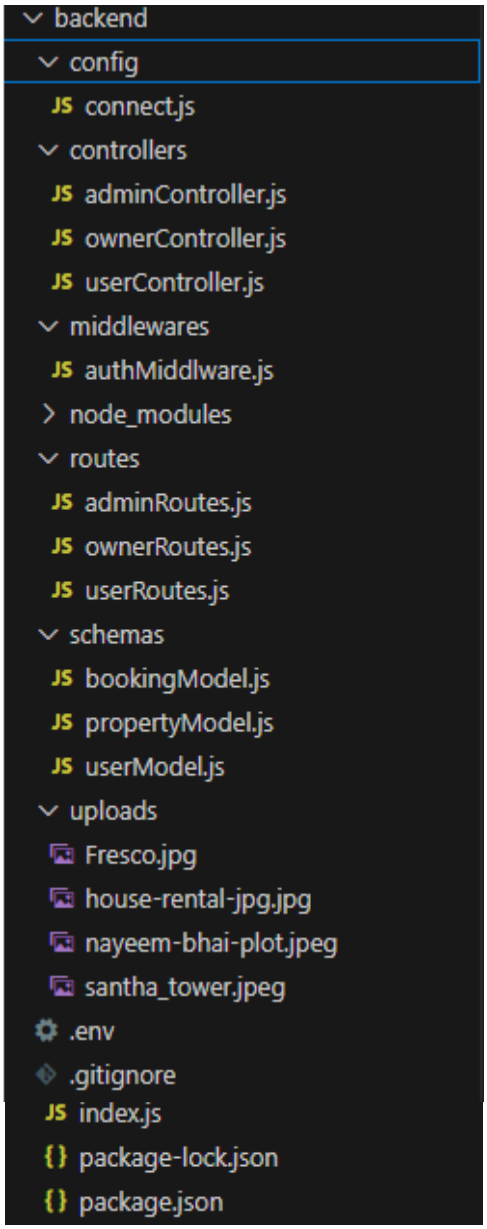
PORT=8001

5. Folder Structure

- **Frontend:**



- **Backend:**



6. Running the Application

• Frontend:

```
npm start
```

• Backend:

```
npm start
```

• Access the app at: http://localhost:3000

7. API Documentation

Base URL: All endpoints are prefixed with /api.

1. User Routes (/api/user):
   a. Register User:
      - **Endpoint:** POST /api/user/register
      - **Request Body:**
        - email (string, required)
        - password (string, required)
        - type (string, required)
      - **Example Request:**
        {
        "email": "example@example.com",
        "password": "securepassword123",
        "type": "Owner"
        }
      - **Example Responses**
        - **200:** { "message": "User already exists", "success": false }
        - **201:** { "message": "Registered Successfully", "success": true }
        - **500:** { "message": "Error details", "success": false }

   b. Login User:
      - **Endpoint:** POST /api/user/login

- **Request Body:**
  - email (string, required)
  - password (string, required)
- **Example Request:**
  {
  "email": "example@example.com",
  "password": "securepassword123"
  }
- **Example Responses:**
  - **200:** {"message": "Login successfully", "success": true, "token": "jwt_token_here", "user": {"email": "example@example.com", "type": "UserType"}}
  - **200:** { "message": "User not found", "success": false } or { "message": "Invalid email or password", "success": false }
  - **500:** { "message": "Error details", "success": false }

c. Forgot Password:
- **Endpoint:** POST /api/user/forgotpassword
- **Request Body:**
  - email (string, required)
  - password (string, required)
- **Example Request:**
  {
  "email": "example@example.com",
  "password": "newsecurepassword123"
  }
- **Example Responses:**
  - **200:** { "message": "Password changed successfully", "success": true }
  - **200:** { "message": "User not found", "success": false }
  - **500:** { "message": "Error details", "success": false }

d. Get All Properties:
- **Endpoint:** GET /api/user/getAllProperties
- **Headers:** Authorization: Bearer <token>
- **Example Request:**
  GET /api/user/getAllProperties
- **Example Responses:**

- **200:** {"success": true, "data": [{"_id": "propertyId1", "name": "Beautiful Beach House", "location": "California", "price": 5000000, "description": "A luxurious beach house with ocean views.", "owner": "User123", "createdAt": "2024-01-15T12:00:00Z"}]}
- **404:** {"success": false, "message": "No properties available"}
- **500:** {"success": false, "message": "An error occurred while fetching properties", "error": "Error details here"}

e. Book Property:

- **Endpoint:** POST /api/user/bookinghandle/:propertyid
- **Request Body:**
  - **UserDetails:** fullName (string), phone (string)
  - status (string)
  - userId (string)
  - ownerId (string)
- **Example Request:**
  ```
  {
    "userDetails": {
      "fullName": "Alice Smith",
      "phone": "987-654-3210"
    },
    "status": "pending",
    "userId": "user789",
    "ownerId": "owner321"
  }
  ```
- **Example Responses:**
  - **200:** {"success": true, "message": "Booking status updated"}
  - **500:** {"success": false, "message": "Error handling booking"}

f. Get All Bookings:

- **Endpoint:** GET /api/user/getallbookings
- **Request Query Parameters:**
  - userId (string, required)
- **Example Request:**
  GET /api/user/bookings?userId=user123
- **Example Responses:**
  - **200:** {"success": true, "data": [{"_id": "bookingId1", "propertyId": "propertyId123", "userID": "user123",

"ownerID": "owner456", "userName": "John Doe", "phone": "123-456-7890", "bookingStatus": "confirmed"}]}

- **500:** {"message": "Internal server error", "success": false}

2. Admin Routes (/api/admin):
   a. Get All Users:
      - **Endpoint:** GET /api/admin/getallusers
      - **Example Request:**
        GET /api/admin/getallusers
      - **Example Responses:**
        - **200:** {"success": true, "message": "All users", "data": [{"_id": "userId1", "email": "user1@example.com", "type": "User"}, {"_id": "userId2", "email": "user2@example.com", "type": "Owner"}]}
        - **401:** {"success": false, "message": "No users present"}
        - **500:** {"success": false, "message": "Error retrieving users"}

   b. Handle User Status:
      - **Endpoint:** POST /api/admin/handlestatus
      - **Request Body:**
        - userid (string, required)
        - status (string, required)
      - **Example Request:**
        {
        "userid": "userId1",
        "status": "active"
        }
      - **Example Responses:**
        - **200:** {"success": true, "message": "User has been active"}
        - **500:** {"success": false, "message": "Error updating user status"}

   c. Get All Properties:
      - **Endpoint:** GET /api/admin/getallproperties
      - **Example Request:**
        GET /api/admin/properties

- **Example Responses:**
    - **200:** {"success": true, "message": "All properties", "data": [{"_id": "propertyId1", "name": "Beachfront Villa", "location": "California", "price": 2000000, "description": "A luxury villa with a stunning view of the beach.", "owner": "owner123", "createdAt": "2024-01-10T12:00:00Z"}]}
    - **401:** {"success": false, "message": "No properties present"}
    - **500:** {"success": false, "message": "Error retrieving properties"}

d. Get All Bookings:
- **Endpoint:** GET /api/admin/getallbookings
- **Example Request:**
  GET /api/admin/bookings
- **Example Responses:**
    - **200:** {"success": true, "data": [{"_id": "bookingId1", "propertyId": "propertyId123", "userID": "user123", "ownerID": "owner456", "userName": "John Doe", "phone": "123-456-7890", "bookingStatus": "confirmed"}]}
    - **500:** {"success": false, "message": "Error retrieving bookings"}

3. Owner Routes (/api/owner)
   a. Add Property:
   - **Endpoint:** POST /api/owner/addproperty
   - **Request Body:**
     - userId (string, required)
     - propertyTitle (string, required)
     - propertyDescription (string, required)
     - propertyType (string, required)
     - price (number, required)
     - location (string, required)
     - propertyImage (Array of files, optional)
   - **Example Request:**
     {"userId": "602d2149e773f2a3990b47f5",
     "propertyTitle": "Luxury Villa in Beverly Hills",

"propertyDescription": "A beautiful luxury villa with ocean views",
"propertyType": "Villa",
"price": 2500000,
"location": "Beverly Hills, CA"
}

- **Example Responses:**
  - **200:** {"success": true, "message": "New Property has been stored"}
  - **500:** {"success": false, "message": "An error occurred while adding the property."}

b. Get All Properties by Owner:

- **Endpoint:** POST /api/owner/getallproperties
- **Request Parameters:**
  - userId (string, required)
- **Example Request:**
  {"userId": "602d2149e773f2a3990b47f5"}
- **Example Responses:**
  - **200:**{"success":true,"data":[{"_id":"propertyId1","propertyTitle":"Luxury Villa in Beverly Hills","propertyDescription":"A beautiful luxury villa with ocean views","propertyType":"Villa","price":2500000,"location":"Beverly Hills, CA","propertyImage":[{"filename":"image1.jpg","path":"/uploads/image1.jpg"},{"filename":"image2.jpg","path":"/uploads/image2.jpg"}],"ownerId":"602d2149e773f2a3990b47f5","ownerName":"John Doe","isAvailable":"Available"}]}
  - **500:** {"success": false, "message": "Internal server error"}

c. Delete Property:

- **Endpoint:** DELETE /api/owner/deleteproperty/:propertyid
- **Path Parameters:**
  - propertyid (string, required)
- **Example Request:**
  DELETE /api/owner/deleteproperty/602d2149e773f2a3990b47f5
- **Example Responses:**
  - **200:** {"success": true, "message": "The property is deleted"}
  - **500:** {"success": false, "message": "Internal server error"}

d. Update Property:
   - **Endpoint:** PUT /api/owner/updateproperty/:propertyid
   - **Path Parameters:**
     - propertyid (string, required)
   - **Request Parameters:**
     - propertyTitle (string, optional)
     - propertyDescription (string, optional)
     - propertyType (string, optional)
     - price (number, optional)
     - location (string, optional)
     - userId (string, required)
   - **Example Request:**
     PUT /api/owner/updateproperty/602d2149e773f2a3990b47f5
     {
       "propertyTitle": "Updated Villa Title",
       "propertyDescription": "Updated description for the luxury villa",
       "price": 2750000,
       "location": "Malibu, CA",
       "userId": "602d2149e773f2a3990b47f5"
     }
   - **Example Responses:**
     - **200:** {"success": true, "message": "Property updated successfully."}
     - **500:** {"success": false, "message": "Failed to update property."}

e. Get All Bookings:
   - **Endpoint:** GET /api/owner/getbookings/:userId
   - **Path Parameters:**
     - userId (string, required)
   - **Example Request:**
     GET /api/owner/getbookings/602d2149e773f2a3990b47f5
   - **Example Responses:**
     - **200:**{"success":true,"data":[{"_id":"bookingId1","propertyId":"propertyId1","propertyName":"Luxury Villa in Beverly Hills","ownerID":"602d2149e773f2a3990b47f5","userId":"603e2149e773f2a3990b47a3","bookingDate":"2024-11-13T00:00:00.000Z","status":"Confirmed"}]}
     - **500:** {"success": false,"message": "Internal server error"}

f. Handle Booking Status:

- **Endpoint:** PUT /api/owner/handlebookingstatus
- **Request Parameters:**
  - bookingId (string, required)
  - propertyId (string, required)
  - status (string, required)
- **Example Request:**

```
{
  "bookingId": "bookingId1",
  "propertyId": "propertyId1",
  "status": "booked"
}
```

- **Example Responses:**
  - **200:** {"success": true, "message": "Changed the status of property to booked"}
  - **500:** {"success": false, "message": "Internal server error"}

8. Authentication

• Token-based Authentication:

- Upon successful login, users receive a JSON Web Token (JWT), which is stored in local storage.

- Protected routes on the backend require a valid token, verified using middleware.

9. User Interface

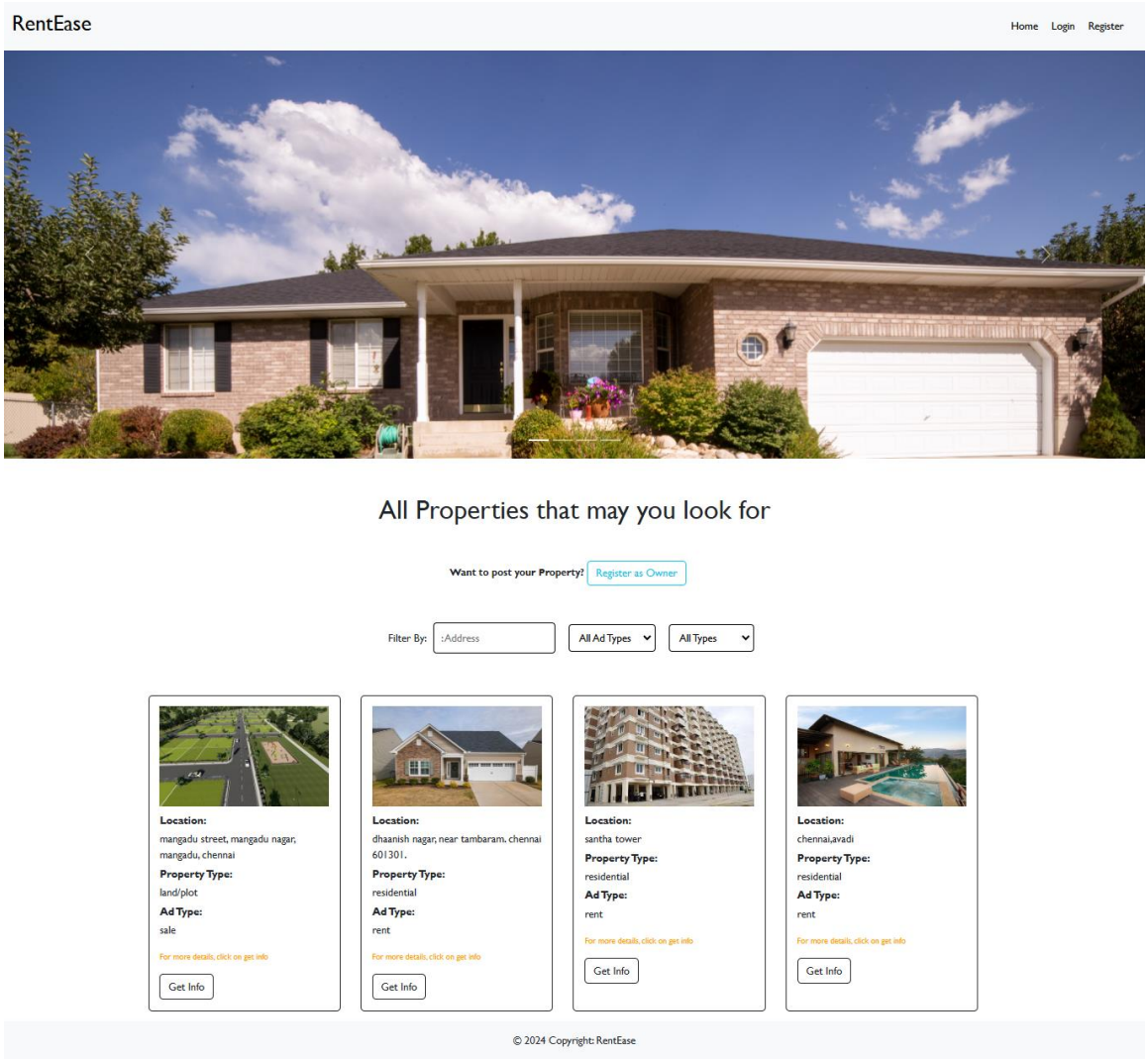
• UI: Screenshots of the main user interface features will be included.

10. Testing

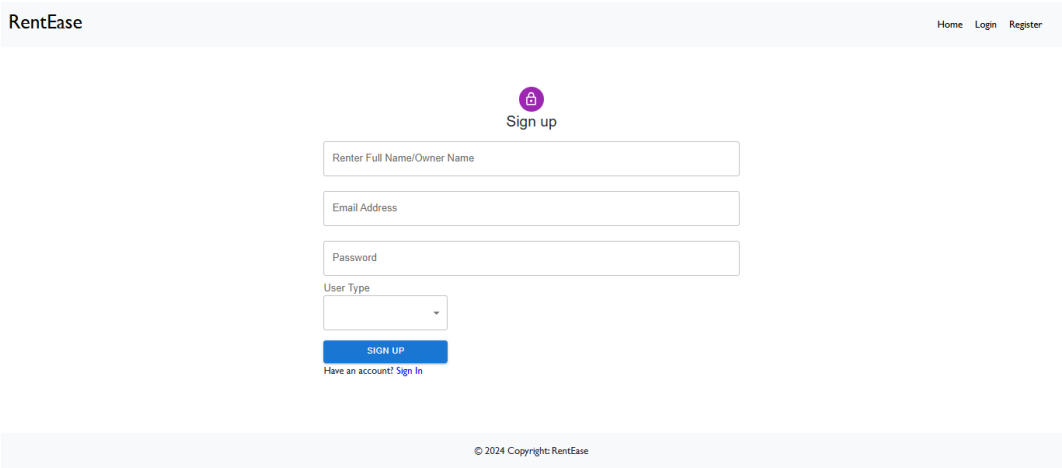
• Testing: Unit and integration tests are implemented using tools like Jest for the frontend and Mocha for the backend.

11. Screenshots or Demo

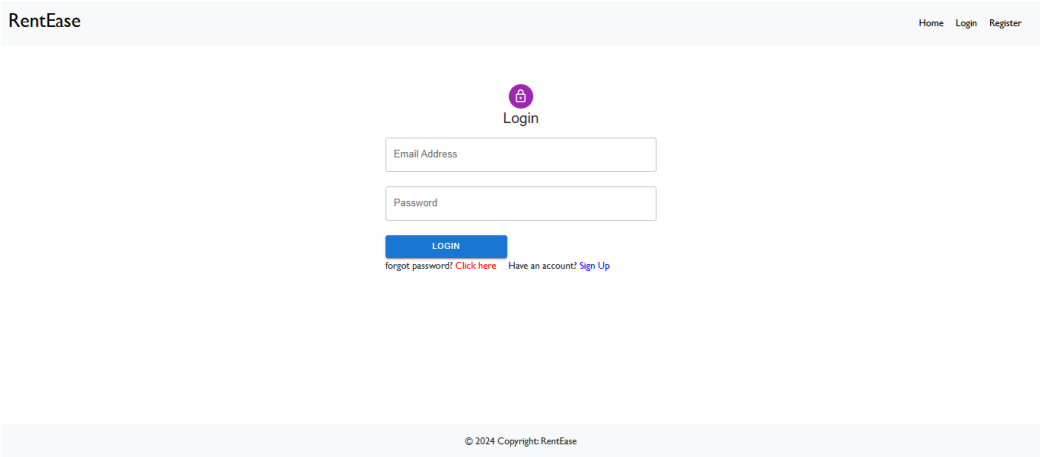
Home page:



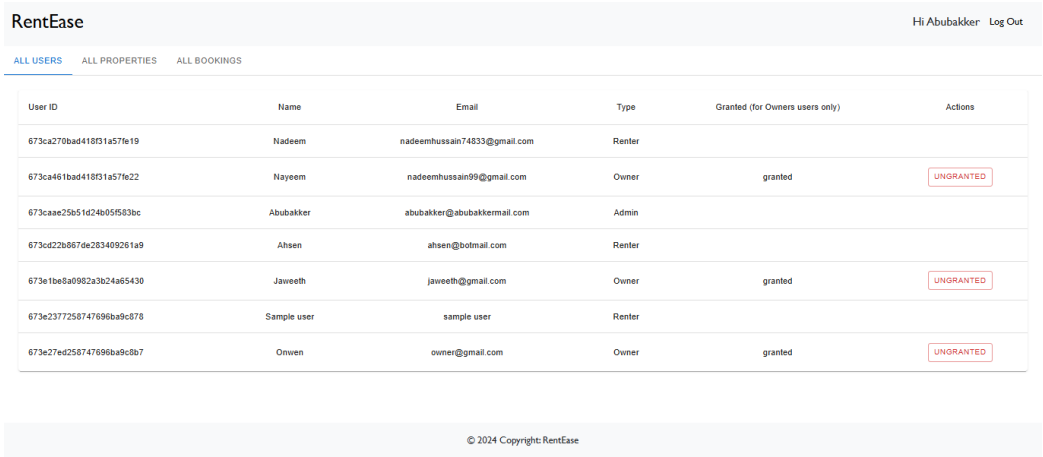
Register page:



Login page:



Admin page:



User ID	Name	Email	Type	Granted (for Owners users only)	Actions
673ca270bad418f31a57fe19	Nadeem	nadeemhussain74833@gmail.com	Renter		
673ca461bad418f31a57fe22	Nayeem	nadeemhussain99@gmail.com	Owner	granted	UNGRANTED
673caae25b51d24b05f583bc	Abubakker	abubakker@abubakkermail.com	Admin		
673cd22b867de283409261a9	Ahsen	ahsen@botmail.com	Renter		
673e1be8a0982a3b24a65430	Jaweeth	jaweeth@gmail.com	Owner	granted	UNGRANTED
673e2377258747696ba9c878	Sample user	sample user	Renter		
673e27ed258747696ba9c8b7	Onwen	owner@gmail.com	Owner	granted	UNGRANTED

Renter page:

RentEase

Hi Sample user Log Out

ALL PROPERTIES BOOKING HISTORY

Booking ID	Property ID	Tenent Name	Phone	Booking Status
673e23b2258747696ba9c87f	673e234b258747696ba9c875	sample	77756863	pending
673e2756258747696ba9c8ac	673cd810867de283409261f3	sa	0	pending
673e2762258747696ba9c8ae	673cd810867de283409261f3	sample	4585484	pending
673e2890258747696ba9c8cf	673e2860258747696ba9c8c6	sample	48546	booked

RentEase

Hi Sample user Log Out

ALL PROPERTIES BOOKING HISTORY

Filter By: :Address All Ad Types All Types

Location:
mangadu street, mangadu nagar, mangadu, chennai
Property Type:
land/plot
Ad Type:
sale
Owner Contact:
9042910801
Availabilty:
Unavailable
Property Amount:
Rs.150000000

Not Available

Location:
dhaanish nagar, near tambaram, chennai 601301.
Property Type:
residential
Ad Type:
rent
Owner Contact:
9876543201
Availabilty:
Available
Property Amount:
Rs.1500

Get More Info of the Property Get Info

Location:
santha tower
Property Type:
residential
Ad Type:
rent
Owner Contact:
8877796563
Availabilty:
Available
Property Amount:
Rs.15000

Get More Info of the Property Get Info

Location:
chennai,avadi
Property Type:
residential
Ad Type:
rent
Owner Contact:
48545646
Availabilty:
Available
Property Amount:
Rs.150000

Get More Info of the Property Get Info

Owner page:

RentEase

Hi Onwen Log Out

ADD PROPERTY ALL PROPERTIES ALL BOOKINGS

Property type: Residential

Property Ad type: Rent

Property Full Address: Address

Property Images: Choose Files No file chosen

Owner Contact No.: contact number

Property Amt.: 0

Additional details for the Property

Submit form

RentEase

Hi Onwen Log Out

ADD PROPERTY ALL PROPERTIES ALL BOOKINGS

Booking ID	Property ID	Tenent Name	Tenent Phone	Booking Status	Actions
673e2890258747696ba9c8cf	673e2860258747696ba9c8c6	sample	48546	pending	Change

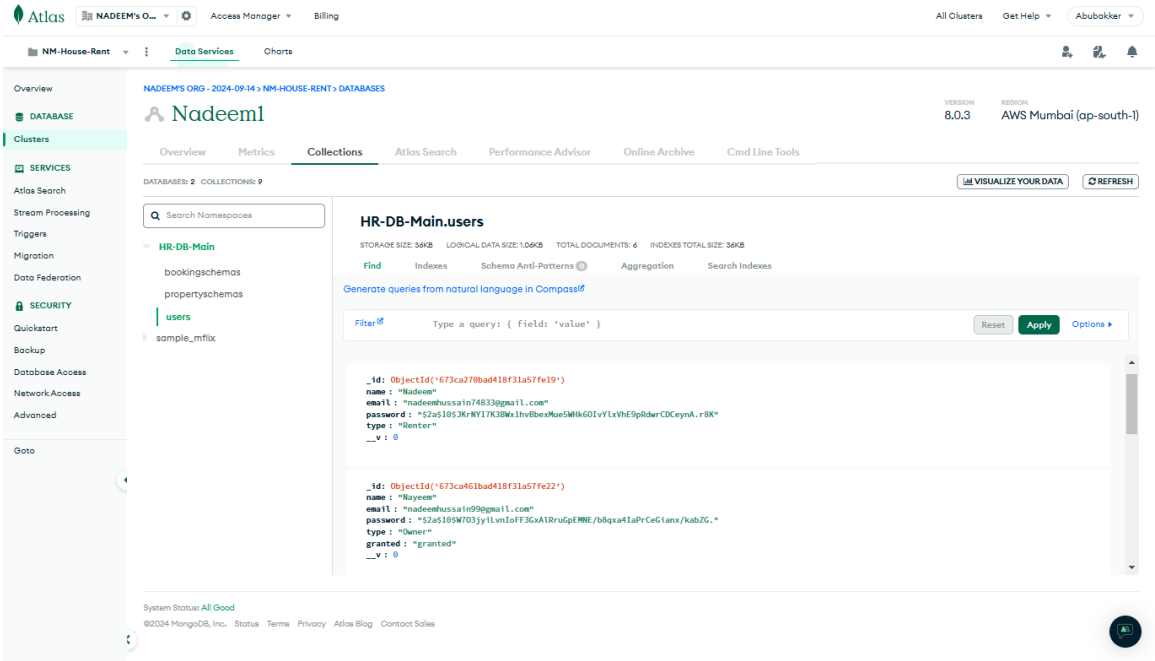
RentEase

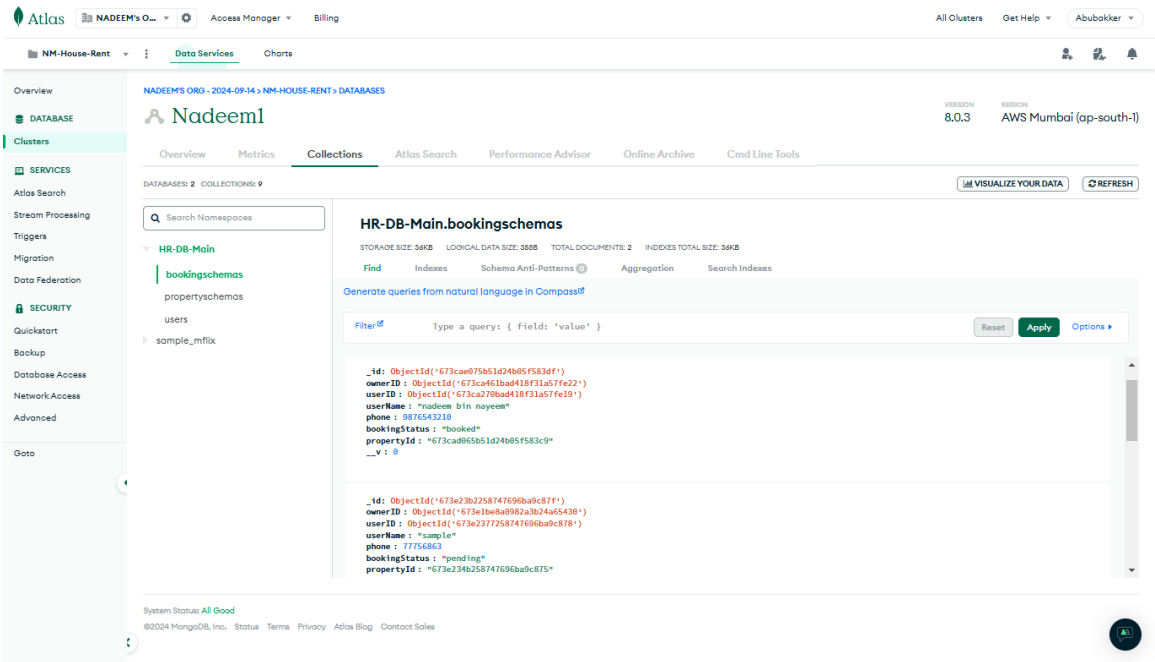
Hi Onwen Log Out

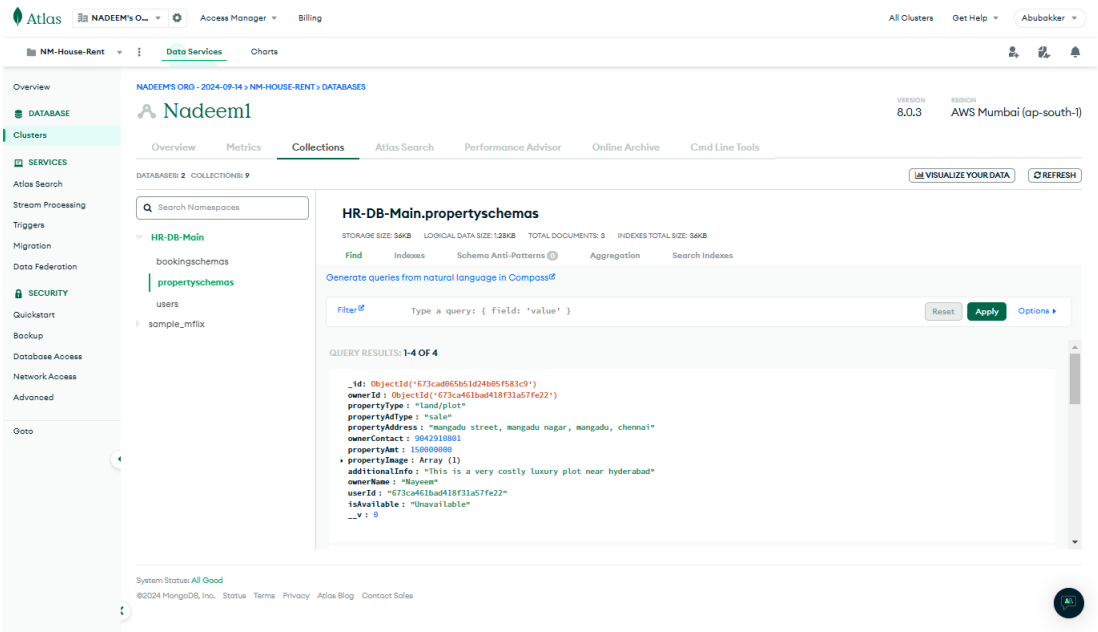
ADD PROPERTY ALL PROPERTIES ALL BOOKINGS

Property ID	Property Type	Property Ad Type	Property Address	Owner Contact	Property Amt	Property Availabilty	Action
673e2860258747696ba9c8c6	residential	rent	chennai,avadi	48545646	150000	Available	Edit Delete

Database:


Atlas
NADEEM's O...
Access Manager
Billing
All Clusters
Get Help
Abubakker
NM-House-Rent
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Goto
NADEEM'S ORG - 2024-09-14 > NM-HOUSE-RENT > DATABASES
Nadeem1
VERSION
8.0.3
REGION
AWS Mumbai (ap-south-1)
Overview
Metrics
Collections
Atlas Search
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 2 COLLECTIONS: 9
VISUALIZE YOUR DATA
REFRESH
Search Namespaces
HR-DB-Main
bookingschemas
propertyschemas
users
sample_mflix
HR-DB-Main.users
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 1.06KB TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
Generate queries from natural language in Compass
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
_id: ObjectId('673ca270bad418f31a57fe19')
name : "Nadeem"
email : "nadeemhussain74833@gmail.com"
password : "$2a$10$JKrNY17K3BWx1hvBbexMueSWHk6OIvYlxVhE9pRdwrCDCeynA.r8K"
type : "Renter"
__v : 0
_id: ObjectId('673ca461bad418f31a57fe22')
name : "Nayeem"
email : "nadeemhussain99@gmail.com"
password : "$2a$10$W7O3jyiLvnIoFF3GxAlRruGpEMNE/b8qxa4IaPrCeGianx/kabZG."
type : "Owner"
granted : "granted"
__v : 0
System Status: All Good
©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales


Atlas
NADEEM's O...
Access Manager
Billing
All Clusters
Get Help
Abubakker
NM-House-Rent
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Goto
NADEEM'S ORG - 2024-09-14 > NM-HOUSE-RENT > DATABASES
Nadeem1
VERSION
8.0.3
REGION
AWS Mumbai (ap-south-1)
Overview
Metrics
Collections
Atlas Search
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 2 COLLECTIONS: 9
VISUALIZE YOUR DATA
REFRESH
Search Namespaces
HR-DB-Main
bookingschemas
propertyschemas
users
sample_mflix
HR-DB-Main.bookingschemas
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 368B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search Indexes
Generate queries from natural language in Compass
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
_id: ObjectId('673cae075b51d24b05f583df')
ownerID : ObjectId('673ca461bad418f31a57fe22')
userID : ObjectId('673ca270bad418f31a57fe19')
userName : "nadeem bin nayeem"
phone : 9876543210
bookingStatus : "booked"
propertyId : "673cad065b51d24b05f583c9"
__v : 0
_id: ObjectId('673e23b225874769 6ba9c87f')
ownerID : ObjectId('673e1be8a0982a3b24a65430')
userID : ObjectId('673e237725874769 6ba9c878')
userName : "sample"
phone : 77756863
bookingStatus : "pending"
propertyId : "673e234b258747696ba9c875"
System Status: All Good
©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



12. Known Issues

- ✓ **Slow Property Search:** Filtering and search results may take longer with a large number of properties.
- ✓ **Image Upload Issues**: Some users experience difficulties when uploading high-resolution images for properties.
- ✓ **Mobile View Layout**: Certain UI elements are not fully responsive, especially on smaller mobile screens.

13. Future Enhancements

- ✓ **Real-Time Chat:** Integrate a messaging feature for direct communication between renters and property owners.
- ✓ **Payment Integration:** Add a payment gateway for handling transactions directly within the app.
- ✓ **Mobile App:** Develop a mobile application to improve accessibility and user experience.