HOUSE LISTING PROJECT

1. How would you say your project exceeds expectations?

Our project is a house listing website that incorporates CRUD (Create, Read, Update, Delete) operations for managing property listings. It features a user-friendly interface for adding, updating, and deleting property details. The website also allows users to search for houses based on their names and locations. To enhance security and user experience, We have implemented session management for user authentication, enabling login and logout functionality. All data is stored using MongoDB, ensuring efficient and organized data management.

2. What is the structure of your project?

Our project follows a well-organized structure, utilizing the nodejs framework to manage routing and handle different endpoints. The application effectively utilizes the Model-View-Controller (MVC) architecture to ensure separation of concerns and maintainable code.

Here's an overview of the project's structure:

a. Routing and Controllers (C): server.js is employed to define and manage various routes, such as sign-up, sign-in, dashboard, adding and updating houses, searching, and more. Each route is associated with a specific controller function, which handles the logic and interacts with the data models.

b. Data Models (M): MongoDB is utilized to create and manage data models. These models define the structure of the stored data, including information about houses, users, and sessions. The Mongoose library facilitates the interaction between the application and the MongoDB database.

c. Views (V) - EJS Templates: The front-end views are rendered using Embedded JavaScript (EJS) templates. These templates dynamically generate HTML content and are populated with data from the controllers. EJS templates ensure a consistent and dynamic user interface.

d. Static Files: Static assets such as CSS stylesheets, images, and client-side JavaScript files are stored in designated folders and served to the client using Express middleware. This ensures a visually appealing and interactive user experience.

e. Session Management: The project incorporates session management using the `express session` middleware. This allows users to securely log in and out, and their session data is stored in the MongoDB database, ensuring persistence across requests.

f. Error Handling: Custom error handling is implemented to handle scenarios such as invalid URLs, server errors, and user-specific errors. This enhances user experience by providing appropriate error messages and status codes.

g. Middleware: Middleware functions are used to perform tasks such as parsing incoming requests, handling file uploads, and authenticating users. These functions enhance the functionality and security of the application.

h. API Endpoints: The project includes API endpoints that provide JSON responses for specific functionalities, such as retrieving house data and owner details. These endpoints enable external applications to interact with the project's data.

By following this structured approach, the project ensures scalability, maintainability, and a seamless user experience. The use of Express.js, EJS templates, MongoDB, and effective separation of concerns contributes to a well-organized and feature-rich house listing website.

3. How are you implementing CRUD operations in your app?

In house listing website, we have implemented CRUD (Create, Read, Update, Delete) operations to allow users to effectively manage house listings. These operations are accessible only to signed-in users, and users can only edit the houses they have added. Here's how each CRUD operation is implemented:

a. Create (Add) Operation:

After a user sign in, they can access the "Add House" functionality. When a user submits the house details through a form, the application processes the data and creates a new house listing. The new house is then saved to the MongoDB database, associated with the user who added it. This ensures that users can contribute new house listings to the platform.

b. Read Operation:

Users can view a list of all available houses on the platform. The "Houses" page displays the house listings, including details such as title, description, price, and location. Users can navigate through the house listings and explore the houses added by various users.

c. Update Operation:

Users are allowed to update the details of houses they have added. When a user accesses the "Update House" functionality and selects a house they own, they can modify the title, description, price, and location. The updated information is then processed and stored in the database, ensuring that users can make changes to their own listings.

d. Delete Operation:

Users can delete houses that they have added. By accessing the "Delete House" functionality and selecting a house they own, users can initiate the deletion process. The selected house is removed from the database, ensuring that users can manage and remove their own listings.

These CRUD operations are implemented with proper validation and authentication checks. Only signed-in users can perform these operations, and they are limited to houses they have added. The use of user-specific sessions and database associations ensures that each user can only interact with and modify their own data, enhancing data security and user experience.

4. How are you managing user logins, and how does your application determine if a user is logged in? (For instance, are you using cookies, sessions, state management, etc.?)

By using sessions, the application maintains user authentication throughout their interaction with the website. The use of sessions ensures that users can access their own data, perform CRUD operations, and enjoy a personalized experience while adhering to security and privacy standards.

5. How are you utilizing the API? (Remember, the API counts for 20% and should be testable by any REST client like Insomnia or Postman.)

In my house listing website, we have implemented a RESTful API with three endpoints, allowing users to interact with the data in the system.

Here's how I'm utilizing the API:

a. Retrieve All Houses:

Endpoint: `GET /api/houses`

This endpoint allows users to retrieve a list of all houses available in the system. It returns a JSON response containing details about each house, including its title, description, price, location, and owner information.

b. Search Houses:

Endpoint: `GET /api/search`

Users can search for houses based on specific criteria, such as the house title or location. This endpoint takes query parameters (`title` and `location`) and returns a JSON response containing houses that match the search criteria.

c. Retrieve House Owner Details:

Endpoint: `GET /api/owner-details/:houseTitle`

Users can retrieve the owner details of a specific house by providing its title as a URL parameter. The endpoint returns a JSON response containing the owner's information and additional details about the house.

We have thoroughly tested these API endpoints using REST clients like Insomnia. This allows me to ensure that the endpoints are functioning as expected, returning the correct data, and handling different scenarios gracefully. Users can interact with the API to retrieve house information, search for specific houses, and access owner details, enhancing the usability and functionality of the application.

6. How do you handle non-existent routes, like when a route `/displayData` isn't registered in your `server.js`?

We have a 404 page that handles any unregistered routes.

7. In what ways do you believe your code is clean, easy to read, and maintainable?

a. Modularization: We divided our code into logical sections and separate files for different functionalities. For instance, you have separate files for models (e.g., `SignUpInModel`, `House`), routes, and utility functions.

b. Descriptive Variable Names: We use meaningful variable names that help readers understand the purpose and content of the variables.

c. Structured Routing: Routes are organized with clear endpoint names and HTTP methods. This makes it easy to understand the different functionalities your application provides.

d. Consistent Formatting: Code follows consistent formatting and indentation throughout, which enhances readability.

e. Comments: Added comments at key sections of your code, explaining what different parts of the code do. This makes it easier for other developers (and your future self) to understand your codebase.

f. Error Handling: We have implemented error handling at various stages of your application, providing helpful error messages and statuses in case of failures.

g. Middleware: The use of middleware, such as `authenticateUser`, helps keepourroute handlers focused on their primary tasks and separates concerns.

h. Expressive Routes: Your route names, such as `/api/houses` and `/api/search`, clearly indicate their purpose.

By following these practices, your codebase is well-organized, easy to understand, and maintainable. It sets a strong foundation for future enhancements, bug fixes, and collaboration with other developers.

8. What considerations should the professor keep in mind when grading your project? Please outline any assumptions or specific details that are important for understanding your work.

When grading my project, here are some considerations, assumptions, and specific details that are important for understanding my work:

a. Session-based Authentication: I've implemented session-based authentication for user logins. Users are required to sign up and log in before accessing certain functionalities. User sessions are managed using the `express session` middleware.

b. User Password Security: For the sake of this project, I've stored user passwords in plain text in the database. In a real-world scenario, password hashing and salting would be implemented for enhanced security.

c. Database Initialization: The application loads sample house data from a JSON file on startup if the database is empty. This simplifies the grading process and demonstrates the application's functionality.

d. File Uploads: Users can upload house images using the `multer` middleware. The images are stored in the `HouseImages` directory, and the file paths are saved in the database.

e. Middleware Usage: Middleware functions are used for authentication (`authenticateUser`) and handling static files. Middleware chaining is used to ensure proper authentication before accessing certain routes.

f. Assumed Dependencies: It's assumed that the necessary dependencies are installed using npm and are accessible in the project directory. These dependencies include `express`, `body-parser`, `express-session`, `ejs`, `path`, `multer`, and `mongoose`.

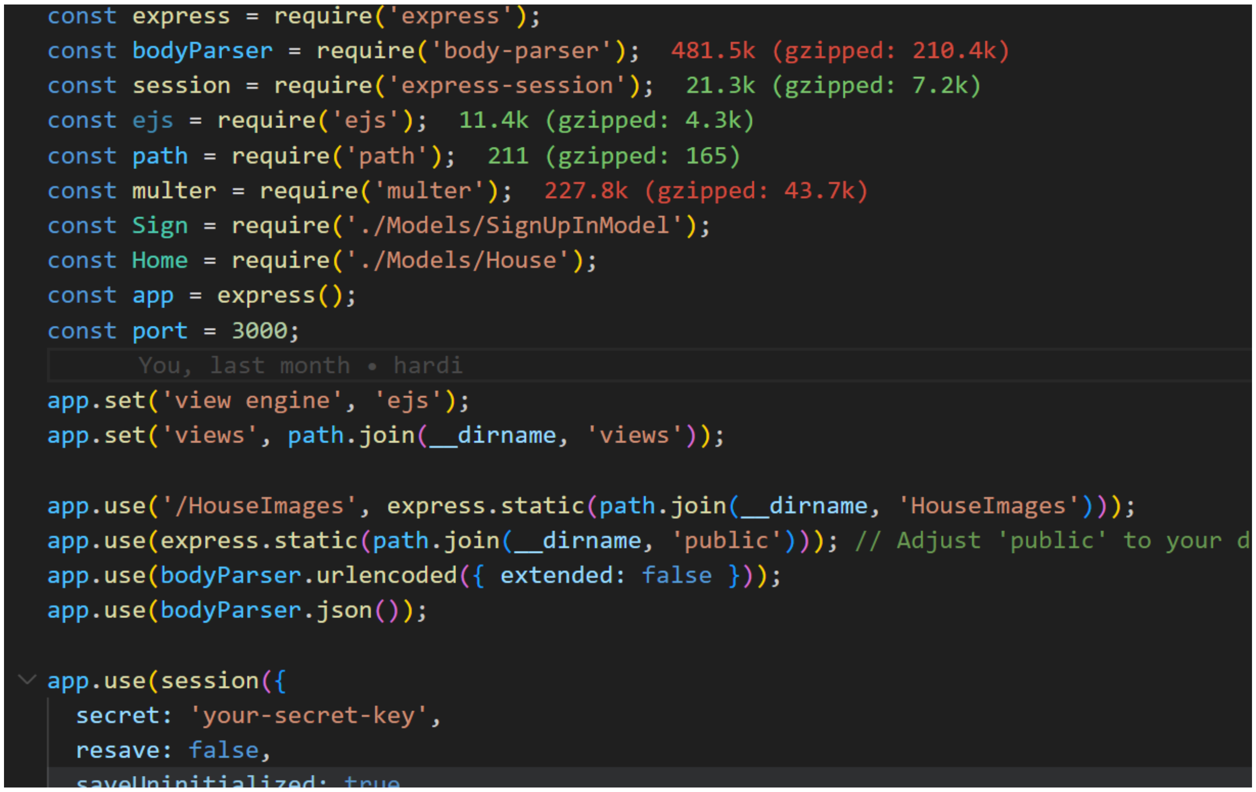
g. Assumed Environment: The project is designed to run in a local development environment. It uses a hardcoded port (`3000`) for the server to listen on.

h. Testing API Endpoints: The provided API endpoints can be tested using tools like Insomnia or Postman, making it easy to verify the functionality and data retrieval of the routes.

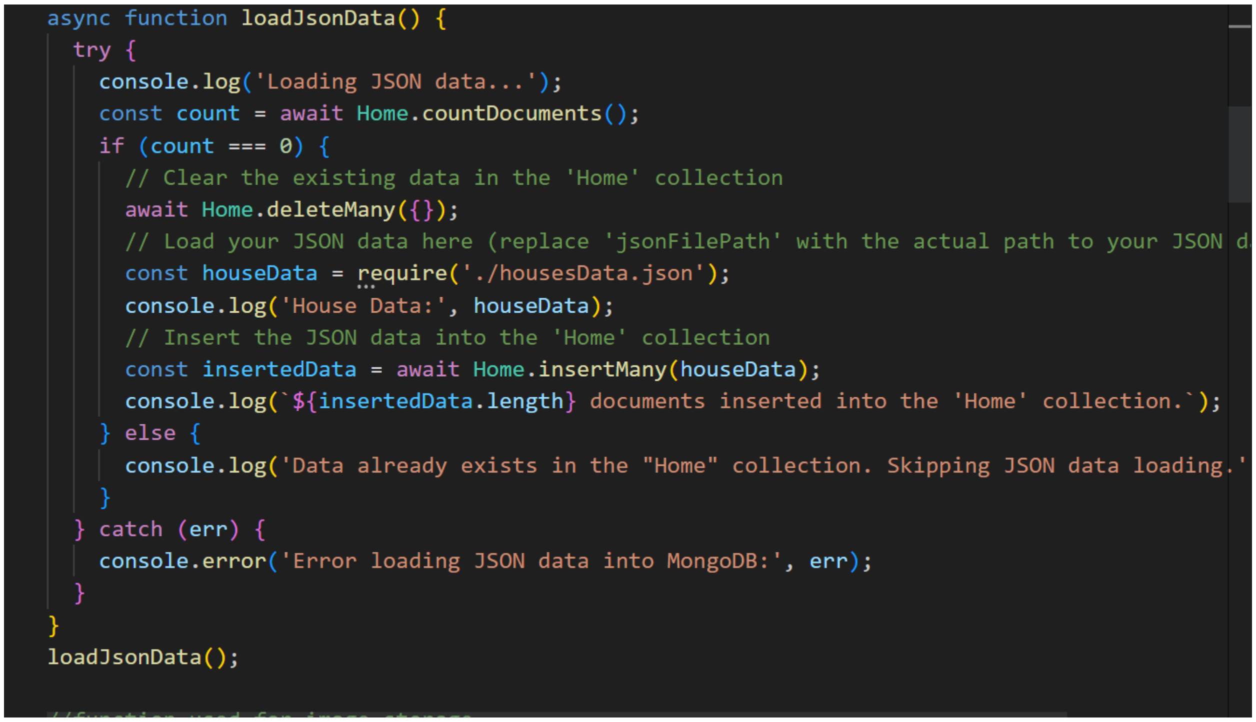
i. Database Connection: The application connects to a MongoDB database using Mongoose. The database connection URL and credentials are assumed to be provided in the `util/config.js` file.

Screenshots:

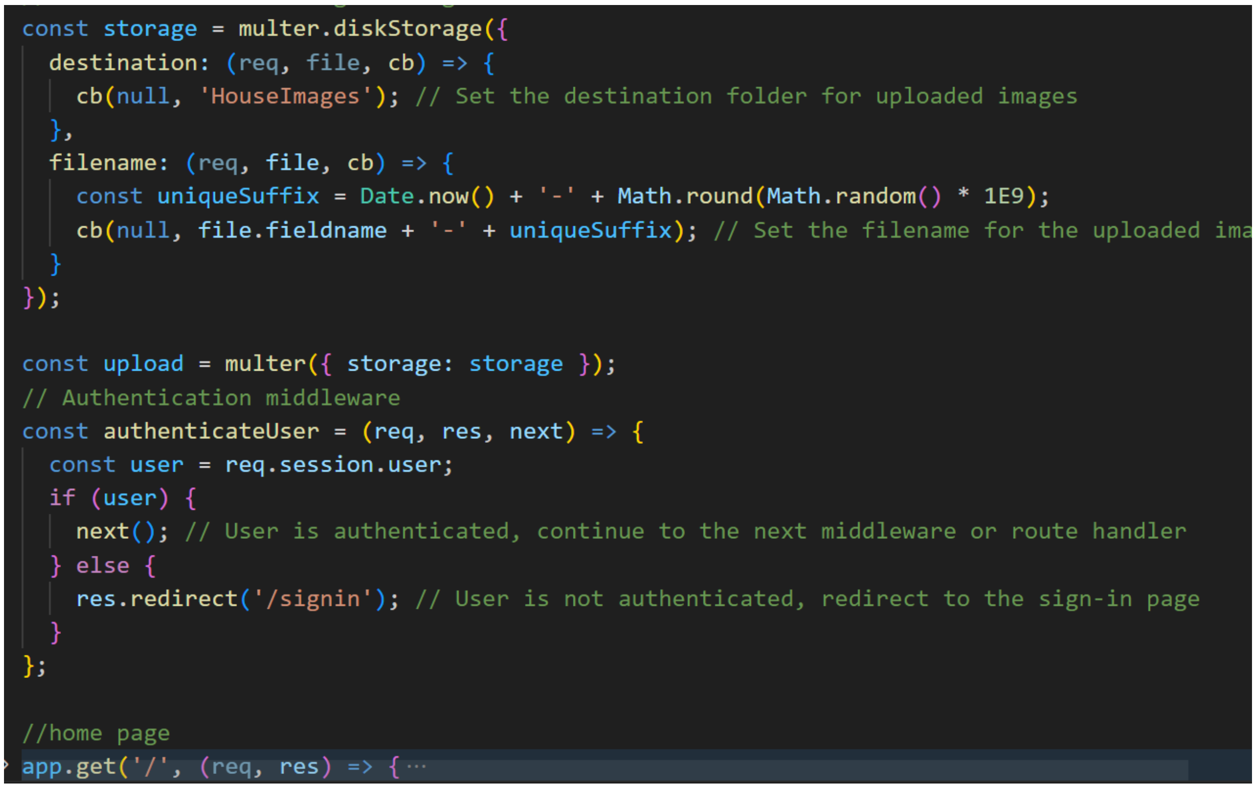
- Screenshot of `server.js`.



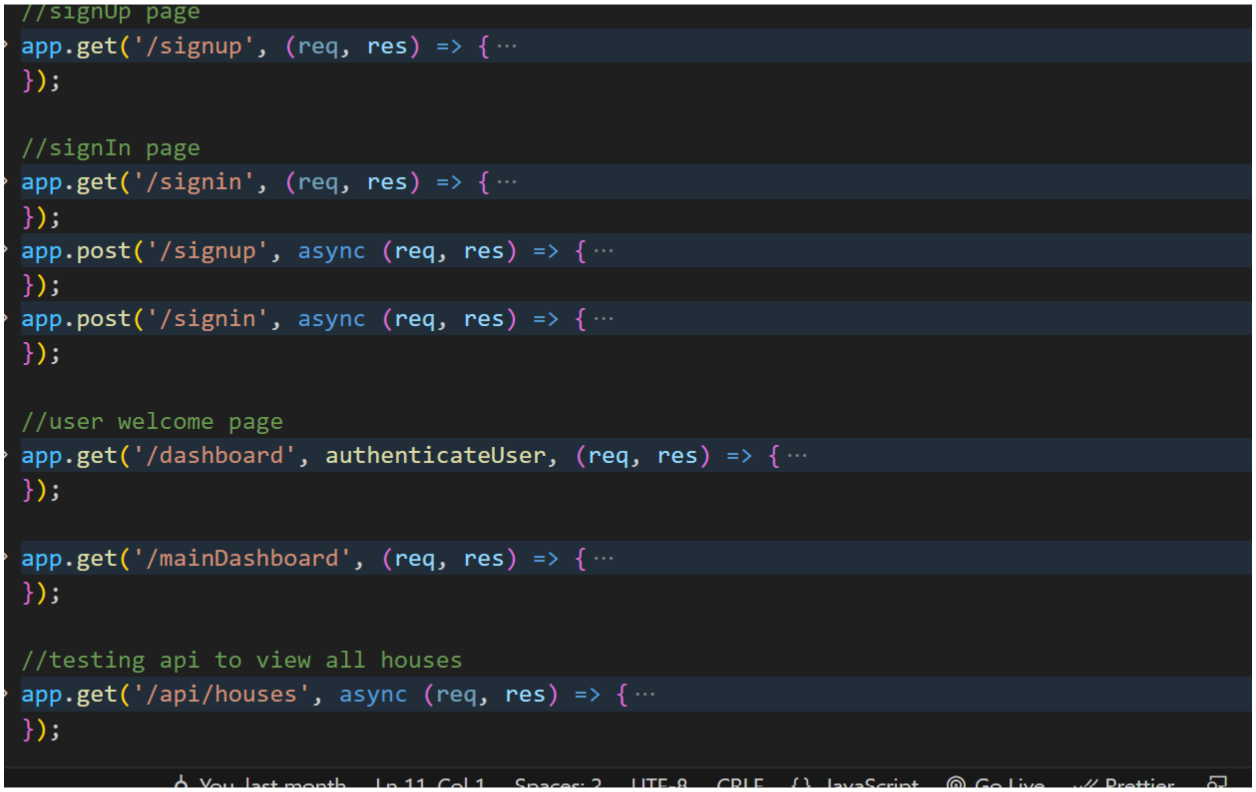
```
const express = require('express');
const bodyParser = require('body-parser');  481.5k (gzipped: 210.4k)
const session = require('express-session');  21.3k (gzipped: 7.2k)
const ejs = require('ejs');  11.4k (gzipped: 4.3k)
const path = require('path');  211 (gzipped: 165)
const multer = require('multer');  227.8k (gzipped: 43.7k)
const Sign = require('./Models/SignUpInModel');
const Home = require('./Models/House');
const app = express();
const port = 3000;
      You, last month • hardi
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.use('/HouseImages', express.static(path.join(__dirname, 'HouseImages')));
app.use(express.static(path.join(__dirname, 'public'))); // Adjust 'public' to your d
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.use(session({
  secret: 'your-secret-key',
  resave: false,
  saveUninitialized: true
```



```
async function loadJsonData() {
  try {
    console.log('Loading JSON data...');
    const count = await Home.countDocuments();
    if (count === 0) {
      // Clear the existing data in the 'Home' collection
      await Home.deleteMany({});
      // Load your JSON data here (replace 'jsonFilePath' with the actual path to your JSON d
      const houseData = require('./housesData.json');
      console.log('House Data:', houseData);
      // Insert the JSON data into the 'Home' collection
      const insertedData = await Home.insertMany(houseData);
      console.log(`${insertedData.length} documents inserted into the 'Home' collection.`);
    } else {
      console.log('Data already exists in the "Home" collection. Skipping JSON data loading.'
    }
  } catch (err) {
    console.error('Error loading JSON data into MongoDB:', err);
  }
}
loadJsonData();
```

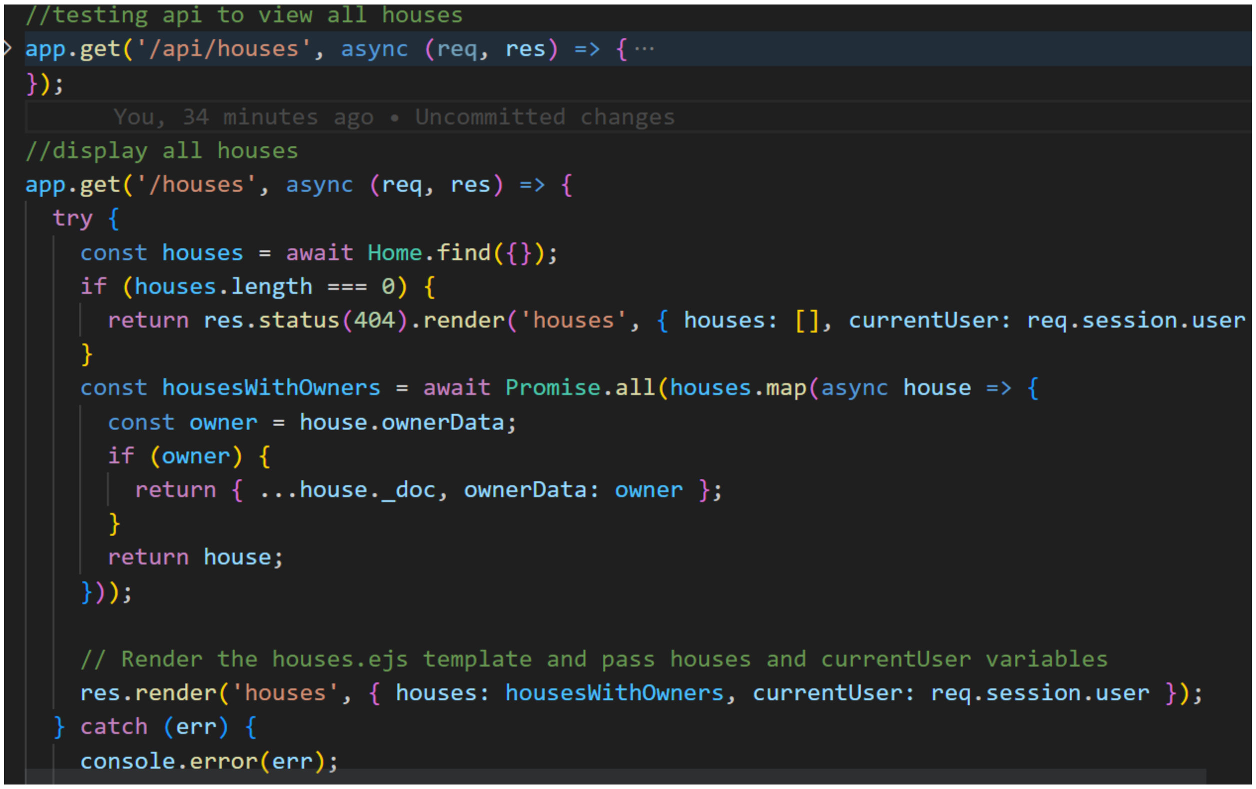


```
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'HouseImages'); // Set the destination folder for uploaded images
  },
  filename: (req, file, cb) => {
    const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9);
    cb(null, file.fieldname + '-' + uniqueSuffix); // Set the filename for the uploaded ima
  }
});

const upload = multer({ storage: storage });
// Authentication middleware
const authenticateUser = (req, res, next) => {
  const user = req.session.user;
  if (user) {
    next(); // User is authenticated, continue to the next middleware or route handler
  } else {
    res.redirect('/signin'); // User is not authenticated, redirect to the sign-in page
  }
};

//home page
app.get('/', (req, res) => {⋯
```

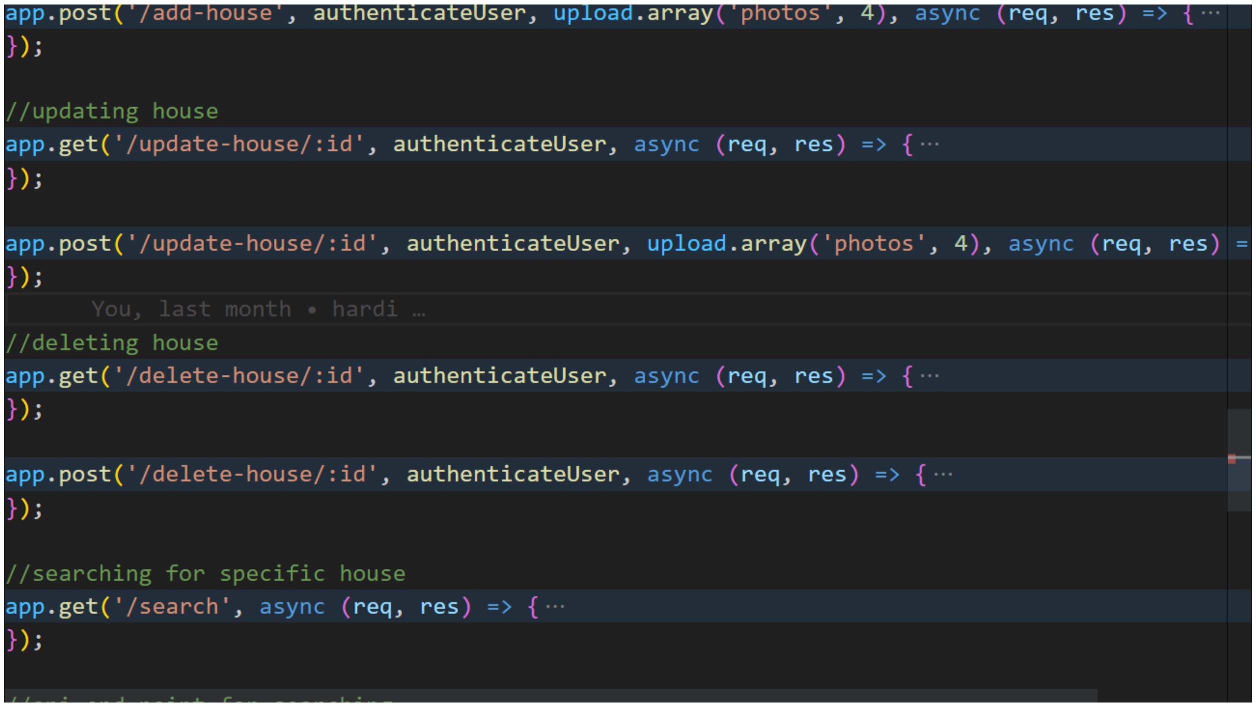


```
//signUp page
app.get('/signup', (req, res) => {⋯
});

//signIn page
app.get('/signin', (req, res) => {⋯
});
app.post('/signup', async (req, res) => {⋯
});
app.post('/signin', async (req, res) => {⋯
});

//user welcome page
app.get('/dashboard', authenticateUser, (req, res) => {⋯
});

app.get('/mainDashboard', (req, res) => {⋯
});

//testing api to view all houses
app.get('/api/houses', async (req, res) => {⋯
});
```

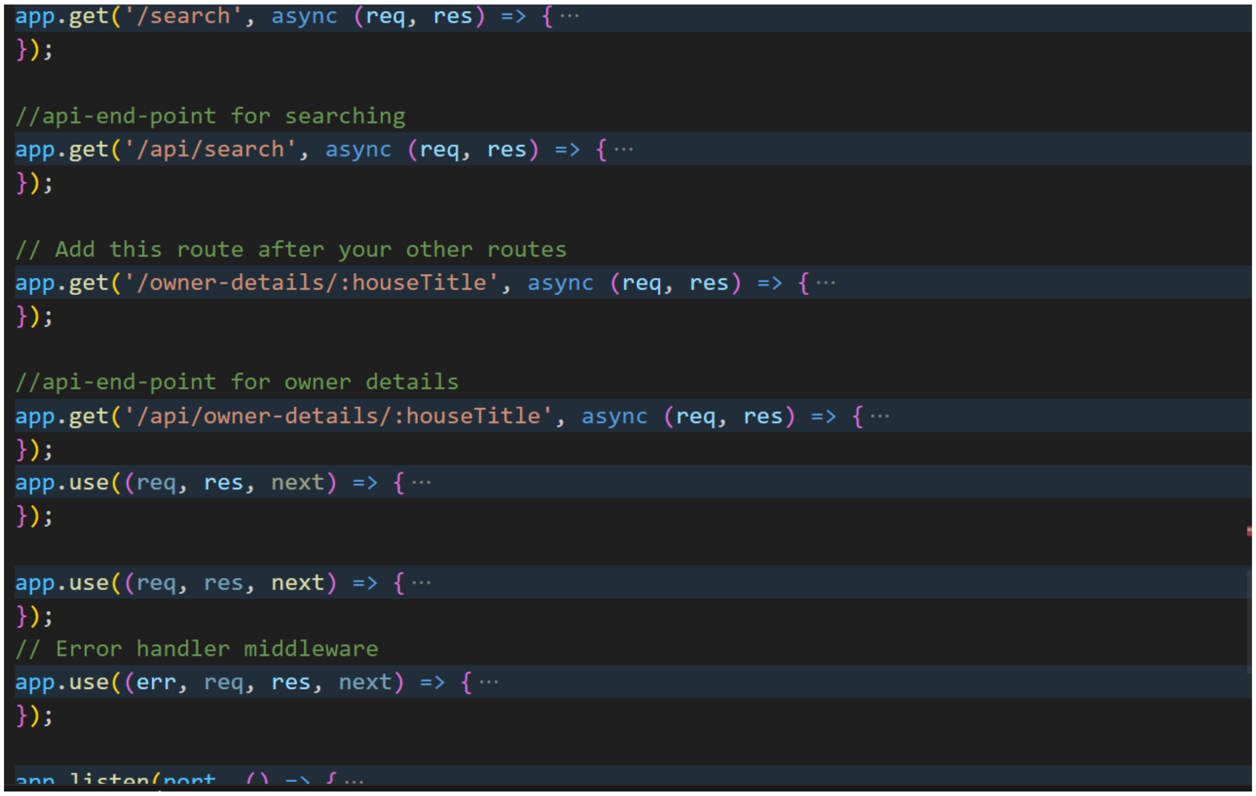
```
You, last month   Ln 11, Col 1   Spaces: 2   UTF-8   CRLF   {} JavaScript   Go Live   Prettier
```



```
//testing api to view all houses
app.get('/api/houses', async (req, res) => {⋯
});
    You, 34 minutes ago • Uncommitted changes
//display all houses
app.get('/houses', async (req, res) => {
  try {
    const houses = await Home.find({});
    if (houses.length === 0) {
      return res.status(404).render('houses', { houses: [], currentUser: req.session.user
    }
    const housesWithOwners = await Promise.all(houses.map(async house => {
      const owner = house.ownerData;
      if (owner) {
        return { ...house._doc, ownerData: owner };
      }
      return house;
    }));

    // Render the houses.ejs template and pass houses and currentUser variables
    res.render('houses', { houses: housesWithOwners, currentUser: req.session.user });
  } catch (err) {
    console.error(err);
```

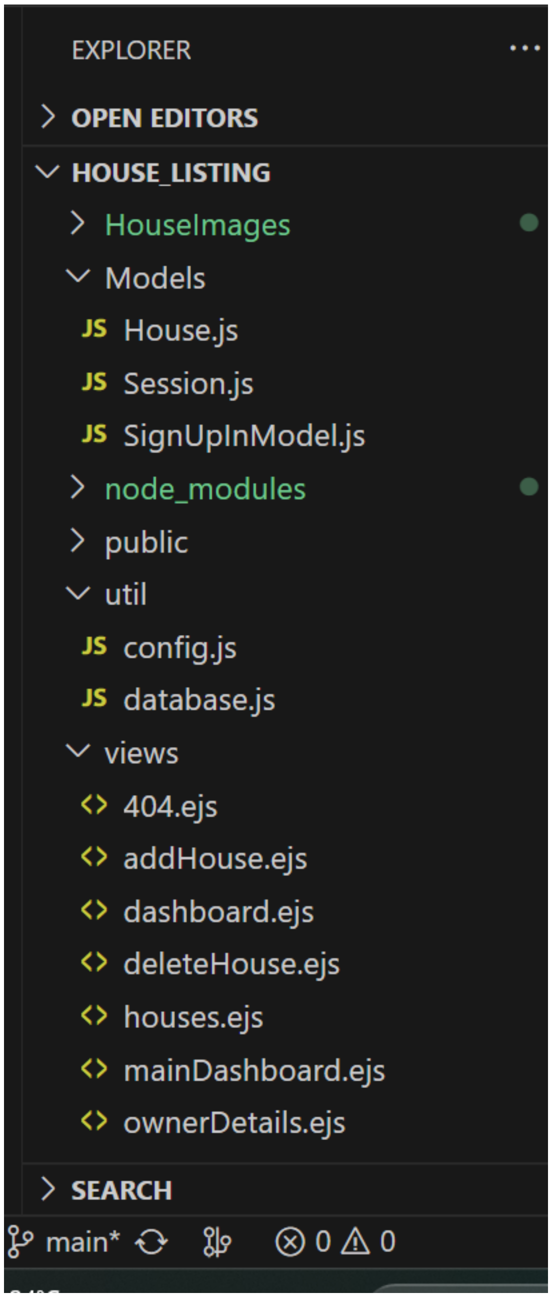


```
app.post('/add-house', authenticateUser, upload.array('photos', 4), async (req, res) => {⋯
});

//updating house
app.get('/update-house/:id', authenticateUser, async (req, res) => {⋯
});

app.post('/update-house/:id', authenticateUser, upload.array('photos', 4), async (req, res) =
});
      You, last month • hardi …
//deleting house
app.get('/delete-house/:id', authenticateUser, async (req, res) => {⋯
});

app.post('/delete-house/:id', authenticateUser, async (req, res) => {⋯
});

//searching for specific house
app.get('/search', async (req, res) => {⋯
});
```

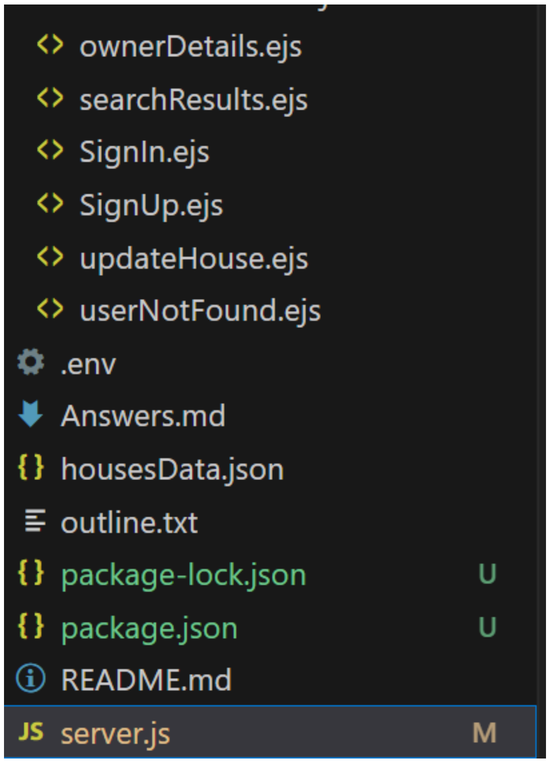


```
app.get('/search', async (req, res) => {⋯
});

//api-end-point for searching
app.get('/api/search', async (req, res) => {⋯
});

// Add this route after your other routes
app.get('/owner-details/:houseTitle', async (req, res) => {⋯
});

//api-end-point for owner details
app.get('/api/owner-details/:houseTitle', async (req, res) => {⋯
});
app.use((req, res, next) => {⋯
});

app.use((req, res, next) => {⋯
});
// Error handler middleware
app.use((err, req, res, next) => {⋯
});
```

- Screenshot of the left side panel showing all your project directories.

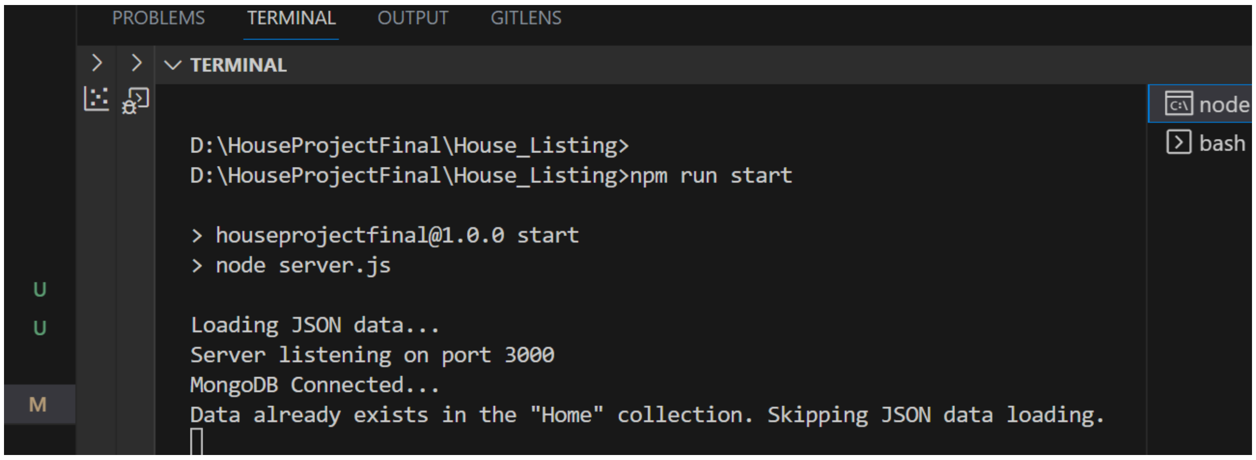


```
EXPLORER
> OPEN EDITORS
∨ HOUSE_LISTING
  > HouseImages
  ∨ Models
    House.js
    Session.js
    SignUpInModel.js
  > node_modules
  > public
  ∨ util
    config.js
    database.js
  ∨ views
    404.ejs
    addHouse.ejs
    dashboard.ejs
    deleteHouse.ejs
    houses.ejs
    mainDashboard.ejs
    ownerDetails.ejs
> SEARCH
main*    ⊗ 0 ⚠ 0
```

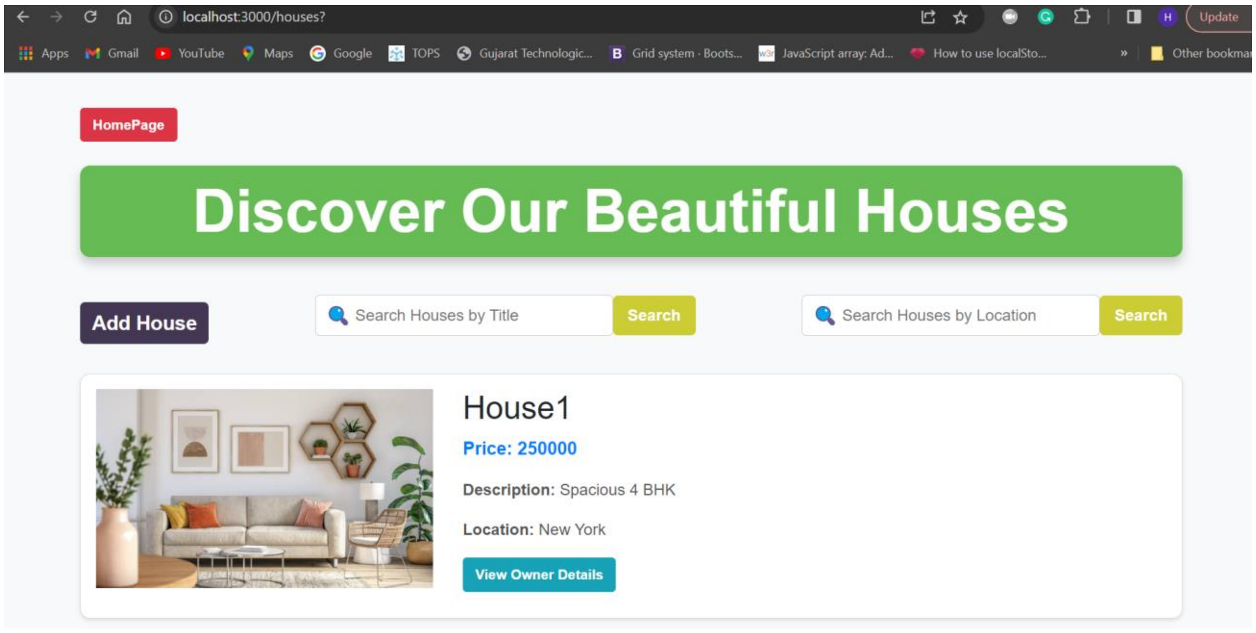


```
ownerDetails.ejs
searchResults.ejs
SignIn.ejs
SignUp.ejs
updateHouse.ejs
userNotFound.ejs
.env
Answers.md
housesData.json
outline.txt
package-lock.json        U
package.json             U
README.md
server.js                M
```

- Screenshots of the frontend of at least two app screens.



```
D:\HouseProjectFinal\House_Listing>
D:\HouseProjectFinal\House_Listing>npm run start

> houseprojectfinal@1.0.0 start
> node server.js

Loading JSON data...
Server listening on port 3000
MongoDB Connected...
Data already exists in the "Home" collection. Skipping JSON data loading.
```


localhost:3000/houses?

HomePage

Discover Our Beautiful Houses

Add House

Search Houses by Title Search

Search Houses by Location Search

House1

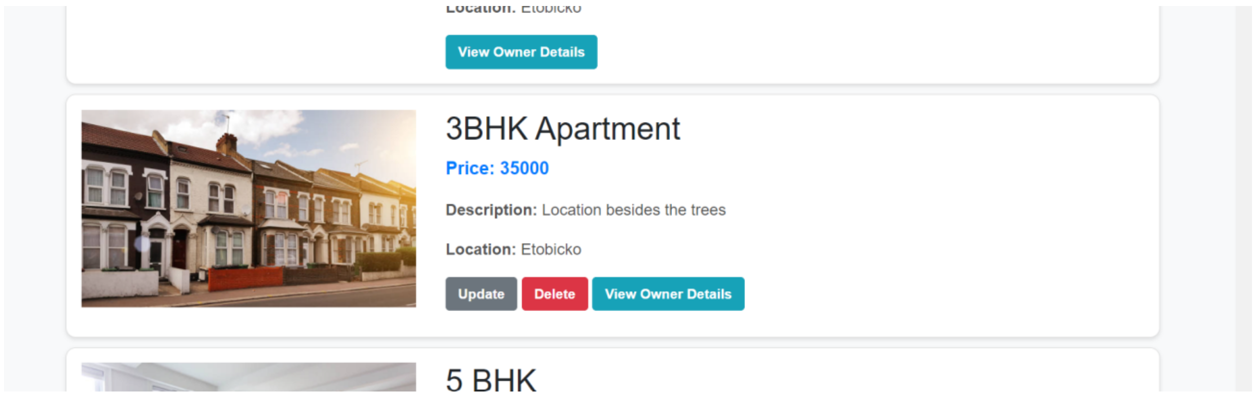
Price: 250000

Description: Spacious 4 BHK

Location: New York

View Owner Details

Signed In user can update and delete houses


Location: Etobicko

View Owner Details

3BHK Apartment

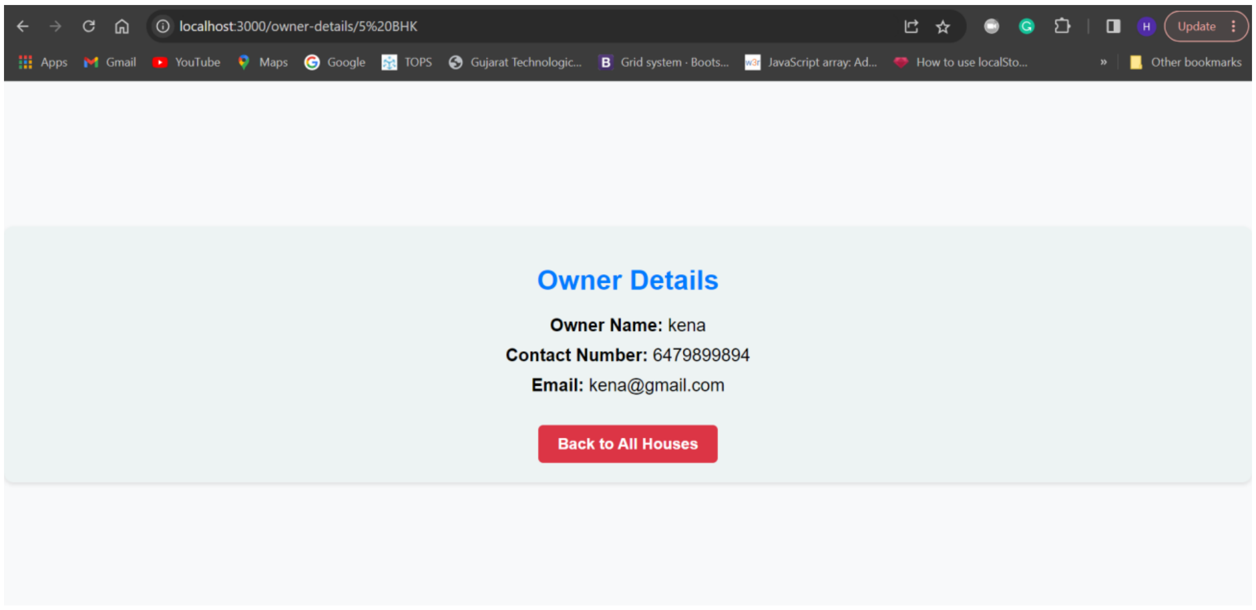
Price: 35000

Description: Location besides the trees

Location: Etobicko

Update Delete View Owner Details

5 BHK



localhost:3000/owner-details/5%20BHK

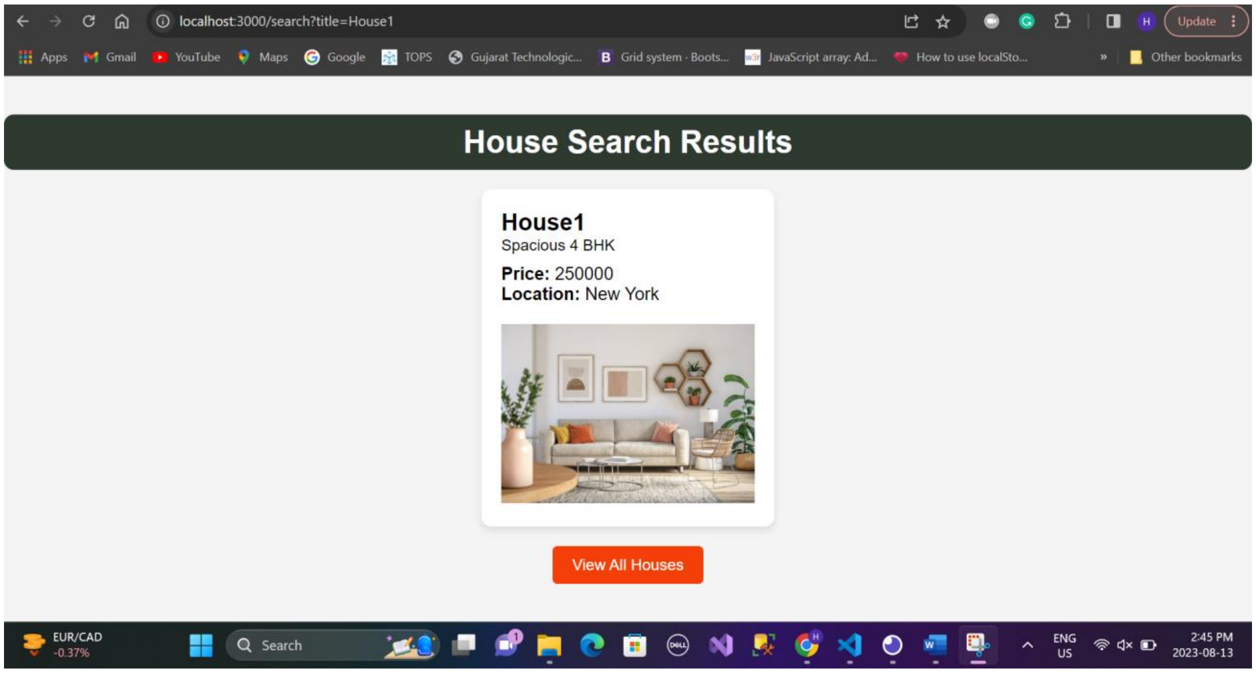
Owner Details

Owner Name: kena

Contact Number: 6479899894

Email: kena@gmail.com

House Search by Title:



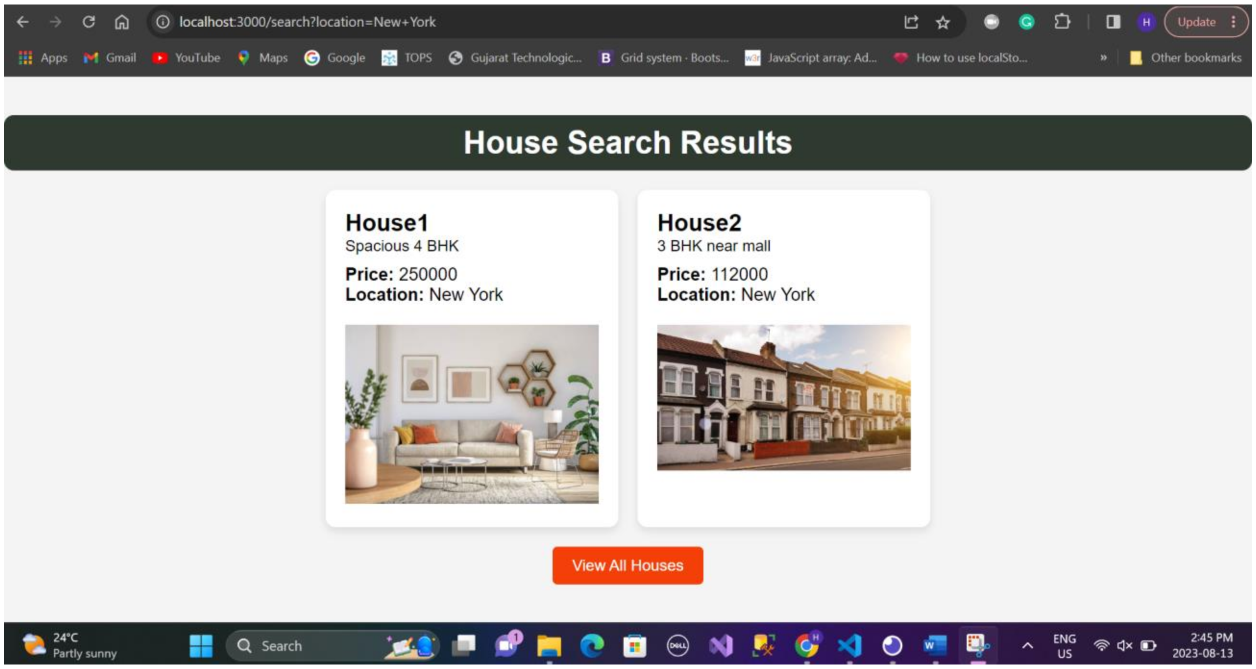
localhost:3000/search?title=House1

House Search Results

House1
Spacious 4 BHK

Price: 250000
Location: New York

House Search by Location:



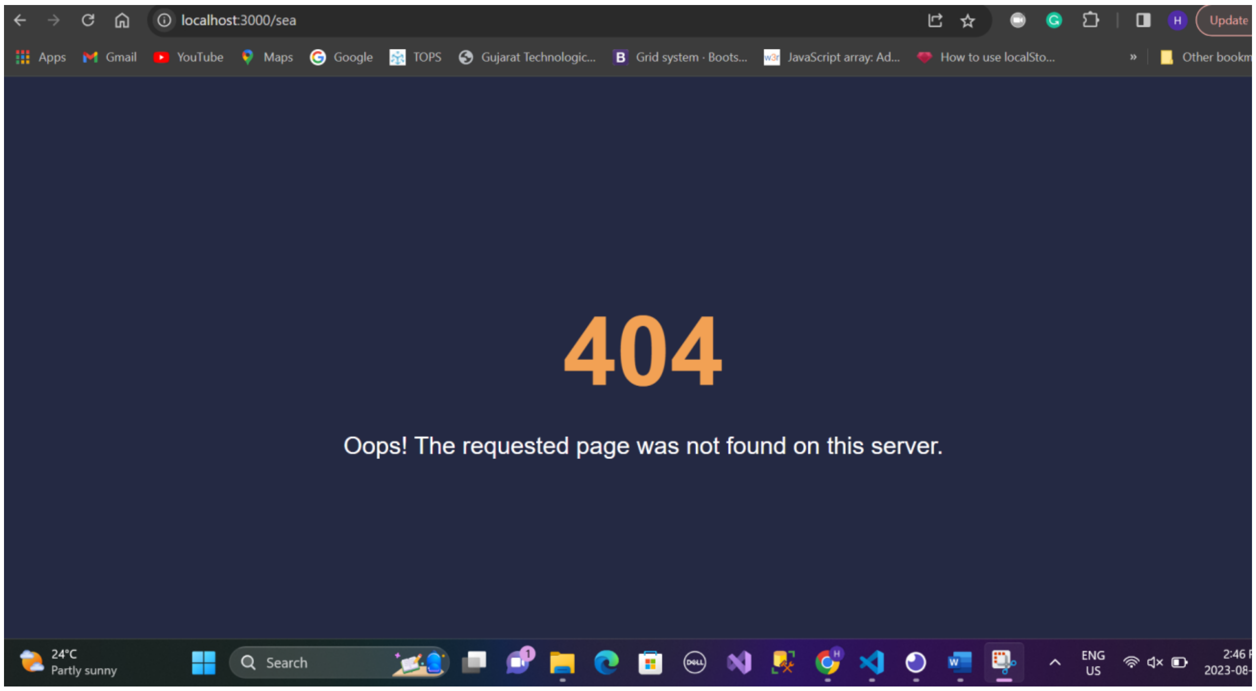
localhost:3000/search?location=New+York

House Search Results

House1
Spacious 4 BHK
Price: 250000
Location: New York

House2
3 BHK near mall
Price: 112000
Location: New York

404 Error:

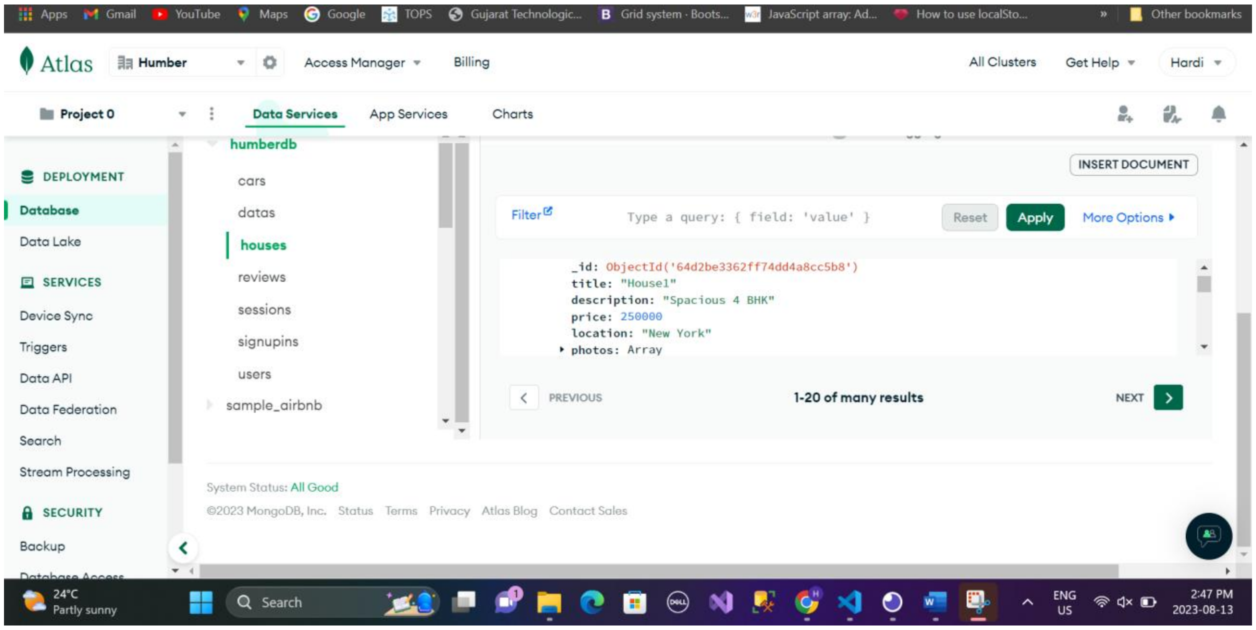


localhost:3000/sea

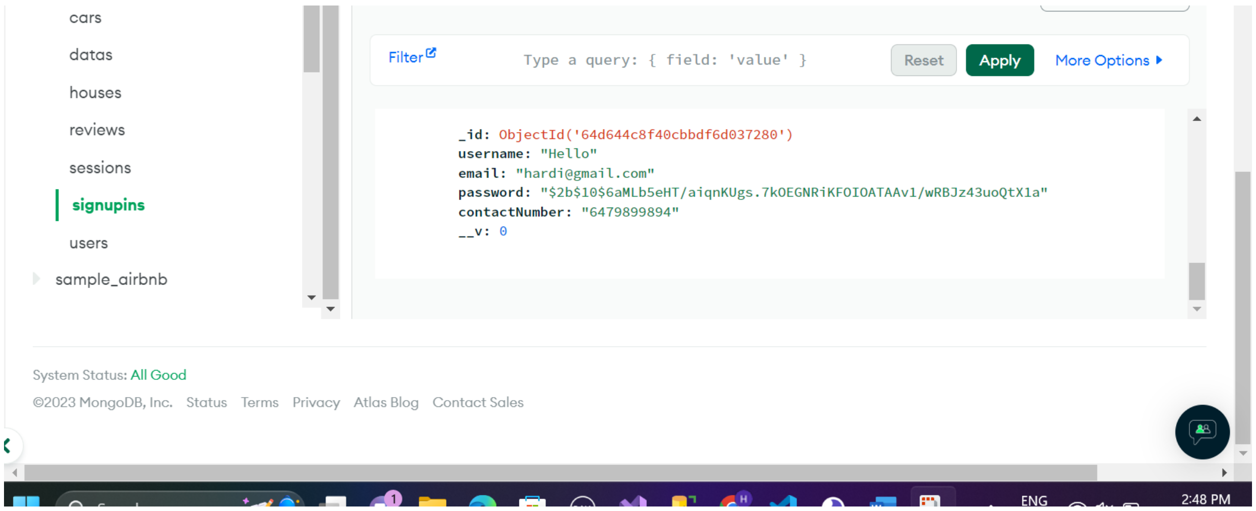
404

Oops! The requested page was not found on this server.

- Mongo Atlas displaying the data used.

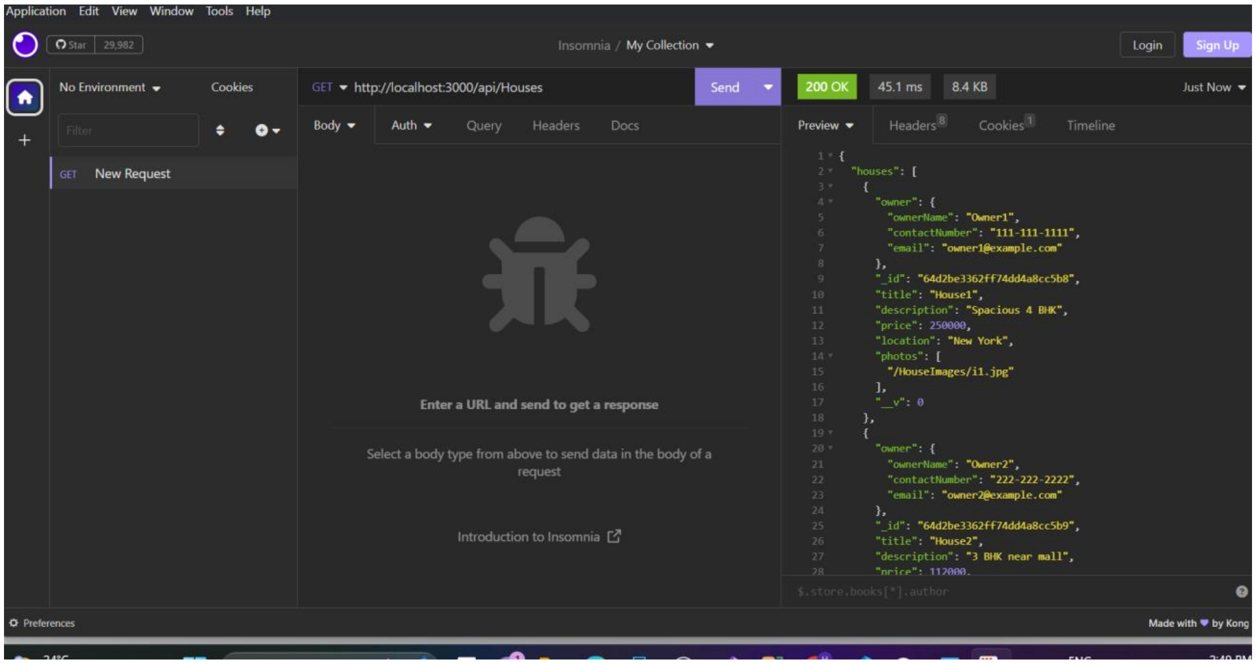


```
_id: ObjectId('64d2be3362ff74dd4a8cc5b8')
title: "House1"
description: "Spacious 4 BHK"
price: 250000
location: "New York"
photos: Array
```



```
_id: ObjectId('64d644c8f40cbbdf6d037280')
username: "Hello"
email: "hardi@gmail.com"
password: "$2b$10$6aMLb5eHT/aiqnKUgs.7kOEGNRiKFOIOATAAv1/wRBJz43uoQtX1a"
contactNumber: "6479899894"
__v: 0
```

Screenshots of end point API: Testing:

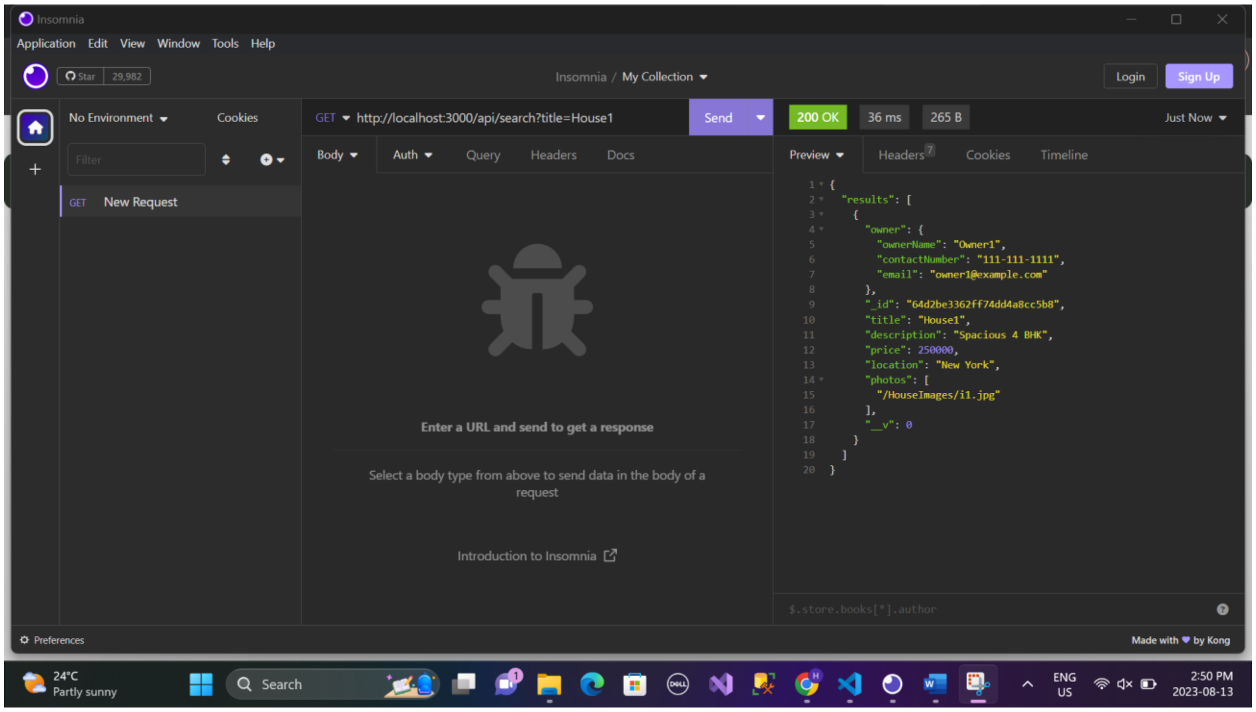


```
GET http://localhost:3000/api/Houses
```

```
200 OK   45.1 ms   8.4 KB
```

```
{
  "houses": [
    {
      "owner": {
        "ownerName": "Owner1",
        "contactNumber": "111-111-1111",
        "email": "owner1@example.com"
      },
      "_id": "64d2be3362ff74dd4a8cc5b8",
      "title": "House1",
      "description": "Spacious 4 BHK",
      "price": 250000,
      "location": "New York",
      "photos": [
        "/HouseImages/i1.jpg"
      ],
      "__v": 0
    },
    {
      "owner": {
        "ownerName": "Owner2",
        "contactNumber": "222-222-2222",
        "email": "owner2@example.com"
      },
      "_id": "64d2be3362ff74dd4a8cc5b9",
      "title": "House2",
      "description": "3 BHK near mall",
      "price": 112000,
```

Application Edit View Window Tools Help
Insomnia / My Collection
Login
Sign Up
No Environment
Cookies
Send
Body Auth Query Headers Docs
Preview Headers 8 Cookies 1 Timeline
Just Now
New Request
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request
Introduction to Insomnia
Preferences
Made with ♥ by Kong

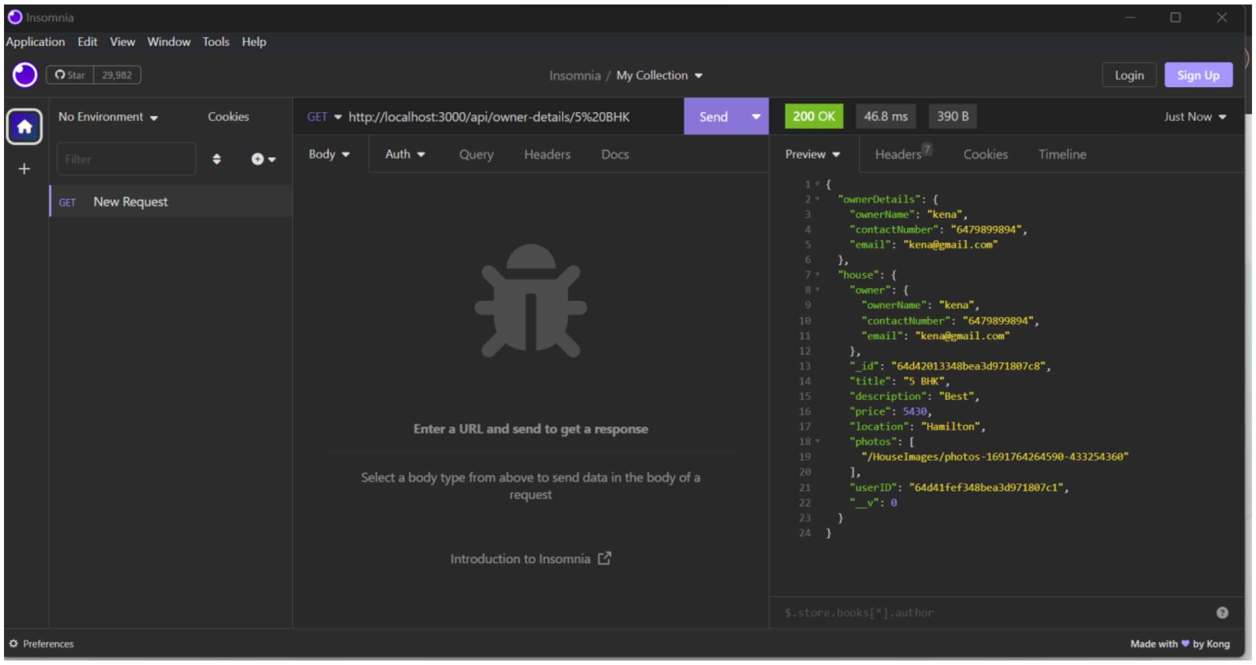


```
GET http://localhost:3000/api/search?title=House1
```

```
200 OK   36 ms   265 B
```

```
{
  "results": [
    {
      "owner": {
        "ownerName": "Owner1",
        "contactNumber": "111-111-1111",
        "email": "owner1@example.com"
      },
      "_id": "64d2be3362ff74dd4a8cc5b8",
      "title": "House1",
      "description": "Spacious 4 BHK",
      "price": 250000,
      "location": "New York",
      "photos": [
        "/HouseImages/i1.jpg"
      ],
      "__v": 0
    }
  ]
}
```

Insomnia
Application Edit View Window Tools Help
Insomnia / My Collection
Login
Sign Up
No Environment
Cookies
Send
Body Auth Query Headers Docs
Preview Headers 7 Cookies Timeline
Just Now
New Request
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request
Introduction to Insomnia
Preferences
Made with ♥ by Kong
24°C Partly sunny
Search
ENG US
2:50 PM 2023-08-13



Insomnia
Application Edit View Window Tools Help
Star 29,982
Insomnia / My Collection
Login
Sign Up
No Environment
Cookies
Filter
GET New Request
Body Auth Query Headers Docs
Enter a URL and send to get a response
Select a body type from above to send data in the body of a request
Introduction to Insomnia
Preview Headers 7 Cookies Timeline
Preferences
Made with ♥ by Kong

GET http://localhost:3000/api/owner-details/5%20BHK Send

200 OK 46.8 ms 390 B Just Now

```
{
  "ownerDetails": {
    "ownerName": "kena",
    "contactNumber": "6479899894",
    "email": "kena@gmail.com"
  },
  "house": {
    "owner": {
      "ownerName": "kena",
      "contactNumber": "6479899894",
      "email": "kena@gmail.com"
    },
    "_id": "64d42013348bea3d971807c8",
    "title": "5 BHK",
    "description": "Best",
    "price": 5430,
    "location": "Hamilton",
    "photos": [
      "/HouseImages/photos-1691764264590-433254360"
    ],
    "userID": "64d41fef348bea3d971807c1",
    "__v": 0
  }
}
```

$.store.books[*].author