# Prompt begin

You are a database assistant.

# Prompt to provide everything

## Stage 1

Here is my task:

“You are required to be able to write code in HTML, JavaScript, Node.js, and Express.js scripting languages to build a fully-fledged web database application. More importantly, you should be able to use mongodb library within a Node.js program. This is broken down into 2 tasks.”

Task 1: Build a simple web database application

The application you are required to build is based on sample\_airbnb database. Within this database I have the collection “listingsAndReviews”.

Within this collection I have the following fields. NOTE I am listing the key pair with the data type but not sample data. Also for objects and arrays, not all are listed as they are not needed.

{

\_id: String,

listing\_url: String,

name: String,

summary: String,

space: String,

description:String,

neighbourhood\_overview:String,

notes:String,

transit:String,

access:String,

interaction:String,

house\_rules:String,

property\_type:String,

room\_type:String,

bed\_type:String,

minimum\_nights:String,

maximum\_nights:String,

cancellation\_policy:String,

last\_scrapped:Date,

calendar\_last\_scraped:Date,

first\_review:Date,

last:review:Date,

accommodates:Int32,

bedrooms:Int32,

beds:Int32,

number\_of\_reviews:Int32,

bathooms:Decimal128,

amenities:Array,

price:Decimal128,

security\_deposit:Decimal128,

cleaning\_fee:Decimal128,

extra\_people:Decimal128,

guests\_included:Decimal128,

images:Object,

host:Object,

address:Object,

street:string,

market:string,

availability:Object,

review\_scores:Object,

reviews:Array,

review\_scores\_rating: int32,

bookings: Array,

0: Object

**booking\_id** :Int32

**arrival\_date** : Date,

**departure\_date** :Date,

**client** : Object

name: String,

email: String,

daytime\_phone: String,

mobile: String,

postal\_address: String,

home\_address:String,

**deposit\_paid**: int32,

**balance\_due** : int32,

**balance\_due\_date** :Date,

**number\_of\_guests**: int32,

guests: Array

0:Object

name: String,

age: int32,

}

The web application will present AirBnB clients with an interface where they can filter listings based on their priorities and then will allow them to choose one listing from the presented list and add a new booking for their requested dates.

The application will have a **minimum of three pages:**

Page 1: **Homepage**

This page will have two parts:

1. The top section will consist of a simple form with three form input fields: Location, the type of the property and the number of bedrooms.
2. The bottom section will initially list some random property listings.

* The **location** is a mandatory input.
* Type of property and number of bedrooms are dropdown lists. However, these two inputs are optional, i.e. the clients can choose to leave them unselected and submit the form.

After the form is submitted, the bottom part of the webpage will get refreshed with property listings that matches with the filtering criteria the client has submitted.

For example, if they have chosen Barcelona as the location and left other two inputs empty, it will display all properties in the Barcelona market (address.market).

If a client had filled all three fields (say, 3-bedroom apartments in Barcelona) then your application will display a further narrowed-down result set.

Each property listing on this page should comprise of the name of the property, summary, daily price, and review score rating (review\_scores.review\_scores\_rating).

**Each property listing’s name is displayed as an active hyperlink**, allowing the client to choose the property and proceed to the next stage (booking stage) of the application.

This hyperlink should carry the listing\_id as a hyperlink query parameter (or URL parameter, e.g.: https://localhost:3000/bookings.html?listing\_id=10083468) and will allow the bookings page to manage the bookings for the chosen property.

Use css style sheet to format the output. A typical listing will appear on this page as shown below:

*{*

*14 Listings that match your preferences*

*Be Happy in Porto*

*Description of listing. E.g. “This is a nice apartment..”*

*Daily Rate: 30*

*Customer rating 97*

*}*

*Note: that “Be Happy in Porto “is 1 listing and is a hyperlink.*

**Page 2:** Booking

This page will also made up of a form which allows the clients to input booking start date, end date, client name, email address, daytime phone number, mobile number, postal address and home address.

To keep your web form simple, it is NOT a requirement in this assignment to enter the other information such as the deposit paid at the booking, the balance due, the due date for the balance payment, and number of guests, and guest details. An example of the booking page’s structure would be like below enclosed in {}.

{

<h2>Let’s book the property</h2>

<h2>Booking Details</h2>

Check In:<Enter check-in date (dd/mm/yyyy)/>

Check out: :<Enter check-in date (dd/mm/yyyy)/>

<h2>Your Details</h2>

Your Name: <Please enter your name (mandatory field)>

Your Email Address: <Please enter your email address (mandatory field)>

Your Mobile No: <Please enter your mobile number: \*\*\*\* \*\*\* \*\*\* (mandatory field)>

Postal Address: <Please provide your postal address>

Residential Address: <Please provide your residential address.>

}

**Page 3:** Booking confirmation Page

After the booking information is submitted and new booking data is stored on the database, a simple booking confirmation will appear. This page will have a simple hyperlink to return to the homepage.

**System Requirements**

Hosting

* This is just an application development exercise, so, it is not required to host it in a proper web hosting platform.
* You can use Visual Studio Code as your interactive development environment and host it locally (say host it on port 3000 on localhost and accessed locally on your browser with homepage URL: localhost:3000/index.html)

Database

Use cloud-based MongoDB Atlas as your database backend.

Database Schema

The sample database “sample\_airbnb” has one document collection called “listingsAndReviews” which contain basic information on property listings. The information for bookings is stored using an embedded approach.

**Technology Stack**

It is a requirement to use Node.js with Express.js as your development platform.

**Can you make the application and it’s associated files.**

Breakdown:

1. The solution.
2. The steps I need to do to complete the task.
3. List hints at each step what I would need to research and now (E.g. JavaScript, Css)

Additional constraints:

* Make the first step creating the directory folder.
* I am also thinking of using express, body-parser and npm init -y as packages. Make this a step after making the directory. Include the path module to avoid hard coding directory paths. Use the fs module to read files. Also specify any other packages which may be relevant to use.
* Make a step where I create the folders after specifying the packages to install.
* All the html pages must be in same folder.
* After creating the pages, I want to look at linking the pages to the MongoDB database.
* I am unsure how to link user input with MongoDB. Can you make a step outlining what I should research for validation and user input.
* I am thinking about using the event Emitter package to handle user input in JavaScript.
* DO not use EJS in any circumstances.

## Stage 2: Display some random listings in the “listings”

In the div with the id “listings” initially display some listings from the listingsAndReviews database.

## 2.1 Edit the code

The code doesn’t display any listings.

## Stage 3: Would be adding user validation

It connects. The issue is validation. I want to have validation on the client side then on server for mongodb. Here is an idea how I would do it:

Server side validation

* After submitting check if a user submitted a value. If false, give a prompt saying to enter a listing E.g. “Be Happy in Porto”. This prompt appears in a div that is invisible at the beginning but filled in false. In the javascript file, there is code to reset div to be blank if user submits again.

# PROMPT: To start from scratch( but include my html files)

**Prompt Step by Step**

I have a directory, test4.

In this directory there is a public folder which contains 4 files: bookings.html, confirmation.html, index.html, styles.css.

* Next I did the following, created json package: **npm init -y**
* Next I Installed the modules: **npm install express mongodb body-parser**

Can you help me with a task?

## Here is my index.html file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Homepage</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Homepage</h1>

    </header>

    <ul>

        <li><a class="active" href="index.html">Homepage</a></li>

        <li><a href="bookings.html">Booking Page</a></li>

    </ul>

    <section class="topSection">

            <form action="index.html" method="POST">

                <fieldset>

                    <h3>Listings?</h3>

                    <div class="divForm"> <label>Location</label><input type="text" id='location' name="location" required> </div>

                    <div class="divForm"><label>Type of property</label>

                        <select>

                        <option value="House">House</option>

                        <option value="Camper/RV">Camper/RV</option>

                        <option value="Cottage">Cottage</option>

                        <option value="Apartment">Apartment</option>

                        <option value="Farm stay">Farm stay</option>

                        <option value="Houseboat">Houseboat</option>

                        <option value="Chalet">Chalet</option>

                        <option value="Aparthotel">Aparthotel</option>

                        <option value="Castle">Castle</option>

                        <option value="Guesthouse">Guesthouse</option>

                        <option value="Townhouse">Townhouse</option>

                        <option value="Nature lodge">Nature lodge</option>

                        <option value="Service apartment">Serviced apartment</option>

                        <option value="Cabin">Cabin</option>

                        <option value="Loft">Loft</option>

                        <option value="Guest suite">Guest suite</option>

                        <option value="Bungalow">Bungalow</option>

                        <option value="Case particular">Case particular (Cuba)</option>

                        <option value="Train">Train</option>

                        <option value="Earth house">Earth house</option>

                        <option value="Heritage hotel">Heritage hotel (India)</option>

                        <option value="Hostel">Hostel</option>

                        </select>

                    </div>

                    <div class="divForm"><label>Number of bedrooms</label>

                        <select>

                            <option value=1>1</option>

                            <option value=2>2</option>

                            <option value=3>3</option>

                            <option value=4>4</option>

                            <option value=5>5</option>

                            <option value=6>6</option>

                            <option value=7>7</option>

                            <option value=8>8</option>

                            <option value=9>9</option>

                            <option value=10>10</option>

                            <option value=11>11</option>

                            <option value=12>12</option>

                            <option value=13>13</option>

                            <option value=14>14</option>

                            <option value=15>15</option>

                            <option value=16>16</option>

                            <option value=17>17</option>

                            <option value=18>18</option>

                            <option value=19>19</option>

                            <option value=20>20</option>

                        </select>

                    </div>

                    <button input type="submit">Submit</button>

                </fieldset>

            </form>

    </section>

    <section class="bottomSection">

        This is for some random listings.

    </section>

</body>

</html>

**Does this make sense. Just answer yes or no.**

## Info about my collection

I am using MongoDB. I have a database called “sample\_airbnb”. Within this database I have the collection “listingsAndReviews”.

Within this collection I have the following fields. NOTE I am listing the key pair with the data type but not sample data. Also for objects and arrays, not all are listed as they are not needed.

{

\_id: String,

listing\_url: String,

name: String,

summary: String,

space: String,

description:String,

neighbourhood\_overview:String,

notes:String,

transit:String,

access:String,

interaction:String,

house\_rules:String,

property\_type:String,

room\_type:String,

bed\_type:String,

minimum\_nights:String,

maximum\_nights:String,

cancellation\_policy:String,

last\_scrapped:Date,

calendar\_last\_scraped:Date,

first\_review:Date,

last:review:Date,

accommodates:Int32,

bedrooms:Int32,

beds:Int32,

number\_of\_reviews:Int32,

bathooms:Decimal128,

amenities:Array,

price:Decimal128,

security\_deposit:Decimal128,

cleaning\_fee:Decimal128,

extra\_people:Decimal128,

guests\_included:Decimal128,

images:Object,

host:Object,

address:Object,

street:string,

market:string,

availability:Object,

review\_scores:Object,

reviews:Array,

review\_scores\_rating: int32,

bookings: Array,

0: Object

**booking\_id** :Int32

**arrival\_date** : Date,

**departure\_date** :Date,

**client** : Object

name: String,

email: String,

daytime\_phone: String,

mobile: String,

postal\_address: String,

home\_address:String,

**deposit\_paid**: int32,

**balance\_due** : int32,

**balance\_due\_date** :Date,

**number\_of\_guests**: int32,

guests: Array

0:Object

name: String,

age: int32,

}

**Does this make sense. Yes or no.**

## Task 1: Create a file to run the application.

I want you to create a file in the root directory that when it runs, connects to MongoDB and opens the index.html file.

I made a mock-up. Edit it to complete the task.

Mock up:

const { MongoClient } = require('mongodb');

//Step 1: We want to connect to cluster, call functions that query db, disconnect cluster

async function main() {

    const uri = 'mongodb+srv://s3722151:Gatesea3@assignment3cluster.kbysd.mongodb.net/';

    //Create instance of mongo

    const client = new MongoClient(uri);

    try {

        //connect to cluster. We wait until the operation is complete.

        await client.connect();

    } catch (e) {

        console.error(e);

    } finally {

        await client.close();

    }

}

//This calls function, but also send an error to catch

main().catch(console.error);

### ErrorFix: Updating server to use styles.css

**Place in server.js**

const { MongoClient } = require('mongodb');

const http = require('http');

const fs = require('fs');

const path = require('path');

// Step 1: Connect to MongoDB and serve the index.html file

async function main() {

const uri = 'mongodb+srv://s3722151:Gatesea3@assignment3cluster.kbysd.mongodb.net/';

// Create instance of MongoDB client

const client = new MongoClient(uri);

try {

// Connect to the cluster

await client.connect();

// Optionally, you can list the databases here

await listDatabases(client);

// Start the server to serve the index.html file

const server = http.createServer((req, res) => {

// Serve index.html for root URL

if (req.url === '/') {

serveFile(res, 'public/index.html', 'text/html');

}

// Serve styles.css for CSS requests

else if (req.url === '/styles.css') {

serveFile(res, 'public/styles.css', 'text/css');

}

// Handle 404 for other URLs

else {

res.writeHead(404, { 'Content-Type': 'text/plain' });

res.end('404 Not Found');

}

});

// Server listens on port 3000

server.listen(3000, () => {

console.log('Server is running on http://localhost:3000');

});

} catch (e) {

console.error(e);

} finally {

// Uncomment the following line if you want to close the connection when done

// await client.close();

}

}

// Function to serve files

function serveFile(res, filePath, contentType) {

fs.readFile(filePath, 'utf8', (err, data) => {

if (err) {

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Error reading the file');

return;

}

res.writeHead(200, { 'Content-Type': contentType });

res.end(data);

});

}

// Step 2: List databases in our cluster

async function listDatabases(client) {

const databasesList = await client.db().admin().listDatabases();

console.log("Databases:");

databasesList.databases.forEach(db => {

console.log(`- ${db.name}`);

});

}

// Call the main function, handling any potential errors

main().catch(console.error);

## Update the prompt

**Prompt Step by Step**

I have a directory, test4.

In this directory there is a public folder which contains 4 files: bookings.html, confirmation.html, index.html, styles.css. In the root directory **I have a server.js** file.

* Next I did the following, created json package: **npm init -y**
* Next I Installed the modules: **npm install express mongodb body-parser**

***Can you help me with a task?***

### Here is my index.html file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Homepage</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Homepage</h1>

    </header>

    <ul>

        <li><a class="active" href="index.html">Homepage</a></li>

        <li><a href="bookings.html">Booking Page</a></li>

    </ul>

    <section class="topSection">

            <form action="index.html" method="POST">

                <fieldset>

                    <h3>Listings?</h3>

                    <div class="divForm"> <label>Location</label><input type="text" id='location' name="location" required> </div>

                    <div class="divForm"><label>Type of property</label>

                        <select>

                        <option value="House">House</option>

                        <option value="Camper/RV">Camper/RV</option>

                        <option value="Cottage">Cottage</option>

                        <option value="Apartment">Apartment</option>

                        <option value="Farm stay">Farm stay</option>

                        <option value="Houseboat">Houseboat</option>

                        <option value="Chalet">Chalet</option>

                        <option value="Aparthotel">Aparthotel</option>

                        <option value="Castle">Castle</option>

                        <option value="Guesthouse">Guesthouse</option>

                        <option value="Townhouse">Townhouse</option>

                        <option value="Nature lodge">Nature lodge</option>

                        <option value="Service apartment">Serviced apartment</option>

                        <option value="Cabin">Cabin</option>

                        <option value="Loft">Loft</option>

                        <option value="Guest suite">Guest suite</option>

                        <option value="Bungalow">Bungalow</option>

                        <option value="Case particular">Case particular (Cuba)</option>

                        <option value="Train">Train</option>

                        <option value="Earth house">Earth house</option>

                        <option value="Heritage hotel">Heritage hotel (India)</option>

                        <option value="Hostel">Hostel</option>

                        </select>

                    </div>

                    <div class="divForm"><label>Number of bedrooms</label>

                        <select>

                            <option value=1>1</option>

                            <option value=2>2</option>

                            <option value=3>3</option>

                            <option value=4>4</option>

                            <option value=5>5</option>

                            <option value=6>6</option>

                            <option value=7>7</option>

                            <option value=8>8</option>

                            <option value=9>9</option>

                            <option value=10>10</option>

                            <option value=11>11</option>

                            <option value=12>12</option>

                            <option value=13>13</option>

                            <option value=14>14</option>

                            <option value=15>15</option>

                            <option value=16>16</option>

                            <option value=17>17</option>

                            <option value=18>18</option>

                            <option value=19>19</option>

                            <option value=20>20</option>

                        </select>

                    </div>

                    <button input type="submit">Submit</button>

                </fieldset>

            </form>

    </section>

    <section class="bottomSection">

        This is for some random listings.

    </section>

</body>

</html>

**Does this make sense. Just answer yes or no.**

### Info about my collection

I am using MongoDB. I have a database called “sample\_airbnb”. Within this database I have the collection “listingsAndReviews”.

Within this collection I have the following fields. NOTE I am listing the key pair with the data type but not sample data. Also for objects and arrays, not all are listed as they are not needed.

{

\_id: String,

listing\_url: String,

name: String,

summary: String,

space: String,

description:String,

neighbourhood\_overview:String,

notes:String,

transit:String,

access:String,

interaction:String,

house\_rules:String,

property\_type:String,

room\_type:String,

bed\_type:String,

minimum\_nights:String,

maximum\_nights:String,

cancellation\_policy:String,

last\_scrapped:Date,

calendar\_last\_scraped:Date,

first\_review:Date,

last:review:Date,

accommodates:Int32,

bedrooms:Int32,

beds:Int32,

number\_of\_reviews:Int32,

bathooms:Decimal128,

amenities:Array,

price:Decimal128,

security\_deposit:Decimal128,

cleaning\_fee:Decimal128,

extra\_people:Decimal128,

guests\_included:Decimal128,

images:Object,

host:Object,

address:Object,

street:string,

market:string,

availability:Object,

review\_scores:Object,

**review\_scores\_rating**: Int32,

reviews:Array,

bookings: Array,

0: Object

**booking\_id** :Int32

**arrival\_date** : Date,

**departure\_date** :Date,

**client** : Object

name: String,

email: String,

daytime\_phone: String,

mobile: String,

postal\_address: String,

home\_address:String,

**deposit\_paid**: int32,

**balance\_due** : int32,

**balance\_due\_date** :Date,

**number\_of\_guests**: int32,

guests: Array

0:Object

name: String,

age: int32,

}

**Does this make sense. Yes or no.**

### Contents of server.js file

This is used to run the application, connect to MongoDB and open index.html.

const { MongoClient } = require('mongodb');

const http = require('http');

const fs = require('fs');

const path = require('path');

// Step 1: Connect to MongoDB and serve the index.html file

async function main() {

    const uri = 'mongodb+srv://s3722151:Gatesea3@assignment3cluster.kbysd.mongodb.net/';

    // Create instance of MongoDB client

    const client = new MongoClient(uri);

    try {

        // Connect to the cluster

        await client.connect();

        // Start the server to serve the index.html file

        const server = http.createServer((req, res) => {

            // Serve index.html for root URL

                //We do this to seperate querying(QUERYING) from static content (HTML,CS)

            if (req.url === '/') {

                serveFile(res, 'public/index.html', 'text/html');

            }

            // Serve styles.css for CSS requests

            else if (req.url === '/styles.css') {

                serveFile(res, 'public/styles.css', 'text/css');

            }

            // Handle 404 for other URLs

            else {

                res.writeHead(404, { 'Content-Type': 'text/plain' });

                res.end('404 Not Found');

            }

        });

        // Server listens on port 3000

        server.listen(3000, () => {

            console.log('Server is running on http://localhost:3000');

        });

    } catch (e) {

        console.error(e);

    } finally {

        // Uncomment the following line if you want to close the connection when done

        // await client.close();

    }

}

// Function to serve files

function serveFile(res, filePath, contentType) {

    fs.readFile(filePath, 'utf8', (err, data) => {

        if (err) {

            res.writeHead(500, { 'Content-Type': 'text/plain' });

            res.end('Error reading the file');

            return;

        }

        res.writeHead(200, { 'Content-Type': contentType });

        res.end(data);

    });

}

// Call the main function, handling any potential errors

main().catch(console.error);

**Does this make sense. Yes or No.**

## Task 2: Display random listings log to console

In the section “bottomSection” in the index.html file, I want to display random listings from the collection.

* Each property listing on this page should comprise of {name: 1, summary:1, price:1,"review\_scores.review\_scores\_rating":1}.
* Each property listing’s name is displayed as an active hyperlink, allowing the client to choose the property and proceed to the next page (booking.html) of the application.
* This hyperlink should carry the listing\_id as a hyperlink query parameter (or URL parameter, e.g.: https://localhost:3000/bookings.html?listing\_id=10083468) and will allow the bookings page to manage the bookings for the chosen property.

An example of a listing would be displayed in brackets.

[

Name = Be Happy In Porto

Summary = *text..*

Price = 89

Customer rating = 97

]

**Make the solution.**

* **For now, just log the results to the console. I just want to check if the value are passed.**
* **Also if there is any javascript code in index.html, just place it in a separate file, index.js.**

Here is a sample framework I have created to edit to the server.js file. Feel free to edit it.

Sample framework:

//https://youtu.be/fbYExfeFsI0

//To connect to MongoDB database

const { MongoClient } = require('mongodb');

//Step 1: We want to connect to cluster, call functions that query db, disconnect cluster

async function main() {

    const uri = 'mongodb+srv://s3722151:Gatesea3@assignment3cluster.kbysd.mongodb.net/';

    //Create instance of mongo

    const client = new MongoClient(uri);

    try {

        //connect to cluster. We wait until the operation is complete.

        await client.connect();

// 6.Find Many Listings

        await findListingsWithMinimumBedroomsBathroomsAndMostRecentReviews(client, {

            minimumNumberOfBedrooms: 4,

            minimumNumberOfBathrooms: 2,

            maximumNumberOfResults: 5

        });

    } catch (e) {

        console.error(e);

    } finally {

        await client.close();

    }

}

//This calls function, but also send an error to catch

main().catch(console.error);

//6. Find multiple documents

async function findListingsWithMinimumBedroomsBathroomsAndMostRecentReviews(client, {

    //Here these are de-structured parameters

    minimumNumberOfBedrooms = 0,

    minimumNumberOfBathrooms = 0,

    maximumNumberOfResults = Number.MAX\_SAFE\_INTEGER

}={}){

    //We use find to look at multiple documents

    const cursor = await client.db("sample\_airbnb").collection("listingsAndReviews").find({

        bedrooms: { $gte: minimumNumberOfBedrooms },

        bathrooms: { $gte: minimumNumberOfBathrooms }

    }).sort({ last\_review: -1 })

        .limit(maximumNumberOfResults);

    const results = await cursor.toArray();

    if (results.length > 0) {

        console.log(`Found listings(s) with at least ${minimumNumberOfBedrooms}

        bedrooms and ${minimumNumberOfBathrooms} bathrooms:`);

        results.forEach((result, i) => {

            date = new Date(result.last\_review).toDateString();

            console.log();

            console.log(`${i + 1}. name: ${result.name}`);

            console.log(`  \_id: ${result.\_id}`);

            console.log(`   bedrooms: ${result.bedrooms}`);

            console.log(`   bathrooms: ${result.bathrooms}`);

            console.log(`   most recent review date: ${new Date(result.last\_review).

                toDateString()}`);

        });

    } else {

        console.log(`No listings found with at least ${minimumNumberOfBedrooms}

        bedrooms and ${minimumNumberOfBathrooms}`);

    }

### ErrorFix: No random bookings is displayed. In console.

**Use the console to check**

***Logging in server.js: Added a console log statement to show the random listings fetched from the database in the server console (console.log('Random Listings:', listings);).***

***Logging in index.html: Added a console log in the fetchRandomListings function to show the listings fetched from the server in the browser console (console.log('Fetched Listings:', listings);).***

***Error Handling in the Listings Section: Included a check for listings.length === 0 to display a message if no listings are available.***

***Solution:***

*Updated* ***server.js***

const { MongoClient } = require('mongodb');

const http = require('http');

const fs = require('fs');

const path = require('path');

// MongoDB connection URI

const uri = 'mongodb+srv://s3722151:Gatesea3@assignment3cluster.kbysd.mongodb.net/';

const client = new MongoClient(uri);

// Create an HTTP server

const server = http.createServer(async (req, res) => {

console.log(`Received request for: ${req.url}`);

// Serve static files from the public directory

if (req.url.startsWith('/public/')) {

const filePath = path.join(\_\_dirname, req.url);

fs.readFile(filePath, 'utf8', (err, data) => {

if (err) {

res.writeHead(404, { 'Content-Type': 'text/plain' });

res.end('404 Not Found');

return;

}

res.writeHead(200);

res.end(data);

});

} else if (req.url === '/') {

serveFile(res, 'public/index.html', 'text/html');

} else if (req.url === '/random-listings') {

try {

await client.connect();

const database = client.db('sample\_airbnb');

const collection = database.collection('listingsAndReviews');

const randomListings = await collection.aggregate([

{ $sample: { size: 5 } },

{ $project: { name: 1, summary: 1, price: 1, "review\_scores.review\_scores\_rating": 1 } }

]).toArray();

console.log('Fetched random listings:', randomListings);

res.writeHead(200, { 'Content-Type': 'application/json' });

res.end(JSON.stringify(randomListings));

} catch (error) {

console.error('Error fetching random listings:', error);

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Error fetching random listings');

}

} else {

res.writeHead(404, { 'Content-Type': 'text/plain' });

res.end('404 Not Found');

}

});

// Function to serve files

function serveFile(res, filePath, contentType) {

fs.readFile(filePath, 'utf8', (err, data) => {

if (err) {

res.writeHead(500, { 'Content-Type': 'text/plain' });

res.end('Error reading the file');

return;

}

res.writeHead(200, { 'Content-Type': contentType });

res.end(data);

});

}

// Start the server

server.listen(3000, () => {

console.log('Server is running on http://localhost:3000');

});

updated **index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Homepage</title>

    <link rel="stylesheet" href="public/styles.css">

</head>

<body>

    <header>

        <h1>Homepage</h1>

    </header>

    <ul>

        <li><a class="active" href="index.html">Homepage</a></li>

        <li><a href="bookings.html">Booking Page</a></li>

    </ul>

    <section class="topSection">

        <form action="index.html" method="POST">

            <fieldset>

                <h3>Listings?</h3>

                <div class="divForm"><label>Location</label><input type="text" id='location' name="location" required>

                </div>

                <div class="divForm"><label>Type of property</label>

                    <select>

                        <option value="House">House</option>

                        <option value="Camper/RV">Camper/RV</option>

                        <option value="Cottage">Cottage</option>

                        <option value="Apartment">Apartment</option>

                        <option value="Farm stay">Farm stay</option>

                        <option value="Houseboat">Houseboat</option>

                        <option value="Chalet">Chalet</option>

                        <option value="Aparthotel">Aparthotel</option>

                        <option value="Castle">Castle</option>

                        <option value="Guesthouse">Guesthouse</option>

                        <option value="Townhouse">Townhouse</option>

                        <option value="Nature lodge">Nature lodge</option>

                        <option value="Service apartment">Serviced apartment</option>

                        <option value="Cabin">Cabin</option>

                        <option value="Loft">Loft</option>

                        <option value="Guest suite">Guest suite</option>

                        <option value="Bungalow">Bungalow</option>

                        <option value="Case particular">Case particular (Cuba)</option>

                        <option value="Train">Train</option>

                        <option value="Earth house">Earth house</option>

                        <option value="Heritage hotel">Heritage hotel (India)</option>

                        <option value="Hostel">Hostel</option>

                    </select>

                </div>

                <div class="divForm"><label>Number of bedrooms</label>

                    <select>

                        <option value=1>1</option>

                        <option value=2>2</option>

                        <option value=3>3</option>

                        <option value=4>4</option>

                        <option value=5>5</option>

                        <option value=6>6</option>

                        <option value=7>7</option>

                        <option value=8>8</option>

                        <option value=9>9</option>

                        <option value=10>10</option>

                        <option value=11>11</option>

                        <option value=12>12</option>

                        <option value=13>13</option>

                        <option value=14>14</option>

                        <option value=15>15</option>

                        <option value=16>16</option>

                        <option value=17>17</option>

                        <option value=18>18</option>

                        <option value=19>19</option>

                        <option value=20>20</option>

                    </select>

                </div>

                <button input type="submit">Submit</button>

            </fieldset>

        </form>

    </section>

    <section class="bottomSection">

        This is for some random listings.

    </section>

    <script src="public/index.js"></script>

</body>

</html>