# 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>

**Index.js**

// Function to fetch and display random listings

async function fetchRandomListings() {

    try {

        const response = await fetch('/random-listings');

        if (!response.ok) {

            throw new Error('Network response was not ok');

        }

        const listings = await response.json();

        displayListings(listings);

    } catch (error) {

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

    }

}

// Function to display listings in the bottom section

function displayListings(listings) {

    const bottomSection = document.getElementById('bottomSection');

    bottomSection.innerHTML = ''; // Clear existing content

    listings.forEach(listing => {

        // Create a div for each listing

        const listingDiv = document.createElement('div');

        listingDiv.className = 'listing';

        // Create the hyperlink

        const listingLink = document.createElement('a');

        listingLink.href = `bookings.html?listing\_id=${listing.\_id}`; // URL with listing ID

        listingLink.innerText = listing.name; // Listing name as hyperlink

        // Create listing summary

        const summary = document.createElement('p');

        summary.innerText = `Summary: ${listing.summary || 'No summary available'}`; // Summary text

        // Create price and rating elements

        const price = document.createElement('p');

        price.innerText = `Price: $${listing.price || 'N/A'}`; // Price text

        const rating = document.createElement('p');

        rating.innerText = `Customer Rating: ${listing.review\_scores?.review\_scores\_rating || 'N/A'}`; // Rating text

        // Append elements to listing div

        listingDiv.appendChild(listingLink);

        listingDiv.appendChild(summary);

        listingDiv.appendChild(price);

        listingDiv.appendChild(rating);

        // Append listing div to bottom section

        bottomSection.appendChild(listingDiv);

    });

}

// Call the fetchRandomListings function when the page loads

window.onload = fetchRandomListings;

## Prompt update: After logging to the console

I have a directory, test4.

In this directory there is a public folder which contains 5 files: bookings.html, confirmation.html, index.html, index.js and 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**

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

### 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="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>

**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, open index.html and find random sample documents in mongodb.

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');

});

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

### Contents of index.js

// Function to fetch and display random listings

async function fetchRandomListings() {

    try {

        const response = await fetch('/random-listings');

        if (!response.ok) {

            throw new Error('Network response was not ok');

        }

        const listings = await response.json();

        displayListings(listings);

    } catch (error) {

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

    }

}

// Function to display listings in the bottom section

function displayListings(listings) {

    const bottomSection = document.getElementById('bottomSection');

    bottomSection.innerHTML = ''; // Clear existing content

    listings.forEach(listing => {

        // Create a div for each listing

        const listingDiv = document.createElement('div');

        listingDiv.className = 'listing';

        // Create the hyperlink

        const listingLink = document.createElement('a');

        listingLink.href = `bookings.html?listing\_id=${listing.\_id}`; // URL with listing ID

        listingLink.innerText = listing.name; // Listing name as hyperlink

        // Create listing summary

        const summary = document.createElement('p');

        summary.innerText = `Summary: ${listing.summary || 'No summary available'}`; // Summary text

        // Create price and rating elements

        const price = document.createElement('p');

        price.innerText = `Price: $${listing.price || 'N/A'}`; // Price text

        const rating = document.createElement('p');

        rating.innerText = `Customer Rating: ${listing.review\_scores?.review\_scores\_rating || 'N/A'}`; // Rating text

        // Append elements to listing div

        listingDiv.appendChild(listingLink);

        listingDiv.appendChild(summary);

        listingDiv.appendChild(price);

        listingDiv.appendChild(rating);

        // Append listing div to bottom section

        bottomSection.appendChild(listingDiv);

    });

}

// Call the fetchRandomListings function when the page loads

window.onload = fetchRandomListings;

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

### @@Task: Change logging to be viewable by user in index.hml

Now I want to change have the results retrieved by mongodb to be displayed within the class “bottomSection”.

* At the top there should be the number of listings that match the listings E.g. (14 Listings match your preferences)
* 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.

## Task: Make the price visible and not just an object

**Solution server.js**

  //This is the randomise function.

            const randomListings = await collection.aggregate([

                { $sample: { size: 3 } },

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

            ]).toArray();

            //There was an issue here. $Object $Object. Solve by converting decimal to

            randomListings.forEach(listing => {

                if (listing.price && listing.price.\_bsontype === 'Decimal128') {

                    listing.price = listing.price.toString(); // Convert Decimal128 to string

                    console.log('Converted price:', listing.price); // Check if conversion happens

                }

            });

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

# Challenge: Solve user input

I have a directory, DatabaseApplicationsA3.

In this directory there is a public folder which contains 5 files: **bookings.html, confirmation.html, index.html, index.js and 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 be an assistant to help me make a form that checks if it exists in a server(mongodb)? Answer yes or no only.

## Here is the contents of my server.js file

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

const http = require('http'); // For creating the HTTP server

const fs = require('fs'); // For file system operations

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');

    }

    // Route for fetching random listings

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

        try {

            await client.connect(); // First connect to the connection string

            const database = client.db('sample\_airbnb'); // Connect to the database

            const collection = database.collection('listingsAndReviews'); // Connect to the collection

            // Get the total count of listings

            const totalListingsCount = await collection.countDocuments();

            // Fetch random listings with a $sample size of 3

            const randomListings = await collection.aggregate([

                { $sample: { size: 3 } },

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

            ]).toArray();

            // Convert Decimal128 to string for price

            randomListings.forEach(listing => {

                if (listing.price && listing.price.\_bsontype === 'Decimal128') {

                    listing.price = listing.price.toString(); // Convert Decimal128 to string

                }

            });

            // Send both random listings and the total listings count

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

            res.end(JSON.stringify({ listings: randomListings, totalCount: totalListingsCount }));

        } catch (error) {

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

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

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

        }

    }

    // Route for searching listings based on filters

    else if (req.url === '/search-listings' && req.method === 'POST') {

        let body = '';

        req.on('data', chunk => {

            body += chunk.toString(); // Convert Buffer to string

        });

        req.on('end', async () => {

            try {

                const query = JSON.parse(body); // Parse the JSON body into an object

                await client.connect(); // Ensure the client is connected

                const database = client.db('sample\_airbnb'); // Connect to the database

                const collection = database.collection('listingsAndReviews'); // Connect to the collection

                // Create the search filter based on location (required) and optional fields

                const filter = {};

                if (query.location) {

                    filter['address.market'] = query.location;

                }

                if (query.property\_type) {

                    filter.property\_type = query.property\_type;

                }

                if (query.bedrooms) {

                    filter.bedrooms = parseInt(query.bedrooms, 10); // Convert bedrooms to an integer

                }

                // Query the database with the filter

                const listings = await collection.find(filter)

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

                    .toArray();

                // Convert Decimal128 to string for price

                listings.forEach(listing => {

                    if (listing.price && listing.price.\_bsontype === 'Decimal128') {

                        listing.price = listing.price.toString(); // Convert Decimal128 to string

                    }

                });

                // If listings are found, send them; otherwise, send an error message

                if (listings.length > 0) {

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

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

                } else {

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

                    res.end(JSON.stringify({ error: 'No listings found for the specified criteria.' }));

                }

            } catch (error) {

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

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

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

            }

        });

    } else {

        //indicates that the server received a request for a URL it couldn’t find or handle

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

        res.end('Error message 2: 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');

});

**Make sense? Yes or no.**

## Here is the contents of 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="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 method="POST" onsubmit="event.preventDefault(); userValidation();">

                <h3>Listings?</h3>

                <div class="divForm"><label>Location</label><input type="text" id='location' placeholder ="Name of listing E.g By Happy in Porto"name="location" required>

                </div>

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

                <select name="propertyType">

                    <option value="" disabled selected>Select a property type</option>

                    <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="Serviced 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="Casa particular (Cuba)">Casa particular (Cuba)</option>

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

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

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

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

                    <option value="Hut">Hut</option>

                    <option value="Resort">Resort</option>

                    <option value="Treehouse">Treehouse</option>

                    <option value="Boutique hotel">Boutique hotel</option>

                    <option value="Hotel">Hotel</option>

                    <option value="Barn">Barn</option>

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

                    <option value="Unknown">Unknown</option>

                    <option value="Bed and breakfast">Bed and breakfast</option>

                    <option value="Condominium">Condominium</option>

                    <option value="Campsite">Campsite</option>

                    <option value="Villa">Villa</option>

                    <option value="Pension (South Korea)">Pension (South Korea)</option>

                    <option value="Boat">Boat</option>

                    <option value="Other">Other</option>

                </select>

            </div>

                </div>

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

                    <select name="bedrooms">

                        <option value="" disabled selected>Select number of bedrooms</option>

                        <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>

                <div class="errorValidation"></div>

        </form>

    </section>

    <section class="bottomSection" id="bottomSection">

        <p id="listingsCount"></p>

        <!-- Placeholder for the number of listings -->

        This is for some random listings.

    </section>

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

    <script>

        document.getElementById('searchForm').addEventListener('submit', async (event) => {

            event.preventDefault(); // Prevent default form submission

            const formData = new FormData(event.target);

            const data = Object.fromEntries(formData.entries());

            const response = await fetch('/search-listings', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json'

                },

                body: JSON.stringify(data)

            });

            const listings = await response.json();

            const bottomSection = document.getElementById('bottomSection');

            bottomSection.innerHTML = ''; // Clear previous listings

            if (response.ok) {

                listings.forEach(listing => {

                    const listingDiv = document.createElement('div');

                    listingDiv.innerHTML = `

                        <h2><a href="bookings.html?listing\_id=${listing.\_id}">${listing.name}</a></h2>

                        <p>${listing.summary}</p>

                        <p>Price: ${listing.price}</p>

                        <p>Rating: ${listing['review\_scores.review\_scores\_rating']}</p>

                    `;

                    bottomSection.appendChild(listingDiv);

                });

            } else {

                bottomSection.innerHTML = `<p>${listings.error}</p>`;

            }

        });

    </script>

</body>

</html>

**Make sense. Yes or no.**

## Here is the contents of my index.js file

// Function to fetch and display random listings

async function fetchRandomListings() {

    try {

        const response = await fetch('/random-listings');

        if (!response.ok) {

            throw new Error('Network response was not ok');

        }

        const { listings, totalCount } = await response.json(); // Destructure the response to get listings and total count

        displayListings(listings); // Listings name of value stored in MongoDB

        // Update listings count

        const listingsCount = document.getElementById('listingsCount');

    } catch (error) {

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

    }

}

// Function to display listings in the bottom section

function displayListings(listings) {

    const bottomSection = document.getElementById('bottomSection'); // This looks at form

    // Clear existing content to allow update when refresh

    bottomSection.innerHTML = '';

    listings.forEach(listing => {

        // Create a div for each listing

        const listingDiv = document.createElement('div');

        listingDiv.className = 'listing';

        // Create the hyperlink

        const listingLink = document.createElement('a');

        listingLink.href = `bookings.html?listing\_id=${listing.\_id}`; // URL with listing ID

        listingLink.innerText = listing.name; // Listing name as hyperlink

        listingLink.className = 'active'; // Add class for active styling

        // Create listing summary

        const summary = document.createElement('p');

        summary.innerText = `Summary: ${listing.summary || 'No summary available'}`; // Summary text

        // Create price and rating elements

        const price = document.createElement('p');

        const priceValue = listing.price ? listing.price.toString() : 'N/A';

        price.innerText = `Price: $${priceValue}`; // Use the converted price or 'N/A'

        const rating = document.createElement('p');

        rating.innerText = `Customer Rating: ${listing.review\_scores?.review\_scores\_rating || 'N/A'}`; // Rating text

        // Append elements to listing div

        listingDiv.appendChild(listingLink);

        listingDiv.appendChild(summary);

        listingDiv.appendChild(price);

        listingDiv.appendChild(rating);

        // Append listing div to bottom section

        bottomSection.appendChild(listingDiv);

    });

}

// Call the fetchRandomListings function when the page loads

window.onload = fetchRandomListings;

/\* ----------------- PART 2: User Validation ----------------------------------------------------------------------------------------------- \*/

async function userValidation() {

    // Step 1: Get values from user input

    const location = document.getElementById('location').value.trim();

    const propertyType = document.querySelector('select[name="propertyType"]').value || null; // Allow null if unselected

    const bedrooms = document.querySelector('select[name="bedrooms"]').value || null; // Allow null if unselected

    // Ensure location is provided (it’s required by HTML, but double-check here)

    if (!location) return;

    try {

        // Step 2: Prepare query object with required location

        const query = { 'address.market': location }; // Location required

        // Add optional filters if provided

        if (propertyType) query.property\_type = propertyType; // Only add if it's selected

        if (bedrooms) query.bedrooms = parseInt(bedrooms, 10); // Only add if it's selected

        // Step 3: Send the query to the server

        const response = await fetch('/search-listings', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify(query),

        });

        // Step 4: Handle response and display results

        if (response.ok) {

            const listings = await response.json();

            // If listings match, display them; if not, show an error message

            if (listings.length > 0) {

                displayListings(listings);

                document.querySelector('.errorValidation').innerText = ''; // Clear error if matches found

            } else {

                // Show error message if no matches

                const errorValidation = document.querySelector('.errorValidation');

                errorValidation.innerText = 'No listings found for the specified criteria.';

            }

        } else {

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

        }

    } catch (error) {

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

    }

}

**Make sense. 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.

## Task: Update the listings

As a user I want to search for a listing in the location input tag.

* If the user selects a matching listing(location), then in the bottomSection, information about the listing would be displayed. {name: 1, summary:1, price:1,"review\_scores.review\_scores\_rating":1}. This will clear the randomly generated listings. **Valid type of property and a valid number of bedrooms can be left null.**
* If the enters a valid listing(location), a valid type of property, then in the bottomSection, information about the listing would be displayed. {name: 1, summary:1, price:1,"review\_scores.review\_scores\_rating":1}. This will clear the randomly generated listings. **Valid number of bedrooms can be left null.**
* If the enters a valid listing(location), a valid number of bedrooms, then in the bottomSection, information about the listing would be displayed. {name: 1, summary:1, price:1,"review\_scores.review\_scores\_rating":1}. This will clear the randomly generated listings. **Valid number type of property can be left null.**
* If the enters a valid listing(location), a valid type of property and a valid number of bedrooms, then in the bottomSection, information about the listing would be displayed. {name: 1, summary:1, price:1,"review\_scores.review\_scores\_rating":1}. This will clear the randomly generated listings.
* **Else if** there are no listings that match what is In mongodb, have a message returned in errorValidation that there is no listings of that name.

## Task

The index.html 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, bottomSection 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.html) of the application.

What is displayed is similar to the function displayListing but based on user input.

## Prompt: Debug issue

I am having the following issue. When I enter a correct listing in the input ‘location’ , bottomSection refreshes the listings from the database but does not display the listing. E.g there is 1 listing called Be Happy in Porto { name: "Be Happy in Porto" }. But when I click submit the form does not refresh and display the listing.

What I want is if the listing exists:

* “id="listingsCount" will be updated to display the number of listings. E.g. documents with the name “İstanbul Birden fazla bölümden oluşan bina” has 2 documents. So the message “This has 2 documents”.
* If the user only entered the location, all relevant information would be displayed. So name of the property, summary, daily price, and review score rating
* If the user entered location and (type of property OR number of bedrooms all relevant information would be displayed. So name of the property, summary, daily price, and review score rating

**Error**

I have the following errors:

**Error 1**

Error fetching listings: TypeError: Cannot set properties of null (setting 'innerText')

at userValidation (index.js:101:41)

* This refers to listingsCount.innerText = `This has ${listings.length} document${listings.length > 1 ? 's' : ''}`;

at async HTMLFormElement.<anonymous> ((index):112:13)

* This refers to console.error('Error fetching listings:', error);

**Error 2**

POST http://localhost:3000/search-listings 404 (Not Found)

* This refers to await userValidation();
* This refers to const response = await fetch('/search-listings',

**Error 3**

Uncaught (in promise)

*{message: 'A listener indicated an asynchronous response by r…age channel closed before a response was received'}*

Please fix the code.

# User input from scratch (added 2 divs in bottomSection)

I have a directory, DatabaseApplicationsA3. In this directory there is a public folder which contains 5 files: bookings.html, confirmation.html, index.html, index.js and 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 be an assistant to help me make my code cleaner using express. Answer yes or no only.

## Here is my server.js file:

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

const http = require('http'); // For creating the HTTP server

const fs = require('fs'); // For file system operations

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');

    }

    // Route for fetching random listings

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

        try {

            await client.connect(); // First connect to the connection string

            const database = client.db('sample\_airbnb'); // Connect to the database

            const collection = database.collection('listingsAndReviews'); // Connect to the collection

            // Get the total count of listings

            const totalListingsCount = await collection.countDocuments();

            // Fetch random listings with a $sample size of 3

            const randomListings = await collection.aggregate([

                { $sample: { size: 3 } },

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

            ]).toArray();

            // Convert Decimal128 to string for price

            randomListings.forEach(listing => {

                if (listing.price && listing.price.\_bsontype === 'Decimal128') {

                    listing.price = listing.price.toString(); // Convert Decimal128 to string

                }

            });

            // Send both random listings and the total listings count

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

            res.end(JSON.stringify({ listings: randomListings, totalCount: totalListingsCount }));

        } catch (error) {

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

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

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

        }

    }

     else {

        //indicates that the server received a request for a URL it couldn’t find or handle

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

        res.end('Error message 2: 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');

});

**Make sense? Answer yes or no only.**

## 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="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" id="topSection">

        <form id="searchForm" method="POST" onsubmit="searchListings(event)">

            <h3>Listings?</h3>

            <div class="divForm">

                <label for="location">Location(required)</label>

                <input type="text" id='location' placeholder="Name of listing E.g By Happy in Porto" name="location"

                    required>

            </div>

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

                <select name="propertyType" id="propertyType">

                    <option value="">Select a property type</option>

                    <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="Serviced 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="Casa particular (Cuba)">Casa particular (Cuba)</option>

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

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

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

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

                    <option value="Hut">Hut</option>

                    <option value="Resort">Resort</option>

                    <option value="Treehouse">Treehouse</option>

                    <option value="Boutique hotel">Boutique hotel</option>

                    <option value="Hotel">Hotel</option>

                    <option value="Barn">Barn</option>

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

                    <option value="Unknown">Unknown</option>

                    <option value="Bed and breakfast">Bed and breakfast</option>

                    <option value="Condominium">Condominium</option>

                    <option value="Campsite">Campsite</option>

                    <option value="Villa">Villa</option>

                    <option value="Pension (South Korea)">Pension (South Korea)</option>

                    <option value="Boat">Boat</option>

                    <option value="Other">Other</option>

                </select>

            </div>

            </div>

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

                <select name="bedrooms" id="bedrooms">

                    <option value="">Select number of bedrooms</option>

                    <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>

            <div class="errorValidation"></div>

        </form>

    </section>

    <section class="bottomSection" id="bottomSection">

             <h2>Entered location</h2>

             <div id="listingEntered"></div>

             <h2>Random Listings</h2>

            <div id="randomListings"></div>

    </section>

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

</body>

</html>

Make sense? **Answer yes or no**

## Here is my index.js file:

// Function to fetch and display random listings

async function fetchRandomListings() {

    try {

        const response = await fetch('/random-listings');

        if (!response.ok) {

            throw new Error('Network response was not ok');

        }

        const { listings } = await response.json(); // Adjusted destructuring if totalCount is not needed

        displayListings(listings); // Display random listings

    } catch (error) {

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

    }

}

// Function to display listings in the bottom section

function displayListings(listings) {

    const bottomSection = document.getElementById('randomListings'); // Target the bottom section

    // Clear existing content

    bottomSection.innerHTML = '';

    listings.forEach(listing => {

        // Create a div for each listing

        const listingDiv = document.createElement('div');

        listingDiv.className = 'listing';

        // Create the hyperlink

        const listingLink = document.createElement('a');

        listingLink.href = `bookings.html?listing\_id=${listing.\_id}`; // URL with listing ID

        listingLink.innerText = listing.name; // Listing name as hyperlink

        listingLink.className = 'active'; // Add class for active styling

        // Create listing summary

        const summary = document.createElement('p');

        summary.innerText = `Summary: ${listing.summary || 'No summary available'}`; // Summary text

        // Create price and rating elements

        const price = document.createElement('p');

        const priceValue = listing.price ? listing.price.toString() : 'N/A';

        price.innerText = `Price: $${priceValue}`; // Price display

        const rating = document.createElement('p');

        rating.innerText = `Customer Rating: ${listing.review\_scores?.review\_scores\_rating || 'N/A'}`; // Rating text

        // Append elements to listing div

        listingDiv.appendChild(listingLink);

        listingDiv.appendChild(summary);

        listingDiv.appendChild(price);

        listingDiv.appendChild(rating);

        // Append listing div to bottom section

        bottomSection.appendChild(listingDiv);

    });

}

// Call the fetchRandomListings function when the page loads

window.onload = fetchRandomListings;

/\* ----------------- PART 2: User Validation ----------------------------------------------------------------------------------------------- \*/

**Make sense. 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.

## Task

I want to create a function that will take a look at the input in the form. This input will consist of location, type of property and number of bedrooms.

* 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, listingEntered will get refreshed with property listings that matches with the filtering criteria the client has submitted.

* For example, if the user entered “Barcelona” as the location and left other two inputs empty, it will display all properties in the Barcelona market (address.market). The associated listings will also be displayed if location is entered and either the type of property or number of bedrooms is entered.

Each property listing in listingEntered 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.html) of the application.

# Create application with express

## Context

I want to create a web application using MongoDB, Express and Node. Can you help me? I will provide more information for the project over time. Do the task and always clarify things unless not specified. Do you understand?

## 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.

## Directory

Here is my director. I have a blank server.js file in the root directory. I have a public folder. In that folder, I have a index.html and styles.css file.

Here is the contents of 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="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" id="topSection">

        <form id="searchForm" method="POST" onsubmit="">

            <h3>Listings?</h3>

            <div class="divForm">

                <label for="location">Location(required)</label>

                <input type="text" id='location' placeholder="Name of listing E.g By Happy in Porto" name="location"

                    required>

            </div>

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

                <select name="propertyType" id="propertyType">

                    <option value="">Select a property type</option>

                    <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="Serviced 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="Casa particular (Cuba)">Casa particular (Cuba)</option>

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

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

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

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

                    <option value="Hut">Hut</option>

                    <option value="Resort">Resort</option>

                    <option value="Treehouse">Treehouse</option>

                    <option value="Boutique hotel">Boutique hotel</option>

                    <option value="Hotel">Hotel</option>

                    <option value="Barn">Barn</option>

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

                    <option value="Unknown">Unknown</option>

                    <option value="Bed and breakfast">Bed and breakfast</option>

                    <option value="Condominium">Condominium</option>

                    <option value="Campsite">Campsite</option>

                    <option value="Villa">Villa</option>

                    <option value="Pension (South Korea)">Pension (South Korea)</option>

                    <option value="Boat">Boat</option>

                    <option value="Other">Other</option>

                </select>

            </div>

            </div>

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

                <select name="bedrooms" id="bedrooms">

                    <option value="">Select number of bedrooms</option>

                    <option value=1>1</option>

                    <option value=2>2</option>

                    <option value=3>3</option>

                </select>

            </div>

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

            <div class="errorValidation"></div>

        </form>

    </section>

    <section class="bottomSection" id="bottomSection">

        <h2>Entered location</h2>

        <div id="listingEntered"></div>

        <h2>Random Listings</h2>

        <div id="randomListings"></div>

    </section>

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

</body>

</html>

Do you understand? Yes or no.

## Overview of the project

Here is an overview of the project. My web application will allow users to filter listings based on entering the form and will allow them to choose one listing from the presented list, enter their details and add a new booking for their requested dates to the database. The application will have a minimum of three pages.

For now I want to focus on the index.html page. Does that make sense? I will provide more details after this.

## Index.html

I want you to do 2 tasks.

The first task is to take in user input, check if it exists in the database and return it in listingEntered.

The second task is to initially display random listings in randomListings.

Here are the details. Index.html is divided into 2 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 relevant listings would be shown.
* If a client had filled location AND (type of property OR number of bedrooms), then relevant listings would be shown.

Structure for each listing would be like this:

* Number of listings E.g. 14 listings match your preferences
* Name of listing: E.g. Be Happy in Porto
* Summary: E.g. “By Happy ..”
* Daily Rate: E.g. 30
* Customer rating: E.g. 97

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, open the (booking.html, and store the values associated with the listing that they choose.

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.

# Booking html

I have a directory, DatabaseApplicationsA3. In this directory there is a public folder which contains 5 files: bookings.html, confirmation.html, index.html, index.js and 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 mongoose -g

Can you be an assistant to help my bookings.html file.

## 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 (ISO 8601),

**departure\_date** : Date (ISO 8601),

**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 (ISO 8601),

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

guests: Array

0:Object

name: String,

age: int32,

}

Does this make sense. Just answer Yes or no only.

## This is my server.js file

// Load environment variables from the .env file

require('dotenv').config();

// Import necessary modules

const express = require('express');

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

const path = require('path');

// Initialize Express app and set port

const app = express();

const PORT = process.env.PORT || 3000;

// MongoDB connection URI from .env

const uri = process.env.MONGO\_URI;

// Middleware for static files and JSON parsing

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use(express.json());

// MongoDB Client and Database Setup

let db;

async function connectToDb() {

    try {

        const client = new MongoClient(uri);

        await client.connect();

        db = client.db('sample\_airbnb'); // Specify your database name here

        console.log('Connected to MongoDB');

    } catch (error) {

        console.error('Error connecting to MongoDB:', error);

    }

}

connectToDb();

// Function to convert price to string

function convertPrice(price) {

    if (price && price.\_bsontype === 'Decimal128') {

        return price.toString();

    }

    return price;

}

// Endpoint to get random listings

app.get('/api/listings/random', async (req, res) => {

    try {

        const listings = await db.collection('listingsAndReviews').aggregate([{ $sample: { size: 5 } }]).toArray();

        // Log to check if the listings are retrieved from server

        //console.log('Random Listings Fetched:', listings);

        const convertedListings = listings.map(listing => ({

            ...listing,

            price: convertPrice(listing.price),

        }));

        res.json(convertedListings);

    } catch (error) {

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

        res.status(500).json({ error: 'Error fetching random listings' });

    }

});

// Endpoint to search listings based on form input

app.post('/api/listings/search', async (req, res) => {

    const { location, propertyType, bedrooms } = req.body;

    const query = {

        // Dynamic Object - only properties with truthy values included in final object

        ...(location && {

            $expr: { $eq: [{ $strcasecmp: ['$name', location] }, 0] } // Case-insensitive exact match for name

        }),

        ...(propertyType && { property\_type: propertyType }),

        ...(bedrooms && { bedrooms: parseInt(bedrooms, 10) }),

    };

    // Used to check what if input is equal to server

    // console.log('Received search query:', query); // Log the search query

    try {

        const listings = await db.collection('listingsAndReviews').find(query).limit(10).toArray();

        // console.log('Listings retrieved from database:', listings); // Log the retrieved listings

        // Convert price before sending response

        //https://www.w3schools.com/jsref/jsref\_map.asp

        const convertedListings = listings.map(listing => ({

            ...listing,

            price: convertPrice(listing.price),

        }));

        res.json(convertedListings);

    } catch (error) {

        console.error('Error searching for listings:', error);

        res.status(500).json({ error: 'Error searching for listings' });

    }

});

// Endpoint to handle booking submission - Booking step Server side processing

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

// Endpoint to handle booking submission

app.post('/api/bookings', async (req, res) => {

    const { checkIn, checkOut, yourName, emailAddress, daytimePhone, mobileNo, postalAddress, residentialAddress, listingId } = req.body;

    //Inspect backlog logic

    console.log('Received booking request:', req.body);

    // Validate date format on the server

    const datePattern = /^\d{4}-\d{2}-\d{2}$/;

    if (!datePattern.test(checkIn) || !datePattern.test(checkOut)) {

        return res.status(400).json({ error: "Dates must be in YYYY-MM-DD format" });

    }

    // Convert dates to ISO 8601 format

    const arrivalDate = new Date(`${checkIn}T00:00:00.000Z`);

    const departureDate = new Date(`${checkOut}T00:00:00.000Z`);

    // Validate listing ID format

    if (!ObjectId.isValid(listingId)) {

        return res.status(400).json({ error: "Invalid listing ID" });

    }

    // Create booking object

    const booking = {

        booking\_id: Math.floor(Math.random() \* 100000),

        arrival\_date: arrivalDate,

        departure\_date: departureDate,

        client: {

            name: yourName,

            email: emailAddress,

            daytime\_phone: daytimePhone,

            mobile: mobileNo,

            postal\_address: postalAddress,

            home\_address: residentialAddress,

        },

        number\_of\_guests: 1 // Adjust this as needed

    };

    try {

        // Update the listing and add the booking

        const result = await db.collection('listingsAndReviews').updateOne(

            { \_id: new ObjectId(listingId) },

            { $push: { bookings: { $each: [booking], $ifNull: [[], []] } } },

            { upsert: true } // This will create a new listing if it doesn't exist

        );

        // Check if update was successful

        if (result.matchedCount === 0) {

            return res.status(404).json({ error: "Listing not found" });

        }

        console.log("Booking successfully submitted:", booking);

        // Send success response

        res.json({ success: true, booking });

    } catch (error) {

        console.error("Error saving booking:", error);

        res.status(500).json({ error: "Failed to save booking" });

    }

});

// Start the server

app.listen(PORT, () => {

    console.log(`Server is running on http://localhost:${PORT}`);

});

Make sense. Answer Yes or no only.

## Bookings file

Here is my bookings.html file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Bookings Page</title>

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

</head>

<body>

    <header>

        <h1>Bookings Page</h1>

    </header>

    <ul>

        <li><a href="index.html" onclick="sessionStorage.removeItem('selectedListing')">Homepage</a></li>

    </ul>

    <section class="topSection">

        <h2>Let's book the property</h2>

        <!-- Section to display booking details -->

        <div id="bookingDetails"></div>

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

            <fieldset>

                <h3>Booking Details</h3>

                <div class="divForm"> <label>Check In:</label><input type="text" id='checkIn' name="checkIn" placeholder="YYYY-MM-DD" required>

                </div>

                <div class="divForm"> <label>Check Out:</label><input type="text" id='checkOut' name="checkOut" placeholder="YYYY-MM-DD" required>

                </div>

                <h3>Your Details</h3>

                <div class="divForm"> <label>Your Name:</label><input type="text" id='yourName' name="yourName"

                        required>

                </div>

                <div class="divForm"> <label>Email Address: </label><input type="text" id='emailAddress'

                        name="emailAddress" required></div>

                <div class="divForm"> <label>Your Daytime Phone:</label><input type="text" id='daytimePhone' name="daytimePhone" required></div>

                <div class="divForm"> <label>Your Mobile No:</label><input type="text" id='mobileNo' name="mobileNo"

                        required></div>

                <div class="divForm"> <label>Postal Address:</label><input type="text" id='postalAddress'

                        name="postalAddress" required></div>

                <div class="divForm"> <label>Residential Address:</label><input type="text" id='residentialAddress'

                        name="residentialAddress" required></div>

                <button type="submit">Book Now</button>

            </fieldset>

        </form>

        <div class="errorValidation"></div>

    </section>

    <script>

        // Step 1: Retrieve the listing data when the page loads

        document.addEventListener('DOMContentLoaded', () => {

            const selectedListing = JSON.parse(sessionStorage.getItem('selectedListing'));

            if (selectedListing) {

                // Display the listing information on bookings.html

                const bookingDetails = document.getElementById('bookingDetails');

                bookingDetails.innerHTML = `

                    <h3>Property Details:</h3>

                    <p><strong>Name:</strong> ${selectedListing.name}</p>

                    <p><strong>Summary:</strong> ${selectedListing.summary}</p>

                    <p><strong>Daily Rate:</strong> $${selectedListing.price}</p>

                    <p><strong>Customer Rating:</strong> ${selectedListing.review\_scores?.review\_scores\_rating || 'N/A'}</p>

                `;

            } else {

                const bookingDetails = document.getElementById('bookingDetails');

                bookingDetails.innerHTML = '<p>No property selected.</p>';

            }

        });

        //Step 2: Client-Side Validation for Dates in YYYY-MM-DD Format

        document.querySelector('form').addEventListener('submit', function (event) {

                const checkIn = document.getElementById('checkIn').value;

                const checkOut = document.getElementById('checkOut').value;

                const datePattern = /^\d{4}-\d{2}-\d{2}$/; // Regex for YYYY-MM-DD format

                const errorDiv = document.querySelector('.errorValidation');

                        // Clear previous error message

                errorDiv.textContent = '';

                // Validate date format

                if (!datePattern.test(checkIn) || !datePattern.test(checkOut)) {

                    event.preventDefault();

                    errorDiv.textContent = "Please enter Check In and Check Out details in YYYY-MM-DD format.";

                    return;

                }

            });

        //Step 3: Re-direct to confirmation.html to store the bookings temporarily

        fetch('/api/bookings', {

            method: 'POST',

            headers: { 'Content-Type': 'application/json' },

            body: JSON.stringify({

                checkIn: document.getElementById('checkIn').value,

                checkOut: document.getElementById('checkOut').value,

                yourName: document.getElementById('yourName').value,

                emailAddress: document.getElementById('emailAddress').value,

                daytimePhone: document.getElementById('daytimePhone').value, // Add daytimePhone here

                mobileNo: document.getElementById('mobileNo').value,

                postalAddress: document.getElementById('postalAddress').value,

                residentialAddress: document.getElementById('residentialAddress').value,

                listingId: selectedListing.\_id,

            })

        })

        .then(response => response.json())

        .then(data => {

            if (data.success) {

                sessionStorage.setItem('bookingDetails', JSON.stringify(data.booking));

                window.location.href = 'confirmation.html';

            } else {

                alert(data.error || "Booking failed. Please try again.");

            }

        })

        .catch(error => console.error("Error:", error));

    </script>

</body>

</html>

Does that make sense. Just answer yes or no.

## Here is my issue:

I think my issue is in bookings.html here:

        fetch('/api/bookings', {

                method: 'POST',

                headers: { 'Content-Type': 'application/json' },

                body: JSON.stringify({

                    checkIn: document.getElementById('checkIn').value,

                    checkOut: document.getElementById('checkOut').value,

                    yourName: document.getElementById('yourName').value,

                    emailAddress: document.getElementById('emailAddress').value,

                    mobileNo: document.getElementById('mobileNo').value,

                    postalAddress: document.getElementById('postalAddress').value,

                    residentialAddress: document.getElementById('residentialAddress').value,

                    listingId: selectedListing.\_id,

                })

            })

                .then(response => response.json())

                .then(data => {

                    if (data.success) {

                        sessionStorage.setItem('bookingDetails', JSON.stringify(data.booking));

                        window.location.href = 'confirmation.html';

                    } else {

                        alert(data.error || "Booking failed. Please try again.");

                    }

                })

                .catch(error => console.error("Error:", error));

And in the server.js file this code:

// Endpoint to handle booking submission - Booking step Server side processing

app.post('/api/bookings', async (req, res) => {

    const { checkIn, checkOut, yourName, emailAddress, mobileNo, postalAddress, residentialAddress, listingId } = req.body;

    // Validate date format on the server as well

    const datePattern = /^\d{4}-\d{2}-\d{2}$/;

    if (!datePattern.test(checkIn) || !datePattern.test(checkOut)) {

        return res.status(400).json({ error: "Dates must be in YYYY-MM-DD format" });

    }

    // Convert dates to ISO 8601 format for MongoDB

    const arrivalDate = new Date(`${checkIn}T00:00:00.000Z`);

    const departureDate = new Date(`${checkOut}T00:00:00.000Z`);

    // Create booking object

    const booking = {

        booking\_id: Math.floor(Math.random() \* 100000), // Random ID for simplicity

        arrival\_date: arrivalDate,

        departure\_date: departureDate,

        client: {

            name: yourName,

            email: emailAddress,

            daytime\_phone: mobileNo,

            postal\_address: postalAddress,

            home\_address: residentialAddress,

        },

        number\_of\_guests: 1, // Example, adjust as needed

    };

    try {

        // Find listing by ID and update

        const result = await db.collection('listingsAndReviews').updateOne(

            { \_id: listingId },

            { $push: { bookings: booking } }

        );

        if (result.matchedCount === 0) {

            return res.status(404).json({ error: "Listing not found" });

        }

        res.json({ success: true, booking });

    } catch (error) {

        console.error("Error saving booking:", error);

        res.status(500).json({ error: "Failed to save booking" });

    }

});

Does that make sense. Just answer yes or no only.

## Now I want to do the following:

In the bookings.html after a user has a submitted the form, there is validation that checks if the **checkIn** and **checkOut** fields are in correct data type (YYYY-MM-DD).

* In the database of MongoDB, date is IS0 8601 (e.g. 2025-03-11T04:00:00.000+00:00). In this case the format YYYY-MM-DD has to be correct and time and timezone have to be inserted into submission.

Then if all fields are entered correctly, submit the information to the database. This will be done in the following steps:

* Check if the bookings array exists for the associated listing name. If not create the array and the object that with it’s fields.
* InsertOne into the bookings array that will: Generate a booking\_id for the object. Insert arrival\_date, departure\_date. Within the client object, insert name, email, daytime\_phone, mobile, postal\_address, home\_address. number\_of\_guests will just be assigned 1.
* Make sure to have a console.log so I check if the form was sent through.

## Confirmation.html file

Here is the contents of my confirmation.html file.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Booking Confirmation</title>

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

</head>

<body>

    <header>

        <h1>Booking Confirmation</h1>

    </header>

    <section="topSection">

        <div id="confirmationDetails"></div>

        <a href="index.html">Return to Homepage</a>

    </section>

    <script>

        // Retrieve booking details from sessionStorage

        document.addEventListener('DOMContentLoaded', () => {

            const bookingDetails = JSON.parse(sessionStorage.getItem('bookingDetails'));

            if (bookingDetails) {

                document.getElementById('confirmationDetails').innerHTML = `

                    <h3>Your Booking:</h3>

                    <p><strong>Name:</strong> ${bookingDetails.client.name}</p>

                    <p><strong>Email:</strong> ${bookingDetails.client.email}</p>

                    <p><strong>Arrival:</strong> ${bookingDetails.arrival\_date}</p>

                    <p><strong>Departure:</strong> ${bookingDetails.departure\_date}</p>

                `;

            }

        });

    </script>

</body>

</html>

Does that make sense. Yes or no?

## Confirmation error

Error I receive when I submit the form on the booking.html form:

Cannot POST /confirmation.html

This is what is in the console of the server:

Uncaught (in promise) {message: 'A listener indicated an asynchronous response by r…age channel closed before a response was received'}

What type of problem is it?

* Did the problem occur on bookings.html?
* Was the routing not specified correctly in server.js?
* What is the solution?

# Embedded or referenced approach

I am building a web application to enter values into the array bookings in monogodb.

## Info about my collection in a embedded approach.

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 (ISO 8601),

**departure\_date** : Date (ISO 8601),

**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 (ISO 8601),

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

guests: Array

0:Object

name: String,

age: int32,

}

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

## Info about my collection in a referenced approach.

This is the collection for “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,

}

There is then another collection which stores documents bookings for listings called “Bookings”.

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: ObjectID,

**booking\_id** :Int32

**arrival\_date** : Date (ISO 8601),

**departure\_date** : Date (ISO 8601),

**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 (ISO 8601),

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

guests: Array

0:Object

name: String,

age: int32,

}

**Out of the referenced or embedded approach in mongodb, which is easier to insert values into for a web application.**