# Prompt begin

You are a database assistant.

# 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.

**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 breakdown:

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

Additional constraints:

* 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 directory.
* I would also like to create the base pages (Homepage, Bookings, Booking confirmation) first and place them in 1 directory. Make this a step. 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 know how to use MongoDB and HTML but unsure how to link user input with MongoDB. Can you make a step outlining what I should research for validation and user input.

# Folder structure

A screenshot of a computer

Description automatically generated

* <https://www.geeksforgeeks.org/folder-structure-for-a-node-js-project/>

1. Public

Purpose: Contains files that anyone can access directly through the web browser.

Examples:

index.html: The main webpage that users see.

script.js: JavaScript code that makes your web pages interactive.

CDN Links: Links to external libraries or frameworks (like jQuery or Bootstrap) that help build your site without hosting them yourself.

2. Routes

Purpose: Contains files that define how the server responds to different requests (like visiting a page or submitting a form).

Examples:

Authentication routes: A file for login or registration, where you define what happens when a user tries to log in or sign up.

3.Models

Purpose: Defines the structure of the data that your application uses (like a blueprint).

Examples: Schemas: Files that specify what data to expect from the client (like user information) and how to validate that data (like ensuring an email is in the correct format).

4. App.js: The main file where everything starts. It sets up the server and connects all parts of the application.

Examples:

Defines the main routes for your application.

Specifies which port the server listens on (like a phone number for your app).

# Steps

**Step 1: Set Up Your Development Environment**

* **Create a Project Directory**:
  + Create a main project folder (e.g., airbnb-app).
* **Initialize a Node.js Project**:
  + Run npm init -y in your terminal within the project directory. This will create a package.json file.

**Research Hints**:

* Look up "how to initialize a Node.js project" and "npm basics".

**Step 2: Install Required Packages**

* **Install Express and Other Packages**:
  + Install necessary packages using npm:

npm install express mongoose body-parser

**Research Hints**:

* Look for "npm packages for Node.js", "Express.js setup", "Mongoose for MongoDB".

<https://www.w3schools.com/nodejs/nodejs_intro.asp>

<https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js>

<https://www.reddit.com/r/javascript/comments/36sef5/what_is_expressjs_exactly_i_keep_hearing_nodejs/>

<https://www.geeksforgeeks.org/npm-mongoose/>

**Step 3: Create the Folder Structure**

* **Create the Following Directory Structure**:

A screenshot of a computer

Description automatically generated

**Research Hints**:

* Learn about "Node.js project folder structure".
* <https://www.geeksforgeeks.org/folder-structure-for-a-node-js-project/>

**Step 4: Create Basic HTML Pages**

* **Create HTML Pages**:
  + **Homepage (index.html)**:
    - Create a form with inputs for Location, Property Type, and Number of Bedrooms.
    - Include a section to display property listings.
  + **Booking Page (bookings.html)**:
    - Create a form to capture booking details (dates, client info).
  + **Confirmation Page (confirmation.html)**:
    - Simple text confirming the booking and a link to return to the homepage.

**Research Hints**:

* Look for "HTML form examples", "HTML structure", "creating forms in HTML". <https://www.w3schools.com/html/html_forms.asp>

**Step 5: Link HTML Pages with Backend**

* **Set Up Express Server**:
  + In app.js, set up a basic Express server and static file serving:

javascript

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static('public'));

// Connect to MongoDB

mongoose.connect('your\_mongodb\_connection\_string\_here', { useNewUrlParser: true, useUnifiedTopology: true });

// Set routes

const apiRoutes = require('./routes/api');

app.use('/api', apiRoutes);

const PORT = 3000;

app.listen(PORT, () => {

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

});

**Research Hints**:

* Search for "Express.js routing", "MongoDB connection with Mongoose".

**Step 6: Set Up Mongoose Model for Bookings**

* **Create Booking Model**:
  + In models/booking.js, create a schema for booking data:

javascript

Copy code

const mongoose = require('mongoose');

const bookingSchema = new mongoose.Schema({

listingId: String,

clientName: String,

email: String,

phone: String,

startDate: Date,

endDate: Date,

postalAddress: String,

homeAddress: String,

});

module.exports = mongoose.model('Booking', bookingSchema);

**Research Hints**:

* Look for "Mongoose schema creation", "Mongoose models".

**Step 7: Implement API for Handling Form Submissions**

* **Create Routes**:
  + In routes/api.js, create endpoints to handle the form submissions for filtering listings and creating bookings.

**Example of Handling Form Submission**:

javascript

Copy code

const express = require('express');

const router = express.Router();

const Booking = require('../models/booking');

// Endpoint to filter listings

router.post('/filter-listings', (req, res) => {

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

// Query the MongoDB database for listings based on the input criteria

});

// Endpoint to create a booking

router.post('/create-booking', (req, res) => {

const newBooking = new Booking(req.body);

newBooking.save((err) => {

if (err) return res.status(500).send(err);

res.redirect('/confirmation.html');

});

});

module.exports = router;

**Research Hints**:

* Look for "Express.js route handling", "Mongoose CRUD operations".

**Step 8: Link User Input with MongoDB**

* **Implement Form Validation**:
  + Research how to validate user input in Express. Use a library like express-validator for easier validation.

**Research Hints**:

* Search for "input validation in Express.js", "express-validator tutorial".

**Step 9: Testing the Application**

* **Run the Application**:
  + Start the server using node app.js and test the functionality by visiting http://localhost:3000/index.html.

**Research Hints**:

* Look for "testing Express.js applications", "how to use Postman with Express".

**Step 10: Additional Features**

* **Enhancements and Error Handling**:
  + Add error handling in your routes.
  + Consider implementing session management for bookings.