1. Create open A.I account,

https://platform.openai.com/docs/overview

2. Create an API KEY.

3. Create a Secret in Wix: name it : OpenAI_API_Key

4. Create a collection, make sure to name it exactly like this because is case sensitive name: Prompt

5. Create a collection make sure to name it exactly like this because is case sensitive name: FieldMappings

6. Add Your Wix Elements (Input Box, text element, repeater,

7. Make sure the ID, for the dynamic page or your repeater is connected to the dataset: dynamicDataset

8. Create a backend file for the code (watch video)

9. Copy the IDs from your collection in the field id in the collection FieldMappigns
10. Add your collection id name, in the field id in the collection: FieldMappigns

CODES ARE BELOW.

FRONT PAGE CODE: (PASTE THIS CODE IN THE PAGE WHERE YOU WANT THE SEARCH TO HAPPEN)

```
import { searchProjects } from 'backend/backendGeneralAi';
import wixData from 'wix-data';

$w.onReady(function () {
    console.log("🔹 Page Loaded");

    // ✅ Handle button click
    $w('#submitBTN').onClick(async () => {
        let query = $w('#InstructionsPromptInputs').value.trim();
        if (!query) {
            console.log("⚠️ Input is empty.");
            $w("#RecommendationTXT").text = "Please enter an issue to get
recommendations.";
            return;
        }

        console.log(`🔹 Searching for: ${query}`);

        // Show loading indicator
        $w('#submitBTN').label = "Searching...";
        $w('#submitBTN').disable();

        try {
            console.log("🔹 Sending query to backend...");
            let searchResults = await searchProjects(query);
            console.log("🔹 Search results received:", searchResults);

            if (Array.isArray(searchResults) && searchResults.length > 0) {
                console.log(`🔹 Displaying ${searchResults.length} results`);

                // ✅ Update the recommendation text
                let recommendationText = `Based on your input, we recommend the
following cake(s):\n\n`;
                searchResults.forEach(result => {
                    recommendationText += `🍰 ${result.field_1}\n`;
                });
                $w("#RecommendationTXT").text = recommendationText;

                // ✅ Update dataset filter for the repeater
                let dataset = $w("#dynamicDataset");
                dataset.setFilter(wixData.filter().hasSome("_id",
searchResults.map(item => item.id)));

                // ✅ Update repeater data dynamically
```

```
                $w("#repeaterResultsAI").data = searchResults;
            } else {
                console.log("⚠️ No matching results found.");
                $w("#RecommendationTXT").text = "We couldn't find a product based on your search. Please contact us.";

                // ✅ Clear dataset filter & repeater
                $w("#dynamicDataset").setFilter(wixData.filter().ne("_id", "dummy_value"));
                $w("#repeaterResultsAI").data = [];
            }
        } catch (error) {
            console.error("❌ Error in search:", error);
            $w("#RecommendationTXT").text = "An error occurred while fetching recommendations.";
        }

        // Reset button label and re-enable
        $w('#submitBTN').label = "Submit";
        $w('#submitBTN').enable();
    });
});
```

BACKEND CODE BELOW

CREATE BACKEND FILE (WATCH VIDEO AND PASTE THIS CODE IN THAT SECTION)
MAKE SURE IT'S NAME: backendGeneralAi
And once added should look like this backendGeneralAi.jsw

```
import { fetch } from 'wix-fetch';
import wixData from 'wix-data';
import { getSecret } from 'wix-secrets-backend';

export async function searchProjects(query) {
    if (!query) throw new Error("Query is required");

    try {
        // ✅ Retrieve API Key from Wix Secrets Manager
        const apiKey = await getSecret("OpenAI_API_Key");
        if (!apiKey) throw new Error("API Key not found in Secrets Manager");

        console.log("🔹 API Key Retrieved Successfully");

        // ✅ Step 1: Retrieve Collection & Field Mappings
        let mappingResults = await wixData.query("FieldMappings").limit(1).find();
        if (mappingResults.items.length === 0) {
            console.error("⚠️ No field mappings found!");
            return [];
        }

        let mapping = mappingResults.items[0];
        let collectionName = mapping.collectionName[0]; // Tags field - takes the
first selected collection
        let fieldIDs = mapping.fieldIDs.split(",").map(field => field.trim()); //
Convert CSV to array

        console.log(`🔹 Using Collection: ${collectionName}`);
        console.log(`🔹 Field IDs: ${fieldIDs.join(", ")}`);

        // ✅ Step 2: Retrieve Data Dynamically from the Collection
        let results = await wixData.query(collectionName).limit(1000).find();

        // ✅ Fix: Check if results exist
        if (!results.items || results.items.length === 0) {
            console.log("⚠️ No data found in the specified collection");
            return [];
        }

        // ✅ Fix: Ensure _id field is present & is a string
        let validItems = results.items.filter(item => item._id && typeof item._id ===
"string");
```

```
        if (validItems.length === 0) {
            console.log("⚠️ No valid items with proper `_id` found in the collection.");
            return [];
        }

        // ✅ Step 3: Infer Field Types from First Item
        let firstItem = validItems[0]; // Get first valid item
        let fieldTypes = {};

        fieldIDs.forEach(field => {
            let value = firstItem[field];

            if (typeof value === "number") {
                fieldTypes[field] = "number";
            } else if (Object.prototype.toString.call(value) === "[object Date]") {
                fieldTypes[field] = "date";
            } else if (typeof value === "boolean") {
                fieldTypes[field] = "boolean";
            } else {
                fieldTypes[field] = "string"; // Default
            }
        });

        console.log(`🔹 Inferred Field Types:`, fieldTypes);

        // ✅ Step 4: Dynamically Extract Field Values & Format Based on Type
        let items = validItems.map(item => {
            let formattedItem = { id: item._id }; // Always include default ID

            fieldIDs.forEach((field, index) => {
                let fieldType = fieldTypes[field] || "string";
                let value = item[field];

                // ✅ Format Values Based on Type
                if (fieldType === "date" && value) {
                    value = new Date(value).toISOString().split("T")[0]; // Convert to YYYY-MM-DD format
                } else if (fieldType === "number" && value !== undefined) {
                    value = parseFloat(value); // Ensure number is formatted correctly
                } else if (fieldType === "boolean") {
                    value = value ? "Yes" : "No"; // Convert boolean to Yes/No
                } else if (!value) {
                    value = "N/A"; // Default for empty fields
                }
```

```
                formattedItem[`field_${index + 1}`] = value;
            });

            return formattedItem;
        });

        console.log(`🔹 Sending ${items.length} formatted items to OpenAI...`);

        // ✅ Step 5: Retrieve AI Prompt & Industry from "Prompt" Collection
        let promptResults = await
wixData.query("Prompt").descending("_createdDate").limit(1).find();
        if (promptResults.items.length === 0) {
            console.error("⚠️ No prompt found in 'Prompt' collection");
            return [];
        }

        let industry = promptResults.items[0].yourIndustry || "General";
        let customPrompt = promptResults.items[0].yourAgentPrompt;
        console.log(`🔹 Using Industry: ${industry}`);
        console.log("🔹 Custom Prompt Retrieved:", customPrompt);

        // ✅ Step 6: Call OpenAI API
        const openAiEndpoint = 'https://api.openai.com/v1/chat/completions';

        const response = await fetch(openAiEndpoint, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${apiKey}`
            },
            body: JSON.stringify({
                model: "gpt-4",
                messages: [
                    { role: "system", content: `You are an expert in ${industry}.` },
                    { role: "user", content: `${customPrompt}\n\nUser Query:
${query}\nDataset: ${JSON.stringify(items)}\n\nIMPORTANT: Respond only with a JSON
array of relevant cakes. Do NOT include explanations or extra text.` }
                ],
                temperature: 0.5
            })
        });

        const data = await response.json();

        // ✅ Ensure response is valid before processing
        if (!data.choices || !data.choices[0] || !data.choices[0].message ||
!data.choices[0].message.content) {
```

```
            console.error("⚠️ OpenAI response is missing expected data:", data);
            return [];
        }

        console.log("🔹 OpenAI Response:", data.choices[0].message.content);

        try {
            let aiResponse = data.choices[0].message.content.trim();

            // ✅ Extract JSON if extra text is included
            const jsonMatch = aiResponse.match(/\[.*\]/s); // Extracts the first valid
JSON array
            if (!jsonMatch) {
                console.error("⚠️ OpenAI response did not contain valid JSON:",
aiResponse);
                return [];
            }

            return JSON.parse(jsonMatch[0]); // ✅ Extract JSON only
        } catch (error) {
            console.error("⚠️ Error parsing OpenAI response:", error);
            return [];
        }

    } catch (error) {
        console.error("❌ Error in searchProjects function:", error);
        return [];
    }
}
```