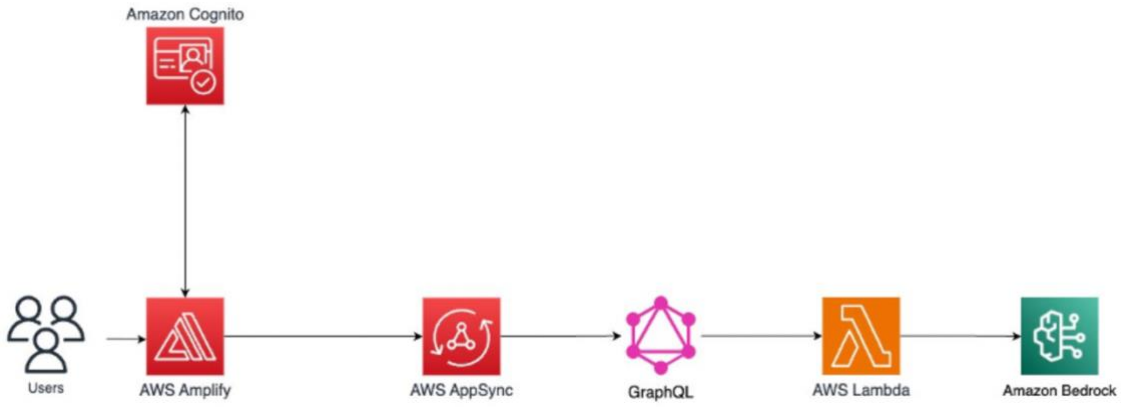
BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

Application Architecture:



AWS AMPLIFY:

Hosting a static website:

- Here we are using AWS Amplify to host our website. It provides Git-based workflow for hosting full-stack serverless web applications with continuous deployment.
- Amplify deploys your app to the AWS global content delivery network (CDN)
- Amplify hosting supports single page application frameworks (SPA) which is basically a web application or website that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of loading entire new pages. The goal is faster transitions.

Step 1: In this step we are going to create a new React application and push it to a Git repository.

- Firstly install node.js and verify the installation of node and npm
  node --version
  npm --version

- In a new terminal or command line window, run the following command to use Vite to create a React application:

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

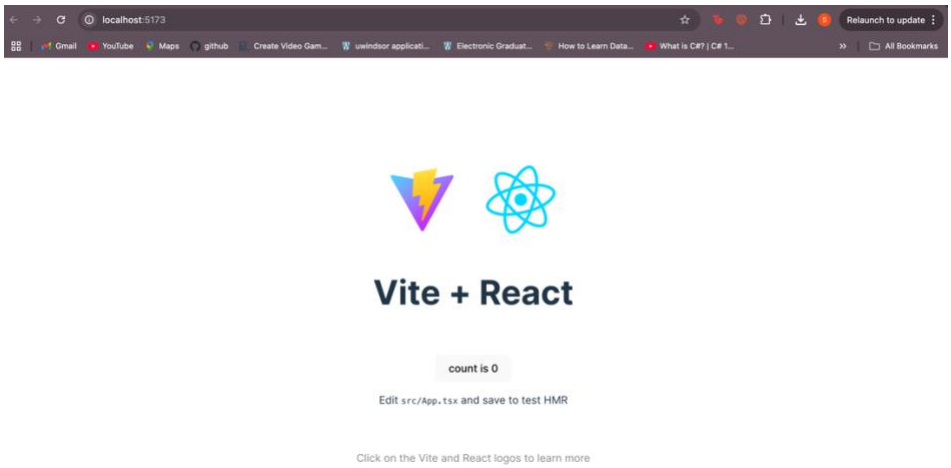
```
npm create vite@latest ai-recipe-generator -- --template react-ts -y
cd ai-recipe-generator
npm install
npm run dev
```

- Open the local link:

```
sailavanyapudi — esbuild ◂ npm run dev TMPDIR=/var/folders/zm/4__fp7...

VITE v6.0.11  ready in 518 ms

➜  Local:   http://localhost:5173/
➜  Network: use --host to expose
➜  press h + enter to show help
```

Step 2:

- The application which is hosted on the localhost:5173



- 5173 is vite's default port

- Create a new GitHub repo with name (ai-recipe-generator)
- Open a new terminal
- Change the directory to the project root folder i.e ai-recipe-generator
- Use these commands to initialise a git and push the created application to the new GitHub Repo from your terminal:
- First you need to check if there is a ssh connection established or not, if not create one ssh key and authenticate.

```
git init
git add .
git commit -m "first commit"
git remote add origin git@github.com:<your user name>/ai-recipe-generator.git
git branch -M main
git push -u origin main
```

which pushes our project files to the ai-recipe-generator github repo

Step 3:

- In this step we are gonna install the amplify packages
- Go to the directory of the root folder
- Run the below command to scaffold a lightweight Amplify project in the app's directory.

  Scaffold: Automated code or project structure.

  ```
  npm create amplify@latest -y
  ```

  you can now find a amplify project folder in the main directory.

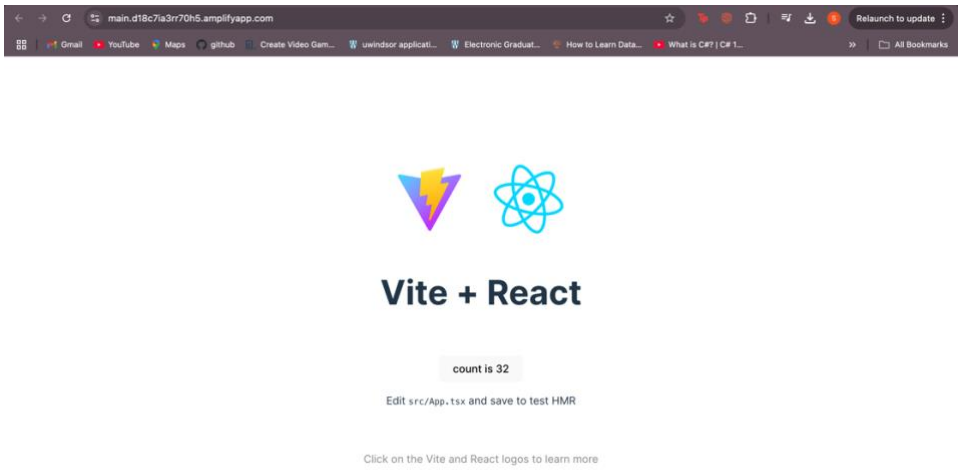
- To modify the changes

  ```
  git add .
  git commit -m 'installing amplify'
  git push origin main
  ```

Step 4: Deploy our app with AWS Amplify

- Open aws amplify
- Deploy your app with GitHub
- Select the correct repository and main branch
- Check the default build settings, save and deploy
- You can now check the displayed URL

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI



- Now check for the DNS which is xyz.amplifyapp.com

MANAGE AUTHENTICATION OF USERS:

In this task we are gonna configure authentication for the app using AWS **Amplify Auth**, which is part of **Amazon Cognito**

Step 1: This app uses email as the default login. When the user signs up, they receive a verification email. We will customize the verification email.

Update the below code to the resource.ts file

TS resource.ts ×

Users > sailavanyapudi > ai-recipe-generator > amplify > auth > TS resource.ts > ...

```ts
import { defineAuth } from "@aws-amplify/backend";

export const auth = defineAuth({
  loginWith: {
    email: {
      verificationEmailStyle: "CODE",
      verificationEmailSubject: "Welcome to the AI-Powered Recipe Generator!",
      verificationEmailBody: (createCode) =>
        `Use this code to confirm your account: ${createCode()}`,
    },
  },
});
```

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

Amazon Bedrock enables users to request access to a variety of Generative AI models. Here we will need access to Claude 3 Sonnet from Anthropic.

Step 2:

- Set up AWS Amazon Bedrock
- Choose the foundation model: Claude by Anthropic
- Request model access for Claude 3 Sonnet

BUILD A SERVERLESS BACKEND

- In this task, you will configure a serverless function using AWS Amplify and AWS Lambda.
- This function takes an input parameter i.e. ingredients to generate a prompt. It then sends this prompt to Amazon Bedrock via an HTTP POST request to the Claude 3 Sonnet model. The body of the request includes the prompt string within a messages array.

Step 1: Creating a Lambda function for handling requests.

Create bedrock.js file in the data folder of amplify and update it with this code:

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

```
TS resource.ts ~/.../auth    JS bedrock.js ×    TS resource.ts ~/.../data ●
Users > sailavanyapudi > ai-recipe-generator > amplify > data > JS bedrock.js > response
export function request(ctx) {
    const { ingredients = [] } = ctx.args;

    // Construct the prompt with the provided ingredients
    const prompt = `Suggest a recipe idea using these ingredients: ${ingredients.join(", ")}.`;

    // Return the request configuration
    return {
      resourcePath: `/model/anthropic.claude-3-sonnet-20240229-v1:0/invoke`,
      method: "POST",
      params: {
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({
          anthropic_version: "bedrock-2023-05-31",
          max_tokens: 1000,
          messages: [
            {
              role: "user",
              content: [
                {
                  type: "text",
                  text: `\n\nHuman: ${prompt}\n\nAssistant:`,
                },
              ],
            },
          ],
        }),
      },
    };
  }

  export function response(ctx) {
    // Parse the response body
    const parsedBody = JSON.parse(ctx.result.body);
    // Extract the text content from the response
    const res = {
      body: parsedBody.content[0].text,
    };
    // Return the response
    return res;
```

- This code is composed of request function and response function in lambda.
- The above code defines the askBedrock query that takes an array of strings called ingredients and returns a BedrockResponse.
- Run the command **npx ampx sandbox** to deploy cloud resources into an isolated development space so you can iterate fast.

```
ements = ["REQUIRES_NUMBERS","REQUIRES_LOWERCASE","REQUIRES_UPPERCASE","REQUIRES
_SYMBOLS"]
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.region = ca-central-
1
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.signupAttributes = [
"email"]
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.socialProviders =
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.userPoolId = ca-cent
ral-1_C93SeWgWH
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.usernameAttributes =
 ["email"]
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.verificationMechanis
ms = ["email"]
amplify-airecipegenerator-sailavanyapudi-sandbox-ff8ab5ee49.webClientId = 17c9o6
kppmgem0aouvh6a81b6l
Stack ARN:
arn:aws:cloudformation:ca-central-1:890742574937:stack/amplify-airecipegenerator
-sailavanyapudi-sandbox-ff8ab5ee49/4f2e75d0-e2a9-11ef-a6e0-0ae15a6a3c8d

✨  Total time: 176.39s

[Sandbox] Watching for file changes...
File written: amplify_outputs.json
```

Troubleshooting:

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

- I forget to configure my AWS account in the terminal. Installed aws cli and added my aws secret access key and aws access key.

❖ We have successfully configured a GraphQL API to define a custom query to connect to Amazon Bedrock and generate recipes based on a list of ingredients.

Note: When to Use What?

✅ Use SQL when you need simple, structured, and fast data retrieval.
✅ Use GraphQL when you need flexible, customized data (like for APIs).

BUILD THE FRONTEND OF THE WEB APPLICATION:

- Here we will enhance the website that was developed by integrating **AWS Amplify's pre-built UI components** to streamline a secure user authentication workflow.

- This includes enabling functionalities like account creation, login, and password recovery. Additionally, you will connect the application to a **GraphQL API** to fetch personalized recipe recommendations by leveraging a query that processes a user-provided list of ingredients, which was the backend API we created in the previous step.

Step1: we will install two amplify libraries for the project.

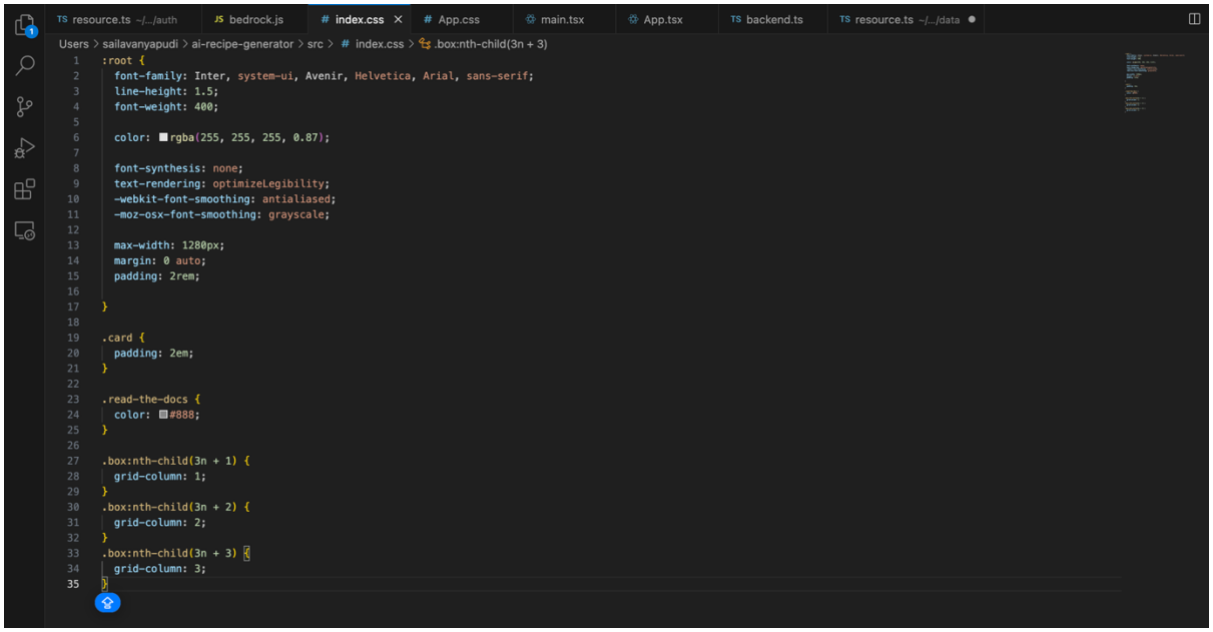
1. The aws-amplify library – which contains all the client side APIs for connecting your app's frontend to the backend
2. Aws-amplify/ui-react – contains framework specific UI components.

Command: **npm install aws-amplify @aws-amplify/ui-react**

Step2: Style the App UI

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

- navigate to the ai-recipe-generator/src/index.css file, and update it with the following code to center the App UI.



❖ Update the src/App.css file with the following code to style the ingredients form.

```
.app-container {

  margin: 0 auto;
  padding: 20px;
  text-align: center;
}

.header-container {
  padding-bottom: 2.5rem;
  margin:  auto;
  text-align: center;

  align-items: center;
  max-width: 48rem;
```

```
}

.main-header {
  font-size: 2.25rem;
  font-weight: bold;
  color: #1a202c;
}

.main-header .highlight {
  color: #2563eb;
}

@media (min-width: 640px) {
  .main-header {
    font-size: 3.75rem;
  }
}

.description {

  font-weight: 500;
  font-size: 1.125rem;
  max-width: 65ch;
  color: #1a202c;
}

.form-container {
  margin-bottom: 20px;
}

.search-container {
  display: flex;
  flex-direction: column;
  gap: 10px;
  align-items: center;
}

.wide-input {
  width: 100%;
```

```
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.search-button {
  width: 100%; /* Make the button full width */
  max-width: 300px; /* Set a maximum width for the button */
  padding: 10px;
  font-size: 16px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.search-button:hover {
  background-color: #0056b3;
}

.result-container {
  margin-top: 20px;
  transition: height 0.3s ease-out;
  overflow: hidden;
}

.loader-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 10px;
}

.result {
  background-color: #f8f9fa;
  border: 1px solid #e9ecef;
  border-radius: 4px;
```

```
  padding: 15px;
  white-space: pre-wrap;
  word-wrap: break-word;
  color: black;



  font-weight: bold;
  text-align: left; /* Align text to the left */
}
```

Step3: To implement the UI

- navigate to the ai-recipe-generator/src/main.tsx file, and update it with the following code.

```
TS resource.ts ~/.../auth   JS bedrock.js   # index.css   # App.css   main.tsx
Users > sailavanyapudi > ai-recipe-generator > src > main.tsx
1   import React from "react";
2   import ReactDOM from "react-dom/client";
3   import App from "./App.jsx";
4   import "./index.css";
5   import { Authenticator } from "@aws-amplify/ui-react";
6
7   ReactDOM.createRoot(document.getElementById("root")!).render(
8     <React.StrictMode>
9       <Authenticator>
10        <App />
11      </Authenticator>
12    </React.StrictMode>
13  );
```

- Open the ai-recipe-generator/src/App.tsx file, and update it with the following code.

```
import { FormEvent, useState } from "react";
import { Loader, Placeholder } from "@aws-amplify/ui-react";
import "./App.css";
import { Amplify } from "aws-amplify";
import { Schema } from "../amplify/data/resource";
import { generateClient } from "aws-amplify/data";
import outputs from "../amplify_outputs.json";



import "@aws-amplify/ui-react/styles.css";
```

```
Amplify.configure(outputs);

const amplifyClient = generateClient<Schema>({
  authMode: "userPool",
});

function App() {
  const [result, setResult] = useState<string>("");
  const [loading, setLoading] = useState(false);

  const onSubmit = async (event: FormEvent<HTMLFormElement>) => {
    event.preventDefault();
    setLoading(true);

    try {
      const formData = new FormData(event.currentTarget);

      const { data, errors } = await amplifyClient.queries.askBedrock({
        ingredients: [formData.get("ingredients")?.toString() || ""],
      });

      if (!errors) {
        setResult(data?.body || "No data returned");
      } else {
        console.log(errors);
      }



    } catch (e) {
      alert(`An error occurred: ${e}`);
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="app-container">
      <div className="header-container">
```

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

```
    <h1 className="main-header">
      Meet Your Personal
      <br />
      <span className="highlight">Recipe AI</span>
    </h1>
    <p className="description">
      Simply type a few ingredients using the format ingredient1,
      ingredient2, etc., and Recipe AI will generate an all-new recipe on
      demand...
    </p>
  </div>
  <form onSubmit={onSubmit} className="form-container">
    <div className="search-container">
      <input
        type="text"
        className="wide-input"
        id="ingredients"
        name="ingredients"
        placeholder="Ingredient1, Ingredient2, Ingredient3,...etc"
      />
      <button type="submit" className="search-button">
        Generate
      </button>
    </div>
  </form>
  <div className="result-container">
    {loading ? (
      <div className="loader-container">
        <p>Loading...</p>
        <Loader size="large" />
        <Placeholder size="large" />
        <Placeholder size="large" />
        <Placeholder size="large" />
      </div>
    ) : (
      result && <p className="result">{result}</p>
    )}
  </div>
</div>
```

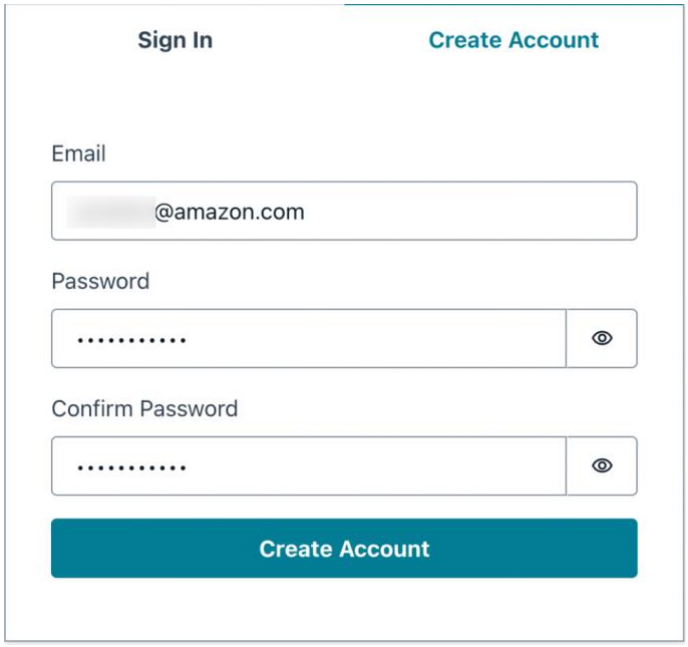
```
  );
}

export default App;
```

- run the following command to launch the app: **npm run dev**
- Select the Local host link to open the Vite + React application.

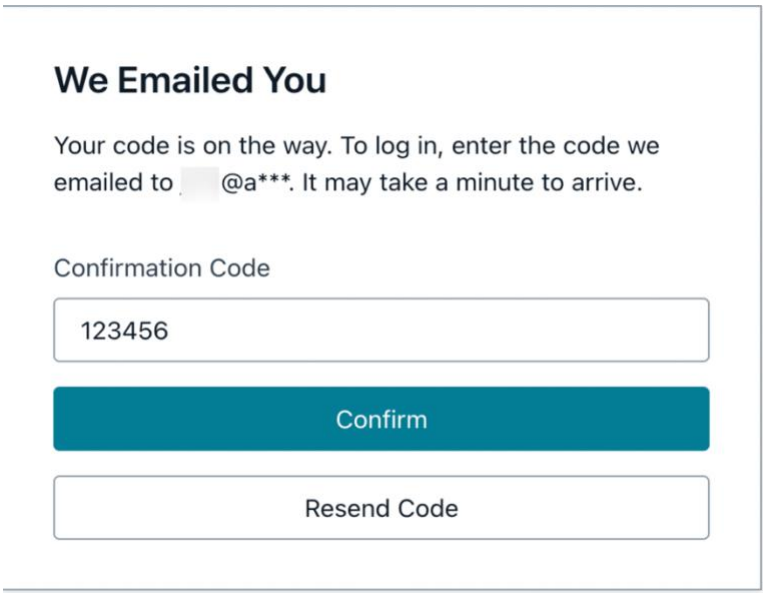
Outputs:

- To create an account

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI

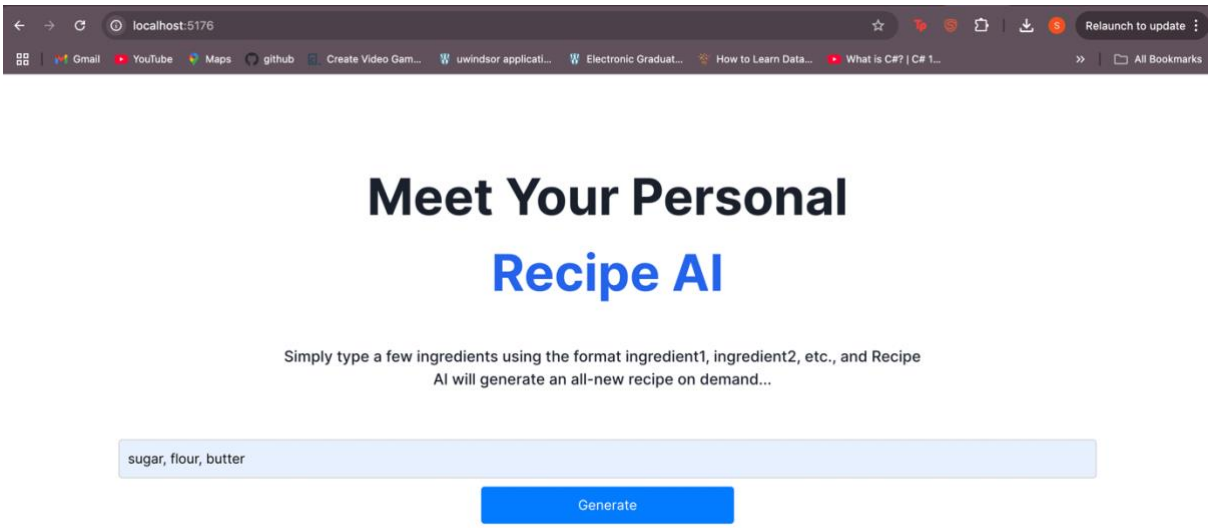


❖ To send the verification code to the email and complete the verification

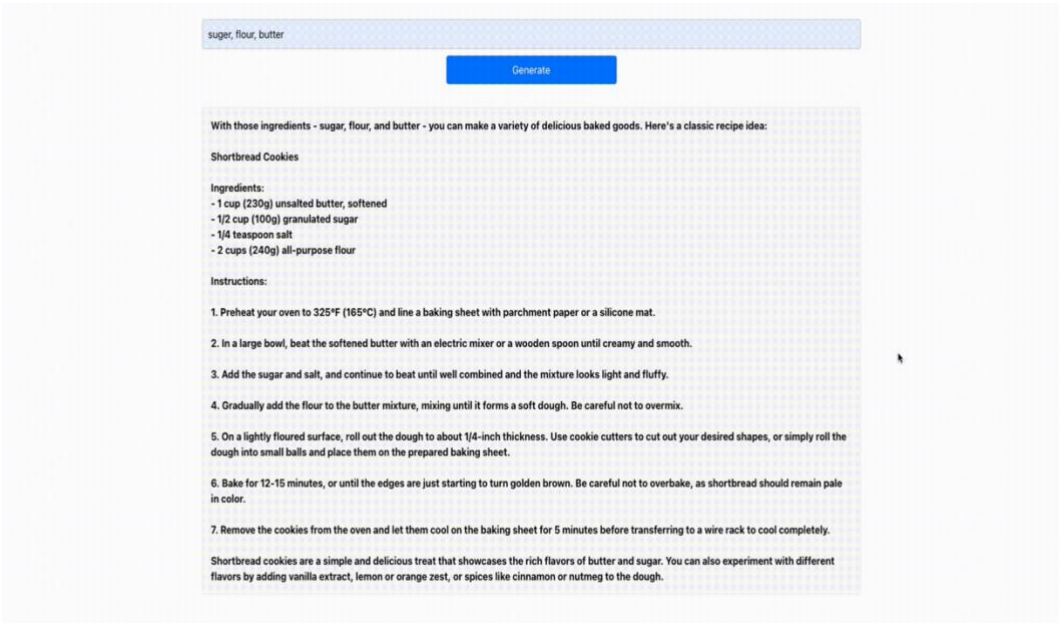


❖ To Generate AI recipes with the given ingredients

BUILD A SERVERLESS WEB APPLICATION USING GENERATIVE AI



Tadaaa… we successfully created a serverless web application using Generative AI



- Finally push the code to the github repo:

```
git add .
git commit -m 'connect to bedrock'
git push origin main
```