

```

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">

```

```

    <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;

```