AI Document Assistant

This repository contains a base web application. It uses Vite + React. To deploy you will run a basic CDK stack using Amazon S3 and Amazon Cloudfront with AWS WAF for security.

Requirements

In order to run and deploy this project, you need to have installed:

- AWS CLI. Refer to Installing the AWS CLI
- AWS Credentials configured in your environment. Refer to Configuration and credential file settings
- Node >= 20.x.x
- pnpm - Fast, disk space efficient package manager
- AWS CDK. Refer to Getting started with the AWS CDK

You also need to have the proper backend stack for your prototype deployed into your account, as well as a valid user configured in Amazon Cognito.

Make sure to deploy the frontend using the same configuration option as your backend.

Developing and running locally

Configuring your environment

In a terminal, run:

```
$ cd webapp/
```

Inside the `webapp/` folder, create a file named `.env`. Copy the environment displayed below and replace the property values with the outputs from your deployed backend stack.

```
VITE_AWS_REGION="<REGION_NAME>"
VITE_COGNITO_USER_POOL_ID="<COGNITO_USER_POOL_ID>"
VITE_COGNITO_USER_POOL_CLIENT_ID="<COGNITO_USER_POOL_CLIENT_ID>"
VITE_COGNITO_IDENTITY_POOL_ID="<COGNITO_IDENTITY_POOL_ID>"
VITE_API_GATEWAY_REST_API_ENDPOINT="<API_GATEWAY_REST_API_ENDPOINT>"
VITE_API_NAME="<API_NAME>"
VITE_APP_NAME="PACE Sample Project"
```

Developing with dev mode

From the `webapp/` folder, you can run the following command in a terminal to run the app in development mode:

```
$ pnpm i
$ pnpm run dev
```

Open http://localhost:5173/ to view it in your browser.

The page will reload when you make changes. You may also see any lint errors in the console.

Developing with watch and hot reloading

In one terminal window, run:

```
$ pnpm run watch
```

In another window, run:

```
$ pnpm run preview
```

This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules. It builds the app for production to the `dist` folder. It correctly bundles React in production mode and optimizes the build for the best performance.

Development Tools

Generating TypeScript Types

The project includes a command to automatically generate TypeScript type definitions from the backend OpenAPI specification:

```
$ pnpm run generate-types
```

What this command does:

This command converts your backend API OpenAPI specification into strongly-typed TypeScript interfaces. It reads the OpenAPI schema from the backend and generates type definitions in the `src/features/documentGeneration/types` directory.

Prerequisites:

- This command expects the backend repository to be cloned at the same level as this frontend repository in your local file structure:

  ```
  /your-projects/
  ├── backend/
  ```

```
│   └── app/api/openapi.yaml
└── frontend/ (this repository)
```

When to use it:

You should run this command whenever:

- The backend API schema changes
- After pulling changes that include API schema updates
- Before starting work on frontend features that interact with the API

After running this command, you can import and use the generated types in your application code to ensure type-safe API interactions.

Deploying the app

Configure your environment

Make sure your webapp environment is configured by creating the `.env` file inside the `webapp/` folder.

The required properties are:

```
VITE_AWS_REGION="<REGION_NAME>"
VITE_COGNITO_USER_POOL_ID="<COGNITO_USER_POOL_ID>"
VITE_COGNITO_USER_POOL_CLIENT_ID="<COGNITO_USER_POOL_CLIENT_ID>"
VITE_COGNITO_IDENTITY_POOL_ID="<COGNITO_IDENTITY_POOL_ID>"
VITE_API_GATEWAY_REST_API_ENDPOINT_JOBS="
<API_GATEWAY_REST_API_ENDPOINT_JOBS>"
VITE_API_GATEWAY_REST_API_ENDPOINT_INDEX_DOCUMENTS="
<API_GATEWAY_REST_API_INDEX_DOCUMENTS>"
VITE_API_GATEWAY_REST_API_ENDPOINT_QUESTION_GENRATOR="
<API_GATEWAY_REST_API_ENDPOINT_QUESTION_GENERATOR>"
VITE_API_NAME="<API_NAME>"
VITE_APP_NAME="PACE Sample Project"
```

You can find the proper values for the environment in your backend stack deployment outputs on the CloudFormation console.

A note on WAF

The provided stack can be geo-restricted by uncommenting and configuring the `geoRestriction` attribute on `/lib/frontend-stack.ts`, in the CloudFront Web Distribution Construct instantiation (check the Docs).

Dependencies

Move to the root folder (`frontend/`).

Use pnpm to install the required dependencies.

```
$ pnpm install
```

At this point you can now synthesize the CloudFormation template for this code.

```
$ cdk synth
```

This will simply generate and print to stout the CloudFormation Template describing the resources. If this succeeds you are ready to deploy.

Deployment

You can deploy the application stack by running:

```
$ cdk deploy --require-approval=never
```

This command will build the Web Application under the `webapp/` folder and deploy it to an Amazon S3 bucket. The application is served via an Amazon Cloudfront distribution protected by a Amazon WAF WebAcl distribution. If you wish to change and redeploy your application, you can run multiple deployment commands.

The URL for your application will be printed to your terminal at the end of the process, but you can always check it again on the CloudFront's console.