Deployment Manual

Introduction

The project has already been deployed and can be found under the following URLs:

- **Frontend (the actual website):** https://purple-coast-0af99a203.4.azurestaticapps.net/
- **Backend:** https://dialogue-hub-b7gsdgf8cqc5f5bc.uksouth-01.azurewebsites.net/

This manual outlines the process for deployment in case you would like to do it in your own resource group.

Development Guide

These instructions outline how to run the project locally, if you would like to skip to only deploying the project, then only follow the Azure Set Up section and keep track of the necessary environment variables.

Prerequisites

- Git
- Docker
- Azure CLI

Clone the GitHub Repository with the following command:

```
git clone https://github.com/nblandos/dialogue-hub.git
```

Environment Variables

In the ‘backend’ directory create a ‘.env’ file, the file should contain the following:

```
SENDER_EMAIL= Confirmation email sender

SENDER_PASSWORD= 16 digit app password for email

KEY_VAULT_NAME=

OPENAI_ENDPOINT_URL=

OPENAI_API_SECRET_NAME=

DEPLOYMENT_NAME=

AZURE_ASSISTANT_ID=
```

AZURE_CLIENT_ID=

AZURE_CLIENT_SECRET=

AZURE_TENANT_ID=

You must set up an email for the confirmation email sender and generate an app password for it. The rest of the Azure related variables can be derived from the next section.

Azure Set Up

1. Login to your azure account in the CLI with the following command:

   ```
   az login
   ```

2. On the Azure portal, create a resource group
3. In Azure create an 'App Registration'
   a. Under 'Manage', 'Certificates and Secrets', create a new 'client secret'
   b. Make sure to copy the value of the client secret which will be AZURE_CLIENT_SECRET
   c. In the 'Overview' section of the App Registration, you can find the 'Application (client) ID' which is AZURE_CLIENT_ID and the 'Directory (tenant) ID' which is AZURE_TENANT_ID
4. Create an 'Azure OpenAI' resource in the resource group
   a. The OPENAI_ENDPOINT_URL can be found in the 'Keys and Endpoints' section under 'Resource Management'
   b. Also copy one of the API KEYS here for later use
   c. Go to the Azure AI Foundry Portal and Navigate to 'Assistants'. Create a new Assistant, AZURE_ASSISTANT_ID is the 'Assistant id' of this assistant
   d. Additionally, the name of the AI model deployment chosen for this assistant is DEPLOYMENT_NAME, e.g. 'gpt-4'
5. Next create a 'Key Vault' resource
   a. Under 'Objects', 'Secrets', add the copied API key for the OpenAI resource
   b. OPENAI_API_SECRET_NAME should be the name of this secret
   c. KEY_VAULT_NAME should be the name of the key vault
   d. Under 'Access Policies', add the App Registration created in 3), with Secret retrieval permissions

Alternatively, if you already have access to our existing resource group and would like to use it, please email nicholas.blandos.23@ucl.ac.uk for the corresponding environment variables.

Running the Project Locally

Once you have created the necessary environment variables, you can run the project locally with the following command:
docker compose up --build

The frontend will be accessible at: http://localhost:3000/

The backend will be accessible at: http://localhost:5001/

Running Tests

Prerequisites

It is expected that you have ‘pip’, ‘Python3’, ‘Node’ and ‘npm’ installed for this section

In the backend, run ‘pip install -r requirements.txt’ preferably in a virtual environment

In the frontend, run ‘npm install’

Commands to run

To run frontend tests and show coverage use the following command in frontend/:

npm run test:coverage

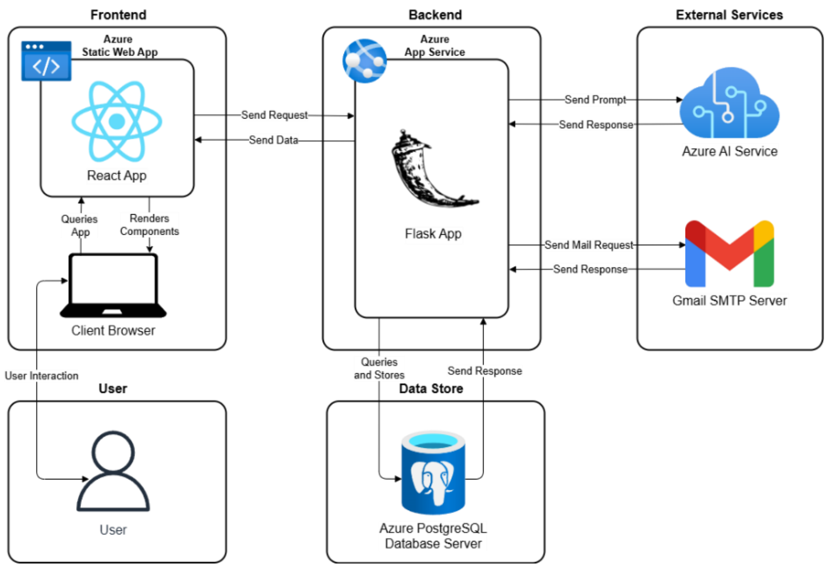
To run backend tests and show coverage use the following command in backend/:

coverage run -m pytest

coverage report

Production Guide

The following instructions describe how to set up Azure resources to deploy the project on the web, we assume that Azure Set Up has already been completed from the Development Guide.



This diagram illustrates a high-level system architecture which represents how the project will be deployed

Backend Deployment

1. In your resource group, create a new resource 'Web App + Database'
    a. For runtime stack, select 'Python 3.12'
    b. For Database Engine, select 'PostgreSQL – Flexible Server'
    c. Under this resource, navigate to 'Settings', 'Environment variables', here you should add all the environment variables mentioned in the Development Guide
    d. While here, Select AZURE_POSTGRESQL_CONNECTIONSTRING and copy the password string after 'Password=' for use later
2. Under your 'App Service' resource, navigate to 'Settings', 'Service Connector'
    a. There should already be a 'DB for PostgreSQL flexible server' connector, select it and press edit
    b. In 'Client Type', select 'Django'
    c. Under 'Authentication', select 'Connection String' and paste the previously copied password
    d. Check 'Store Secret in Key Vault' and under 'Key Vault Connection', 'Create New', finally select your existing Key Vault and finish creation

3. Finally, navigate to ‘Development Tools’, ‘SSH’ under the App Service resource
    a. Press ‘Go’ and you should be taken to a terminal
    b. Run ‘flask db upgrade’ to perform database migrations
4. The backend should now be deployed as an App Service

Frontend Deployment

1. In your resource group, create a new resource ‘Static Web App’
2. For the source control section, you need to have access to the GitHub repository, either email nicholas.blandos.23@ucl.ac.uk to be added to the repository or fork the existing repository and select it here and create the resource
3. Under this Static Web App, navigate to ‘Settings’, ‘Environment variables’, add a variable called ‘VITE_API_URL’ which holds the base URL of the backend web app
4. Under the previously created App Service for the backend, add an environment variable called ‘FRONTEND_URL’ which holds the URL of the Static Web App.

The website should now be deployed and accessible at the URL of the Static Web App