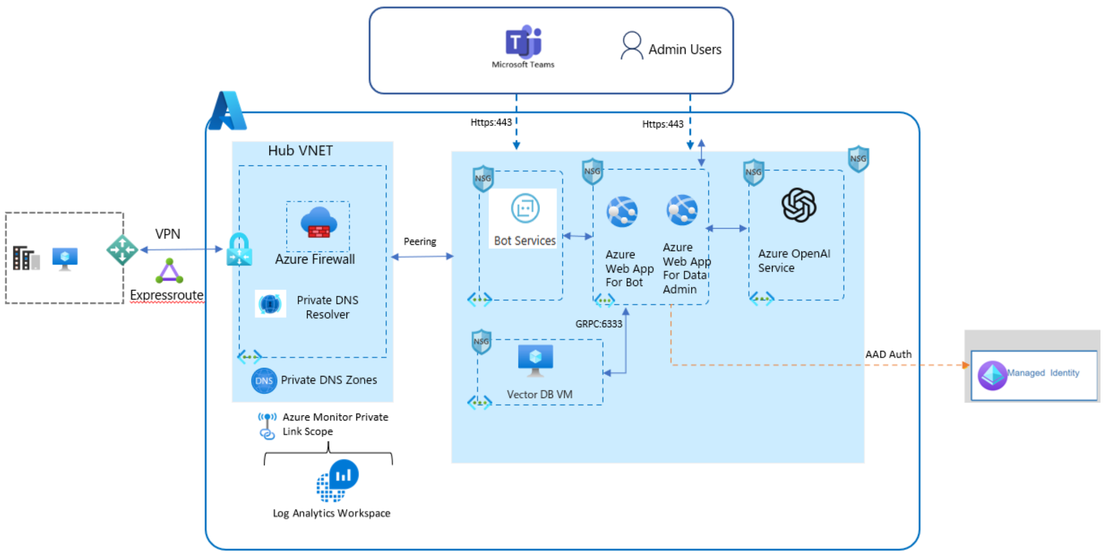
Intro:

This project demonstrates how to build a ChatGPT-like chatbot using Embedding with user's own data. The project features two different styles of chatbots: the Bot page is based on the Bot Framework SDK,supporting applications like Teams and Facebook; the Chat page is a standard web chat application. This project is based on the following Azure services:

Azure OpenAI, Azure Virtual Machine (for deploying qdrant vector database) Azure Web App , Azure Bot service. After deployment, you can upload PDF documents containing user data through the website, and then chat with the chatbot immediately. Below is the deployment documentation for this project. For the operation of the front-end and back-end, please refer to the usage instructions in the corresponding directories.

The following is a architect for this pilot:



Deployment Document

This deployment document will guide you through the following tasks:

Create the required Azure services
Install Qdrant vector database
Deploy React frontend
Install .NET backend

1. Create the required Azure services

Create the following services in the Azure portal:

Azure OpenAI service
Azure Bot service
Azure Web Apps

a. Create Azure OpenAI service

Log in to the Azure portal.
Click on + Create a resource in the left menu bar.
Enter "OpenAI" in the search box and select OpenAI service.
Click Create.
Fill in the following information:
Subscription: Select your Azure subscription.
Resource group: Select an existing resource group or create a new one.
Name: Enter the service name.
Location: Select the region where the service is located.
Pricing tier: Select the pricing tier that suits your needs.
Click Review + create, then click Create.
You need to create the following models in Model Deployments:
text-embedding-ada-002
gpt-35-turbo-16k
text-davinci-003(replaced by gpt-35-trubo)

b. Create Azure Bot service

Log in to the Azure portal.
Click on + Create a resource in the left menu bar.
Enter "Bot" in the search box and select Azure Bot.
Click Create.
Fill in the following information:
Subscription: Select your Azure subscription.
Resource group: Select an existing resource group or create a new one.
Name: Enter the Bot name.
Location: Select the region where the service is located.
Once you created the azure bot service, then go to the bot service
Select Bot Profile, fill in "Display Name" which you want to show in the team channel
Select Channel, click Microsoft Team as the channel
Select configuration, write down "Microsoft App ID", click Manage Passwrod", then select "Add New Client Secret" and get the Value when it's created.
After creation, you need to configure the Bot service settings to obtain the Microsoft App ID and password, and add the Directline channel to obtain the Directline token and secret. These will be used in subsequent deployment steps.
Select

c. Create Azure Web App

Log in to the Azure portal.

Click on + Create a resource in the left menu bar.
Enter “Web App” in the search box and select Web App
Click Create
Fill in the following information:
Subscription: Select your Azure subscription.
Resource group: Select an existing resource group or create a new one.
Name: Enter the Web App name.
Publish: Code
Runtime Stack: .Net 7(STS)
Operating System: Windows
Region: East Asia

We need to create two web app, one is for Bot applications and works as backend, another is for data management as frontend
The same way to create another web app. Except the Runtime stack needs to be selected as:
Runtime stack: Node 18
For data management web application

2. Install Qdrant vector database

a. Create a Linux virtual machine on Azure

Log in to the Azure portal.
Click on + Create a resource in the left menu bar.
Enter "Linux virtual machine" in the search box and select Linux virtual machine.
Click Create.
Fill in the following information:
Subscription: Select your Azure subscription.
Resource group: Select an existing resource group or create a new one.
Virtual machine name: Enter the virtual machine name.
Region: Select the region where the virtual machine is located.
Image: Select the desired Linux distribution.
Size: Select the virtual machine size that suits your needs.
Administrator account: Enter the administrator username and password or SSH public key.
In the Networking tab, ensure that a public IP address is assigned to the virtual machine and set the DNS name.
In the Networking tab, select Advanced for the NIC network security group, and then add a new inbound port rule to open port 6333 on Azure.
Click Review + create, then click Create.

b. Log in to the virtual machine and install Docker

Connect to your virtual machine using SSH. You can use the following command:
ssh <username>@<vm_public_ip>
Make sure to replace <username> with your administrator username and <vm_public_ip>

with the public IP address of the virtual machine. Enter the password to access the Linux environment.
Install Docker:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

c. Start Docker

```
sudo systemctl start docker
```

d. Load Qdrant image

```
sudo docker pull qdrant/qdrant
```

e. Run Qdrant container

Create a directory on the virtual machine to store Qdrant data, such as /var/lib/qdr:

```
mkdir -p /path/to/data
```

Run the Qdrant container:

```
sudo docker run -p 6333:6333 -v /path/to/data:/qdrant/storage qdrant/qdrant
```

Now, the Qdrant vector database is successfully installed and running on port 6333 of the Azure virtual machine. If you want to automatically run the service when the virtual machine restarts, you need to create a service with the following steps:

Create a new Systemd service file:

```
sudo nano /etc/systemd/system/qdrant.service
```

Paste the following content into the file, making sure to replace /path/to/data with the actual path:

```
[Unit]
Description=Qdrant Docker Service
After=docker.service
Requires=docker.service

[Service]
Type=simple
ExecStart=/usr/bin/docker run -p 6333:6333 -v /path/to/data:/qdrant/storage qdrant/qdrant
ExecStop=/usr/bin/docker stop qdrant
Restart=always

[Install]
WantedBy=multi-user.target
```

Save and exit the file. Then run the following commands to enable and start the service:

```
sudo systemctl enable qdrant.service
sudo systemctl start qdrant.service
```

Check the service status:

sudo systemctl status qdrant.service
If the service is running, you will see output similar to the following:

```
● qdrant.service - Qdrant Docker Service
   Loaded: loaded (/etc/systemd/system/qdrant.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-11-22 12:34:56 UTC; 1min 5s ago
 Main PID: 12345 (docker)
    Tasks: 15 (limit: 4915)
   CGroup: /system.slice/qdrant.service
           └─12345 /usr/bin/docker run -p 6333:6333 -v /path/to/data:/qdrant/storage
qdrant/qdrant
```

3. Deploy React frontend

a. Install Node.js

Visit the Node.js official website.
Download the latest LTS version of the Node.js installer for your operating system.
Run the installer and follow the prompts to complete the Node.js installation.

4. Install .NET backend

a. Install ASP.NET Core Runtime 7.0

Visit the .NET official website.
Download the ASP.NET Core Runtime installer for your operating system.
Run the installer and follow the prompts to complete the ASP.NET Core Runtime installation.

b. Build the production version:

```
npm run build
```

c. Deploy to Azure Web Service

deploying to Azure Web Service, install the Azure CLI, compress the "build" folder to "build.zip", and run the following command:

```
az webapp deployment source config-zip --src "<path/to/build.zip>" --name
<MyWebApp> --resource-group <MyResourceGroup>
for example:
```

Frontend> az webapp deployment source config-zip --src "./build.zip" --name webadminforteambot --resource-group OpenAI-Pilot6

b. Set Bot-related parameters

Configure the Messaging endpoint in Azure Bot to https://<backend_server_name>/api/messages.
Configure the client_id, secret_key (create one if not available), and Directline secret obtained from Azure Bot in appsettings.json's MicrosoftAppId, MicrosoftAppPassword, and MicrosoftDirectlineSecret.

c. Configure SearchClientSettings, BlobStorageClientSettings, and AzureOpenAIClientSettings in appsettings.json

Open the appsettings.json file.
Update the following settings with the relevant information obtained from the Azure portal:

```
{
   "SearchClientSettings": {
      "ServiceName": "<your_azure_search_service_name>",
      "ApiKey": "<your_azure_search_api_key>"
   },

   "AzureOpenAIClientSettings": {
      "ApiKey": "<your_azure_openai_api_key>"
   },
"QdrantEndpoint": "<your_qdrant database address>",
"MicrosoftAppId": "<your bot application Id>",
      "MicrosoftAppPassword": "<you bot secret",
"MicrosoftDirectlineSecret": "<directline secret>",
}
```

d. Run .NET backend

1. Open the command prompt or terminal.
2. Navigate to your .NET project directory:
3. `cd /path/to/your/dotnet/project`
4. Development environment: Run the following command directly:
5. `dotnet run`
6. Production environment: Execute the following command to publish:

```
dotnet publish --configuration Release --output ./publish
```

e. Deploy to VM or Azure Web Service

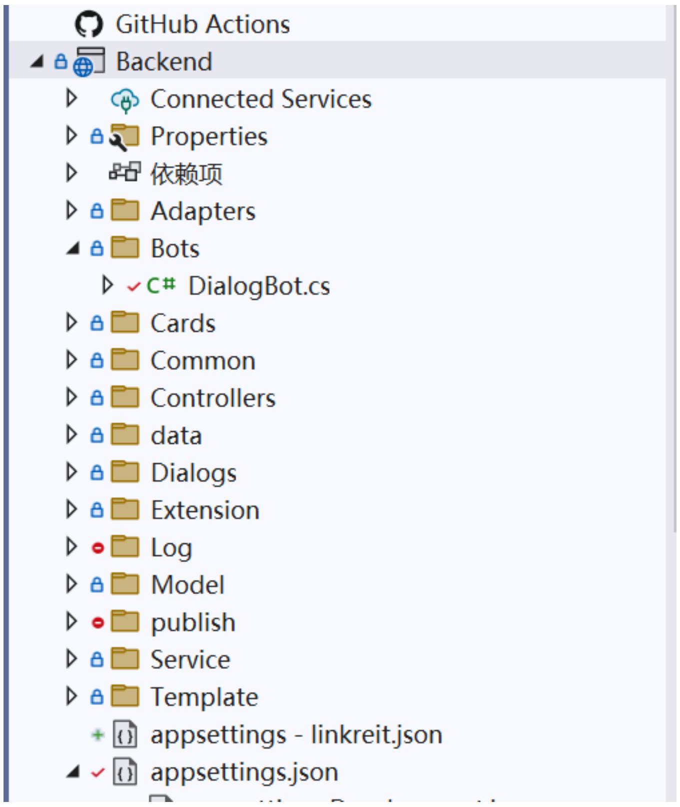
1. If deploying to Azure Web Service, install the Azure CLI, compress the "publish" folder to "publish.zip", and run the following command:

```
az webapp deployment source config-zip --src "<path/to/publish.zip>" --name
<MyWebApp> --resource-group  <MyResourceGroup>

for example:
```

Backend>az webapp deployment source config-zip --src "./publish.zip" --name teamschatbotforlinkreit --resource-group OpenAI-Pilot6

5. Code structure

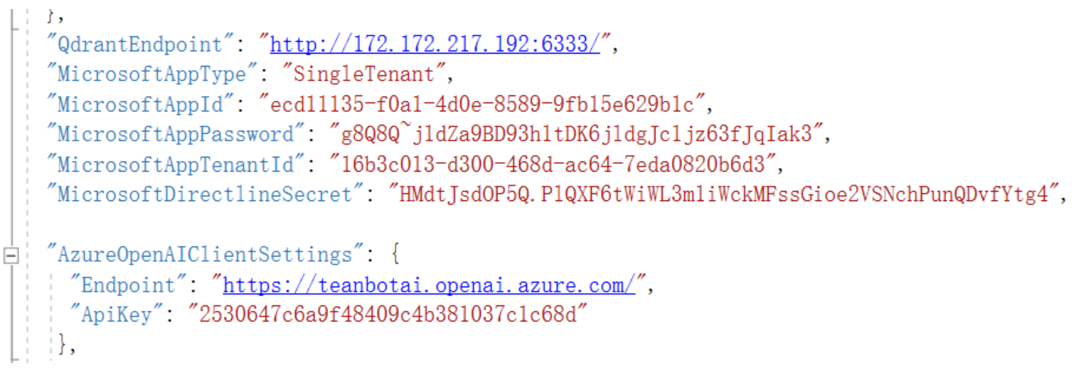


```
GitHub Actions
◢ Backend
    ▷ Connected Services
    ▷ Properties
    ▷ 依赖项
    ▷ Adapters
    ◢ Bots
        ▷ DialogBot.cs
    ▷ Cards
    ▷ Common
    ▷ Controllers
    ▷ data
    ▷ Dialogs
    ▷ Extension
    ▷ Log
    ▷ Model
    ▷ publish
    ▷ Service
    ▷ Template
      appsettings - linkreit.json
    ◢ appsettings.json
```

Edit appsettings.json and fill in the parameters for azure services created:

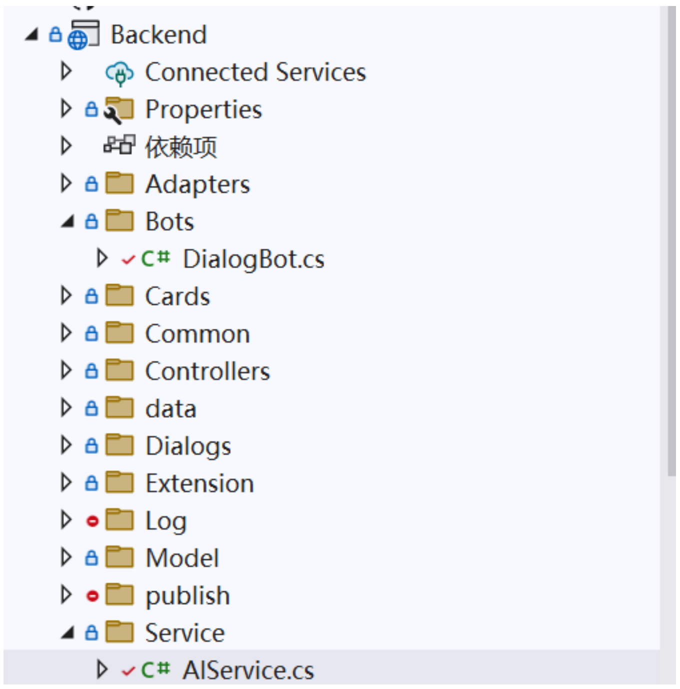
QdrantEndpoint: QdrantEndpoint VM address and port
Bot services information:

MicrosoftAppID, MicrosoftAppPawwsord, MicrosoftAppTenanID, MicrosoftDirectlineSecret
Azure OpenAI Endpoing and APIkey



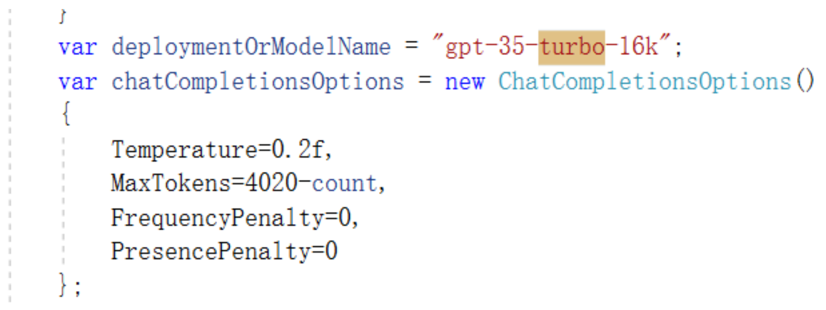
```
},
"QdrantEndpoint": "http://172.172.217.192:6333/",
"MicrosoftAppType": "SingleTenant",
"MicrosoftAppId": "ecd11135-f0a1-4d0e-8589-9fb15e629b1c",
"MicrosoftAppPassword": "g8Q8Q~j1dZa9BD93h1tDK6j1dgJc1jz63fJqIak3",
"MicrosoftAppTenantId": "16b3c013-d300-468d-ac64-7eda0820b6d3",
"MicrosoftDirectlineSecret": "HMdtJsdOP5Q.P1QXF6tWiWL3m1iWckMFssGioe2VSNchPunQDvfYtg4",

"AzureOpenAIClientSettings": {
  "Endpoint": "https://teanbotai.openai.azure.com/",
  "ApiKey": "2530647c6a9f48409c4b381037c1c68d"
},
```



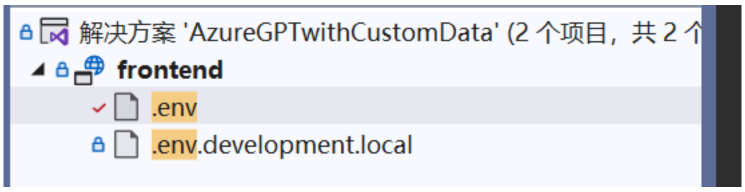
```
Backend
  Connected Services
  Properties
  依赖项
  Adapters
  Bots
    DialogBot.cs
  Cards
  Common
  Controllers
  data
  Dialogs
  Extension
  Log
  Model
  publish
  Service
    AIService.cs
```

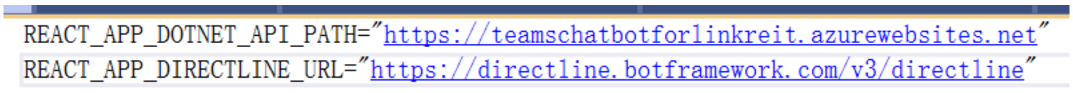
AIService.cs defines the parameters about OpenAI, such as deployment model, Temperature etc.



```
}
var deploymentOrModelName = "gpt-35-turbo-16k";
var chatCompletionsOptions = new ChatCompletionsOptions()
{
    Temperature=0.2f,
    MaxTokens=4020-count,
    FrequencyPenalty=0,
    PresencePenalty=0
};
```

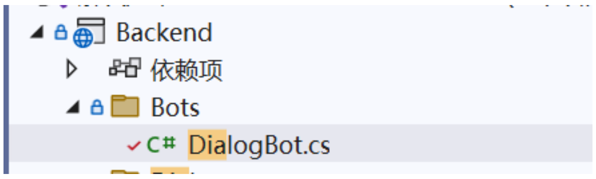
Before you deploy the web application to azure web app, define the backend web domain in .env file:

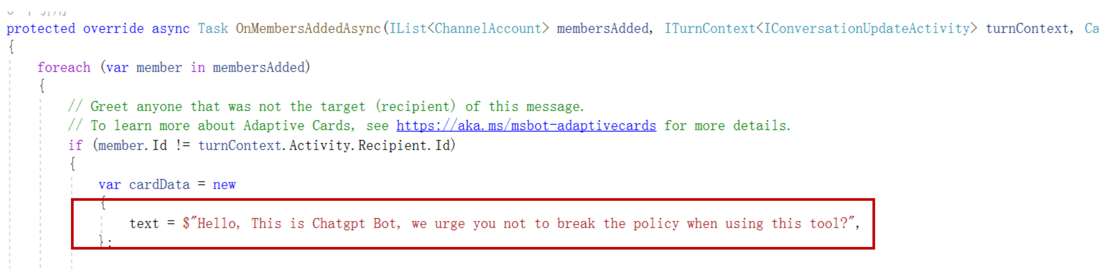




```
REACT_APP_DOTNET_API_PATH="https://teamschatbotforlinkreit.azurewebsites.net"
REACT_APP_DIRECTLINE_URL="https://directline.botframework.com/v3/directline"
```

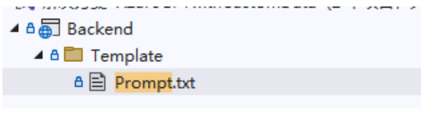
If you want to change the Prompt or welcome message of Bot, you can change the DialogBot.cs file





```
protected override async Task OnMembersAddedAsync(IList<ChannelAccount> membersAdded, ITurnContext<IConversationUpdateActivity> turnContext, Ca
{
    foreach (var member in membersAdded)
    {
        // Greet anyone that was not the target (recipient) of this message.
        // To learn more about Adaptive Cards, see https://aka.ms/msbot-adaptivecards for more details.
        if (member.Id != turnContext.Activity.Recipient.Id)
        {
            var cardData = new
            {
                text = $"Hello, This is Chatgpt Bot, we urge you not to break the policy when using this tool?",
            };
```

After that, you need redeploy the backend web app.



This define the prompt used for openai.