

# 雲端計算期末報告

組別：第八組

系級：資訊四甲

組員：陳柏宇、許仕弦、吳添聖

# 目錄

User Story .....	3
提供文本分析服務 .....	3
提供翻譯服務 .....	3
提供 OpenAI 中 Chatgpt 的服務 .....	3
System Architecture .....	3
架構圖 .....	4
Step 1 .....	4
Step 2 .....	5
Step 3 .....	5
Step 4 .....	7
Step 5 .....	8
Step 6 .....	11
Step 7 .....	13
Step 8 .....	14
Step 9 .....	14
Cloud Properties.....	15
Feedbacks.....	15

# User Story

在我們的 cloud application 中，只要點擊網址即可使用，無需任何複雜操作。  
功能如下：

## 提供文本分析服務

可以分析文章中的語言、語氣。

可以擷取出關鍵字。

將文章中的詞分類成：日期、時間、地點、產品和形容詞。

## 提供翻譯服務

可以選擇多個語言翻譯

## 提供 OpenAI 中 Chatgpt 的服務

輸入任何問題，透過 Chatgpt 的輸出，顯示在結果畫面中。

使用者只需要將想分析的文章、想翻譯的文字和想詢問的事情，輸入在首頁的文字框框中，並點擊分析、翻譯（如需要翻譯，可以先選擇好欲翻譯成的語言）和詢問。

以下為 Demo 影片連結與 QRCode：

<https://youtu.be/mA2lDD9SqYA>



以下為實際網站之網址與 QRCode（只有在 Azure App Services 有啟動時才會運作）：

<https://team8-final-project.azurewebsites.net>



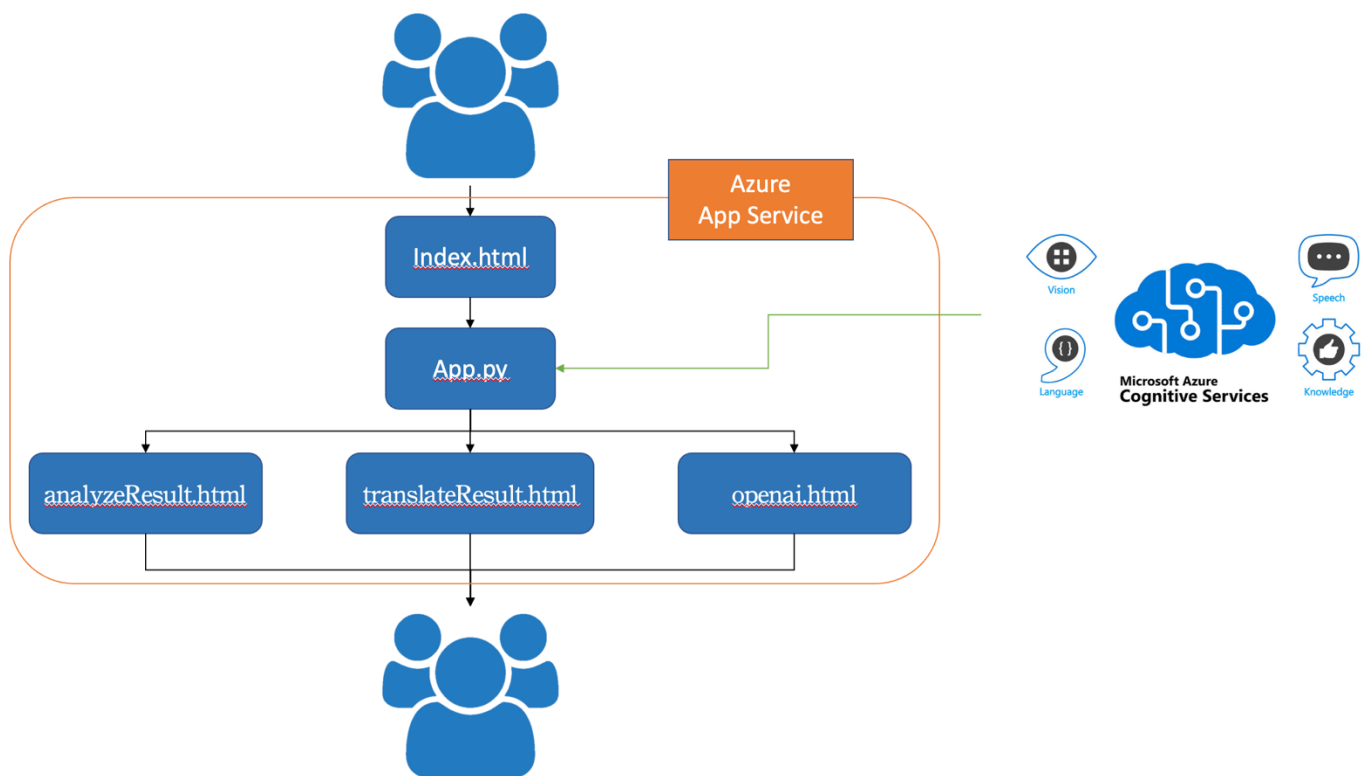
# System Architecture

在系統架構中，我們透過 Flask 來建構一個 Web Application，並且使用 Azure App Service 來將其放到網頁上提供任何人使用。

Web Application 包含 app.py、analyzeResult.html、index.html、openai.html、translateResult.html 和.env。在此 Web Application 中架構出一個虛擬的 python 環境，並在.env 中輸入 Azure Cognitive service 的金鑰、端點與區域，來為翻譯與文本分析做準備。

## 架構圖

架構圖中，讓程式運作在 Azure 的 App Service，使用者可以直接透過網址訪問網站，進行操作。取得使用者的請求之後，分辨需求，給予相對應的輸出。如是文本分析與文字翻譯，則會透過 Azure 的 Cognitive Service，取得各自的結果並呈現給使用者。如是詢問 OpenAI，則會透過 API 來將使用者的輸入傳送給 OpenAI，並取得回應後，顯示在結果畫面中。



## Step 1

首先，需要下載 python，在 VS Code 中建立一個專案目錄，建構 python 虛擬環境。

# Windows

# Create the environment

```
python -m venv venv
```

# Activate the environment

```
.\venv\scripts\activate
```

```
# macOS or Linux
# Create the environment
python -m venv venv
# Activate the environment
source ./venv/bin/activate
```

## Step 2

新增一個 requirements.txt，並加入以下：

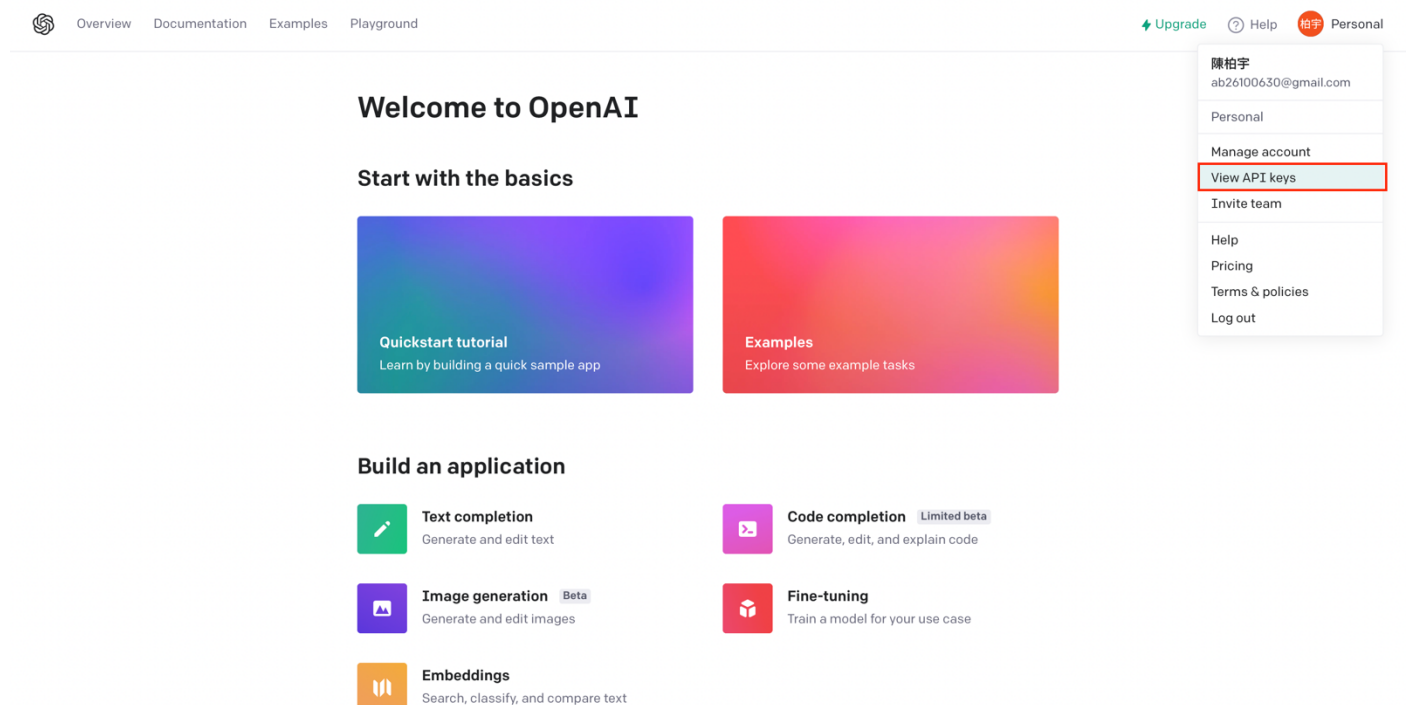
```
flask
python-dotenv
requests
openai
azure-core
azure-ai-textanalytics
```

在 terminal 中輸入：

```
pip install -r requirements.txt
```

## Step 3

註冊 OpenAI，取得自己的 API\_KEY，將其複製到下方 'YOU\_API\_KEY'。



The screenshot shows the OpenAI website interface. At the top, there's a navigation bar with links for Overview, Documentation, Examples, and Playground. On the right, there's a user profile section with a dropdown menu. The dropdown menu is open, showing options like Personal, Manage account, View API keys (highlighted with a red box), Invite team, Help, Pricing, Terms & policies, and Log out. The main content area has a 'Welcome to OpenAI' heading, followed by 'Start with the basics' which includes 'Quickstart tutorial' and 'Examples'. Below that is 'Build an application' with options for Text completion, Code completion, Image generation, Fine-tuning, and Embeddings.

先新增一個檔案，命名為 app.py，加入以下程式碼。

```
from flask import Flask, redirect, url_for, request, render_template, session
from dotenv import load_dotenv
import os
```

```

import requests,json
import openai
openai.api_key = 'YOU_API_KEY'
# Import namespaces
from azure.core.credentials import AzureKeyCredential
from azure.ai.textanalytics import TextAnalyticsClient

app = Flask(__name__)
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

```

接者新增一個名為 templates 的資料集，新增 index.html，加入以下程式碼。

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0j lfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
    <title>Text Analysis</title>
</head>
<body>
    <div class="container">
        <h1>Text Analysis service</h1>
        <div>
            <form method="POST">
                <div class="form-group">
                    <textarea name="text" cols="20" rows="10" class="form-control"></textarea>
                </div>
                <div class="form-group">
                    <label for="language">Choose Language if need translate:</label>
                    <select name="language" class="form-control">
                        <option value="en">English</option>
                        <option value="it">Italian</option>
                        <option value="ja">Japanese</option>
                        <option value="ru">Russian</option>
                        <option value="de">German</option>
                        <option value="zh-Hant">Chinese</option>
                        <option value="fr">French</option>

```

```

        </select>
    </div>
    <div>
        <button type="submit" class="btn btn-success" name="type"
value="analyze">Analyze!</button>
        <button type="submit" class="btn btn-success" name="type"
value="translate">Translate!</button>
        <button type="submit" class="btn btn-success" name="type"
value="ask">Ask!</button>
    </div>
</form>
</div>
</div>
</body>
</html>

```

此時，可以試著測試程式碼是否正常運行，輸入：

# Windows

```
set FLASK_ENV=development
```

# Linux/macOS

```
export FLASK_ENV=development
```

接著輸入：

```
flask run
```

連線到 terminal 顯示的網誌查看，應該可以看到以下樣式：

## Text Analysis service

Choose Language if need translate:

English
⌵

Analyze!
Translate!
Ask!

## Step 4

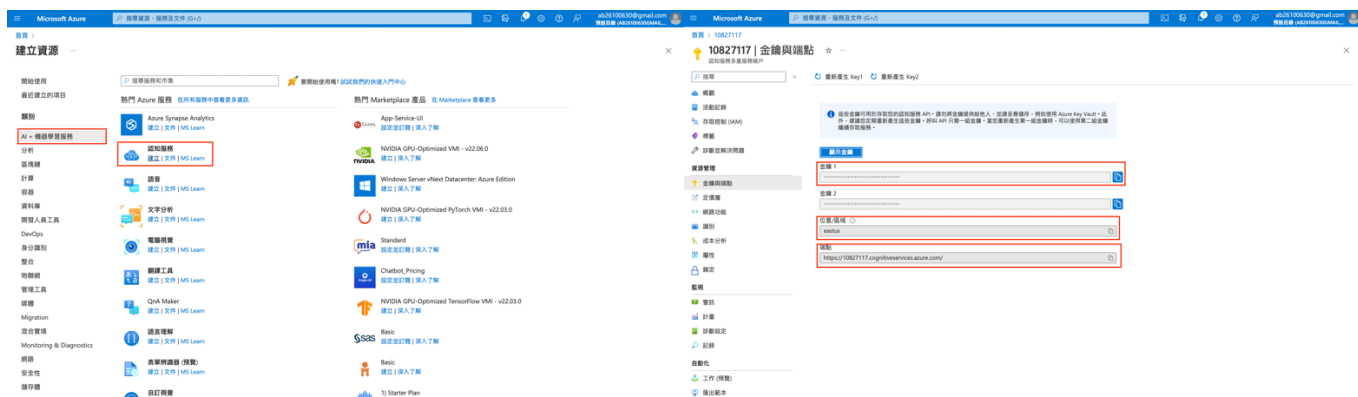
建立新檔案，命名為.env，數入以下程式碼。

```

COG_SERVICE_ENDPOINT=
COG_SERVICE_KEY=
COG_SERVICE_REGION=

```

建立一個認知服務，複製其 key、endpoint 和 region。



將紅色框框三項分別複製到.env 中。

## Step 5

完成 app.py 中的程式碼，輸入以下：

```
@app.route('/', methods=['POST'])
def index_post():
    if request.form['type'] == "analyze":
        return analyzePage()
    elif request.form['type'] == 'translate':
        return translatePage()
    elif request.form['type'] == 'ask':
        return chatgptPage()

def chatgptPage() :
    msg = request.form['text']
    response = openai.Completion.create(
        engine='text-davinci-003',
        prompt=msg,
        max_tokens=3999,
        temperature=0.5
    )
    completed_text = response['choices'][0]['text']
    return render_template('openai.html',
                           original=msg,
                           response=completed_text)

def analyzePage():
    lan = ""
    sent = ""
    phrases_list = []
    entities_list = []
```



```

link_list = []
try:      # Get Configuration Settings
    load_dotenv()
    cog_endpoint = os.getenv('COG_SERVICE_ENDPOINT')
    cog_key = os.getenv('COG_SERVICE_KEY')
    # Create client using endpoint and key
    credential = AzureKeyCredential(cog_key)
    cog_client = TextAnalyticsClient(
        endpoint=cog_endpoint, credential=credential)
    text = request.form['text']
    # Get language
    detectedLanguage = cog_client.detect_language(documents=[text])[0]
    lan = '\n{}'.format(detectedLanguage.primary_language.name)
    # # Get sentiment
    sentimentAnalysis = cog_client.analyze_sentiment(documents=[text])[0]
    # print("\nSentiment: {}".format(sentimentAnalysis.sentiment))
    sent = "\n{}".format(sentimentAnalysis.sentiment)
    # # Get key phrases
    phrases = cog_client.extract_key_phrases(documents=[text])[
        0].key_phrases
    if len(phrases) > 0:
        # print("\nKey Phrases:")
        for phrase in phrases:
            # print('\t{}'.format(phrase))
            phrases_list.append('\t{}'.format(phrase))
    # # Get entities
    entities = cog_client.recognize_entities(documents=[text])[0].entities
    if len(entities) > 0:
        for entity in entities:
            entities_list.append('\t{} ({} )'.format(
                entity.text, entity.category))
    # # Get linked entities
    entities = cog_client.recognize_linked_entities(documents=[text])[
        0].entities
    if len(entities) > 0:
        for linked_entity in entities:
            link_list.append('\t{} ({} )'.format(
                linked_entity.name, linked_entity.url))
    return render_template('analyzeResult.html',
        original=text,
        language=lan,
        sentiment=sent,

```

```

        phrases=phrases_list,
        entities=entities_list,
        link=link_list)

except Exception as ex:
    print(ex)

def translatePage():
    try:
        # Get Configuration Settings
        load_dotenv()
        cog_key = os.getenv('COG_SERVICE_KEY')
        cog_region = os.getenv('COG_SERVICE_REGION')
        translator_endpoint = 'https://api.cognitive.microsofttranslator.com'
        text = request.form['text']

        def GetLanguage(text):
            # Default language is English
            language = 'en'

            # Use the Translator detect function
            # Use the Translator detect function
            path = '/detect'
            url = translator_endpoint + path
            # Build the request
            params = {
                'api-version': '3.0'
            }
            headers = {
                'Ocp-Apim-Subscription-Key': cog_key,
                'Ocp-Apim-Subscription-Region': cog_region,
                'Content-type': 'application/json'
            }
            body = [{
                'text': text
            }]
            # Send the request and get response
            request = requests.post(
                url, params=params, headers=headers, json=body)
            response = request.json()
            # Parse JSON array and get language
            language = response[0]["language"]
            # Return the language
            return language

        def Translate(text, source_language, target_lang):

```

```

translation = ''
# Use the Translator translate function
# Use the Translator translate function
path = '/translate'
url = translator_endpoint + path
# Build the request
params = {
    'api-version': '3.0',
    'from': source_language,
    'to': [target_lan]
}
headers = {
    'Ocp-Apim-Subscription-Key': cog_key,
    'Ocp-Apim-Subscription-Region': cog_region,
    'Content-type': 'application/json'
}
body = [{
    'text': text
}]
# Send the request and get response
request = requests.post(
    url, params=params, headers=headers, json=body)
response = request.json()
# Parse JSON array and get translation
translation = response[0]["translations"][0]["text"]
# Return the translation
return translation

# Get language
lan = GetLanguage(text)
target_lan = request.form['language']
# Translate if not already English
translatedText = Translate(text, lan, target_lan)
return render_template('translateResult.html',
                       original=text,
                       language=lan,
                       translated=translatedText)

except Exception as ex:
    print(ex)

```

## Step 6

在 templates 中新增檔案，命名為 analyzeResult.html，加入以下程式碼：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
  integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkGIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
  <title>Result</title>
</head>
<body>
  <div class="container">
    <h2>Results</h2>
    <div>
      <strong>Original text:</strong> {{ original }}
    </div>
    <div>
      <strong>Language:</strong> {{ language }}
    </div>
    <div>
      <strong>Sentiment:</strong> {{ sentiment }}
    </div>
    <div>
      <strong>Phrases:</strong><br>
      {% for p in phrases %}
        {{p}}<br>
      {% endfor %}
      <br>
    </div>
    <div>
      <strong>Entities:</strong><br>
      {% for e in entities %}
        {{e}}<br>
      {% endfor %}
      <br>
    </div>
    <div>
      <strong>Link:</strong><br>
      {% for l in link %}
        {{l}}<br>
      {% endfor %}
    </div>
  </div>
</body>
</html>
```

```

        <br>
    </div>
    <div>
        <a href="{{ url_for('index') }}">Try another one!</a>
    </div>
</div>
</body>
</html>

```

## Step 7

在 templates 中新增檔案，命名為 translateResult.html，加入以下程式碼：

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
        integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0j lfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
    <title>Result</title>
</head>
<body>
    <div class="container">
        <h2>Results</h2>
        <div>
            <strong>Original text:</strong> {{ original }}
        </div>
        <div>
            <strong>Language:</strong> {{ language }}
        </div>
        <div>
            <strong>Translated text:</strong> {{ translated }}
        </div>
        <div>
            <a href="{{ url_for('index') }}">Try another one!</a>
        </div>
    </div>
</body>
</html>

```

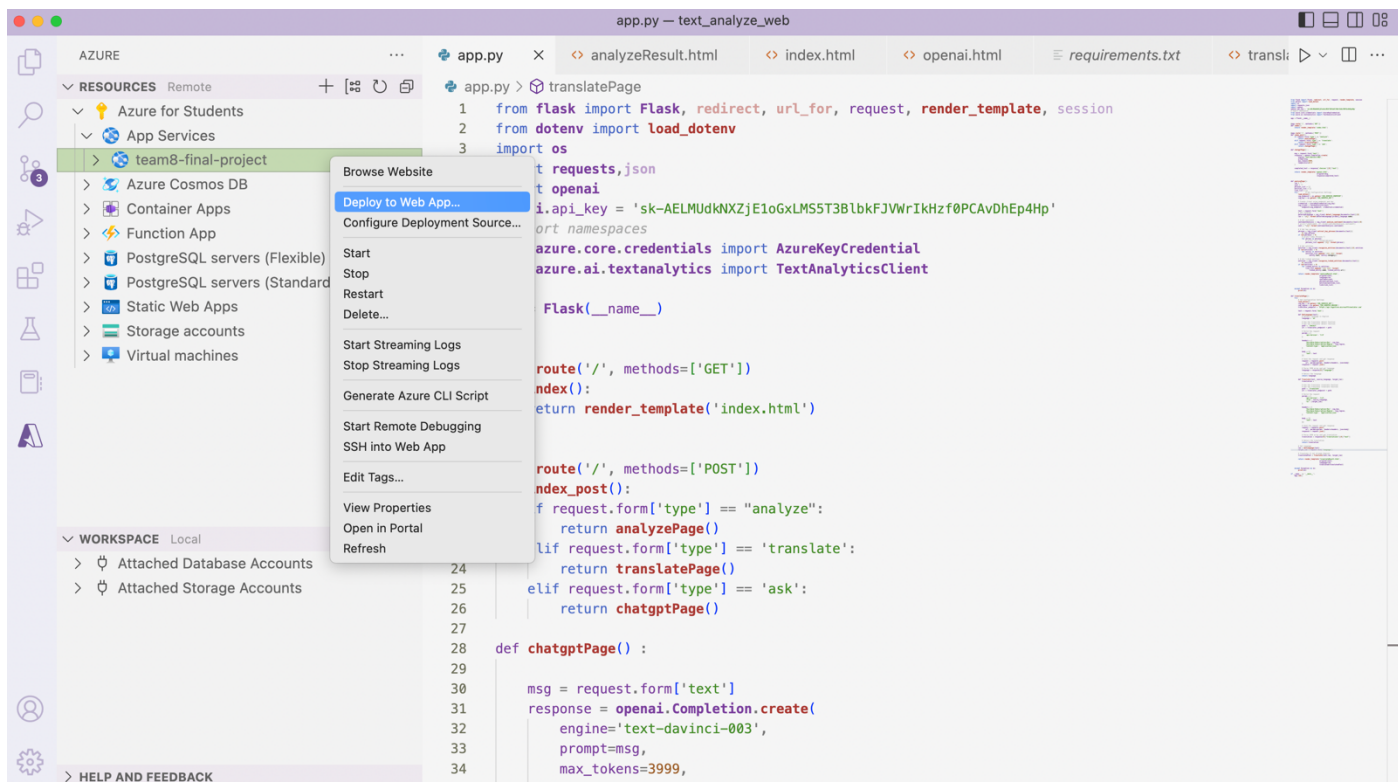
## Step 8

在 templates 中新增檔案，命名為 openai.html，加入以下程式碼：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
  integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
  <title>Result</title>
</head>
<body>
  <div class="container">
    <h2>Results</h2>
    <div>
      <strong>Original text:</strong> {{ original }}
    </div>
    <div>
      <strong>Response:</strong> {{ response }}
    </div>
    <div>
      <a href="{{ url_for('index') }}">Try another one!</a>
    </div>
  </div>
</body>
</html>
```

## Step 9

建立 App Service，將此資料夾 Deploy 到 App Service。



完成後即可點擊 Browse Website，並開始使用。

連結到網頁後，會看到一個文字輸入匡，在其中輸入使用者想輸入的文字，下方有三個可以點擊的案件，包括 Analyze、Translate 與 Ask，依照自己的需求來使用不同功能。

## Cloud Properties

在這一次的專案中，我們實作了 Paas 架構的 App service，將所有的功能、更新與網站架設全部實作出來，使用者只需要到網站上做操作即可。

## Feedbacks

Microsoft 提供了很棒的學習環境，服務也是非常齊全，可以根據使用者需求來使用。但是在部分的 Learning Path 上，似乎沒有設計好，隨著版本的更新，會出現無法相容的問題，還希望可以解決這部分的問題，除此之外並沒有其他的建議。