

**中原大學 雲端計算平台實務**

**Final Project -書面報告**

**Serverless Gamble Chatroom**

**資訊四乙 10727211 林彥輝**

**資訊四乙 10727216 李品毅**

**授課教師：鍾武君 教授**

**中華民國一一一年一月**

目錄

[壹、使用情境 3](#_Toc92379685)

[貳、系統架構 4](#_Toc92379686)

[一、資源使用及介紹 4](#_Toc92379687)

[1. Azure Function 4](#_Toc92379688)

[2. Azure CosmosDB 4](#_Toc92379689)

[3. Azure Storage Account - Queue Storage 4](#_Toc92379690)

[4. Azure SignalR 4](#_Toc92379691)

[5. Azure App Service 5](#_Toc92379692)

[6. Azure Container Registry 5](#_Toc92379693)

[二、系統架構圖 5](#_Toc92379694)

[三、成果展示 6](#_Toc92379695)

[參、雲端屬性 7](#_Toc92379696)

[肆、開發過程所遇問題 8](#_Toc92379697)

[一、Azure Environment 8](#_Toc92379698)

[二、SignalR 8](#_Toc92379699)

[三、Durable Function 8](#_Toc92379700)

[四、WebApp Extension 9](#_Toc92379701)

[五、Error – Create CosmosDB Container 9](#_Toc92379702)

[六、官方文件錯誤 9](#_Toc92379703)

[七、Encode問題 10](#_Toc92379704)

[伍、心得 11](#_Toc92379705)

[陸、參考資料 12](#_Toc92379706)

# 壹、使用情境

本專題基於Azure Function之無伺服器服務搭建網路聊天室，透過此聊天室可與其他使用者於線上進行多人即時對話，在普通對話之餘還有點數博弈機制，使用者可透過指令進行點數博弈以賺取點數。

此專題透過WebApp Service、Azure Function、SignalR、CosmosDB、Queue Storage來搭建此服務，使用者只須透過前端的WebApp下達對話，後端即可根據前端所輸入之對話進行運算，並將結果輸出回前端。

使用者可以透過四種方式使用聊天室，功能如下所述。

GM : “Hello World!”

1. Normal Chat

為正常訊息，對話將直接顯示於聊天視窗。

1. Initial Point

GM : !init

Bot : init successful

使用者輸入「!init」初始化點數，

初始化完成系統將同步發出訊息。

1. Check Point

GM : !init

Bot : GM's points is : 1000

使用者輸入「!point」發出要求查看點數，

系統將回覆該使用者所剩點數。

1. Gamble Point

使用者輸入「!gamble」下注金額，

GM : !gamble 10

bot : GM has won 10 points!

GM : !gamble 50%

bot : GM Failed!505 points!(σ′▽‵)′▽‵)σ

GM : !gamble All

bot : GM has won 505 points!

系統經過random()決定成功與否，

最後輸出最終結果。

# 貳、系統架構

## 一、資源使用及介紹



資源使用如下：

### Azure Function

Azure Function提供應用程式以Serverless的方式在雲端服務。Final Project使用Python開發許多Azure Function作為後端的計算資源，並使用不同的Trigger做觸發（HttpTrigger、CosmosDB Trigger[[1](#CosmosDB_Trigger)]、Queue Storage Trigger[[2](#Queue_Trigger)]），也將計算結果透過定義Output Binding輸出至後續的資源做下一階段的處理。

### Azure CosmosDB

Azure CosmosDB是一個NoSQL類型的資料庫，提供高擴充能力與低延遲的服務供使用者開發。Final Project中採用Azure CosmosDB做為Points的存儲中心，透過Azure Function與資料庫執行Insert、Query的動作[[3](#SQL_Query)]。

### Azure Storage Account - Queue Storage

Azure Storage Account包含各式的Storage服務（Blob、File Shares、Queues、Tables、Disks），Final Project中採用Queue Storage作為前端聊天室輸出的存儲，這些存放在Azure Storage中的Base64編碼訊息會透過定義Azure Function中的Queue Storage Trigger將訊息傳送至Azure Function進行運算[[4](#Queue_Storage_USE)]。

### Azure SignalR

Azure SignalR是一系列技術的抽象概念，提供WebSocket供Client、Server保有連線，Client就不須透過Pooling的方式定期向Server取得資料。Final Project使用SignalR做為後端與前端的訊號傳遞服務，這些資訊包含User、Bot的對話訊息以及CosmosDB Trigger的通知[[5](#SignalR)]。

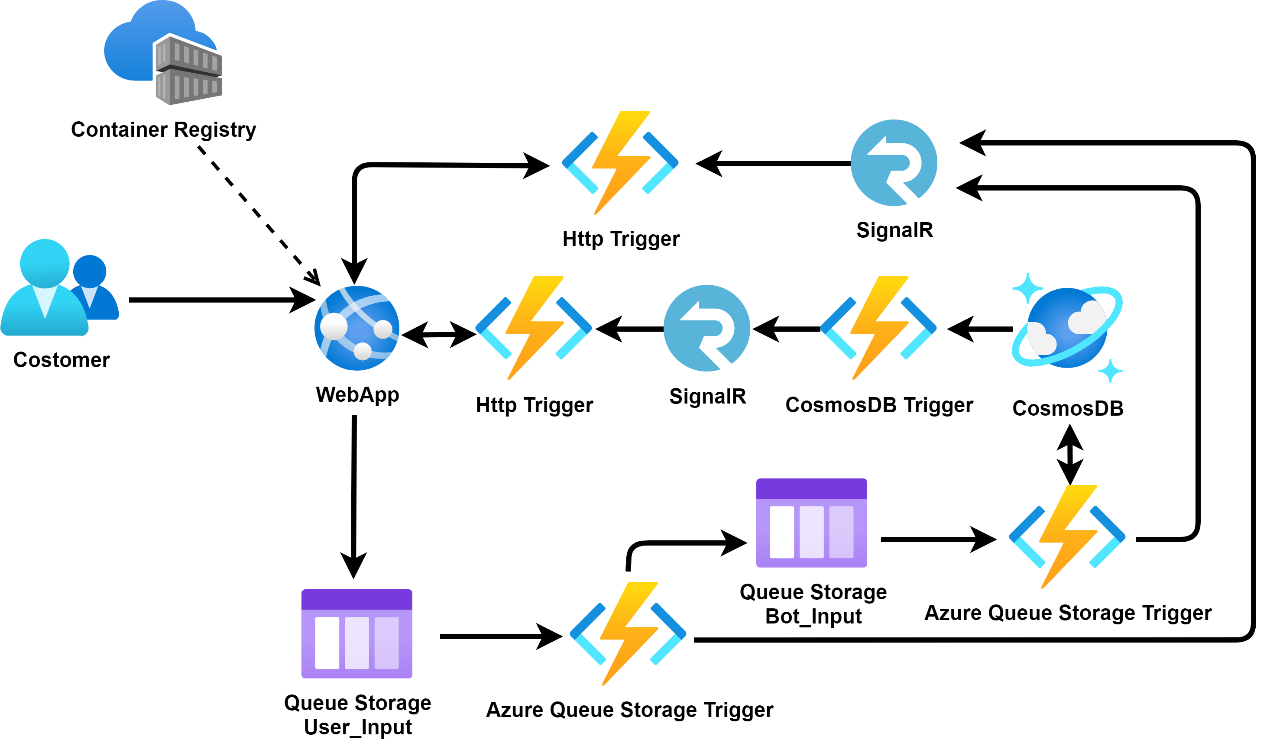
### Azure App Service

Azure App Service供Web App服務，並提供.Net、Java、Node.js、PHP、Python等語言。Final Project中使用Python的Flask做開發[[6](#Flask)]，並在網頁中加入JavaScript製作動態網頁及CSS做為美化[[7](#Bootstrap)]。在開發網頁時希望網頁中所有的元素都能透過非同步的方式進行開發，不須透過重新整理的方式進行更新，因此透過Google查詢到Ajax(Asynchronous JavaScript and XML，AJAX)的方法進行開發提供使用者更好的使用方式[[8](#ajax)]，同時排行榜的功能使用Fetch API抓取來自Flask的JSON資料[[9](#FetchAPI)]。

### Azure Container Registry

Azure Container Registry供映像檔之存儲。由於Azure App中的環境對於任何Azure Linux Base的開發環境皆不友善，難以增加extension，因此採用Docker Container做為Azure App的開發工具，維持開發環境的預期結果能與實際部署成果相同[[10](#Docker_AppService)][[11](#DockerHub)]。

## 二、系統架構圖



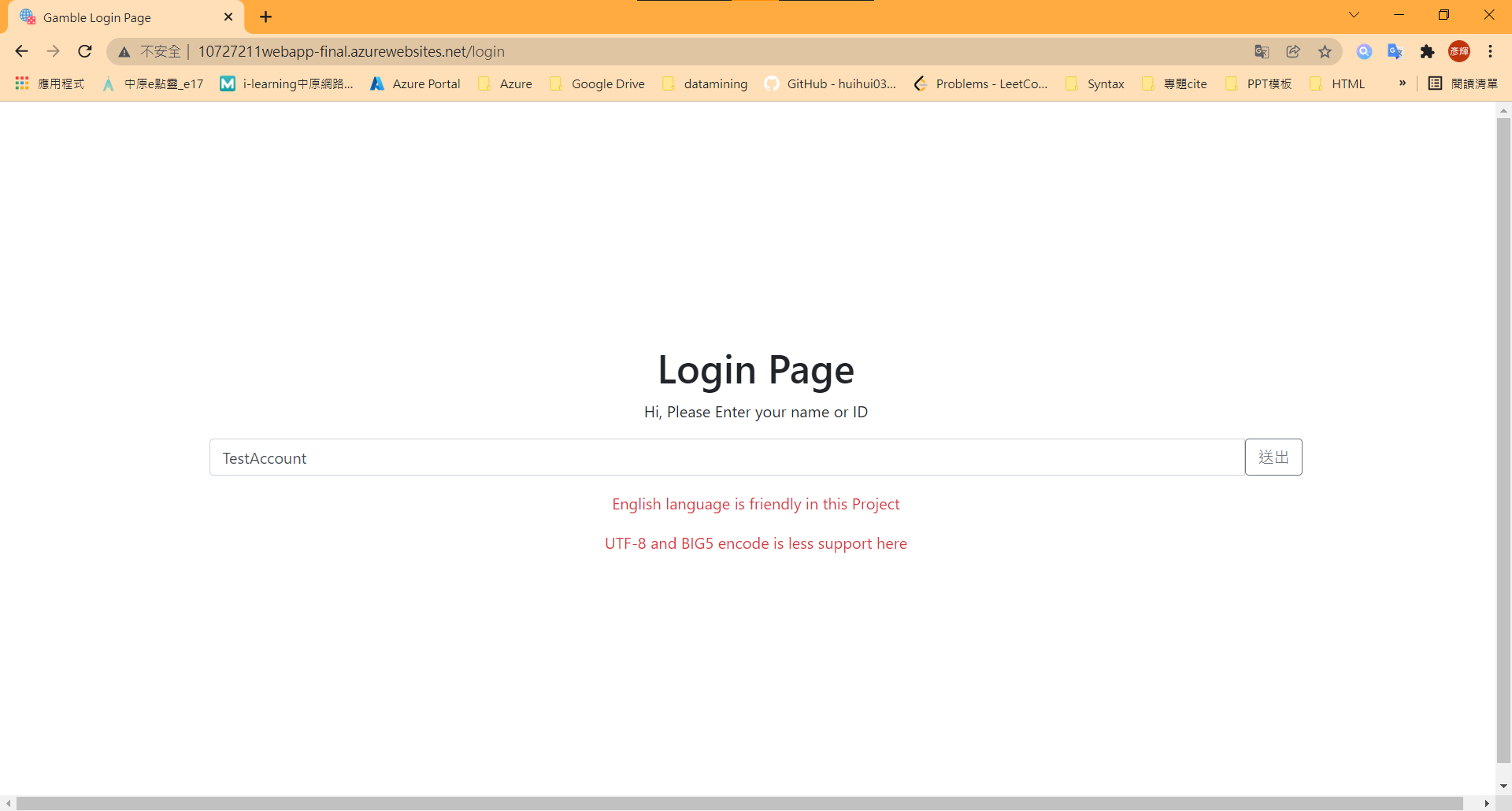
在後端部分，以Queue Storage做為起始點，前端會將使用者傳送至Queue Storage中，接著透過第一個Azure Function中定義Queue Storage接收使用者訊息，在第一個Azure Function中會判斷使用者訊息是否為指令，若是則將訊息傳至第二個Queue Storage做為Bot的訊息輸入進而觸發第二個Azure Function，在此Azure Function中會根據指令的需求與CosmosDB做連動，這些Azure Function執行完畢定義Output Binding將訊號輸出至SignalR。

SignalR再將訊息透過Azure Function的API將WebSocket傳送至Web，CosmosDB若有任何的更新也會透過CosmosDB Trigger將訊號透過一樣的方式傳送至Web。Web接收到這些訊號就會採用非同步的方式即時更新網頁的聊天室及排行榜，而非需要重新刷新頁面用輪詢的方式獲取最新的值。

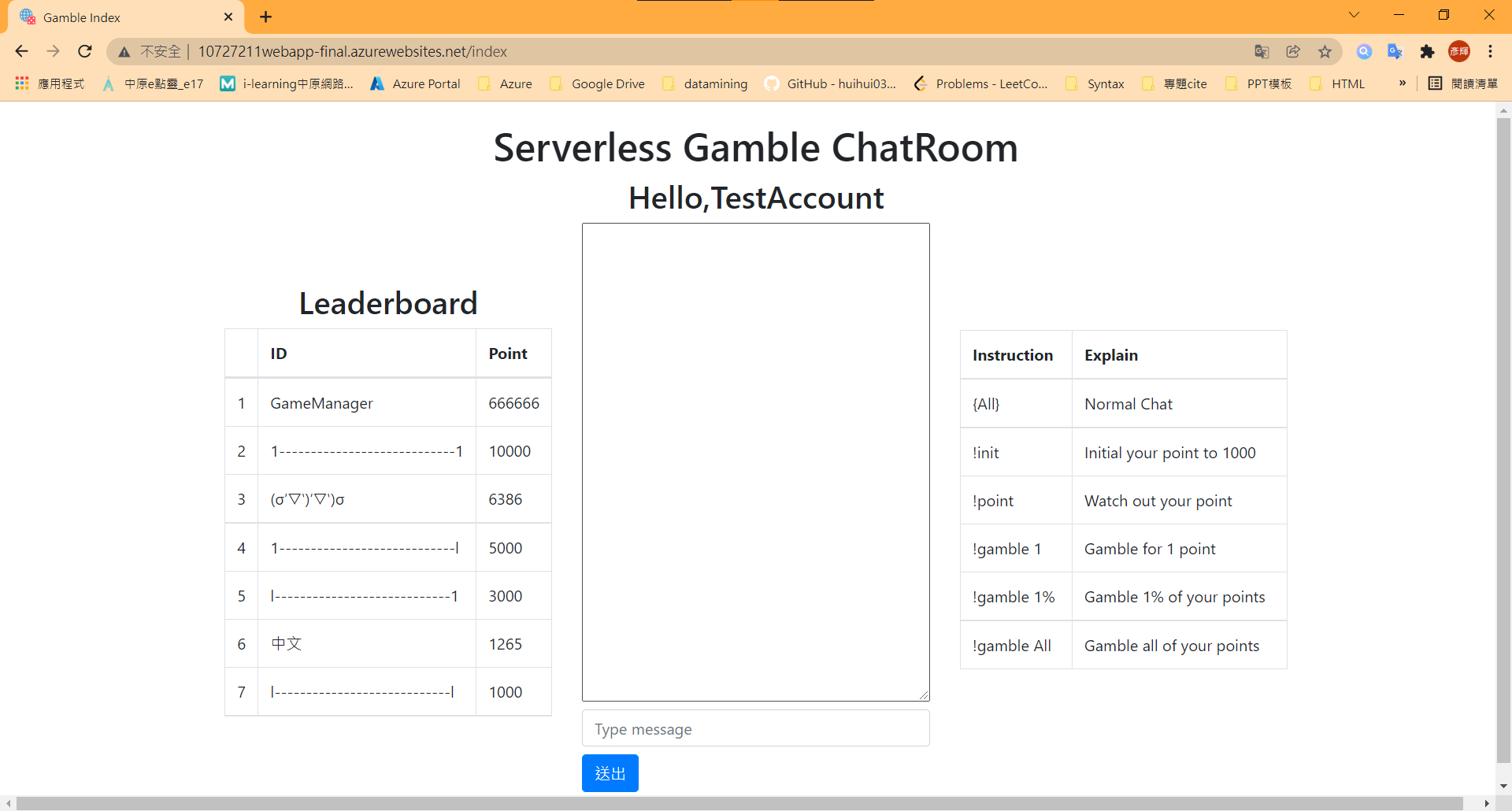
## 三、成果展示

網頁：<http://10727211webapp-final.azurewebsites.net>

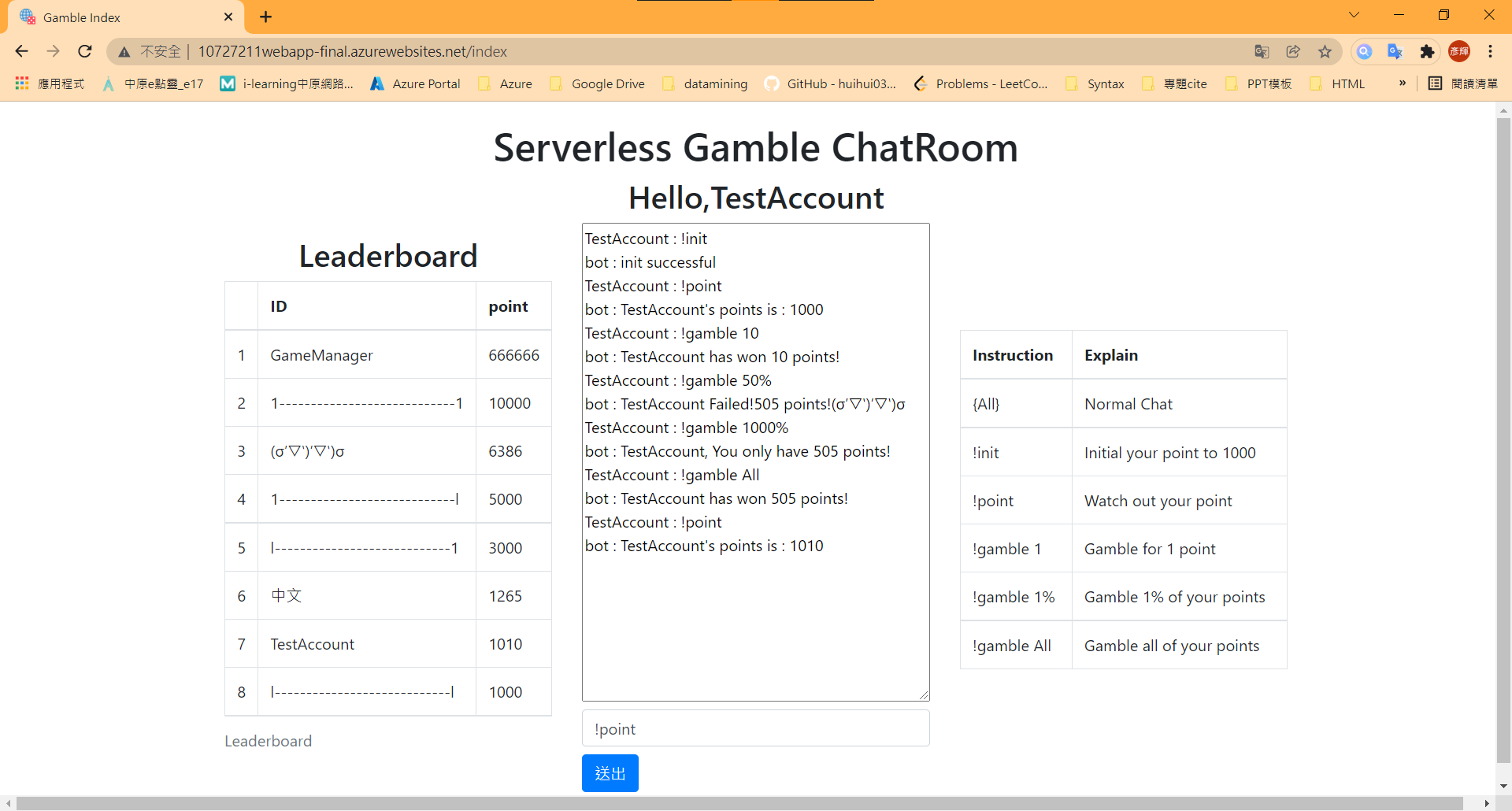
第一步：開啟網頁並要求使用者登記ID。



第二步：接著進到聊天室頁面，使用者可在中下方的區域進行文字輸入，結果會輸出至中間的文字區，左方為Point之排行榜，值會隨著資料庫更動而更新；右方為指令介紹區，提醒使用者可根據「!init、!point、!gamble」等相關指令使用point。



第三步：使用者在初始化之後即可接續使用「!point」、「!gamble」等指令與bot作互動，過程訊息皆會輸出至聊天室，互動過程隨著資料庫的更新，左方排行榜也會做即時更新。



# 參、雲端屬性

1. Availability & Reliability

Azure SLA實現高Availability，且由於Azure Function是部署在雲端上，任何基於Azure Function底層的機器擁有高可用性。透過Function as a Service，可以讓系統具備容錯率，具備Reliability的特性。

1. Performance Optimization

Azure Function、WebApp Service的基本特性包含Load-Balance，能自動調整後端設備運算量，因此皆可達到High Performance & Improvement的特性。

1. Accessibility Portability

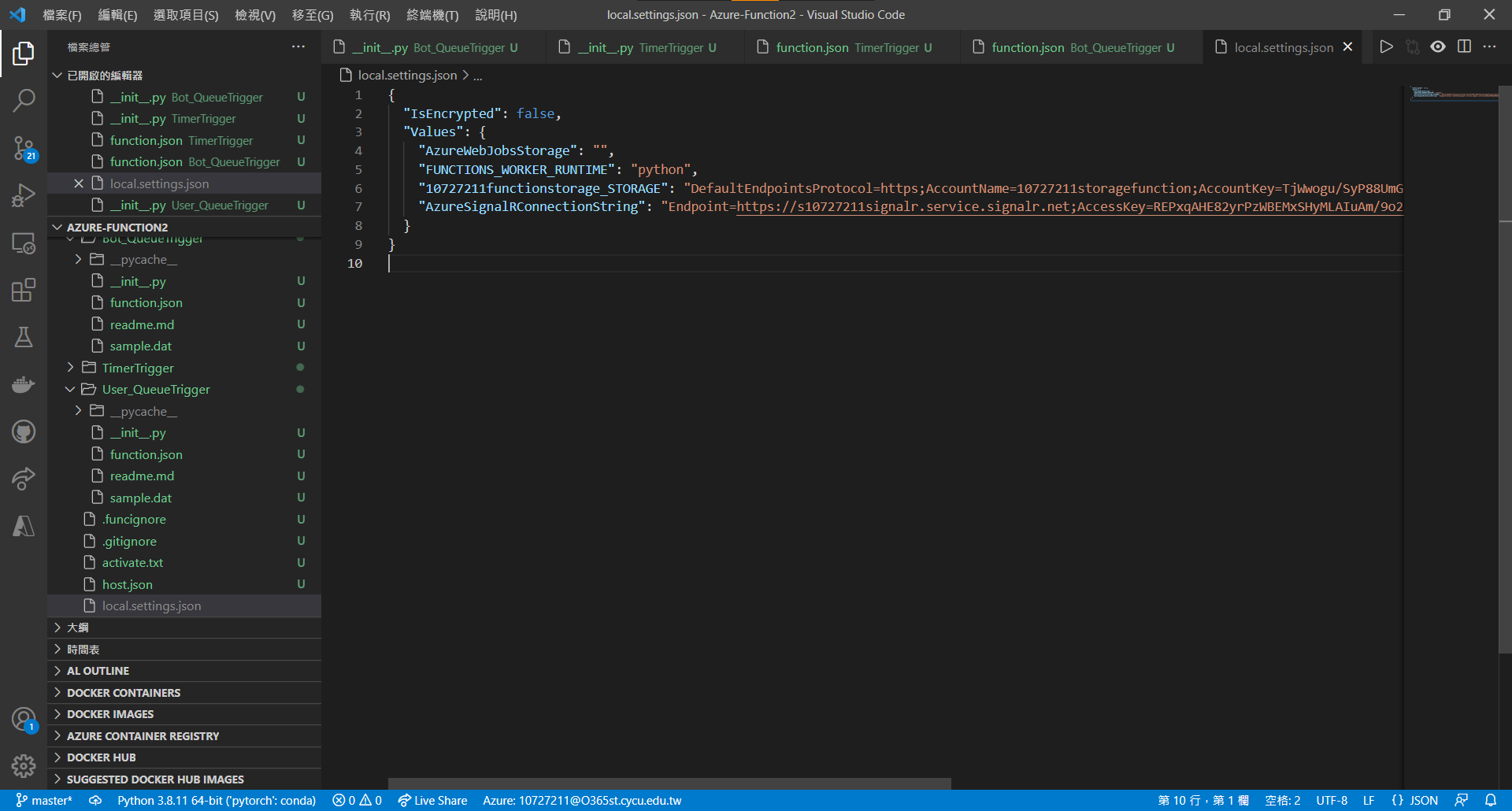
現今多數的設備具備連網能力且安裝瀏覽器能夠瀏覽網頁，故多數的Mobile Device只需連網就能使用。

1. Manageability Interoperability

Azure Function在正常閒置狀態時是無狀態的，且付費方式為pay-per-use，基於Azure Function的聊天室能夠根據聊天室的使用量來決定付費的多寡。

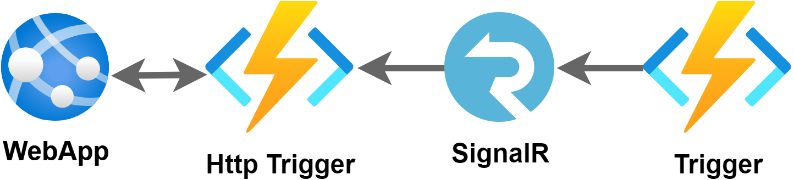
# 肆、開發過程所遇問題

## 一、Azure Environment



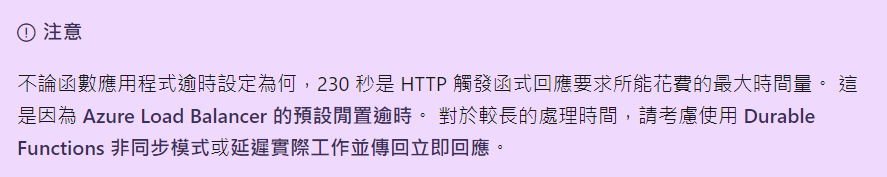
Azure Portal本身並不提供IDE給Python做為開發環境，因此需要在本地IDE做好開發再部署至Azure，有關環境參數問題需要額外多做設定（”Azure Functions: Upload Local Settings…”），才能成功將本地端參數（”local.settings.json”檔案）部署至雲端（”Application Settings”）運作。

## 二、SignalR



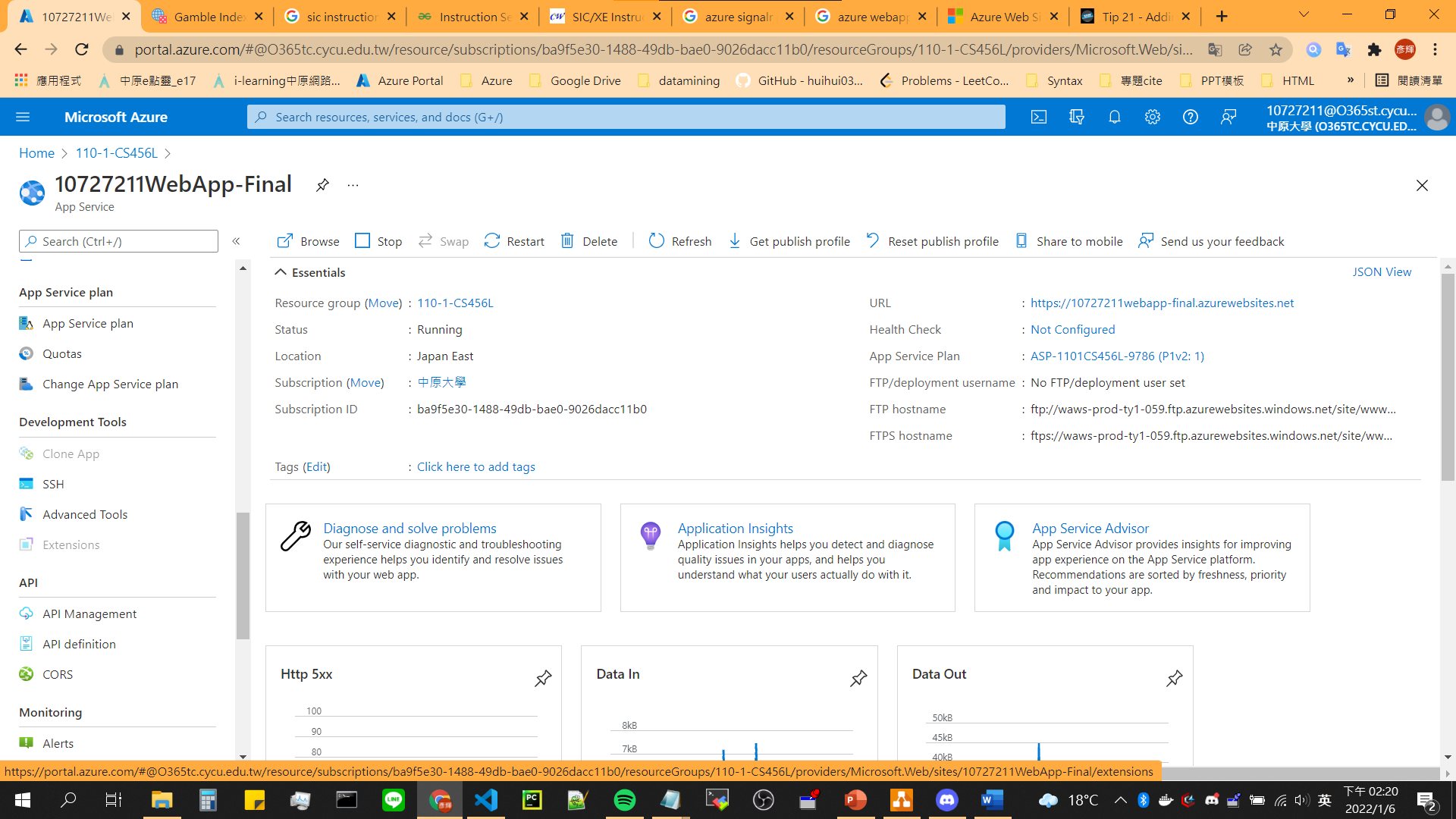
Azure SignalR中雖然可以提供serverless的方式進行連接，但經幾次嘗試後發現只有一種方法能夠接收Azure SignalR訊號：透過Azure Function定義SingalR的Input Binding，再透過HTML中的JavaScript使用WebSocket發送連線至Azure Function之api url，Azure Function接收到HttpTrigger之後隨之將SignalR連線所需資訊回傳到JavaScript中。

## 三、Durable Function



原先採用Durable Function做為後端Azure Function的運行框架，但探究Durable Function運行原理後發現使用動機不符，Durable Function雖然可以針對Http Request Timeout的問題有良好的解法，但過程會透過new()方法新增一連串的協調器，耗時過長，無法達到即時的效果，因此捨棄此方法。

## 四、WebApp Extension

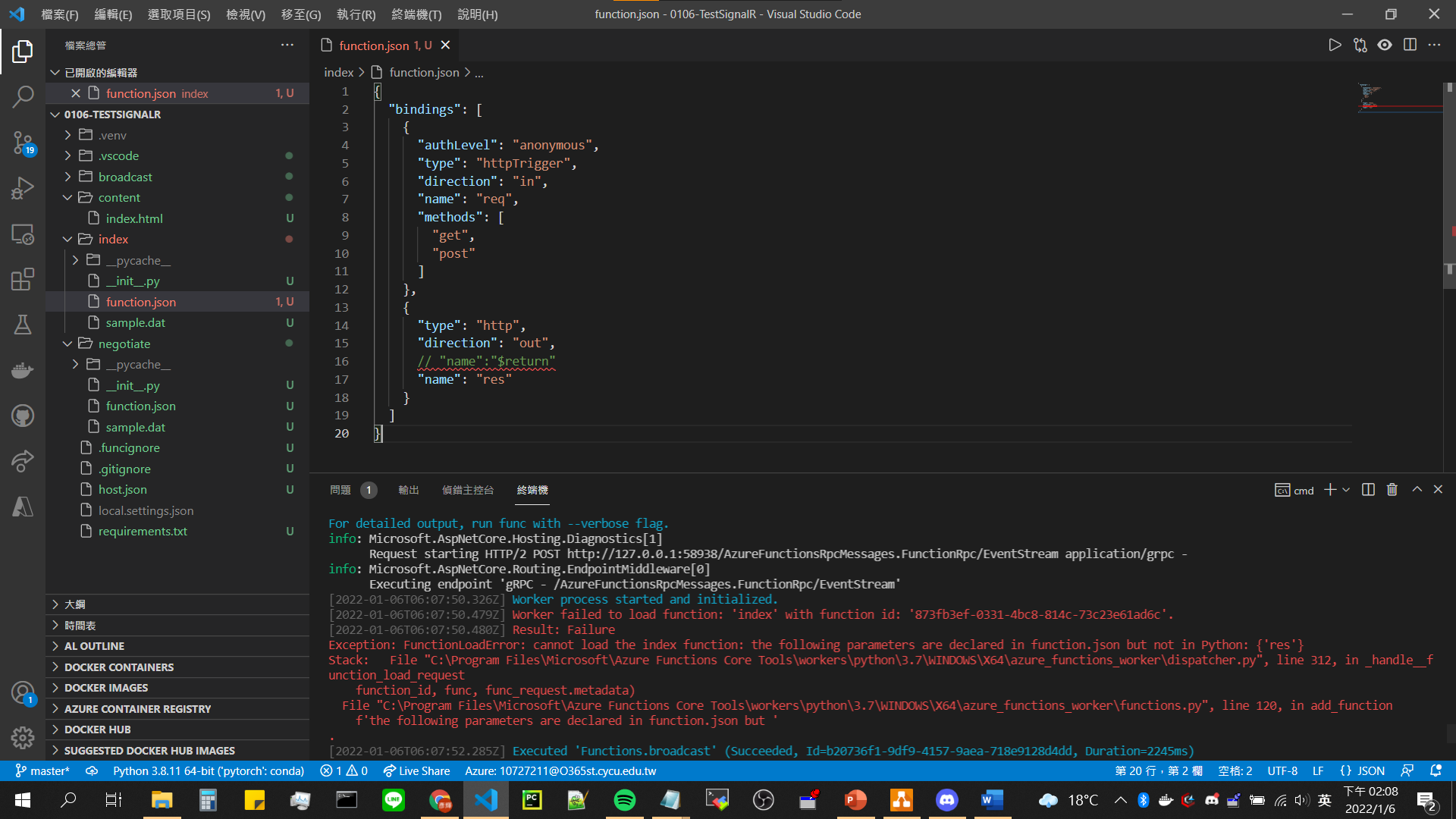


使用Python語言做Azure App Service的開發語言會有環境問題。在新增Azure 資源時，Azure提供給Python的運行環境中限制只能允許Linux OS，但對於extension有許多限制，在網路中的解法中只有針對PHP做解答[[12](#LinuxWebAppExtension)]，未能找到有關Python；而在Windows的環境開發時，官方提供GUI介面新增Extension[[13](#AddingExtension)]。因此在開發網頁部署至Azure時採用Docker Container的部署方式，指定Image並在DockerFile中新增Extension。

## 五、Error – Create CosmosDB Container

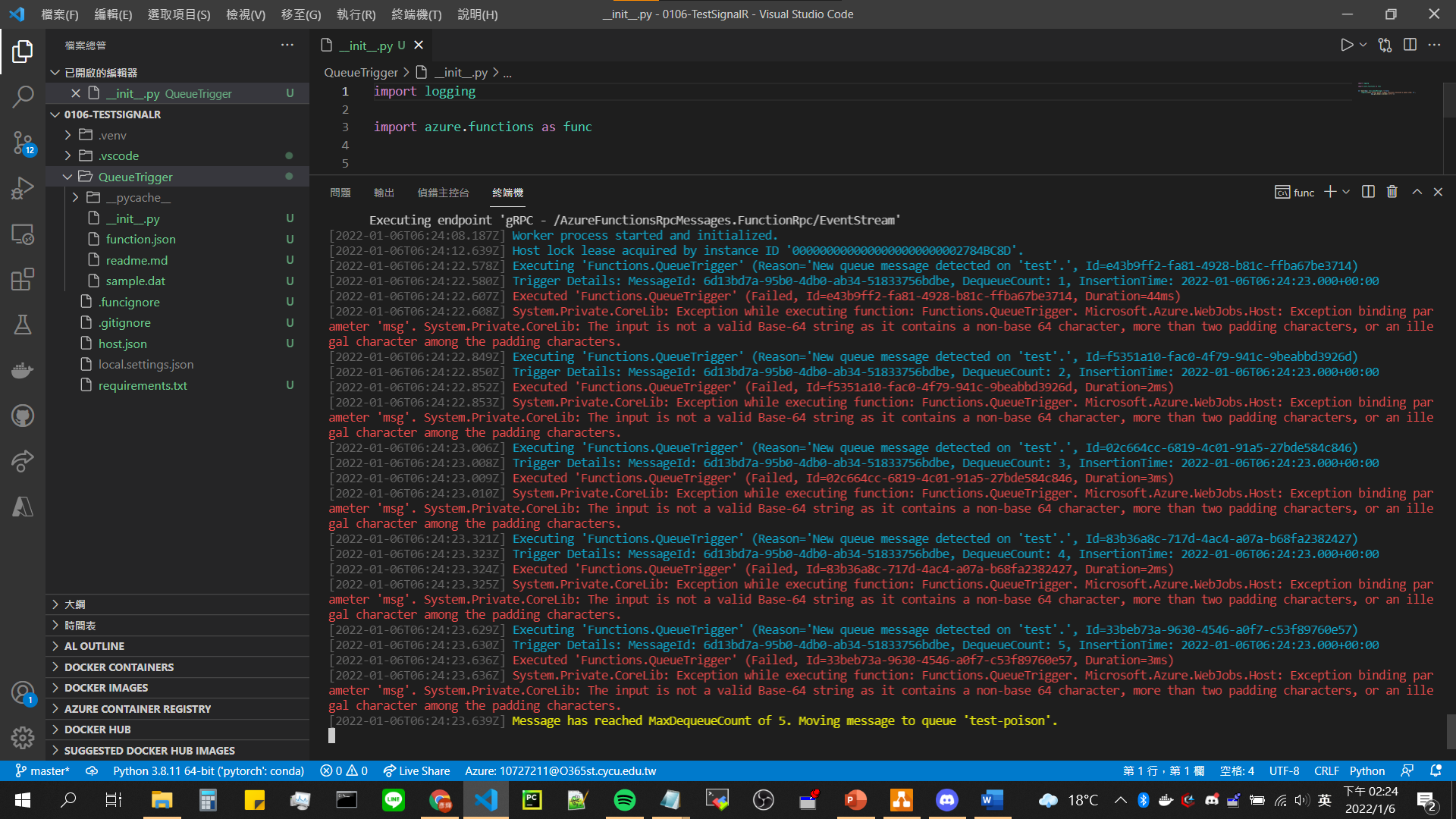
使用Azure Portal之GUI介面無法順利創立DataBase，Container，需要使用API方式使用程式建立。

## 六、官方文件錯誤



https://docs.microsoft.com/zh-tw/azure/azure-signalr/signalr-quickstart-azure-functions-python中的index/function.json output binding有誤：”name”之binding應該為”$return”，否則程式將出現報錯。

## 七、Encode問題



在Azure Queues Storage中傳送/接收訊息的方法必須透過Base64編碼傳送，使得中文格式變得難以處理，但在Azure Function中嘗盡各種Decode/Encode方法皆無法使中文順利處理。

# 伍、心得

後端是第一步著手的點，起初因Azure對於Python不友善的環境，使得起初在架設環境中就接觸到不少Error，架構中起初嘗試Durable Function做後端的環境，從中了解同步非同步的概念，yield的概念使得記憶體能更精準的縮減記憶體的使用，Durable Function過程的傳值也是一大課題，彼此Function的Payload需要控制得已，需要使用json做傳遞，因此Function中傳值需要先將各值使用dict做mapping，再透過json.dumps()、json.loads()工具將dict與json格式做轉換，在越往深入探討Durable Function，卻發現Durable Function的用途並不適合此專題，Durable Function雖然可以執行聊天室的功能，但基於聊天室需要及時的特性，並不適合Durable Function的方法，Durable Function適合長時間的工作或是CPU Time過大的工作，透過Durable Function可以避免Http Request Timeout的問題。

抽換Durable Function後，仍有其他的新知，如第一次接觸DataBase，透過這次的經驗學到如何針對DataBase做Insert、Query，透過API的方式可以針對DataBase做操作，恰巧避免掉Azure GUI操作DataBase異常的現象；在Queue Storage中碰到了Encode/Decode的問題，Azure Queue Storage中雖然可以儲存類似中文等Big5等字體，但Azure Queue Storage與Azure Function過程溝通的方式需要透過Base64編碼，使得中文的問題相當棘手。

透過SignalR接觸到WebSocket，翻閱許多官方的文件，最終發現Azure SignalR並不支援Python，且Python的WebSocket工具也不能解析DNS，因此使用官方教學的方式在Html中的JavaScript成功進行WebSocket；但過程中發現不管是Azure Function、Azure SignalR都有Cold Start問題，儘管使用了進階資源，且官方文件宣稱Kept-Warm，但Cold Start的問題仍舊相當大，最長需要1分鐘來啟動。

前端部分，透過Python的Flask簡易打造了網頁，透過Flask換個角度探討Post/Get的概念，儘管在Microsoft Learn中曾經透過Http Trigger的範例簡單玩到GET的概念，但當時不知GET跟POST具體的差異，透過Flask與Html、JavaScript、CSS的開發，了解GET/POST的具體差異，此外，透過本次的Project第一次接觸到前端，開發了Html，使用JavaScript做動態網頁的開發、最後再使用Bootstrap進行網頁的美化作業，過程獲益良多！

最後因為Azure Extension不支援Python的關係，使用Docker Container將所需環境打包成一個映像檔，最終成功部署至WebApp。Python做為熱門的程式語言，希望官方可以針對Python做更多的支援。

# 陸、參考資料

1. *Create a Function Triggered by Azure Cosmos DB*. (2021). https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-cosmos-db-triggered-function
2. *Azure Queue Storage Trigger for Azure Functions*. (2021). https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-queue-trigger?tabs=python
3. *Getting Started with SQL Queries*. (2021). https://docs.microsoft.com/en-us/azure/cosmos-db/sql/sql-query-getting-started
4. *How to Use Azure Queue Storage from Python*. (2021). https://docs.microsoft.com/en-us/azure/storage/queues/storage-python-how-to-use-queue-storage?tabs=python
5. *Quickstart: Create an App Showing GitHub Star Count with Azure Functions and SignalR Service Using Python*. (2021). https://docs.microsoft.com/en-us/azure/storage/queues/storage-python-how-to-use-queue-storage?tabs=python
6. Ithomemhjao. (2019, September 28). *第 20 天：Flask：基礎網頁製作*. ttps://ithelp.ithome.com.tw/articles/10222132
7. Bootstrap. https://getbootstrap.com/docs/5.1/getting-started/introduction/
8. *Flask Form Submission without Page Reload*. (2021, July 17). https://www.geeksforgeeks.org/flask-form-submission-without-page-reload/
9. *How to Use the JavaScript Fetch API to Get Data?* (2021, July 20). https://www.geeksforgeeks.org/how-to-use-the-javascript-fetch-api-to-get-data/
10. *Deploy a Container Image to Azure App Service*. (2021). https://docs.microsoft.com/en-us/azure/developer/python/tutorial-deploy-containers-02
11. *Azure Functions Python*. (2021).https://hub.docker.com/\_/microsoft-azure-functions-python
12. *Adding Extension of Composer to Linux Web App*. (2020, December 6). https://docs.microsoft.com/en-us/answers/questions/187690/adding-extension-of-composer-to-linux-web-app.html
13. *Adding Extensions for Azure App Service*. (2020). https://dailydotnettips.com/adding-extensions-for-azure-app-service/