# 2022 年臺灣國際科學展覽會
# 優勝作品專輯

作品編號　**190023**

參展科別　電腦科學與資訊工程

作品名稱　程式語言學習系統

得獎獎項

就讀學校　新竹市立光華國民中學

指導教師　王建豪

作者姓名　洪翊俊

關鍵詞　**Python、Django、程式解題系統**

# 作者簡介

我是洪翊俊。國小時接觸到程式設計，因為能夠親手用程式碼打造出實用或有趣的功能而對程式設計產生興趣。這次打造出能讓老師和學生互動的程式語言學習系統，結合之前學到的網頁前端設計 HTML & JavaScript 等並另外用 Python 模組 Django 學習製作網頁後端。科展過程中使我更加熟悉網頁程式語言的設計模式，很幸運能有機會將這個網站的原型分享給大家。

# 摘要

『程式解題系統』是提供不同難易的題目給學習程式設計者使用的系統，主要透過測試資料來驗證程式碼的正確性和效能姓。然而，這類系統大多為私人用。台灣著名的程式解題系統有台中女中程式解題系統及高中生解題系統(Zero Judge)等，題目雖然繁多卻分類雜亂，不能讓教師客製化題目給學生，只能從眾多題目中，零散的挑出適合學生的題目，給學生練習。基於以上理由，此研究目的是做出不同於一般程式解題系統的功能，希望能讓教師彈性增刪題目。使用 Python 為基礎語言，後端採用 Django 框架，已經架構出系統的原型，並上架至 Heroku(zeaf.herokuapp.com)。希望循序漸進地讓學習者有成就感，也讓教師能監督每一個學生學習狀況。

# Abstract

Online Judges refers to websites containing a system that provides quizzes with specific difficulty for users learning coding, and judges the code's efficacy with test data. However, Online Judges are mostly private. While public Online Judges contain a lot of problems, the problems sometimes are considered disorganized, and teachers are not allowed to customize quizzes for students, forcing the teachers to pick quizzes individually for students. Based on the reasons above, this research aims to build an Online Judge system with features different from others, providing teachers the ability to add and remove quizzes by themselves. In our study, the Online Judge system has been built with Python and the Django module. The prototype of the judge system has been constructed and hosted on a server via Heroku. We hope to make the users learn coding step by step and offer teachers a more convenient way of managing students and their accounts.


# 壹、研究動機

學習程式設計的課堂上，發現老師常需要親自來到學生面前觀察學生的學習狀況，老師也很難立即性的知道學生程式碼哪裡有錯誤，於是我開始想利用網頁程式設計以及 Python 後端的 Django 模組相關知識，架構一個簡易且兼顧教學功能的解題系統，使教師更方便地指派題目、確認程式碼對錯。

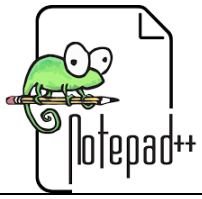
# 貳、研究目的

一、透過 Django 的 MVT(Model, View, Template)設計模式，架構後端伺服器。
二、規劃網頁佈局，建立網頁模板(Template)資料庫(Database)。
三、讓網站支援學生與教師用帳號，實做登入系統和學生群組功能。
四、限時裁判系統與測試結果回報。

# 參、研究設備及器材

一、器材：電腦設備（電腦、滑鼠等）。

二、程式編輯器：Visual Studio Code、Notepad++

| | |
|---|---|
|  |  |
| Visual Studio Code | Notepad ++ |

# 肆、研究過程及方法

一、名詞定義：

(一)、函式(Function)與物件導向(Object-Oriented Programming)專有名詞：

此專案使用大量的函式及物件導向概念。函式是將一段常用到的程式碼封裝，宣告(Declare)函式時，會宣告接收參數(Parameters)以及所回傳(Return)結果。物件導向的概念是宣告新類別(Class)，類別含有屬性(Property)以及方法(Method)，可以透過呼叫該類別宣告新的物件(Object)，新的物件在初始化(Initialize)時常會定義屬性，如題目類別，屬性就有標題、文字等，初始化時就會以參數形式傳入初始化函式__init__()該題目的標題與文字。當要對某特定類別的物件進行相同的程序時，會封裝成方法，也就是說方法即是針對某類別的函式，可以修改該物件的屬性。值得一提的是，方法比起一般函式宣告要多一個參數，通常記為 self，當作第一個參數。myObject.myMethod(v1,v2)實際上會呼叫 myObject 所對應類別內的方法 myMethod(myObject, v1, v2)，也就是說 self 參數接受呼叫此方法的物件。當某個新類別要使用現有的類別之方法和屬性時，常會繼承(Inherit)原有類別，這時新類別的物件將可以使用原類別的方法和屬性，新的類別也可以定義新的方法和屬性，甚至在新類別的方法內再套用舊類別的方法。

(二)、網路程式語言與 Django 模組相關名詞：

瀏覽器要進入網站，即是對該網站後端發送請求(Request)。請求可以依傳送參數方法分為 GET 和 POST 兩種。另外網站分為靜態(Static)和動態(Dynamic)，其中動態網站會搭配伺服器與資料庫(Database)，是指使用者可以與網頁做互動編譯之網站，靜態則相反。

(三)、程式判定基本術語：

系統對程式碼進行判定會利用固定的測資點(Judge, 即測試用輸入資料)，程式上傳者通常不知道測資點有哪些，需要透過撰寫可以處理一般性的輸入之程式來通過系統判定。

二、選擇適合的後端架設框架：

Python 程式語言是一個易學、強大、具豐富套件庫的程式語言，Python 可應用於建立網路後端，有兩個主流框架分別是 Django 和 Flask。選用適合的框架，是必須第一個思考的問題。框架的特性應選適合開發大型網站的特質，具可維護、易修改的特性。程式解題系統需要多功能與龐大資料庫，才能達到理想的功能。以下將針對 Django 和 Flask 兩者的進行比較：

| Django | Flask |
|---|---|
| 具有專門給管理人員可客製化的後端 | 沒有功能可以處理管理員任務 |
| 具有 HTML Template 引擎 | 具有 Template 引擎 Jinja2，以 Django 的 HTML Template 引擎為基礎 |
| 允許將專案劃分成數個小的應用程式，使其容易開發和維護 | 一個專案一個應用程式，但可套用上不同模型(models)和檢視函式(views) |
| 提供了許多現成的功能並減少構建複雜應用程式的時間。 | Flask 是架一個小型網站好的開始 |
| 適用於需要大量功能的大型專案。對於簡單的項目，功能可能會過多 | 小型應用程式可以輕鬆構建，不需要太多的程式 |

由上表可知，Django 適合處理大型專案，適合應用於後端架設，且具有良好的可維護性；Flask 適合快速開發出小型專案，但是隨著系統擴充時可維護性不佳。因此選擇使用 Django 會比較適合本研究。(以上節錄自 Flask vs Django in 2021: Which Framework to Choose?，見參考資料一)

三、探討 Django 的 MVT 設計模式：

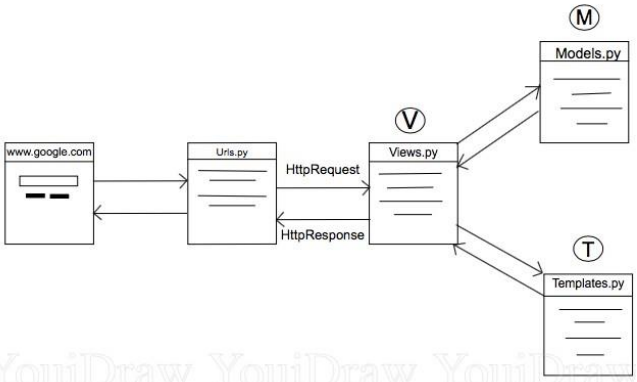
Django 使用 MVT 設計模式，分別是模型(Models), 檢視函式(Views)與模板(Templates)。

模型(models.py)：通常包含內建的基本資料模型(Fields)和類別方法，Django 提供了簡單的方式在資料庫中自由新增、刪除、更新和檢索物件。

檢視函式(views.py)：它會接受 HttpRequest 並回傳 HttpResponse。一個檢視函式可以與資料庫、HTML 模板和其他檔案互動，後端程式碼都寫在檢視函式中。

模板(templates.py)：製作同一種網頁版型的 HTML，為了方便生成動態網頁，Django 提供了 HTML 模板語言，並有內建 render()函式用以渲染(Render, 即將資料嵌入)這些模板。

瀏覽器發出請求後，會去呼叫定義網址與其對應檢視函式的 urls.py，找到適合的 views.py 中的檢視函式(View function)後，該函式會接受該請求的參數後以 HttpResponse 物件方法回傳純文字內容或是利用渲染方法以 HTML 將資料呈現出來。見(圖一)：




圖一：MVT 設計模式示意圖(參考資料二)

四、理想功能

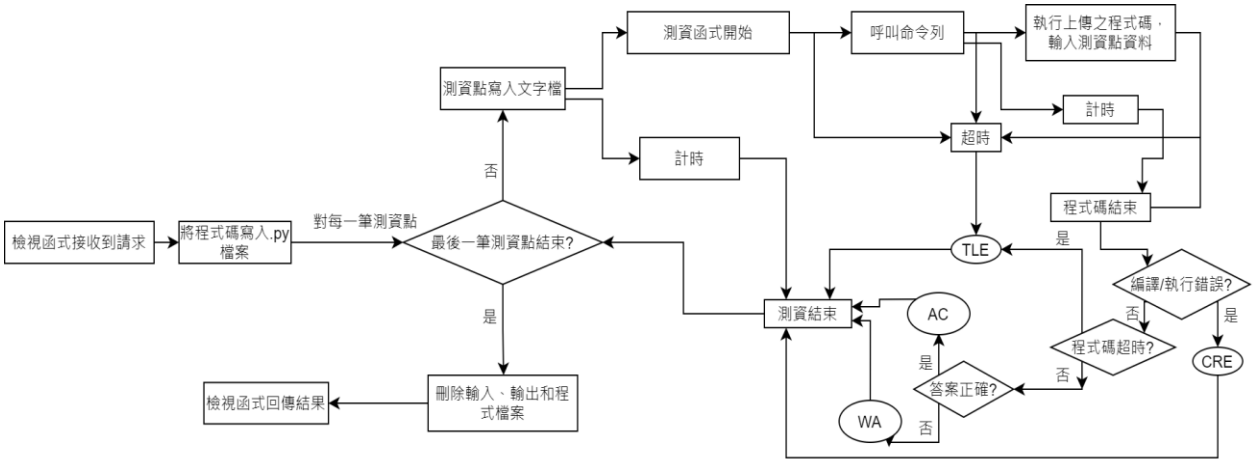
（一） 上傳及判定程式碼：

1. 上傳程式碼至檢視函式接收請求 ：

網頁請求(Request)傳參數方式有分兩種：GET 和 POST。GET 是將參數放到連結，POST 則是隱藏參數。當使用者上傳程式碼，為了避免 GET 方式儲存程式碼提出請求會使連結過長，使用 POST 方式儲存程式碼提出請求。

2. 判定程式碼

檢視函式會將上傳的程式碼寫入.py 檔，同步叫出命令列(Subprocess shell)執行檔案和計時，輸入給定的測試資料(測資點)後，偵測輸出是否正




圖二：理想程式判定流程圖

確和超時與否。重複對不同比輸入資料做上述過程，紀錄判定結果(答案正確記為 AC[Accepted]，錯誤 WA[Wrong Answer]，過時 TLE[Time Limit Exceeded]，程式出現錯誤 RCE[Runtime Error/Compile Error]。RCE 在此為 RE 和 CE 之總稱)，最後再回傳結果(全數正確 AC，否則回傳非 AC 發生最多次的結果)。

（二）自創題目功能

題目類別定義於 Models.py。其屬性除了標題、文字、創建者帳號、唯一的 ID、創建時間戳章(Timestamp)以外，還有測資點(Judge)。測資點是 Python 二維清單轉成的文字基本類別(TextField)。長度不定，清單中的項是長度為二的清單，第一筆資料是輸入、第二筆是預期正確的輸出。網站首頁的導覽列(Navbar)有導向新增題目頁面的連結，在該頁面提交按鈕按下後會提出 POST 請求，後端會檢查測資點是否符合格式並在資料庫創建題目。

（三）帳號群組功能

帳號功能繼承至 Django 內建的帳號類別，登入、登出時 HTML 模板渲染的差別則是用 Django HTML 模板語言寫進。由於實做時前端 HTML 模板有許多重複的程式碼因此不在此贅述。

群組功能可以使老師創建學生群組，加題目至群組使學生可以方便的上傳程式碼，測試結果則可以給老師直接瀏覽。為了讓學生方便加入群組，群組生成時有一組唯一的連結。學生端會有 HTML 網頁可以輸入此連結。

在首頁，學生可以看見有標題的題目連結(題目來自所有加入的群組)，使教師能指派適當題目給學生，群組頁面也有可以導向該群組題目的連結，兩者的差別在於若有多個群組用同一個題目，在首頁上傳的紀錄和測試結果所有的群組管理者都可以看到；若點進一個群組頁面則只有該群組管理者看得到上傳紀錄和判定結果。老師端則可以看見加入群組的學生、群組上傳紀錄與結果。

（四）錯誤回報功能：

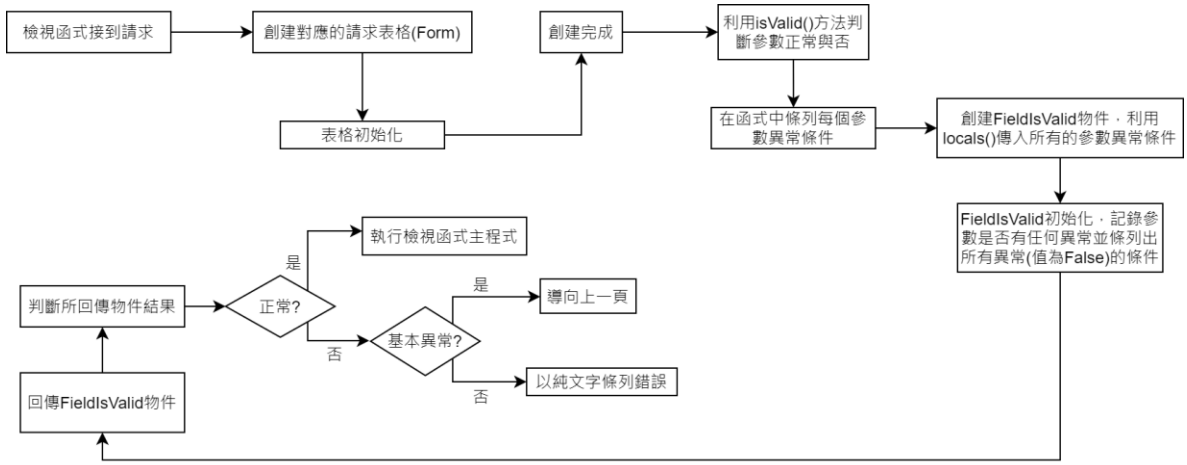
錯誤回報係指提出請求後，檢視函式接收到的 POST/GET 參數異常。在剛開始建此專案時都是在各個檢視函式中檢查各參數，但隨著專案規模擴大，不同檢視函式應該有不同的請求表格(Form)類別，以便使檢視函式的主程式可讀性增加、減少類似的程式碼，在主程式中只須用 form.isValid()辨識請求參數異常與否，以及異常的參數是哪些。請求表格類別的程式碼則應和主程式 views.py 分隔，獨立成一檔案再導入(Import)主程式。

所有請求表格的都將繼承至類別 RequestField。RequestField 類別的初始化函式會將物件賦予兩個屬性：method(POST/GET 方法)和 data(該請求方式的資料)。另外有 FieldIsValid 類別，初始化時接收所有判斷參數異常與否的布林值，若全部為真(即參數正常)該物件的屬性 result 為真，否則為假且屬性 reason 會列出所有的參數異常。前端 HTML 送出請求前會把關一些錯誤(如標題不能為空白)，但是有些需要後端判斷是否符合要求(如測資點是否符合格

式、用戶名稱是否已經選走等)。當後端發現請求參數異常時，如果是屬於基本情況(即上述用後端判斷的情況)會重新導向上一頁，否則(惡意請求)則會回傳純文字條列出問題(圖三)。

當要創建新的請求表格，初始化函式可直接省略(繼承 RequestField)或是使用 super 函式，並且 isValid()中只需條列出所有參數驗證的條件，再回傳 FieldIsValid 傳入所有區域變數 locals()即可。

需要製作請求表格來驗證的檢視函式有不少，登入、註冊帳號、自訂問題和群組等都會需要請求表格使主程式更簡潔。




圖三：錯誤回報

四、實現理想功能：

（一）、前置作業：

以下各使用的 Python 版本為 3.7.4，Django 版本 3.1。

要開始一個 Django 專案，首先要建立虛擬環境(Virtual Environment)將專案內的程式獨立，避免受到電腦上現有的其他 Python 模組或設定干擾。

$ virtualenv venv-zeaf (建立虛擬環境)

$ Scripts\activate.bat (起始虛擬環境)

啟動虛擬環境後，就可以建立新的 Django 專案了。專案命名為 zeaf。

$ django-admin startproject zeaf (建立專案)

此專案需要建立一個應用程式：

$ django-admin startapp app_questions

若需在 app 中套用新的模型，編輯 models.py 後輸入 ：

$ manage.py makemigrations [app name]

$ manage.py migrate [app name]

（二）、題目頁面：

題目頁面是給學生閱讀題目敘述並上傳的頁面。包含了程式編輯器，題目標題與敘述，後兩者會根據不同題目連結改變。

1. 程式編輯器：

程式編輯器是一個輸入欄，並且顯示會目前程式碼行數(HTML 模板見

附錄一)。原始的想法是使用 HTML Textbox，但是考慮到之後為了方便讓特定程式碼保留字變色，使前端更加彈性，而 HTML Textbox 無法實現這點，於是改用 Contenteditable Div 實作。

2. 題目標題與敘述：

首先要建立題目模型。題目要含有唯一的 ID 讓使用者可以存取，還有標題、敘述以及測資點。

```
class Question(models.Model):
    title = models.TextField()

    text = models.TextField() # question content
    unique_id = models.IntegerField(default=-1) # unique id

    judges = models.TextField()
    created = models.DateTimeField(auto_now_add=True)
```

圖四：題目模型 models.py

將專案 migrate 後，為了使管理員(即我們自己)方便管理資料庫，還會在 admin.py 建立題目物件排序的方法等(admin.py 見附錄二)。

題目敘述會根據題目 ID 不同而不同，為了動態產生網站，需要使用 Django 的 HTML 模板(Template)語言，加上在檢視函式渲染此 HTML 時必須要有題目的資訊傳給渲染函式。當收到連結/questions/{number} 時，Urls.py 要能接收此連結並傳給檢視函式(圖五)，檢視函式收到 number 時要從資料庫找到有此 ID 的問題物件並將此物件的題目敘述和標題傳進渲染函式(圖六)，最後就根據模板語言將題目標題與敘述放進 HTML(HTML 模板見附錄一)並回傳給瀏覽器。

```
from django.contrib import admin
from django.urls import path
from app_questions.views import *

urlpatterns = [
    path('questions/<int:q_num>', show_question),
]
```

圖五：urls.py

```
def show_question(request, q_num):
    question = Question.objects.get(unique_id=q_num)

    title = question.title
    text = question.text.replace("\n","<br>")

    return render(request, 'question_app/show.html', locals())
```

圖六 ：views.py

（三）、(未計時)程式碼判定功能：

首先要建立程式類別。程式類別的屬性有程式碼、編號、對應的問題 ID、程式碼判定結果、建立時間。為了防止後端出現錯誤導致程式物件有重複的 ID，在類別中加上唯一 ID 的約束(constraint)。(同時也將 Question 類別加上相同的約束)

```python
class Code(models.Model):
    code = models.TextField() # code content
    number = models.IntegerField() # unique id

    for_question = models.IntegerField() # the question user submittted such code for

    result = models.TextField(default="No Result") # the submit result

    created = models.DateTimeField(auto_now_add=True) # submit time

    '''unique number constraint'''

    class Meta:
        constraints = [
            models.UniqueConstraint(fields=['number'], name='unique_number_constraint')
        ]
```

圖七：程式類別模型 models.py

```python
def noself(f):
    @classmethod
    def new_f(self, *args, **kwargs):
        return f(*args, **kwargs) # ignore the the self param
    return new_f
'''
The classes below was created for managing and distinguishing view functions.
Each class methods will call as view functions, hence the "self" parameter above is ignorable.
The function decorator noself() was created to avoid misplacement of parameters in function.
Each class methods above should use this decorator.
'''
```

圖八：noself 函式裝飾器

當按鈕按下，將程式碼以 POST 方式儲存導向/questions/[number]/submit，將程式碼儲存再以 GET 傳其 ID 導向/submit-result/[number] (圖七)，最後再測試程式碼。在建立此功能時 views.py 的檢視函式程式碼未分類而顯得雜亂，因此建立 Script_check 類別分類相同目的函式，它的兩個方法(Script_check.upload , Script_check.judge ) 都是和程式上傳與判定相關的檢視函式。而之前做的 show_question 名稱則改為 Qs_page.show。

程式碼測試之功能實現即是利用 for 迴圈將每一筆測資點都驗證過後回傳正確題目(圖九)。這種做法是無法計時的。萬一程式時間過長(甚至不會結束)，測試時間也會過長(不會結束)。原因是 os.system()要等到執行完 shell script 才會結束。檔案名稱都依據程式碼編號，是為了避免同時有兩個檔案上傳時造成判定錯誤。

```
@noself
@csrf_exempt
def judge(request, q_num, number):
    judges = Question.objects.get(number=q_num).get_judges()

    corrects = 0
    wrongs = 0

    with open(".judging/uploaded_script/uploaded_script_{}.py".format(number), "w", encoding="UTF-8") as f:
        code = Code.objects.get(number=number).code.replace('&amp;', '&').replace('&lt;', '<').replace('&gt;', '>').replace('&quot;', '"').replace('&#39;', "'")

        f.write(code)

    for judge in judges:
        with open(".judging/io/input_{}.txt".format(number), "w") as f:
            f.write(judge[0])

        os.system("python .judging/uploaded_script/uploaded_script_{}.py < .judging/io/input_{}.txt > .judging/io/output_{}.txt".format(number, number, number))

        time.sleep(1)

        with open(".judging/io/output_{}.txt".format(number), "r") as f:
            if f.read()==judge[1]:
                corrects += 1
            else:
                wrongs += 1

    os.remove(".judging/io/input_{}.txt".format(number))
    os.remove(".judging/io/output_{}.txt".format(number))
    os.remove(".judging/uploaded_script/uploaded_script_{}.py".format(number))

    if corrects == len(judges):
        return HttpResponse("AC")
    else:
        return HttpResponse("You scored {} correct answers out of {} questions.".format(corrects, len(judges)))
```

圖九：原始程式判定系統

（四）、帳號功能：

帳號模型不需要自行重頭寫。Django 提供的 AbstractUser 可供繼承使用。唯一需要加的屬性只有 isTeacher，紀錄帳號是教師用或是學生用。

```
class RequestField(object):
    def __init__(self, request):
        self.method = request.method

        if self.method=="GET":
            self.data = request.GET
        elif self.method=="POST":
            self.data = request.POST

class FieldIsValid(object):
    def __init__(self, eachResult):
        self.reason = [key for key in eachResult if not eachResult[key] and key[:2] != "__"]
        self.result = (self.reason == [])

'''
Example:

class MyRequestField:
    def __init__(self, request):
        super(MyReqeustField, self).__init__(request)

    def isValid(self):
        MethodValid = (self.method=="GET")
        NonemptyValid = (self.data!=[])

        return FieldIsValid(locals()) # <- receives all local variables, i.e., MethodValid & NonemptyValid
'''
```

圖十：RequestField 及 FieldIsValid 錯誤回報

申請帳號的欄位有使用者名稱、密碼、信箱、勾選是否為教師帳號的勾選框。其頁面對應的檢視函式為 Register.show，申請後創建帳號的檢視函式則是 Register.register。此時要判斷請求參數是否異常的條件多變及複雜，於是我開始打造錯誤回報系統：RequestField 及 FieldIsValid(圖十)。註解中寫的是如何運用這兩個類別，快速造出新的請求表格類別之舉例。

有快速打造請求表格類別的方式之後，後端用以檢查請求參數異常與否的程式就可以更簡潔(圖十一~十二)。創建帳號若發現異常，並且屬於基本異常(見討論一)，會以 GET 方式回傳錯誤原因至帳號申請頁面，而該頁會提示錯誤原因(如：未勾選為學生帳號或教師帳號)，如果不是，則會以純文字回傳錯誤訊息。

```
class Register:
    ''' /register '''

    @noself
    def show(request):
        if request.user.is_authenticated:
            return redirect("/")

        rf = RegisterPageErr_RequestField(request)

        valid = rf.isValid()

        if valid.result:
            return render(request, 'register/register.html', {"failed": rf.failed, "fail_status": rf.fail_status})
        else:
            return HttpResponse(" ".join(valid.reason))
```

圖十一：Register.show 申請帳號前端

(圖十二) Register.register 函式中有兩種基本異常 ：MailREValid 及 teacherSectionValid。前者是以正規表達式(Regular expression)所判定(圖十三)，雖然方便但也有風險(regex 來源見參考資料三，風險請見參考資料四)。

登入頁面的前端渲染和後端 Login.show 及 Login.login 也類似 Register 類別，請求表格也有判斷請求異常的方式、錯誤提示，只差創建帳號的程式碼變成登入帳號(利用 Django login 和 authenticate 函式)(申請帳號及登入之 HTML 模板見附錄一、Login 檢視函式見附錄三、請求表格見附錄四)。

```python
@noself
@csrf_exempt
def register(request):
    rf = RegisteringErr_RequestField(request)

    valid = rf.isValid()

    if valid.result:
        proper = rf.isProper()
        if proper.result:
            username = rf.data["username"]
            password = rf.data["password"]
            mail = rf.data["mail"]
            isTeacher = rf.isTeacher

            if User.objects.filter(username=username).exists():
                return redirect("/register?fail=1")

            try:
                new_user = User.objects.create_user(username=username, email=mail, password=password, isTeacher=isTeacher)
                new_user.save()

                user=authenticate(username=username,password=password)

                login(request, user)

                return redirect("/")

            except BaseException as err:
                print("Error in Register.register(): {}".format(err))

                return redirect("/register?fail=4")

        else:
            if "teacherSectionValid" in proper.reason:
                return redirect("/register?fail=2")
            else: # mailREValid
                return redirect("/register?fail=3")

    else:
        return HttpResponse(" ".join(valid.reason))
```

圖十二：Register.register 申請帳號後端

```python
class RegisteringErr_RequestField(RequestField):
    ''' Register.register() '''

    def __init__(self, request):
        super(RegisteringErr_RequestField, self).__init__(request)

    def isValid(self):
        methodValid = (self.method=="POST")

        # username/password attr. exists and valid(non-empty string)

        usernameValid = self.data.get("username", None) is not None
        usernameLengthValid = usernameValid and ( 4 <= len(self.data.get("username", "")) <= 30 )

        passwordValid = self.data.get("password", None) is not None
        passwordREValid = passwordValid and (re.search("[a-zA-Z0-9!@#$%^*_|]{8,30}", self.data.get("password", "")) is not None)

        mailExistingValid = self.data.get("mail", None) is not None
        mailNonemptyValid = mailExistingValid and self.data.get("mail", None) != ""

        return FieldIsValid(locals())

    def isProper(self):
        mailREValid = ( re.search('^\w+((-\w+)|(\.\w+))*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)*\.[A-Za-z]+$', self.data.get("mail", "")) is not None)

        teacherSectionValid = (self.data.get("account-type1", False) is not False) != (self.data.get("account-type2", False) is not False)

        return FieldIsValid(locals())

    @property
    def isTeacher(self):
        return (self.data.get("account-type1", "off") == "on")
```

圖十三：Register.register 對應的請求表格與用正規表達式判定信箱

（五）、題目菜單、導覽列與自訂題目：

題目菜單，也是此網站的首頁，會條列所有目前資料庫內的題目標題，使用者可以自由點選要練習的題目。導覽列是網站最上面的區域，包含許多文字連結，如左上有 Logo & Questions，兩者都導向首頁題目選單。

而最右上角有登入及創建帳號連結，即 Login & Register，如果登入則會變為 [username] & Logout。

教師帳號要能自創題目，接下來實現的群組功能中，老師端才可以將題目放到群組內。創建題目的連結將會放置到導覽列上，變為 Logo & Questions & Create(其中 Create 導向創建題目連結)。創建題目的頁面有三個輸入欄位 ：標題、題目敘述及測資點。測資點是二維清單，總長度不定(有幾筆測試資料就多長)，每項是長度為二的清單，其第一項為輸入、第二項為預期正確輸出，兩者皆為字串型態。上傳後，後端的檢視函式會將請求整理成新的請求表單物件(由於和創建帳號的程式碼類似，程式碼將放至附錄三)，自定義的請求表單類別中有方法可以判斷測資點是否正常(符合清單格式，且都是由清單和字串組成)(圖十四)。

```
def judgeIsProper(self):
    try:
        data = ast.literal_eval(self.data.get("judge", ""))
    except BaseException as e:
        return False

    return all(len(entry) == 2 and all(isinstance(entry2, str) for entry2 in entry)\
        if isinstance(entry,list) else False for entry in data) if isinstance(data, list)\
            else False
```

圖十四：判斷測資點符合格式

完成自創題目的後端之前，順帶將 Question 類別新增屬性 author(創建者)以紀錄題目創建者。Code 類別也可以新增相同屬性(代表程式上傳者)。

（六）、群組功能：

群組功能是讓教師可以建立群組，並讓學生加入群組後，可以一併新增題目給所有學生的功能。用戶群組的模型屬性有群組名稱、創建者用戶名、學生用戶名清單、題目 ID 清單(兩者皆以字串形式儲存)、歷史紀錄(學生上傳之程式碼進行判定後，對應的群組會記錄該上傳之程式碼物件 ID)、群組連結、創建時間。群組模型(圖十五)繼承自 Django 的基本模型類別，為了每次於資料庫中新增群組物件時都產生隨機的連結，需要定義 save 函式，而非直接繼承父類別。此時需要用到 super 函式。

```python
class UserGroup(models.Model):
    title = models.TextField()
    author = models.CharField(max_length=25) # author username

    ''' convert by json module '''
    userJson = models.TextField(default="[]") # store usernames
    questionJson = models.TextField(default="[]") # store the unique number of question objects
    history = models.TextField(default="[]") # store the index of code objects

    ''' unique 8-digit url generated by django.utils.crypto.get_random_string '''
    urlCode = models.CharField(max_length=8)

    created = models.DateTimeField(auto_now_add=True) # time created

    def save(self, *args, **kwargs):
        if self._state.adding: # adding into DB the first time
            self.urlCode=randStr(length=8) # generate url for the group

        super(UserGroup, self).save(*args, **kwargs)
```

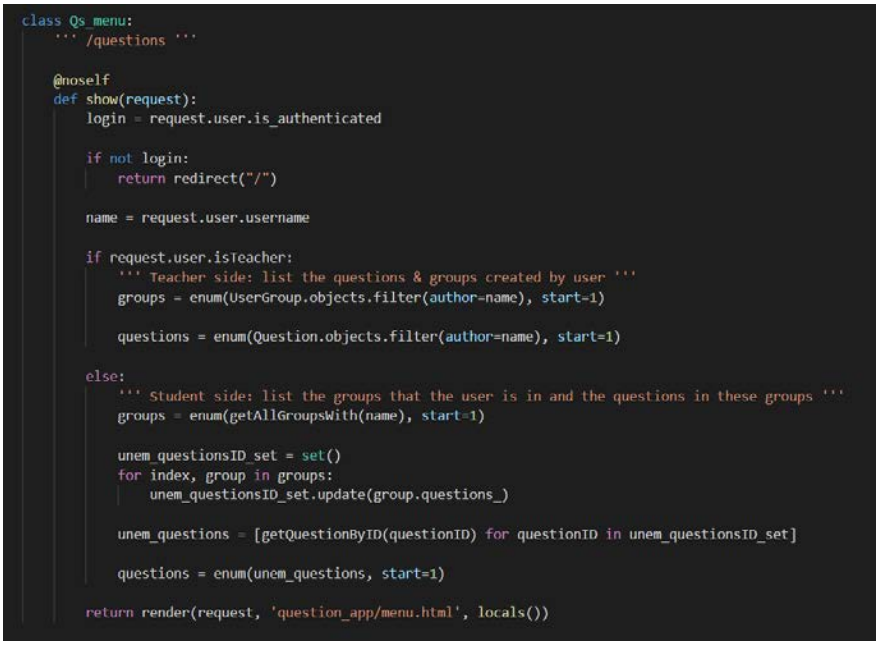
圖十五：models.py 之 UserGroup 類別(節錄主要部份)
(randStr 為 django.utils.crypto.get_random_string)

新增群組的連結也會放在導覽列上，變為 Logo & Questions & Create & Group。新增群組的表單內有群組名稱輸入欄以及一個按鈕，按下去後會利用 jQuery 向/api/questions/created 發送 AJAX 請求(圖十六)(JavaScript 中的 onReturn-Question 宣告方式見附錄一)，回傳值會是該使用者創建的所有題目 ID，並產生上面寫有題目標題的 div。div 中有以該問題 ID 命名的勾選格，當該表格被傳送時會記錄所有勾選的題目並傳給後端。由於實做上和新增問題的功能差不多，程式碼將附註在(附錄三)。

```
<label>Questions ID: </label><div class="button-browse"><p onclick="
    document.getElementById('question-browser').style.display='';
    $.get('/api/questions/created', (data)=>{
        onReturnQuestion(JSON.parse(data));
    });
"class="no-select" style="transform: none;">Browse questions...</p></div>
```

圖十六：用於新增群組的頁面前端 HTML 模板節錄(發送 AJAX 請求並產生 div)

當新的群組物件增至資料庫，save 函式將生成群組的連結，而學生帳號要能根據此連結加入群組。在學生端，導覽列的連結會是 Logo & Questions & Join，其中 Join 導向輸入群組連結的頁面(而 Logo & Questions 仍導向首頁)。輸入群組連結於此頁面後，就會將該用戶的名稱寫入群組的 userJson 屬性。學生的題目來源應該是所加入的群組提供的所有題目，因此首頁中的題目應該從資料庫中的所有題目改為所有加入群組的 questionJson 的聯集，而首頁所列的群組就是所有加入的群組；老師端所列的則是所有創建的問題與群組(圖十七)。圖中的 enum()是 1~n 與任意長度為 n 清單的卡氏積。由於 Django 的模板語言 for 迴圈不支援 Python 的 enumerate 物件，改用了替代方式(enum 函式之宣告方法見附錄三)。



```
class Qs_menu:
    ''' /questions '''

    @noself
    def show(request):
        login = request.user.is_authenticated

        if not login:
            return redirect("/")

        name = request.user.username

        if request.user.isTeacher:
            ''' Teacher side: list the questions & groups created by user '''
            groups = enum(UserGroup.objects.filter(author=name), start=1)

            questions = enum(Question.objects.filter(author=name), start=1)

        else:
            ''' Student side: list the groups that the user is in and the questions in these groups '''
            groups = enum(getAllGroupsWith(name), start=1)

            unem_questionsID_set = set()
            for index, group in groups:
                unem_questionsID_set.update(group.questions_)

            unem_questions = [getQuestionByID(questionID) for questionID in unem_questionsID_set]

            questions = enum(unem_questions, start=1)

        return render(request, 'question_app/menu.html', locals())
```

圖十七：渲染首頁 HTML 的檢視函式完整版(列出題目及群組)

群組還有個重要功能，那就是使教師能看見學生答題狀況。當學生進入題目頁面，若是從群組頁面，會導向題目並以 GET 方式儲存該群組的連結(如：/questions/5?group =iCQOyMoO)；否則從首頁點選題目，則不會有 GET 參數。前者上傳程式碼只儲存一份，且該程式物的 group 屬性會是群組連結；否則若有不同的群組使用同一份題目，則會將程式碼物件複製給所有群組儲存，每份程式碼內容都一樣，只差 group 屬性會是不同的群組連結(圖十八)。

在群組頁面中，有個只有老師端可以看到的連結導向該群組的測試紀錄，會顯示學生的程式碼判定結果。程式碼判定完後，會將結果同步到不同群組但相同內容的程式碼，因此點選首頁的題目作答，所有具有該問題且使用者加入的群組之管理員都會看得到程式碼判定結果。

## （七）、計時程式碼判定功能

舊式的判定方式無法限時。為了解決此問題，試著套用 asyncio 模組，我寫出了完整的計時判定設計圖(圖二)，並做出一個小型的 asyncio 模組測試(圖十九)，以模擬程式判定過程。

```python
@noself
@csrf_exempt
def upload(request, q_num):
    if not (request.user.is_authenticated):
        return redirect("/")

    name = request.user.username

    num_li = []

    if request.method=="POST":
        groups = ""

        if "group" in request.GET and request.GET["group"] not in [None, '']: #1
            group = request.GET["group"]
        else:
            if request.user.isTeacher: #2
                group = ""
            else: #3
                groups = getAllGroupsWithUserAndQuestionID(name, q_num)

        if not groups: #1 2

            code = Code(author=name, code=request.POST["code"], for_question=q_num, group=group)
            code.save()

            if group: #1
                getGroupByUrlCode(request.GET["group"]).new_code_history(code_index=code.number)
        else: #3

            for group in groups:
                code = Code(author=name, code=request.POST["code"], for_question=q_num, group=group.urlCode)
                code.save()

                num_li.append(str(code.number))

                group.new_code_history(code_index=code.number)

        return redirect("/questions/submit-result/"+str(code.number)+\
            (("?li="+str(' '.join(num_li))) if num_li!=[] else '')
        )
```

圖十八：Script.upload 完整版(多群組的程式碼上傳)

```python
import os, time, asyncio

async def cmd():
    os.system("my.py > my.txt") # longer than 1 sec

async def run():
    try:
        await asyncio.wait_for(cmd(), 1) # should raise TimeoutError
        print("A")
    except asyncio.TimeoutError:
        os.system("killall python")
        print("B")

asyncio.run(run())
```

圖十九：asyncio 模組測試

我預設的執行過程是『當 os.system() 執行程式碼，由於 my.py 執行過程超過一秒鐘，asyncio.wait_for 將引發 TimeoutError 錯誤。因此會印出 B』。但這個測試並不成功。在國外的網站上面提問後(參考資料五)，才發現不成功的原因。

os.system()屬於非異步的程序(synchronous/blocking operation)，即函式會等到接收到結果後才會交回控制。這種情況下，asyncio 模組無法限制 os.system 程序。如果要實現限時功能，應改用 asyncio.create_subprocess_shell，產生異步(asynchronously/unblocking) 程序，此時使用 async 函式才有效。

```
@noself
@csrf_exempt
def judge(request, number):
    q_num = Code.objects.get(number=number).for_question

    question = Question.objects.get(unique_id=q_num)

    judges = question.get_judges()

    AC = 0
    WA = 0
    TLE = 0
    CRE = 0

    code_obj = Code.objects.get(number=number)
    code = code_obj.code.replace('&amp;', '&').replace('&lt;', '<').replace('&gt;', '>').replace('&quot;', '"').replace('&#39;', "'")

    with open(".judging/uploaded_script/uploaded_script_%d.py" % number, mode="w", encoding="UTF-8") as f:
        f.write(code)

    for judge in judges:
        with open(".judging/io/input_%d.txt" % number, "w") as f:
            f.write(judge[0])

        async def judge_process():
            try:
                proc = await asyncio.create_subprocess_shell(
                    "python .judging/processor.py "+\
                    str(number)+" "+\
                    "< .judging/io/input_"+str(number)+".txt "+\
                    "> .judging/io/output_"+str(number)+".txt",
                    shell=True,
                    stdout=asyncio.subprocess.DEVNULL,
                    stderr=open(".judging/io/errlog_%s.txt" % str(number), "w+"))
                await proc.communicate()

            except asyncio.CancelledError:
                proc.terminate()
                print("Process cancelled!")
```

圖二十：Script.judge 完整版之一(限時程式碼判定)

知道原因和解法後，就可以將系統測得的結果分類更細。AC 代表正確答案、WA 錯誤答案、RCE 執行錯誤、TLE 時間過長全部都可以被系統測試出來(圖二十～二十三)。

(圖二十)中的.judge/processor.py 是負責開啟執行程式碼的檔案。該檔案將程式碼讀入後會用內建的 exec 執行(並且不傳入任何全域變數)。(圖二十一)中的 PROCESS_TIME_BOUND 為基本測試時間(見討論二)，而上架至 Heroku.com 時使用 asyncio.ProactorEventLoop()會出現問題，因此為了支援不同系統，寫了另外替代的程式。(圖二十二)中 result 是回傳到頁面給上傳者的結果，而另一個 short_result 則是當教師看到群組的紀錄時會看到的縮寫。最後程式是為了同步結果至其它相同內容的程式，接收以 GET 方式傳遞其餘程式碼編號的參數。

```
        ac_time_limit = 1

        timeout = False

        async def run():
            try:
                await asyncio.wait_for(judge_process(), PROCESS_TIME_BOUND + ac_time_limit)
                print("Process ended!")

            except asyncio.TimeoutError:
                print("Process timeout! (If this happened a lot, alter PROCESS_TIME_BOUND) ")

                timeout = True


        if os.name == 'nt':
            loop = asyncio.ProactorEventLoop()
        else:
            loop = asyncio.get_event_loop()

        asyncio.set_event_loop(loop)
        loop.run_until_complete(run())

        with open(".judging/io/errlog_%d.txt" % number, "r") as errlogf:
            if errlogf.read()!="":
                CRE += 1
            elif timeout == True:
                TLE += 1
            else:
                with open(".judging/io/log_%d.txt" % number, "r") as logf:
                    time = float(logf.read())
                    if time <= ac_time_limit:
                        with open(".judging/io/output_%d.txt" % number, "r") as outputf:
                            output = outputf.read()
                            if output == judge[1]:
                                AC += 1
                            else:
                                WA += 1
                    else:
                        TLE += 1
```

圖二十一：Script.judge 完整版之二(限時程式碼判定)

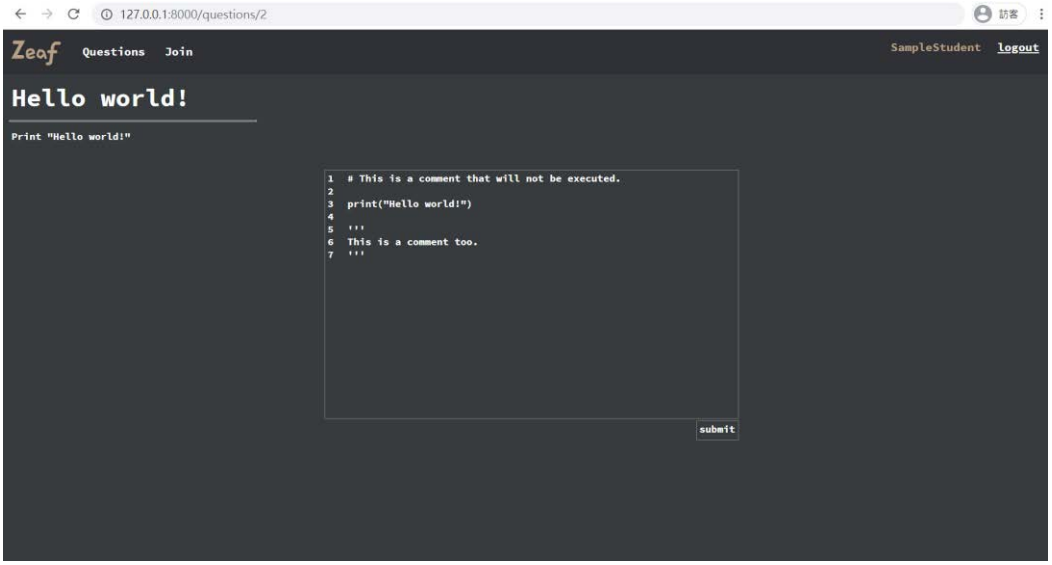
```
os.remove(".judging/io/input_%d.txt" % number)
os.remove(".judging/io/output_%d.txt" % number)

if os.path.exists(".judging/io/log_%d.txt" % number):
    os.remove(".judging/io/log_%d.txt" % number)

if os.path.exists(".judging/io/errlog_%d.txt" % number):
    os.remove(".judging/io/errlog_%d.txt" % number)

os.remove(".judging/uploaded_script/uploaded_script_%d.py" % number)

if AC == len(judges):
    short_result = "AC"

    result = "AC(Accepted)"
elif WA >= CRE and WA >= TLE:
    short_result = "WA: %d/%d" % (AC, len(judges))

    result = "WA(Wrong Answer): You scored %d correct answers out of %d questions.\nWrong answers: %d\nTime limit exceeded: %d\nCompile/Runtime error: %d"\
        % (AC, len(judges), WA, TLE, CRE)
elif CRE >= TLE:
    short_result = "CRE: %d/%d" % (AC, len(judges))

    result = "CE/RE(Compile/Runtime Error): You scored %d correct answers out of %d questions.\nWrong answers: %d\nTime limit exceeded: %d\nCompile/Runtime error: %d"\
        % (AC, len(judges), WA, TLE, CRE)
else:
    short_result = "TLE: %d/%d" % (AC, len(judges))

    result = "TLE(Time Limit Exceeded): You scored %d correct answers out of %d questions.\nWrong answers: %d\nTime limit exceeded: %d\nCompile/Runtime error: %d"\
        % (AC, len(judges), WA, TLE, CRE)

code_obj.result = short_result
code_obj.save()

if "li" in request.GET:
    if request.GET["li"] != '':
        # valid li

        li = request.GET["li"].split()
        for l in li:
            code = Code.objects.get(number=int(l))
            if code.author == code_obj.author and code.code == code_obj.code and code.for_question == code_obj.for_question:
                # author, code content and for_question attr. are the same

                code.result = short_result

                code.save()

return HttpResponse(result, content_type="text/plain")
```

圖二十二：Script.judge 完整版之三(限時程式碼判定)

# 伍、研究結果

一、題目頁面：

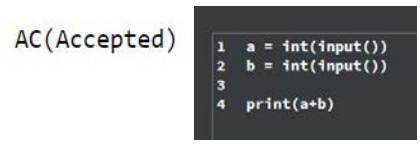
在程式編輯器輸入程式碼，JavaScript 會產生目前程式碼行數。當後端渲染此 HTML 模板，會根據連結(/questions/[ID])渲染具該 ID 的題目之標題與敘述。最終版本已經含有 navbar 及帳號系統，因此會看到右上角有登入的用戶名稱(圖二十四)。



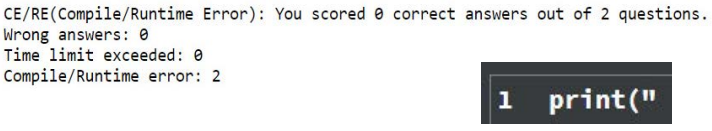

圖二十四：題目頁面最終版

二、程式碼判定：

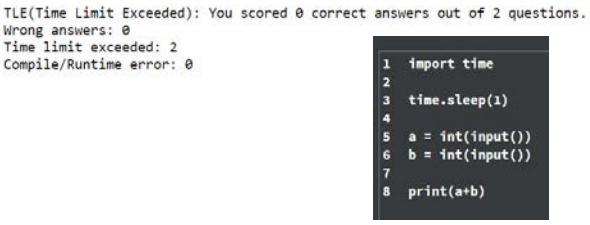
系統可以判定出四種結果，另外也可以支援多人上傳及上傳程式碼給多個群組。以下(圖二十五~二十八)為示範(範例題目皆相同)。



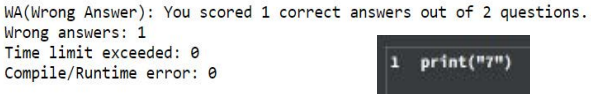

圖二十五：判定 AC




圖二十六：判定 CE/RE



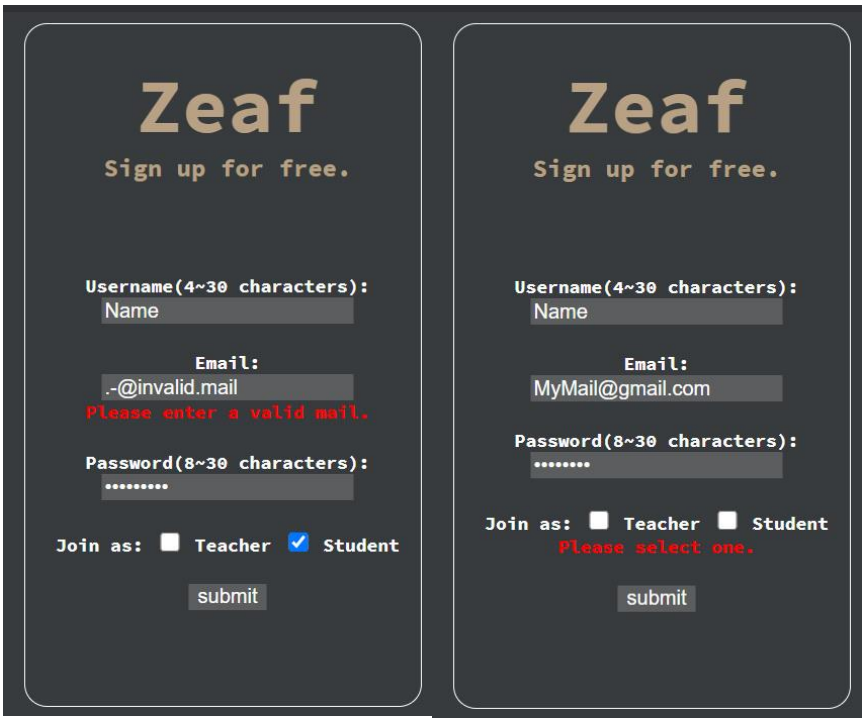

圖二十七：判定 TLE




圖二十八：判定 WA

三、登入、申請帳號頁面：

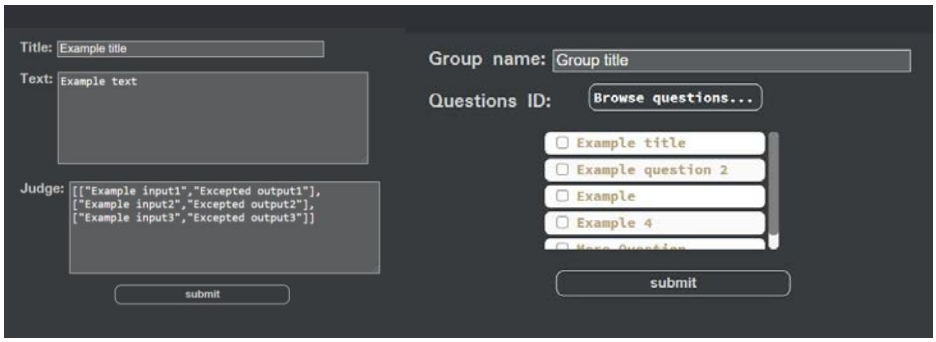
申請帳號和登入的介面都有基本異常提示，即當後端偵測基本異常時，會以 GET 方式將錯誤編號儲存(如/register?fail=2)，並導向該頁面，而頁面會有紅字提示(圖二十九)。




圖二十九：申請帳號頁面的錯誤提示

四、自訂題目、群組：

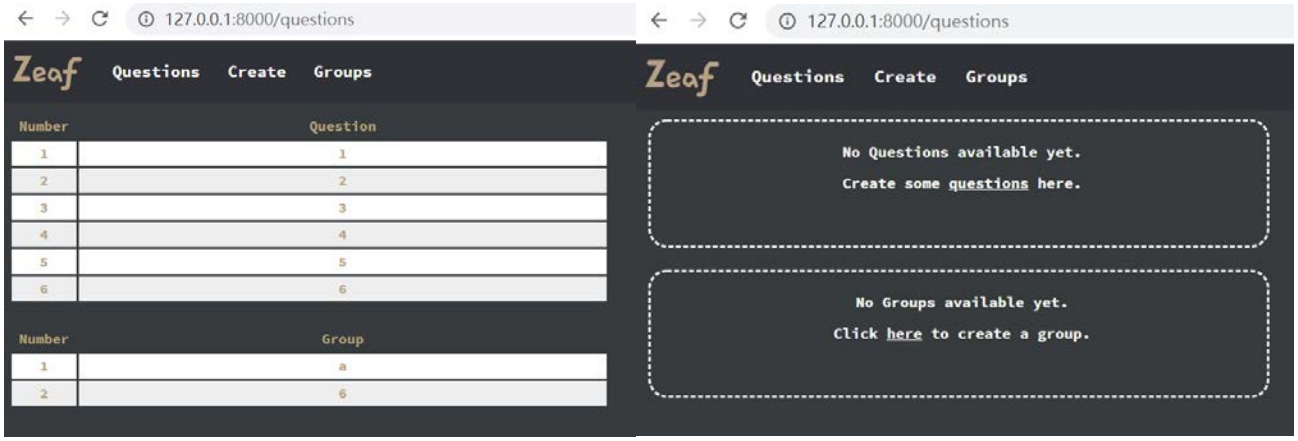
老師可以自訂題目與群組，一樣有基本異常提示。自訂群組的頁面會供使用者選擇題目加入，旁邊有捲動軸可以滑動，按住 shift 可以連選題目。這些功能的程式碼都寫在 HTML 模板內或是 Django 的 Static files 中。




圖三十：自訂題目與群組的介面

五、首頁：

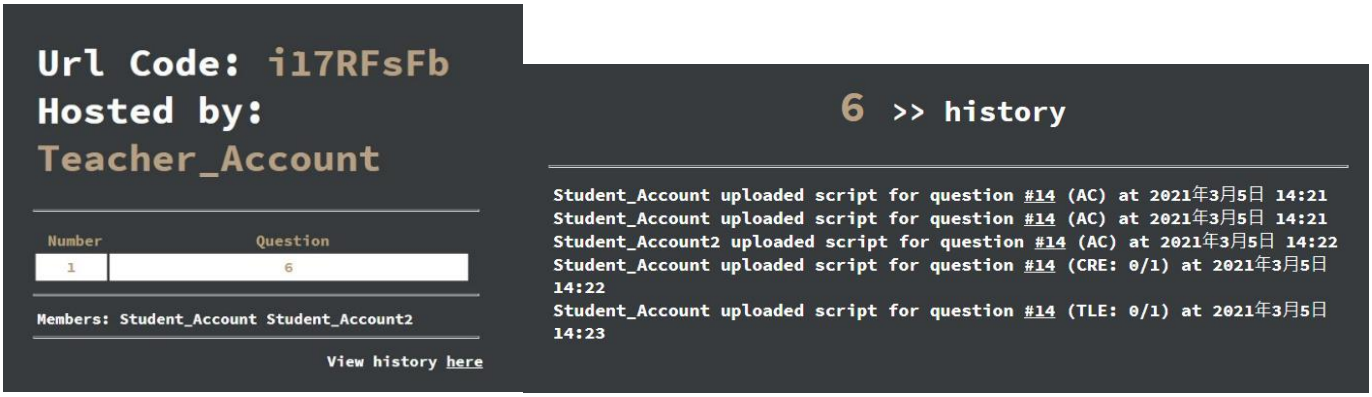
首頁(圖三十一)裡，老師端會列出所有創建的群組及題目(左)；學生端則是所有加入的群組以及群組內的題目。如果沒有任何群組或是題目，則會顯示預設頁面(右)。




圖三十一：首頁

六、群組功能：

每個群組都有自己的頁面(圖三十二)，以及只有老師端可以看見的程式碼判定結果，包含該學生、題目連結、上傳時間以及測試結果。




圖三十二：群組頁面及群組歷史

# 陸、討論

一、網頁請求、檢視函式接收請求：

由於檢視函式至開發中期會顯得雜亂，在開發過程中新增類別(Class)以分類這些檢視函式。實際上呼叫這些函式時會以 MyClass.myViewFunction()呼叫。其中 myViewFunction 在 MyClass 中的定義方式則必須要接收比原本多一個參數，即 myViewFunction(self, v1, ..., vn)，其中 self 會是 MyClass 本身。為了避免宣告檢視函式產生錯誤及增加可讀性，另外定義了函式裝式器 noself，使宣告函式時不需寫 self 參數。

當檢視函式接收網頁請求(Request)時，如果上一頁應該有參數(不論以 GET 或是 POST 方式)，則該檢視函式將檢查此請求是否正常。若正常則執行原後端功能，否則將判斷是否為基本異常。若是基本異常，即在正常使用下有可能產生該異常(如申請帳號時名稱重複、新增問題時測資點異常)，此時對應的請求表格物件會判定 result 為正常，但 proper 為異常，並會回到上一頁並以 GET 方式傳錯誤代碼，否則不屬於基本異常，則請求表格 result 異常，該請求有可能是不是正常方式發出或是惡意的，會直接以文字回傳錯誤名稱。

另外，重要請求前端都會有 CSRF Token，若後端驗證該 CSRF Token 失敗則直接終止該請求，為了防止跨站請求偽造(Cross-site request forgery)。跨站請求偽造是其他網站的惡意連結，導向此網站，且該連結以使用者名義做出使用者不知道的請求(參考資料六：CSRF Token)。通常這是不會被允許的，就算在沒有 CSRF Token 的情況下，因為此專案不允許跨網站請求。

二、關於程式判定系統：

程式碼上傳後的第一個連結會將程式碼存進資料庫，得到其索引值，再以 GET 方式儲存索引值並導向另一個連結，而其對應的檢視函式會對程式碼進行判定。

流程圖(圖二)中可以看見如果程式碼超時，會回傳 TLE。其中兩次的『計時』有區別。第一次的計時是 Python 的 asyncio.wait_for() 的時間限制，在程式判定過程的任一時刻如果超時則會直接將其終止。第二次則是呼叫程式碼進行判定的過程，『計時』只是確定程式碼結束和起始時間相減的差距不大於時間限制。

最後版本的計時程式判定有常數 PROCESS_TIME_BOUND，是為了讓系統有時間運行該 async 函式(運行 Python 檔案)，由於在本機端測試時需要大約 3.4 秒讓此 shell script 開始執行，為了避免 async 函式本身的時間限制影響到測試結果，特定設立了此常數。

另外，群組內儲存的題目都是以 ID 儲存；程式碼上傳到的群組也是以連結儲存。程式中常常需要用到實際物件而非 ID 或連結，此時就會將其轉換成物件；有時相反，只需要其 ID 或連結。為了節省兩者重複轉換的時間，另外寫了 object_filters.py，將要存取 ID 及物件的函式分開寫。例如 getAllGroupsWith 和 getAllGroupsUrlWith 兩者回傳值雖然可以用迴圈互相轉換，但是考慮到效率還是將兩者分開寫。

建立題目時，要確認上傳的測資點是否符合格式，若直接使用 eval()函式可能將非清單及字串類別的物件回傳，造成系統發生錯誤(參考資料七：eval 函式的危險性)。為了避免這種事，將文字轉換成 Python 物件時應改用 ast.literal_eval。

三、Heroku 上架：

在本機端執行 manage.py runserver 即可架設區域網路，但若要上架，就需要利用 Heroku。註冊 Heroku 帳號後，新增專案，即可用 git 上架至 Heroku 了。上架時需要將 settings.py 中的資料庫設定從 SQLite 改為 PostgreSQL，由於 Heroku 使用零時文件系統 (Ephemeral filesystem)，若使用 SQLite 將使資料庫每天都會丟失一次(參考資料八)。目前專案已經成功上架至 zeaf.herokuapp.com，可供測試。

# 柒、結論

製作此專案比我原想得還要難且繁複許多，遇到不少問題，但在論壇上發布問題後也都一一解決了。不僅讓我熟悉網頁前端與後端架構、網站請求相關的知識與其它安全性議題，更學習如何問一個使人能快速理解的問題、自己找到可性的資訊來源。原只是學生角色的我，現在也能利用所學知識協助更多教師及學生。

# 捌、未來展望

優化群組功能問題與學生的管理、加入深度學習功能以提供學生適合的題目。

# 玖、參考資料與附錄


參考資料 ：

一、Flask vs Django in 2021- Which Framework to Choose?
https://hackr.io/blog/flask-vs-django

二、Django Framework MVT Architecture
https://medium.com/@ajitdhulam/django-framework-d5f8fccd1176

三、Regular Expression - Email 表單驗證 https://ithelp.ithome.com.tw/articles/10094951

四、How to validate an email address using a regular expression? Can it cause harm to validate email addresses with a RegEx?
https://stackoverflow.com/questions/201323/how-to-validate-an-email-address-using-a-regular-expression
https://stackoverflow.com/questions/48055431/can-it-cause-harm-to-validate-email-addresses-with-a-regex

五、Using asyncio to run command in a time interval and terminate it afterwards (自行發問)
https://stackoverflow.com/questions/66065010/using-asyncio-to-run-command-in-a-time-interval-and-terminate-it-afterwards

六、What is a CSRF token? What is its importance and how does it work?
https://stackoverflow.com/questions/5207160/what-is-a-csrf-token-what-is-its-importance-and-how-does-it-work

七、Why is using 'eval' a bad practice?

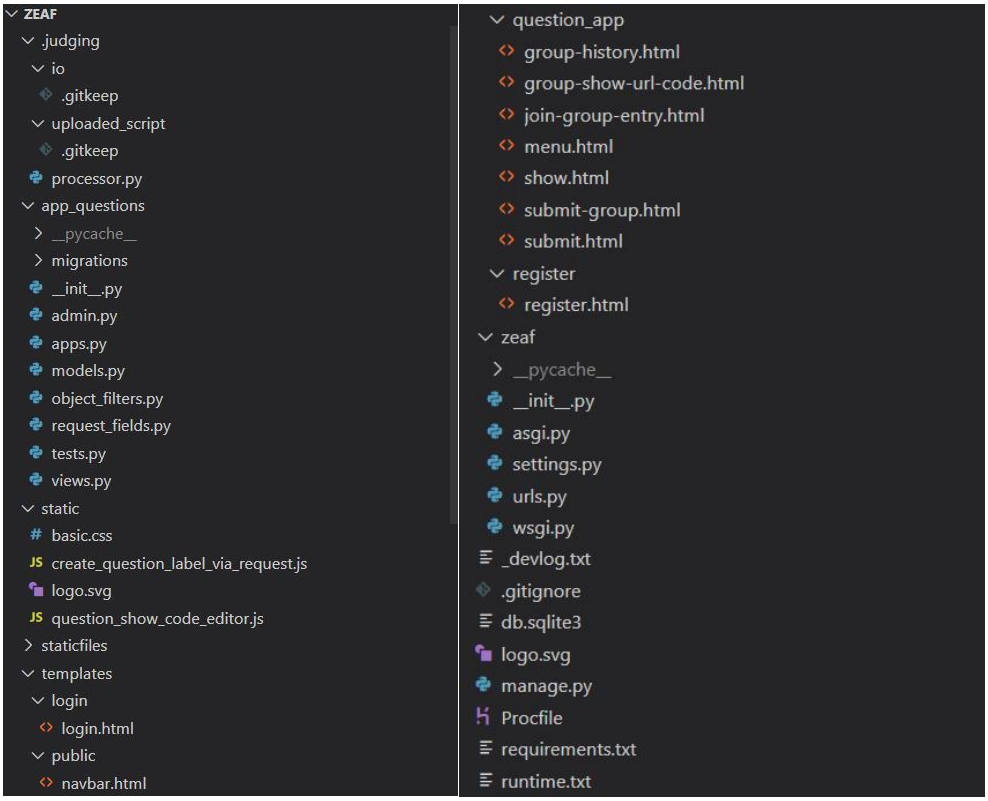
https://stackoverflow.com/questions/1832940/why-is-using-eval-a-bad-practice

八、Python Django - Internal Error ProgrammingError relation does not exist (自行發問)

https://stackoverflow.com/questions/63646532/python-django-internal-error-programmingerror-relation-does-not-exist

附錄：

檔案總覽：




一、HTML 模板

(一) question_app/show.html 及對應 static files(題目敘述及程式碼上傳模板)

```
<!DOCTYPE html>
<html>
    <head>
        <title>{{ title }} - Zeaf Questions</title>
        <meta charset="UTF-8">
        <link href="https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@700&amp;display=swap" rel="stylesheet">

        {% load static %}
        <link rel="stylesheet" type="text/css" href="{% static 'basic.css' %}">

        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300&display=swap" rel="stylesheet">

        <meta name="google" value="notranslate">
    </head>
    <body>
        {% include 'public/navbar.html' %}

        <div id="question-container" style="width: 0px; height: 0px;">
            <div id="title-container"style="width: 500px; height: 40px;">
                <p style="font-size: 30px;">{{ title }}</p>
            </div>
            <div id="hr-container" style="width: 300px; height: 10px;">
            <hr>
            </div>
            <div id="text-container" style="width: 300px; height: 0px;">
                <p>{{ text | safe}}</p>
            </div>
        </div>

        <div id="textbox-full">
            <div id="textbox-container">
                <div id="row-text-container" style="width: 0px; height: 0px;">
                    <p id="row-text" style="width: 0px;"></p>
                </div>
                <div id="textbox" cols="50" rows="5" contenteditable="true"></div>
            </div>
            <div id="submit-button" style="transform: translate(200px,110px);" onclick="submit();">
                <p id="submit-text" >submit</p>
            </div>
        </div>

        <div id="token-container" style="display: none;">
            {% csrf_token %}
        </div>

        <div id="q_num" style="display: none;">{{q_num}}</div>

        <style>…
        </style>

        <script>var group_code = "{{group.urlCode}}";</script>

        <script src="{% static 'question_show_code_editor.js' %}"></script>
    </body>
</html>
```

```
var textbox = document.getElementById("textbox");

var row_text = document.getElementById("row-text");

var text, code, hiddenField;

async function childs(){
    code = textbox.innerHTML.split("<div>").join("\n").split("</div>").join("").split("<br>").join("");

    form = document.createElement("form");
    await form.setAttribute("method", "POST");
    await form.setAttribute("action", "/questions/"+document.getElementById("q_num").textContent+((group_code!=null)?`/submit?group=${group_code}`:`/submit`));

    hiddenField = document.createElement("input");
    await hiddenField.setAttribute("type", "hidden");
    await hiddenField.setAttribute("name", "code");
    await hiddenField.setAttribute("value", ""+code);

    await form.appendChild(hiddenField);

    await form.appendChild(document.getElementById("token-container").childNodes[0]);

    await form.appendChild(document.getElementById("token-container").childNodes[0]);
}

function submit(){
    childs().then(()=>{
        document.body.appendChild(form);

        form.submit();
    });

}
```

```
var text;

function update(){
    let row_text_inner = "";

    raw_html = textbox.innerHTML.replace(/<span style="[^]*">/, "").replace(/<\/span>/, "");

    text_raw = raw_html.startsWith("<div>")? textbox.innerHTML : `<div>${textbox.innerHTML.split("<div>")[0]}</div>`+textbox.innerHTML.split("<div>").slice(start=1).join("<div>")

    text = text_raw.split("<br>").join("").split("<div>").join("").split("</div>").join("\n");

    let totalRow = text.split("\n").length-1;

    totalRow = totalRow>0 ? totalRow : 1;

    for(i=1;i<=totalRow;i++){
        row_text_inner+=i+"\n";
    }

    setTimeout(function(){row_text.innerHTML=row_text_inner;}, 0);

    setTimeout(function(){row_text.style.top = `-${textbox.scrollTop}px`;}, 1);
    setTimeout(function(){row_text.style.left = `-${textbox.scrollLeft}px`;}, 1);
}

document.getElementById("textbox").addEventListener('input', (e) => {update();});

textbox.addEventListener("paste", function(e) {
    // cancel paste
    e.preventDefault();

    // get text representation of clipboard
    var text = (e.originalEvent || e).clipboardData.getData('text/plain');

    // insert text manually
    document.execCommand("insertHTML", false, text);
});

update();
```

(二) create_question_label_via_request.js (onReturnQuestion 宣告)

```
var last_selected = 0;

function raw_uncheck(index){
    document.getElementsByClassName('checkbox')[index].checked=!document.getElementsByClassName('checkbox')[index].checked;
}

function set_check(index, status){
    document.getElementsByClassName('checkbox')[index].checked = status;
}

function uncheckInterval(i1, i2, status){
    if ( i1 > i2 ){
        uncheckInterval(i2-1, i1, status);
        return;
    }

    for(i=i1+1; i<=i2; i++){
        set_check(i, status);
    }
}

function uncheck(event, index){
    if(event.shiftKey)
        uncheckInterval(last_selected, index, !document.getElementsByClassName('checkbox')[index].checked);
    else{
        raw_uncheck(index);
        last_selected = index;
    }
}

var loaded_flag = false;
```

```
function onReturnQuestion(data){
    if(loaded_flag)
        return;

    loaded_flag = true;

    browser = document.getElementById("question-browser");

    if(data.status == -1)
        return;

    for(index=0;index<data.data.length;index++){
        html = `<div class="section" style="background-color: ${index%2==0?'#ffffff':'#fafafa'};">
    <div class="inner-section no-select" onclick="uncheck(event, ${index});">
        <input name="qs_${data.data[index][1]}" class="checkbox" type="checkbox" onchange='this.checked=!this.checked'/>
        <p style="display: inline; color: #B8A17D; transform: none; top: -2px;">
        ${data.data[index][0]}
        </p>
    </div>
</div>
`
        new_section = document.createElement('div');
        new_section.innerHTML = html;

        browser.appendChild(new_section);
    }
}
```

(三) register/register. html (申請帳號頁面模板)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Register - Zeaf</title>
        <meta charset="UTF-8">
        <link href="https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@700&amp;display=swap" rel="stylesheet">

        {% load static %}
        <link rel="stylesheet" type="text/css" href="{% static 'basic.css' %}">

        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300&display=swap" rel="stylesheet">

        <meta name="google" value="notranslate">
    </head>
    <body>
        {% include 'public/navbar.html' %}

        <div id="register-board" style="text-align: center; border-radius: 15px; position: relative; width: 260px; left:50%; transform: translateX(-130px); height: 450px;
        background-color: none; border: 1px white solid;">
            <p style="color: #B8A184; font-size: 50px; top: 20px; display: inline-block;">
                Zeaf
            </p>
            <br>
            <p style="color: #B8A184; font-size: 16px; top: 20px; display: inline-block;">
                Sign up for free.
            </p>

            <div id="form-container" style="position: relative; top: 80px;">
                <form action="/register/forward" method="post" style="position: relative; font-size: 16px; top: 20%; display: inline-block;">

                    <p>Username(4~30 characters):</p>
                    <p><input autocomplete="off" class="form-input" type="text" name="username" minlength="4" maxlength="30"
                        {% if failed and fail_status == "USERNAME_EXISTED" %}
                            style="margin: auto; border: 1px red solid;"
                        {% else %}
                            style="margin: auto;"
                        {% endif %}
                    required></p>
                    {% if failed and fail_status == "USERNAME_EXISTED" %}
                        <p style="color: red;" >Username already exist!</p>
                    {% endif %}

                    <br>
```

```
                <p>Email:</p>
                <p><input class="form-input"  type="email" name="mail" style="margin: auto;" required></p>
                {% if failed and fail_status == "MAIL_RE_ERROR" %}
                <p style="color: red;" >Please enter a valid mail.</p>
                {% endif %}

                <br>

                <p>Password(8~30 characters):</p>
                <p><input autocomplete="off" class="form-input"  type="password" name="password" style="margin: auto;" pattern="[a-zA-Z0-9!@#$%^*_|]{8,30}" required></p>

                <br>


                <p>Join as:
                    <input type="checkbox" id="teacher-select" name="account-type1" style="display: inline;" onchange="document.getElementById('student-select').checked=false;">
                        Teacher
                    <input type="checkbox" id="student-select" name="account-type2" style="display: inline;" onchange="document.getElementById('teacher-select').checked=false;">
                        Student
                </p>
                {% if failed and fail_status == "TEACHER_SECTION_ERROR" %}
                    <p style="color: red;" >Please select one.</p>
                {% endif %}

                <br>

                <p><input id="form-submit" type="submit" class="form-input" value="submit"></p>

                {% csrf_token %}
            </form>
        </div>
        <style>
            input{
                border-radius: 3px;
                border: 1px black solid;
            }
            .form-input{
                color: #ffffff;
                border: none;
                outline: none;
                background-color:#5C5D5E;
                border-radius: 0px;
            }
        </style>
    </div>

</body>
</html>
```

(四) login/login.html (登入頁面模板)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Login - Zeaf</title>
        <meta charset="UTF-8">
        <link href="https://fonts.googleapis.com/css2?family=Source+Code+Pro:wght@700&amp;display=swap" rel="stylesheet">

        {% load static %}
        <link rel="stylesheet" type="text/css" href="{% static 'basic.css' %}">

        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300&display=swap" rel="stylesheet">

        <meta name="google" value="notranslate">
    </head>
    <body>
        {% include 'public/navbar.html' %}

        <div id="register-board" style="text-align: center; border-radius: 15px; position: relative; width: 260px; left:50%; transform: translateX(-130px); height: 450px;
        background-color: none; border: 1px white solid;">
            <p style="color: #B8A184; font-size: 50px; top: 20px; display: inline-block;">
                Zeaf
            </p>
            <br>
            <p style="color: #B8A184; font-size: 16px; top: 20px; display: inline-block;">
                Login
            </p>

            <div id="form-container" style="position: relative; margin-top: 100px;">
                <form action="/login/forward" method="post" style="position: relative; font-size: 16px; top: 20%; display: inline-block;">

                    <p>Username</p>
                    <p><input autocomplete="off" class="form-input" type="text" name="username" maxlength="30"
                        {% if failed and fail_status == "USERNAME_EXISTED" %}
                            style="margin: auto; border: 1px red solid;"
                        {% else %}
                            style="margin: auto;"
                        {% endif %}
                    required></p>
                    {% if failed and fail_status == "USER_NOT_FOUND" %}
                        <p style="color: red;" >Username not found.</p>
                    {% endif %}
                    {% if failed and fail_status == "PASSWORD_DOES_NOT_MATCH" %}
                        <p style="color: red;" >Incorrect username or password.</p>
                    {% endif %}
                    {% if failed and fail_status == "UNKNOWN_ERROR" %}
                        <p style="color: red;" >Unknown error!</p>
                    {% endif %}
```

```
                <br>

                <p>Password</p>
                <p><input autocomplete="off" class="form-input"  type="password" name="password" style="margin: auto;" pattern="[a-zA-Z0-9!@#$%^*_|]{8,30}" required></p>

                <br>

                <p><input id="form-submit" type="submit" class="form-input" value="submit"></p>

                {% csrf_token %}
            </form>
        </div>

        <p style="position: relative; margin-top: 30px;">No accounts yet? <a href="/register" style="color: purple;">Sign up</a></p>

        <style>
            input{
                border-radius: 3px;
                border: 1px black solid;
            }
            .form-input{
                color: #ffffff;
                border: none;
                outline: none;
                background-color:#5C5D5E;
                border-radius: 0px;
            }
        </style>
    </div>

</body>
</html>
```

二、admin.py

```
from django.contrib import admin

from .models import Code, Question, User, UserGroup

def adminize(model, displays):
    class ObjAdmin(admin.ModelAdmin):
        list_display = displays

    admin.site.register(model, ObjAdmin)

adminize(Code,['number'])
adminize(Question,['unique_id'])
adminize(User,['email'])
adminize(UserGroup,['title'])
```

三、views.py 檢視函式、enum 函式宣告

(一) views.py imports(匯入之模組與簡稱)

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
'''
render(): render HTMLfiles in template folder as a response
redirect(): redirect page to other url
HttpResponse(): pure text response
'''

from django.views.decorators.csrf import csrf_exempt
''' function decorator, for verifying csrf token '''

from django.contrib.auth import authenticate as _auth, login as _login, logout as _logout
'''
Example:
- login(request, _auth(username, password))
- logout(request)
'''

from django.contrib.auth import get_user_model
User = get_user_model() # return the user model that is activated

from .models import * # import models

import os, time, re, ast, json, asyncio
'''
os      | creates, deletes, and runs python files for judge system
time    | stops the script for seconds. Used in judge system
re      | regular expression matching
ast     | converts text expression to python list type with all nodes in string and sub-lists
json    | converts string <=> list
asyncio | run function in async way
'''

from app_questions.request_fields import * # import request_fields.py
from app_questions.object_filters import * # import object_filters.py

def noself(f):
    @classmethod
    def new_f(self, *args, **kwargs):
        return f(*args, **kwargs) # ignore the the self param
    return new_f
'''
The classes below was created for managing and distinguishing view functions.
Each class methods will call as view functions, hence the "self" parameter above is ignorable.
The function decorator noself() was created to avoid misplacement of parameters in function.
Each class methods above should use this decorator.
'''
```

(二) Login 類別：登入

```python
@noself
@csrf_exempt
def login(request):
    rf = AuthenticationErr_RequestField(request)

    valid = rf.isValid()

    if valid.result:
        #authenticate checks if credentials exists in db

        if not User.objects.filter(username=request.POST['username']).exists():
            return redirect("/login?fail=1")
        else:
            user = _auth(username=request.POST['username'],password=request.POST['password'])

            if user is not None:
                try:
                    _login(request, user)

                    return redirect("/")

                except BaseException as err:
                    print("Error in Login.login(): {}".format(err))

                    return redirect("/login?fail=3")

            else:
                return redirect("/login?fail=2")
    else:
        return HttpResponse(" ".join(valid.reason))
```

(三) Group 類別：自訂群組功能

```python
@noself
@csrf_exempt
def submit(request):
    if not (request.user.is_authenticated and request.user.isTeacher):
        return redirect("/groups/new-group?fail=1")

    name = request.user.username

    rf = GroupSubmitErr_RequestField(request)

    valid = rf.isValid()

    if valid.result:
        question = rf.validQuestions

        try:
            name = request.user.username

            new_group = UserGroup(
                title = rf.data["title"],
                questionJson = json.dumps(rf.validQuestions),
                author = name,
            )

            new_group.save()

            return redirect("/groups/group/{}".format(new_group.urlCode))

        except BaseException as err:
            print("Error in Group_submit_form.submit(): {}".format(err))

            return redirect("/groups/new-group?fail=2")

    else:
        return HttpResponse(" ".join(valid.reason))
```

(四) enum 函式宣告

```python
def enum(li, start=1):
    return [[i, li] for i, li in enumerate(li, start=start)] # to support django HTML template iterations
''' Django HTML template does not support enumerate() iterations in {% for %} blocks. This is an alternate function to replace enumerate() '''
```

四、request_fields.py 請求表單 LoginPageErr_RequestField 類別

```python
class LoginPageErr_RequestField(RequestField):
    ''' Login.show() '''

    def __init__(self, request):
        super(LoginPageErr_RequestField, self).__init__(request)

    def isValid(self):
        methodValid = (self.method=="GET")

        try:
            statusMinValid = (int(self.data.get('fail', -1))==-1 or int(self.data.get('fail', -1))>0)
            statusMaxValid = (int(self.data.get('fail', -1))<=3)
            statusValid = True

        except ValueError:
            statusMinValid = True
            statusMaxValid = True
            statusValid = False

        return FieldIsValid(locals())

    @property
    def failed(self):
        return "fail" in self.data

    @property
    def fail_status(self):
        if self.failed:
            status_dict = {
                '1': "USER_NOT_FOUND",
                '2': "PASSWORD_DOES_NOT_MATCH",
                '3': "UNKNOWN_ERROR",
            }

            fail_status = status_dict.get(self.data["fail"], "UNKNOWN_KEY")

            return fail_status

        return None
```

# 【評語】190023

1. 有很好的系統平台程式撰寫知識能支援程式教學平台。

2. 只是寫研究報告像在寫使用手冊 使用者使用滿意度如何？系統功能完整度與使用者回應與評量？使用本系統是否學生學習成效有增加。