

Normal Online Judge (NOJ) 系統測試報告書

專案名稱	Normal Online Judge (NOJ)
文件版本	V2.2
測試版本	Release Candidate
報告日期	2025 年 12 月 29 日

文件修訂歷史

版本	日期	修訂者	修訂內容
V1.0	2025/12/8	李子頤	初始架構
V2.0	2025/12/15	鐘珮甄	重構架構
V2.1	2025/12/22	呂凱崴、鐘珮甄	格式調整
V2.2	2025/12/29	呂凱崴	最後修正

目錄

- 摘要
- 測試目標
- 測試環境
- 測試結果摘要
- 詳細測試案例與執行結果
- 安全性測試
- 結論與建議
- 附錄

1. 摘要

本報告書為 **Normal Online Judge (NOJ)** 系統之正式測試報告，涵蓋系統功能測試、整合測試及安全性測試之執行結果。

測試期間自 **2025 年 12 月 1 日 (Week 14)** 至 **2025 年 12 月 29 日 (Week 18)**，共計 **4 週 + 1 日**。測試團隊依據《測試計畫書 V2.1》所規範之測試案例，針對 Feature 1 至 Feature 5 進行完整測試驗證。

測試結果總覽

項目	數值
總測試案例數	126
通過數量	126
失敗數量	0
阻塞數量	0
通過率	100%

結論：系統已通過所有功能測試案例，品質符合上線標準，**可進行正式發布**。

2. 測試目標

本次測試之主要目標如下：

- 驗證 NOJ 系統之所有功能需求是否正確實作
- 確認各模組間之整合運作無誤
- 驗證不同使用者角色（學生、老師、助教、管理員）使用系統時的正確性
- 確保系統穩定性與可靠性達到上線標準

3. 測試環境

3.1 前端測試環境

項目	說明
作業系統	Ubuntu 22.04
框架	Vue Router / Pinia / Vue l18n
建置工具	Vite / pnpm

3.2 後端測試環境

項目	說明
作業系統	Windows 10
架構	Flask (Python Web Framework)
WSGI 伺服器	Gunicorn
測試框架	Pytest + Pytest-cov
資料庫	MongoDB
容器環境	Windows Docker

3.3 ??????????????

4. 測試結果摘要

4.1 測試結果統計

4.1.1 整體統計

系統通過所有 126 個測試案例

4.1.2 各 Feature 測試結果

Feature	測試案例數	通過	失敗	通過率
Feature 1 - AI 助教	22	22	0	100%
Feature 2 - 類 LeetCode 練習	17	17	0	100%
Feature 3 - API Token	30	30	0	100%
Feature 4 - Pipeline	35	35	0	100%
Feature 5 - 討論區	22	22	0	100%
總計	126	126	0	100%

5. 詳細測試案例與執行結果

Feature 1 AI-TA

T-F1-001: 老師可以新增API KEY

項目	內容
測試編號	T-F1-001
測試項目	老師可以新增API KEY
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在Add New API Key的地方輸入Key Name : "teacherkey1"，以及正確API Key Value 4. 點擊「+ ADD KEY」 5. Existing Keys 列表應出現一筆新的 API Key : NAME = "teacherkey1" Masked = "(中間有被****遮擋)" Created By = "TEACHER" Active = 開啟 二、後端 API 回應驗證 6. 檢查送出的 Request 是否正確：API : POST /api/course/test/ai/key JSON : { "key_name":"teacherkey1", "value":"(正確API Key Value)", "is_active":true, "created_by":"TEACHER" } 7. 檢查 Response : status code = 200 JSON : { "data": { "id": "(隨機產生)" }, "message": "Key added.", "status": "ok" }
預期結果	1.列表新增一筆API Key且內容正確 2.後端回應200
執行結果	PASS 200

T-F1-002: 新增API KEY流程錯誤 (重複的Key Name)

項目	內容
測試編號	T-F1-002
測試項目	新增API KEY流程錯誤 (重複的Key Name)
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有一個已存在的API Key Name："teacherkey1" 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在Add New API Key的地方輸入Key Name："teacherkey1"，以及正確API Key Value 4. 點擊「+ ADD KEY」 5. 應跳出錯誤訊息"Key name 'teacherkey1' already exists in this course." 二、後端 API 回應驗證 6. 檢查送出的 Request 是否正確：API：POST /api/course/test/ai/key JSON： { "key_name":"teacherkey1", "value":"(正確API Key Value)", "is_active":true, "created_by":"TEACHER" } 7. 檢查 Response：status code = 400 JSON： { "data": null, "message": "Key name 'teacherkey1' already exists in this course.", "status": "err" }

項目	內容
預期結果	1.列表沒有新增一筆API Key且顯示錯誤訊息 2.後端回應400
執行結果	PASS 400

T-F1-003: 新增API KEY流程錯誤 (沒有輸入Key Name)

項目	內容
測試編號	T-F1-003
測試項目	新增API KEY流程錯誤 (沒有輸入Key Name)
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在Add New API Key的地方不輸入Key Name， 以及正確輸入API Key Value 4. 點擊「+ ADD KEY」 5. 跳出錯誤提示"Key name and value are required."
預期結果	1.列表沒有新增一筆API Key且有正確跳出錯誤訊息
執行結果	PASS 跳出錯誤訊息"Key name and value are required."

T-F1-004: 新增API KEY流程錯誤 (沒有輸入Key Value)

項目	內容
測試編號	T-F1-004
測試項目	新增API KEY流程錯誤 (沒有輸入Key Value)
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在Add New API Key的地方輸入Key Name = "test"，然後不輸入API Key Value 4. 點擊「+ ADD KEY」 5. 跳出錯誤提示"Key name and value are required."
預期結果	1.列表沒有新增一筆API Key且有正確跳出錯誤訊息
執行結果	PASS 跳出錯誤訊息"Key name and value are required."

T-F1-005: 老師可以修改API KEY名稱

項目	內容
測試編號	T-F1-005
測試項目	老師可以修改API KEY名稱
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有一個已存在的API Key Name："teacherkey1" 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在下方Existing Keys列表找到 "teacherkey1"，並修改為 "new_teacherkey1" 4. 點擊「SAVE」 5. Existing Keys 列表中該API Key 的 Name 應該被更新為 "new_teacherkey1" 6. 跳出提示 "Key updated." 二、後端 API 回應驗證 7. 檢查送出的 Request 是否正確：API：PATCH /api/course/test/ai/key/(ID) JSON： { "key_name":"new_teacherkey1", "is_active":true } 7. 檢查 Response：status code = 200 JSON： { "data": null, "message": "Key updated.", "status": "ok" }
預期結果	1.列表成功修改API Key Name且內容正確 2.後端回應200

項目	內容
執行結果	PASS 200

T-F1-006: 老師可以修改API KEY active

項目	內容
測試編號	T-F1-006
測試項目	老師可以修改API KEY active
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有一個已存在的API Key Name："new_teacherkey1"，且Active為開啟狀態 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在下方Existing Keys列表找到 "new_teacherkey1" 4. 點擊「Active」按鈕 5. Existing Keys 列表中該API Key 的 Active應為關閉狀態 6. 跳出提示 "Key updated." 二、後端 API 回應驗證 7. 檢查送出的 Request 是否正確：API：PATCH /api/course/test/ai/key/(ID) JSON： { "key_name":"new_teacherkey1", "is_active":false } 7. 檢查 Response：status code = 200 JSON： { "data": null, "message": "Key updated.", "status": "ok" }
預期結果	1.列表中該 API Key 的active被正確關閉 2.後端回應200

項目	內容
執行結果	PASS 200

T-F1-007: 老師可以刪除API KEY

項目	內容
測試編號	T-F1-007
測試項目	老師可以刪除API KEY
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher， 且有一個已存在的API Key Name："new_teacherkey1" 2. 前端 AI Setting 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → AI Setting」 3. 在下方Existing Keys列表找到 "new_teacherkey1" 4. 點擊「DELETE」按鈕，然後按下OK 5. Existing Keys 列表中該API Key 會消失 6. 跳出提示 "Key deleted." 二、後端 API 回應驗證 7. 檢查 Response：status code = 200 JSON： { "data": null, "message": "Key deleted.", "status": "ok" }
預期結果	1.列表中該 API Key 正確被刪除 2.後端回應200
執行結果	PASS 200

T-F1-008: 老師新增 Problem 時可以選擇是否開啟AI-TA

項目	內容
測試編號	T-F1-008
測試項目	老師新增 Problem 時可以選擇是否開啟AI-TA
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「COURSE → Problems」 3. 點擊「+ NEW」按鈕 4. 點擊「Set Configuration」按鈕 5. 可以看到「AI VTuber」按鈕，且可被點擊
預期結果	1.老師頁面可以成功看到「AI VTuber」按鈕，且可被點擊， 點擊後會展開頁面
執行結果	PASS 可以成功開啟功能

T-F1-009: 老師設置AI-TA功能時，可以選擇AI Model

項目	內容
測試編號	T-F1-009
測試項目	老師設置AI-TA功能時，可以選擇AI Model
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → Problems」 點擊「+ NEW」按鈕 點擊「Set Configuration」按鈕 可以看到「AI VTuber」按鈕，且可被點擊 將預設的AI Model (gemini 2.5 flash lite) 修改成gemini 2.5 pro
預期結果	1.老師可以選擇不同的AI Model並成功建立Problem
執行結果	PASS 有三種模型可供選擇

T-F1-010: 老師設置AI-TA功能時，可以看到active與非active的API Key

項目	內容
測試編號	T-F1-010
測試項目	老師設置AI-TA功能時，可以看到active與非active的API Key
先前條件	- 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有一個狀態為active的API Key，Name為："teacherkey1"，另一個狀態為inactive的API Key，Name為："nothing1" - 前端 Problems 頁面可正常操作 - 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → Problems」 點擊「+ NEW」按鈕 點擊「Set Configuration」按鈕 可以看到「AI VTuber」按鈕，且可被點擊 可以看到下方的 Select API Keys中，左邊的active keys有一個Name為"teacherkey1"的API Key，右邊的inactive keys有一個Name為"nothing1"的API Key
預期結果	1.老師可以正確看到不同狀態的API Key
執行結果	PASS 可以正確顯示

T-F1-011: 老師設置AI-TA功能時，可以選擇多個API Key

項目	內容
測試編號	T-F1-011
測試項目	老師設置AI-TA功能時，可以選擇多個API Key
先前條件	- 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有一個狀態為active的API Key，Name為："teacherkey1"，另一個狀態為inactive的API Key，Name為："nothing1" - 前端 Problems 頁面可正常操作 - 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → Problems」 點擊「+ NEW」按鈕 點擊「Set Configuration」按鈕 可以看到「AI VTuber」按鈕，且可被點擊 可以看到下方的 Select API Keys中，左邊的active keys有一個Name為"teacherkey1"的API Key，右邊的inactive keys有一個Name為"nothing1"的API Key 可以同時選擇"teacherkey1"、"nothing1"這兩個API Key
預期結果	1.可以同時選擇多個API Key並成功建立Problem
執行結果	PASS 可以同時選擇多個且會顯示數量

T-F1-012: 老師設置AI-TA功能時，可以看到Key的設置建議

項目	內容
測試編號	T-F1-012
測試項目	老師設置AI-TA功能時，可以看到Key的設置建議
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → Problems」 點擊「+ NEW」按鈕 點擊「Set Configuration」按鈕 可以看到「AI VTuber」按鈕，且可被點擊 可以看到下方的 Select API Keys中，點擊旁邊的問號按鈕， 會顯示系統的Suggested keys以及Student count跟Reason
預期結果	1.系統可以根據Student count提出Suggested keys與理由
執行結果	PASS 可以成功顯示建議數量

T-F1-013: 老師設置AI-TA功能時，可以設置學生Token上限

項目	內容
測試編號	T-F1-013
測試項目	老師設置AI-TA功能時，可以設置學生Token上限
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → Problems」 點擊「+ NEW」按鈕 點擊「Set Configuration」按鈕 可以看到「AI VTuber」按鈕，且可被點擊 (還沒看到流程)
預期結果	1.老師可以設置學生Token上限
執行結果	PASS

T-F1-014: 老師可以看到各個 API Key 的 Token 用量

項目	內容
測試編號	T-F1-014
測試項目	老師可以看到各個 API Key 的 Token 用量
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有設定一個可以使用AI-TA功能的Problem，Problem名稱為test00（使用API Key Name為"teacherkey0"的Key來建立Problem） 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入 使用學生帳號操作test00中的AI-TA功能 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → AI Setting → Usage」 在下方API Usage (test)列表找到 "teacherkey0" 點擊 "teacherkey0"，可以看到RANK 1 的 Problem Name為test00且有顯示Token Used
預期結果	1.老師可以正確看到Token用量
執行結果	PASS 可以成功顯示Token用量

T-F1-015: 老師可以看到API Key在幾個題目中被使用

項目	內容
測試編號	T-F1-015
測試項目	老師可以看到API Key在幾個題目中被使用
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有設定兩個可以使用AI-TA功能的Problem，Problem名稱分別為test01、test02（使用API Key Name為"teacherkey1"的Key來建立Problem） 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → AI Setting → Usage」 在下方API Usage (test)列表找到 "teacherkey1" 檢查旁邊的訊息是否為Used by 2 problems
預期結果	Usage頁面可以正確顯示API Key被使用於多少個Problems之中
執行結果	PASS 可以成功顯示用於多少題目之中

T-F1-016: 老師可以看到單一API Key下的用量排名且正確顯示

項目	內容
測試編號	T-F1-016
測試項目	老師可以看到單一API Key下的用量排名且正確顯示
先前條件	- 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且有設定兩個可以使用AI-TA功能的Problem，Problem名稱分別為test01、test02（使用API Key Name為"teacherkey1"的Key來建立Problem） - 前端 Problems 頁面可正常操作 - 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入 使用學生帳號操作AI-TA功能，且 test02 所使用的Token用量需大於 test01 以老師帳號 TEACHER，密碼為 teacher 登入 進入「COURSE → AI Setting → Usage」 在下方API Usage (test)列表找到 "teacherkey1" 點擊 "teacherkey1"，可以看到RANK 1 的Problem Name為 test02，RANK 2 的Problem Name為 test01
預期結果	1.老師可以正確看到Token用量排名
執行結果	PASS 可以成功顯示用量與排名

T-F1-017: AI助教功能關閉後看不到AI對話框

項目	內容
測試編號	T-F1-017
測試項目	AI助教功能關閉後看不到AI對話框
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher，且建立了一個課程test1中有一個Problem叫做test（未開啟AI-TA功能）。同時有一個學生帳號 STUDENT，密碼為 student被加入在此課程當中。 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入 使用學生帳號查看課程test1中的Problem-test， 檢查是否看不到AI-TA對話框
預期結果	學生看不到沒有開啟AI-TA功能的對話框
執行結果	PASS 對話框未顯示

T-F1-018: 老師可以在AI-TA功能設置的地方改變APIKey的active狀態

項目	內容
測試編號	T-F1-018
測試項目	老師可以在AI-TA功能設置的地方改變APIKey的active狀態
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher， 且有一個課程為test1 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以老師帳號 TEACHER，密碼為 teacher 登入 進入course → test1 → Problems → +New → Set Configuration → AI-TA 將active keys中的APIKey拖移至右邊的inactive keys submit (儲存Problem) 之後，回去AI Management的地方檢查該APIKey是否變成inactive
預期結果	老師可以成功在Problem的AI- TA的設置介面改變APIKeys的active狀態
執行結果	PASS 200

以下測試預計使用手動測試

T-F1-019: 學生可以跟AI-TA對話

項目	內容
測試編號	T-F1-019
測試項目	學生可以跟AI-TA對話
先前條件	1. 有一個已註冊的學生帳號 STUDENT，密碼為 student， 有一個有AI-TA功能的Problem 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入 使用學生帳號操作AI-TA功能
預期結果	1.學生可以在有AI-TA功能的Problem中使用AI-TA並產生對話
執行結果	PASS

T-F1-020: AI-TA的回覆需正確且為題目相關內容

項目	內容
測試編號	T-F1-020
測試項目	AI-TA的回覆需正確且為題目相關內容
先前條件	1. 有一個已註冊的學生帳號 STUDENT，密碼為 student， 有一個有AI-TA功能的Problem 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入，有一個有AI-TA功能的Problem 使用學生帳號操作AI-TA功能 檢查AI-TA的回答合理性
預期結果	1.學生可以在有AI-TA功能的Problem中使用AI-TA並產生對話， 並且回答內容與題目內容相關
執行結果	PASS

T-F1-021: AI-TA不會直接提供給學生答案

項目	內容
測試編號	T-F1-021
測試項目	AI-TA不會直接提供給學生答案
先前條件	1. 有一個已註冊的學生帳號 STUDENT，密碼為 student， 有一個有AI-TA功能的Problem 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入 使用學生帳號操作AI-TA功能 檢查AI-TA的回答合理性且不會提供學生答案
預期結果	1.學生可以在有AI-TA功能的Problem中使用AI-TA並產生對話， 並且回答內容與題目內容相關，但不會直接提供給學生答案
執行結果	PASS

T-F1-022: AI-TA的表情正常顯示

項目	內容
測試編號	T-F1-022
測試項目	AI-TA的表情正常顯示
先前條件	1. 有一個已註冊的學生帳號 STUDENT，密碼為 student， 有一個有AI-TA功能的Problem 2. 前端 Problems 頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 以學生帳號 STUDENT，密碼為 student 登入 使用學生帳號操作AI-TA功能 檢查使用功能的過程中AI-TA的表情能正常切換並顯示
預期結果	1.AI-TA可以依照不同情緒有不同的表情顯示
執行結果	PASS

Feature 2 Leetcode

T-F2-001: 老師可設定problem

項目	內容
測試編號	T-F2-001
測試項目	老師可設定problem
先前條件	老師已建立Course且老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊New 5.在Name的輸入框輸入Prob1 6.在Quotas的輸入框輸入-1 7.在Description的輸入框輸入plus 8.在Input的輸入框輸入10 20 9.在Output的輸入框輸入30 10.在Hint的輸入框輸入+ 11.在subtask case上傳Prob1.zip 12.在subtask分數的輸入框輸入100 13.點擊Submit
預期結果	成功Create。
執行結果	pass

T-F2-002: Admin可設定problem

項目	內容
測試編號	T-F2-002
測試項目	Admin可設定problem
先前條件	Admin已建立Course且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊New 5.在Name的輸入框輸入Prob1 6.在Quotas的輸入框輸入-1 7.在Description的輸入框輸入plus 8.在Input的輸入框輸入10 20 9.在Output的輸入框輸入30 10.在Hint的輸入框輸入+ 11.在subtask case上傳Prob1.zip 12.在subtask分數的輸入框輸入100 13.點擊Submit
預期結果	成功Create。
執行結果	pass

T-F2-003: 學生可看見problem

項目	內容
測試編號	T-F2-003
測試項目	學生可看見problem
先前條件	problem已開放且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1的PID
預期結果	學生可看到設定之Name、Quotas、Description、Input、Output、Hint、subtask 分數，與老師設定的一致。
執行結果	pass

T-F2-004: 學生提交已開放problem之程式碼

項目	內容
測試編號	T-F2-004
測試項目	學生提交已開放problem之程式碼
先前條件	problem已開放且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1的PID 5.點擊Submit 6.貼上add.cpp程式碼 7.點擊Submit
預期結果	成功Submit，並給予分數。
執行結果	pass

T-F2-005: 學生提交未開放problem之程式碼

項目	內容
測試編號	T-F2-005
測試項目	學生提交未開放problem之程式碼
先前條件	problem未開放(建立後Hidden) 且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems
預期結果	學生無法看到該Prob2。
執行結果	pass

T-F2-006: 學生提交未開放homework之程式碼

項目	內容
測試編號	T-F2-006
測試項目	學生提交未開放homework之程式碼
先前條件	homework未開放(過期) 且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Homeworks 4.點擊Hw1中的Prob1
預期結果	學生無法看到Submit 按鈕。
執行結果	pass

T-F2-007: 學生提交有Quotas之程式碼

項目	內容
測試編號	T-F2-007
測試項目	學生提交有Quotas之程式碼
先前條件	Quotas為3之problem且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submit 6.貼上add.cpp程式碼 7.點擊Submit 6.第五到第七重複三次
預期結果	學生可提交該problem三次。
執行結果	pass

T-F2-008: 學生提交已達Quotas之程式碼

項目	內容
測試編號	T-F2-008
測試項目	學生提交已達Quotas之程式碼
先前條件	Quotas為3之problem且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submit 6.貼上add.cpp程式碼 7.點擊Submit 6.第五到第七重複四次
預期結果	學生可提交該problem三次，但第四次不可提交。
執行結果	pass

T-F2-009: 老師上傳公開測資

項目	內容
測試編號	T-F2-009
測試項目	老師上傳公開測資
先前條件	problem已建立且老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊New 5.在Problem name 的輸入框輸入Prob1 6.點擊Set Configuration 7.在language 中選擇py 8.點擊Trail Mode 9.在Upload Public Test Data (.zip)上傳TestCase.zip 10.在Upload AC files上傳ACcode.py 11.點擊Set Test Data 12.在Upload Test Data Zip上傳TestCase.zip 13.在Score 的輸入框輸入100 14.點擊Submit
預期結果	老師可成功上傳公開測資zip
執行結果	pass

T-F2-010: Admin上傳公開測資

項目	內容
測試編號	T-F2-010
測試項目	Admin上傳公開測資
先前條件	problem已建立且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊New 5.在Problem name 的輸入框輸入Prob1 6.點擊Set Configuration 7.在language 中選擇py 8.點擊Trail Mode 9.在Upload Public Test Data (.zip)上傳TestCase.zip 10.在Upload AC files上傳ACcode.py 11.點擊Set Test Data 12.在Upload Test Data Zip上傳TestCase.zip 13.在Score 的輸入框輸入100 14.點擊Submit
預期結果	Admin可成功上傳公開測資zip
執行結果	pass

T-F2-011: 學生查看公開測資

項目	內容
測試編號	T-F2-011
測試項目	學生查看公開測資
先前條件	公開測資 zip已上傳且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊TEST 6.View Public
預期結果	學生看到公開測資 zip內容與上傳的內容一致。
執行結果	pass

T-F2-012: 學生測試程式

項目	內容
測試編號	T-F2-012
測試項目	學生測試程式
先前條件	測試模式已開啟且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊TEST 6.貼上ACcode.py 7.Run
預期結果	成功Submit，並給予分數。
執行結果	pass

T-F2-013: 學生查看Submissions

項目	內容
測試編號	T-F2-013
測試項目	學生查看Submissions
先前條件	已存在Submissions且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submissions
預期結果	學生看到Submissions，資料與後端資料一致。
執行結果	pass

T-F2-014: 老師查看Submissions

項目	內容
測試編號	T-F2-014
測試項目	老師查看Submissions
先前條件	已存在Submissions且老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submissions
預期結果	老師看到Submissions，資料與下載內容一致。
執行結果	pass

T-F2-015: Admin查看Submissions

項目	內容
測試編號	T-F2-015
測試項目	Admin查看Submissions
先前條件	已存在Submissions且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submissions
預期結果	Admin看到Submissions，資料與下載內容一致。
執行結果	pass

T-F2-016: 老師下載Submissions

項目	內容
測試編號	T-F2-016
測試項目	老師下載Submissions
先前條件	已存在problem且老師已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submissions 6.點擊Download
預期結果	老師下載Submissions，資料與下載內容一致。
執行結果	pass

T-F2-017: Admin下載Submissions

項目	內容
測試編號	T-F2-017
測試項目	Admin下載Submissions
先前條件	已存在problem且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊Problems 4.點擊Prob1 5.點擊Submissions 6.點擊Download
預期結果	Admin下載Submissions，資料與下載內容一致。
執行結果	pass

Feature 3 API Token 驗證系統

T-F3-001: 學生可以建立Token

項目	內容
測試編號	T-F3-001
測試項目	學生可以建立Token
先前條件	1. 有一個已註冊的學生帳號 STUDENT，密碼為 student 2. 前端 Token 管理頁面可正常操作 3. 後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 點擊「+ Create new secrect key」 4. 輸入 Name：stu_token_001' 5. 選擇 Scope：read:user 6. 選取到期日：2025-12-27 7. 點擊「Create」 8. Token 列表應出現一筆新的 Token： NAME = "stu_token_001" OWNER = "STUDENT" STATUS = "active" CREATED = 測試日期 LAST USED = N/A DUE_TIME = 2026-12-27 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確：API：POST /profile/api_token/create JSON： { "data":{ "Due_Time": "2025-12-27", "Name": "stu_token_001" "Scope": ["read:user"] } } 10. 檢查 Response：status code = 200 OK JSON： { "data":{ "Message": "Token Created", "Token": "(隨機產生)", "Type": "OK" } "Message": "Token Created"

項目	內容
	"Token": "(隨機產生)" "Type": "OK" "message": "Token Created" "status": "ok" }
預期結果	1.列表新增一筆 Token且內容正確 2.後端回應200
執行結果	PASS 200

T-F3-002: 老師可以建立Token

項目	內容
測試編號	T-F3-002
測試項目	老師可以建立Token
先前條件	1. 有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2. 前端 Token 管理頁面可正常操作 3. 後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「PROFILE → API Token」 3. 點擊「+ Create new secrect key」 4. 輸入 Name : teacher_token_002 5. 選擇 Scope : read:user 6. 選取到期日 : 2025-12-27 7. 點擊「Create」 8. Token 列表應出現一筆新的 Token : NAME = "teacher_token_002" OWNER = "TEACHER" STATUS = "active" CREATED = 測試日期 LAST USED = N/A DUE_TIME = 2026-12-27 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/create JSON : { "data":{ "Due_Time": "2025-12-27", "Name": "teacher_token_002" "Scope": ["read:user"] } } 10. 檢查 Response : status code = 200 OK JSON : { "data":{ "Message": "Token Created", "Token": "(隨機產生)", "Type": "OK" } "Message": "Token Created"

項目	內容
	"Token": "(隨機產生)" "Type": "OK" "message": "Token Created" "status": "ok" }
預期結果	1.列表新增一筆 Token且內容正確 2.後端回應200
執行結果	PASS 200

T-F3-003: 助教可以建立Token

項目	內容
測試編號	T-F3-003
測試項目	助教可以建立Token
先前條件	1. 有一個已註冊的助教帳號 TA，密碼為 ta 2. 前端 Token 管理頁面可正常操作 3. 後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以助教帳號 TA，密碼為 ta 登入 2. 進入「PROFILE → API Token」 3. 點擊「+ Create new secrect key」 4. 輸入 Name : ta_token_003 5. 選擇 Scope : read:user 6. 選取到期日 : 2025-12-27 7. 點擊「Create」 8. Token 列表應出現一筆新的 Token : NAME = "ta_token_003" OWNER = "TA" STATUS = "active" CREATED = 測試日期 LAST USED = N/A DUE_TIME = 2026-12-27 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/create JSON : { "data":{ "Due_Time": "2025-12-27", "Name": "ta_token_003" "Scope": ["read:user"] } } 10. 檢查 Response : status code = 200 OK JSON : { "data":{ "Message": "Token Created", "Token": "(隨機產生)", "Type": "OK" } "Message": "Token Created"

項目	內容
	"Token": "(隨機產生)" "Type": "OK" "message": "Token Created" "status": "ok" }
預期結果	1.列表新增一筆 Token且內容正確 2.後端回應200
執行結果	PASS 200

T-F3-004: 管理員可以建立Token

項目	內容
測試編號	T-F3-004
測試項目	管理員可以建立Token
先前條件	1. 有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2. 前端 Token 管理頁面可正常操作 3. 後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以管理員帳號 first_admin · 密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 點擊「+ Create new secrect key」 4. 輸入 Name : admin_token_004 5. 選擇 Scope : read:user 6. 選取到期日 : 2025-12-27 7. 點擊「Create」 8. Token 列表應出現一筆新的 Token : NAME = "admin_token_004" OWNER = "first_admin" STATUS = "active" CREATED = 測試日期 LAST USED = N/A DUE_TIME = 2026-12-27 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/create JSON : { "data":{ "Due_Time": "2025-12-27", "Name": "admin_token_004" "Scope": ["read:user"] } } 10. 檢查 Response : status code = 200 OK JSON : { "data":{ "Message": "Token Created", "Token": "(隨機產生)", "Type": "OK" }

項目	內容
	"Message": "Token Created" "Token": "(隨機產生)" "Type": "OK" "message": "Token Created" "status": "ok" }
預期結果	1.列表新增一筆 Token且內容正確 2.後端回應200
執行結果	PASS 200

T-F3-005: 建立 Token 測試 (錯誤流程：未輸入Name)

項目	內容
測試編號	T-F3-005
測試項目	建立 Token 測試 (錯誤流程：未輸入Name)
先前條件	1. 有一個已註冊的學生帳號 STUDENT，密碼為 student 2. 前端 Token 管理頁面可正常操作 3. 後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 點擊「+ Create new secrect key」 4. 直接點擊「Create」 5. 跳出錯誤提示："Please remember to name your token"
預期結果	1. 前端跳出錯誤提示 2. 列表沒有新增Token
執行結果	錯誤訊息：Please remember to name your token

T-F3-006: 學生可以編輯自己的 Token Scope

項目	內容
測試編號	T-F3-006
測試項目	學生可以編輯自己的 Token Scope
先前條件	1.有一個已註冊的學生帳號 STUDENT，密碼為 student 2.有一個已經建立成功的Token，名稱為stu_token_001，Scope為["read:user"]，OWNER為STUDENT 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "stu_token_001"的row 4. 點擊該row的編輯按鈕 5. 選擇 Scope : write:user 6. 點擊「UPDATE」 7. 點擊該row的「VIEW」按鈕 8. 檢查Scope是否為 : write:user 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/edit JSON : <pre>{ "data":{ "Due_Time": "2025-12-27", "Name": "stu_token_001" "Scope": ["write:user"], } }</pre> 10. 檢查 Response : status code = 200 OK JSON : <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的Scope列表有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-007: 老師可以編輯自己的 Token Scope

項目	內容
測試編號	T-F3-007
測試項目	老師可以編輯自己的 Token Scope
先前條件	1.有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2.有一個已經建立成功的Token，名稱為teacher_token_002，Scope為["read:user"]，OWNER為TEACHER 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "teacher_token_002"的row 4. 點擊該row的編輯按鈕 5. 選擇 Scope : write:user 6. 點擊「UPDATE」 7. 點擊該row的「VIEW」按鈕 8. 檢查Scope是否為 : write:user 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/edit JSON : <pre>{ "data":{ "Due_Time": "2025-12-27", "Name": "teacher_token_002" "Scope": ["write:user"], } }</pre> 10. 檢查 Response : status code = 200 OK JSON : <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的Scope列表有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-008: 助教可以編輯自己的 Token Scope

項目	內容
測試編號	T-F3-008
測試項目	助教可以編輯自己的 Token Scope
先前條件	1.有一個已註冊的助教帳號 TA，密碼為 ta 2.有一個已經建立成功的Token，名稱為ta_token_003，Scope為["read:user"]，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以助教帳號 TA，密碼為 ta 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "ta_token_003"的row 4. 點擊該row的編輯按鈕 5. 選擇 Scope : write:user 6. 點擊「UPDATE」 7. 點擊該row的「VIEW」按鈕 8. 檢查Scope是否為 : write:user 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/edit JSON : <pre>{ "data":{ "Due_Time": "2025-12-27", "Name": "ta_token_002" "Scope": ["write:user"], } }</pre> 10. 檢查 Response : status code = 200 OK JSON : <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的Scope列表有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-009: 管理員可以編輯自己的 Token Scope

項目	內容
測試編號	T-F3-009
測試項目	管理員可以編輯自己的 Token Scope
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有一個已經建立成功的Token，名稱為admin_token_004，Scope為["read:user"]，OWNER為first_admin 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以管理員帳號 first_admin · 密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "admin_token_004"的row 4. 點擊該row的編輯按鈕 5. 選擇 Scope : write:user 6. 點擊「UPDATE」 7. 點擊該row的「VIEW」按鈕 8. 檢查Scope是否為 : write:user 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確 : API : POST /profile/api_token/edit JSON : <pre>{ "data":{ "Due_Time": "2025-12-27", "Name": "admin_token_004" "Scope": ["write:user"], } }</pre> 10. 檢查 Response : status code = 200 OK JSON : <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status": "ok" }</pre>
預期結果	1.Token的Scope列表有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-010: 管理員可以編輯其他人的 Token Scope

項目	內容
測試編號	T-F3-010
測試項目	管理員可以編輯其他人的 Token Scope
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有三個已經建立成功的Token： 第一個名稱為stu_token_001，Scope為["write:user"]，OWNER為STUDENT 第二個名稱為teacher_token_002，Scope為["write:user"]，OWNER為TEACHER 第三個名稱為ta_token_003，Scope為["write:user"]，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	以相同步驟分別測試修改stu_token_001、teacher_token_002、ta_token_003 一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "stu_token_001"的row 4. 點擊該row的編輯按鈕 5. 選擇 Scope：read:courses 6. 點擊「UPDATE」 7. 點擊該row的「VIEW」按鈕 8. 檢查Scope是否為：write:user 二、後端 API 回應驗證 9. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-27", "Name": "stu_token_001" "Scope": ["read:courses"], } }</pre> 10. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的Scope列表都有被正確更新 2.後端回應200

項目	內容
執行結果	PASS 200

T-F3-011: 學生可以編輯自己的 Token 到期日

項目	內容
測試編號	T-F3-011
測試項目	學生可以編輯自己的 Token 到期日
先前條件	1.有一個已註冊的學生帳號 STUDENT，密碼為 student 2.有一個已經建立成功的Token，名稱為stu_token_001，DUE TIME為2025-12-27，OWNER為STUDENT 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "stu_token_001"的row 4. 點擊該row的編輯按鈕 5. 選擇日期：2025-12-30 6. 點擊「UPDATE」 7. 檢查該Token的DUE TIME是否為：2025-12-30 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： { "data":{ "Due_Time": "2025-12-30", "Name": "stu_token_001" "Scope": ["write:user"], } } 9. 檢查 Response：status code = 200 OK JSON： { "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }
預期結果	1.Token的DUE TIME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-012: 老師可以編輯自己的 Token 到期日

項目	內容
測試編號	T-F3-012
測試項目	老師可以編輯自己的 Token 到期日
先前條件	1.有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2.有一個已經建立成功的Token，名稱為teacher_token_002，DUE TIME為2025-12-27，OWNER為TEACHER 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "teacher_token_002"的row 4. 點擊該row的編輯按鈕 5. 選擇日期：2025-12-30 6. 點擊「UPDATE」 7. 檢查該Token的DUE TIME是否為：2025-12-30 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： { "data":{ "Due_Time": "2025-12-30", "Name": "teacher_token_002" "Scope": ["write:user"], } } 9. 檢查 Response：status code = 200 OK JSON： { "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }
預期結果	1.Token的DUE TIME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-013: 助教可以編輯自己的 Token 到期日

項目	內容
測試編號	T-F3-013
測試項目	助教可以編輯自己的 Token 到期日
先前條件	1.有一個已註冊的助教帳號 TA，密碼為 ta 2.有一個已經建立成功的Token，名稱為ta_token_003，DUE TIME為2025-12-27，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以助教帳號 TA，密碼為 ta 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "ta_token_003"的row 4. 點擊該row的編輯按鈕 5. 選擇日期：2025-12-30 6. 點擊「UPDATE」 7. 檢查該Token的DUE TIME是否為：2025-12-30 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-30", "Name": "ta_token_003" "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的DUE TIME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-014: 管理員可以編輯自己的 Token 到期日

項目	內容
測試編號	T-F3-014
測試項目	管理員可以編輯自己的 Token 到期日
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有一個已經建立成功的Token，名稱為teacher_token_002，DUE TIME為2025-12-27，OWNER為first_admin 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以管理員帳號 first_admin · 密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "admin_token_004"的row 4. 點擊該row的編輯按鈕 5. 選擇日期：2025-12-30 6. 點擊「UPDATE」 7. 檢查該Token的DUE TIME是否為：2025-12-30 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： { "data":{ "Due_Time": "2025-12-30", "Name": "admin_token_004" "Scope": ["write:user"], } } 9. 檢查 Response：status code = 200 OK JSON： { "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }
預期結果	1.Token的DUE TIME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-015: 管理員可以編輯其他人的 Token 到期日

項目	內容
測試編號	T-F3-015
測試項目	管理員可以編輯其他人的 Token 到期日
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有三個已經建立成功的Token： 第一個名稱為stu_token_001，DUE TIME為2025-12-30，OWNER為STUDENT 第二個名稱為teacher_token_002，DUE TIME為2025-12-30，OWNER為TEACHER 第三個名稱為ta_token_003，DUE TIME為2025-12-30，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	以相同步驟分別測試修改stu_token_001、teacher_token_002、ta_token_003 一、前端操作 1. 以管理員帳號 first_admin，密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "stu_token_001"的row 4. 點擊該row的編輯按鈕 5. 選擇日期：2025-12-28 6. 點擊「UPDATE」 7. 檢查該Token的DUE TIME是否為：2025-12-28 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data": { "Due_Time": "2025-12-28", "Name": "stu_token_001", "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data": { "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status": "ok" }</pre>
預期結果	1.Token的DUE TIME都有被正確更新 2.後端回應200

項目	內容
執行結果	PASS 200

T-F3-016: 學生可以編輯自己的 Token 名稱

項目	內容
測試編號	T-F3-016
測試項目	學生可以編輯自己的 Token 名稱
先前條件	1.有一個已註冊的學生帳號 STUDENT，密碼為 student 2.有一個已經建立成功的Token，名為stu_token_001，OWNER為STUDENT 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "stu_token_001" 的row 4. 點擊該row的編輯按鈕 5. 修改NAME為 = "new_stu_token_001" 6. 點擊「UPDATE」 7. 檢查該Token的NAME是否有被正確更新為 "new_stu_token_001" 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-30", "Name": "new_stu_token_001" "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的NAME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-017: 老師可以編輯自己的 Token 名稱

項目	內容
測試編號	T-F3-017
測試項目	老師可以編輯自己的 Token 名稱
先前條件	1.有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2.有一個已經建立成功的Token，名稱為teacher_token_002，OWNER為TEACHER 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "teacher_token_002" 的row 4. 點擊該row的編輯按鈕 5. 修改NAME為 = "new_teacher_token_002" 6. 點擊「UPDATE」 7. 檢查該Token的NAME是否有被正確更新為 "new_teacher_token_002" 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-30", "Name": "new_teacher_token_002" "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的NAME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-018: 助教可以編輯自己的 Token 名稱

項目	內容
測試編號	T-F3-018
測試項目	助教可以編輯自己的 Token 名稱
先前條件	1.有一個已註冊的助教帳號 TA，密碼為 ta 2.有一個已經建立成功的Token，名為ta_token_001，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以助教帳號 TA，密碼為 ta 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "ta_token_003" 的row 4. 點擊該row的編輯按鈕 5. 修改NAME為 = "new_ta_token_003" 6. 點擊「UPDATE」 7. 檢查該Token的NAME是否有被正確更新為 "new_ta_token_003" 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-30", "Name": "new_ta_token_003" "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的NAME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-019: 管理員可以編輯自己的 Token 名稱

項目	內容
測試編號	T-F3-019
測試項目	管理員可以編輯自己的 Token 名稱
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有一個已經建立成功的Token，名稱為stu_token_001， 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	一、前端操作 1. 以管理員帳號 first_admin · 密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "admin_token_004" 的row 4. 點擊該row的編輯按鈕 5. 修改NAME為 = "new_admin_token_004" 6. 點擊「UPDATE」 7. 檢查該Token的NAME是否有被正確更新為 "new_admin_token_004" 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-30", "Name": "new_admin_token_004" "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的NAME有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-020: 管理員可以編輯其他人的 Token 名稱

項目	內容
測試編號	T-F3-020
測試項目	管理員可以編輯其他人的 Token 名稱
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有三個已經建立成功的Token： 第一個名稱為new_stu_token_001，OWNER為STUDENT 第二個名稱為new_teacher_token_002，OWNER為TEACHER 第三個名稱為new_ta_token_003，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常

項目	內容
測試方法與流程	以相同步驟分別測試修改new_stu_token_001、new_teacher_token_002、new_ta_token_003 一、前端操作 1. 以管理員帳號 first_admin，密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "new_stu_token_001"的row 4. 點擊該row的編輯按鈕 5. 修改NAME為 = "new2_stu_token_001" 6. 點擊「UPDATE」 7. 檢查該Token的NAME是否有被正確更新為 "new2_stu_token_001" 二、後端 API 回應驗證 8. 檢查送出的 Request 是否正確：API：POST /profile/api_token/edit JSON： <pre>{ "data":{ "Due_Time": "2025-12-30", "Name": "new2_stu_token_001" "Scope": ["write:user"], } }</pre> 9. 檢查 Response：status code = 200 OK JSON： <pre>{ "data" :{ "Message": "Token updated", "Type": "OK" } "message": "Token updated" "status":"ok" }</pre>
預期結果	1.Token的NAME都有被正確更新 2.後端回應200

項目	內容
執行結果	PASS 200

T-F3-021: 學生可以撤銷自己的 Token

項目	內容
測試編號	T-F3-021
測試項目	學生可以撤銷自己的 Token
先前條件	1.有一個已註冊的學生帳號 STUDENT，密碼為 student 2.有一個已經建立成功的Token， 名稱為new2_stu_token_001，OWNER為STUDENT 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "new2_stu_token_001" 的row 4. 點擊該row的撤銷按鈕 5. 檢查該Token的STATUS是否有被正確更新為 "deactivated" 二、後端 API 回應驗證 6. 檢查 Response : status code = 200 OK JSON : <pre>{ "data" :{ "Message": "Token revoked", "Type": "OK" } "message": "Token revoked" "status": "ok" }</pre>
預期結果	1.Token的STATUS有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-022: 老師可以撤銷自己的 Token

項目	內容
測試編號	T-F3-022
測試項目	老師可以撤銷自己的 Token
先前條件	1.有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2.有一個已經建立成功的Token， 名稱為new2_teacher_token_002，OWNER為TEACHER 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "new2_teacher_token_002"的row 4. 點擊該row的撤銷按鈕 5. 檢查該Token的STATUS是否有被正確更新為 "deactivated" 二、後端 API 回應驗證 6. 檢查 Response：status code = 200 OK JSON： <pre>{ "data": { "Message": "Token revoked", "Type": "OK" }, "message": "Token revoked", "status": "ok" }</pre>
預期結果	1.Token的STATUS有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-023: 助教可以撤銷自己的 Token

項目	內容
測試編號	T-F3-023
測試項目	助教可以撤銷自己的 Token
先前條件	1.有一個已註冊的助教帳號 TA，密碼為 ta 2.有一個已經建立成功的Token，名稱為new2_ta_token_003，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以助教帳號 TA，密碼為 ta 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "new2_ta_token_003" 的row 4. 點擊該row的撤銷按鈕 5. 檢查該Token的STATUS是否有被正確更新為 "deactivated" 二、後端 API 回應驗證 6. 檢查 Response：status code = 200 OK JSON： <pre>{ "data": { "Message": "Token revoked", "Type": "OK" }, "message": "Token revoked", "status": "ok" }</pre>
預期結果	1.Token的STATUS有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-024: 管理員可以撤銷自己的 Token

項目	內容
測試編號	T-F3-024
測試項目	管理員可以撤銷自己的 Token
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有一個已經建立成功的Token， 名稱為new2_admin_token_004，OWNER為first_admin 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以管理員帳號 first_admin，密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "new2_admin_token_004"的row 4. 點擊該row的撤銷按鈕 5. 檢查該Token的STATUS是否有被正確更新為 "deactivated" 二、後端 API 回應驗證 6. 檢查 Response : status code = 200 OK JSON : <pre>{ "data" :{ "Message": "Token revoked", "Type": "OK" } "message": "Token revoked" "status":"ok" }</pre>
預期結果	1.Token的STATUS有被正確更新 2.後端回應200
執行結果	PASS 200

T-F3-025: 管理員可以撤銷其他人的 Token

項目	內容
測試編號	T-F3-025
測試項目	管理員可以撤銷其他人的 Token
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有三個已經建立成功的Token： 第一個名稱為stu_token_001，OWNER為STUDENT 第二個名稱為teacher_token_002，OWNER為TEACHER 第三個名稱為ta_token_003，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	以相同步驟分別測試修改stu_token_001、teacher_token_002、new_ta_token_003 一、前端操作 1. 以管理員帳號 first_admin，密碼為 firstpasswordforadmin 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "stu_token_001"的row 4. 點擊該row的撤銷按鈕 5. 檢查該Token的STATUS是否有被正確更新為 "deactivated" 二、後端 API 回應驗證 6. 檢查 Response：status code = 200 OK JSON： <pre>{ "data": { "Message": "Token revoked", "Type": "OK" }, "message": "Token revoked", "status": "ok" }</pre>
預期結果	1.Token的NAME都有被正確更新 2.後端回應200

項目	內容
執行結果	PASS 200

T-F3-026: 學生可以依名稱搜尋 Token

項目	內容
測試編號	T-F3-026
測試項目	學生可以依名稱搜尋 Token
先前條件	1.有一個已註冊的學生帳號 STUDENT，密碼為 student 2.有三個已經建立成功的Token： 第一個名稱為stu_token_001，OWNER為STUDENT 第二個名稱為new_stu_token_001，OWNER為STUDENT 第三個名稱為nothing1，OWNER為STUDENT 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到搜尋框，輸入 = stu 4. 檢查下方的token列表中是否每一個token中皆有"stu"
預期結果	1.可以正確搜尋到擁有關鍵字Token
執行結果	PASS 正確顯示含有關鍵字的Token

T-F3-027: 老師可以依名稱搜尋 Token

項目	內容
測試編號	T-F3-027
測試項目	老師可以依名稱搜尋 Token
先前條件	1.有一個已註冊的老師帳號 TEACHER，密碼為 teacher 2.有三個已經建立成功的Token： 第一個名稱為teacher_token_002，OWNER為TEACHER 第二個名稱為new_teacher__token_001，OWNER為TEACHER 第三個名稱為nothing2，OWNER為TEACHER 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以老師帳號 TEACHER，密碼為 teacher 登入 2. 進入「PROFILE → API Token」 3. 找到搜尋框，輸入 = teacher 4. 檢查下方的token列表中是否每一個token中皆有"teacher"
預期結果	1.可以正確搜尋到擁有關鍵字Token
執行結果	PASS 正確顯示含有關鍵字的Token

T-F3-028: 助教可以依名稱搜尋 Token

項目	內容
測試編號	T-F3-028
測試項目	助教可以依名稱搜尋 Token
先前條件	1.有一個已註冊的助教帳號 TA，密碼為 ta 2.有三個已經建立成功的Token： 第一個名稱為ta_token_003，OWNER為TA 第二個名稱為new_ta_token_003，OWNER為TA 第三個名稱為nothing3，OWNER為TA 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以助教帳號 TA，密碼為 ta 登入 2. 進入「PROFILE → API Token」 3. 找到搜尋框，輸入 = ta 4. 檢查下方的token列表中是否每一個token中皆有"ta"
預期結果	1.可以正確搜尋到擁有關鍵字Token
執行結果	PASS 正確顯示含有關鍵字的Token

T-F3-029: 管理員可以依名稱搜尋 Token

項目	內容
測試編號	T-F3-029
測試項目	管理員可以依名稱搜尋 Token
先前條件	1.有一個已註冊的管理員帳號 first_admin，密碼為 firstpasswordforadmin 2.有三個已經建立成功的Token： 第一個名稱為admin_token_004，OWNER為first_admin 第二個名稱為new_admin_token_004，OWNER為first_admin 第三個名稱為nothing4，OWNER為first_admin 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 管理員帳號 first_admin，密碼為 firstpasswordforadmin 2. 進入「PROFILE → API Token」 3. 找到搜尋框，輸入 = admin 4. 檢查下方的token列表中是否每一個token中皆有"admin"
預期結果	1.可以正確搜尋到擁有關鍵字Token
執行結果	PASS 正確顯示含有關鍵字的Token

T-F3-030: Token 過期應視為被撤銷

項目	內容
測試編號	T-F3-030
測試項目	Token 過期應視為被撤銷
先前條件	1.有一個已註冊的學生帳號 STUDENT，密碼為 student 2.有一個已經建立成功的Token，名稱為nothing1，DUE TIME為2025-12-27 3.前端 Token 管理頁面可正常操作 4.後端 API 與資料庫連線正常
測試方法與流程	一、前端操作 1. 以學生帳號 STUDENT，密碼為 student 登入 2. 進入「PROFILE → API Token」 3. 找到Token名稱為NAME = "nothing1" 的row 4. 點擊該row的編輯按鈕 5. 選擇日期：2025-12-01 6. 點擊「UPDATE」 7. 檢查該Token的STATUS是否為"deactivated" 二、後端 API 回應驗證 8. 檢查 Response：status code = 200 OK JSON： <pre>{ "data": { Created "(測試日期)" Due_Time "2025-12-01" ID "(依各Token不同)" Last_Used null Name "nothing1" Owner "STUDENT" Scope ["read:user"] "Status "Deactivated" } "message": "OK" "status": "ok" }</pre>

項目	內容
預期結果	1.可以正確判斷Token的過期狀態
執行結果	PASS 200

Feature 4

T-F4-001: Admin 可展開 Pipeline 區塊

項目	內容
測試編號	T-F4-001
測試項目	Admin 可展開 Pipeline 區塊
先前條件	已登入 admin，course_for_test 存在
測試方法與流程	進入課程 Problems → New 點擊 Set Pipeline 摺疊標題
預期結果	區塊展開並顯示 File Access、Execution Mode、Library Restrictions、Custom Checker、Custom Scoring Script
執行結果	pass

T-F4-002: 切換 **Function Only** 後顯示 **Makefile** 上傳區

項目	內容
測試編號	T-F4-002
測試項目	切換 Function Only 後顯示 Makefile 上傳區
先前條件	Pipeline 區塊已展開
測試方法與流程	展開 Pipeline 區塊 點選 Execution Mode 的 "Function Only" , 並確認該選項被勾選 驗證顯示 "Upload Makefile.zip" 上傳區、存在 .zip 上傳欄位 , 且顯示 "Not Uploaded"
預期結果	出現 Upload Makefile.zip , 標示 Not Uploaded 且有 .zip 輸入
執行結果	pass

T-F4-003: 切到 **Interactive** 顯示 **Teacher First** 與 **Upload Teacher Code**

項目	內容
測試編號	T-F4-003
測試項目	切到 Interactive 顯示 Teacher First 與 Upload Teacher Code
先前條件	Pipeline 區塊已展開
測試方法與流程	展開 Pipeline 區塊 點選 "Interactive" , 並確認該選項被勾選 驗證顯示 "Teacher First" toggle 驗證顯示 Upload Teacher Code 上傳區
預期結果	Teacher First toggle 與 Teacher Code 上傳欄位可見
執行結果	pass

T-F4-004: Imports 區塊因 Python 未啟用而禁用

項目	內容
測試編號	T-F4-004
測試項目	Imports 區塊因 Python 未啟用而禁用
先前條件	Pipeline 展開，開啟 Library Restrictions
測試方法與流程	展開 Pipeline 並啟用 Library Restrictions 進入 Imports Restrictions 區塊並檢查其狀態 若顯示 Disable 遮罩，驗證出現「Imports Restrictions Disabled」及提示「Python language must be enabled」； 否則驗證 Whitelist 按鈕可見且可操作
預期結果	若未啟用 Python顯示「Imports Restrictions Disabled」與 「Python language must be enabled」； 若已啟用則按鈕可操作
執行結果	pass

T-F4-005: Headers 區塊因 C/C++ 未啟用而禁用

項目	內容
測試編號	T-F4-005
測試項目	Headers 區塊因 C/C++ 未啟用而禁用
先前條件	Pipeline 展開，開啟 Library Restrictions
測試方法與流程	展開 Pipeline 並啟用 Library Restrictions 進入 Headers Restrictions 區塊並檢查其狀態 若顯示 Disable 遮罩，驗證出現「Headers Restrictions Disabled」及提示「C or C++ language must be enabled」；否則驗證 Whitelist 按鈕可見且可操作
預期結果	未啟用 C/C++ 時顯示「Headers Restrictions Disabled」與 「C or C++ language must be enabled」；啟用後按鈕可操作
執行結果	pass

T-F4-006: Imports Blacklist 加入 os

項目	內容
測試編號	T-F4-006
測試項目	Imports Blacklist 加入 os
先前條件	Python 語言啟用；Library Restrictions 開啟
測試方法與流程	展開 Pipeline 並確保 Python 已啟用（如需要） 啟用 Library Restrictions → 進入 Imports Restrictions → 切換至 Blacklist 在 Add import 輸入 os 並送出
預期結果	出現紅色 os 標籤
執行結果	pass

T-F4-007: Headers Blacklist 加入 algorithm

項目	內容
測試編號	T-F4-007
測試項目	Headers Blacklist 加入 algorithm
先前條件	Library Restrictions 開啟且 C/C++ 啟用
測試方法與流程	展開 Pipeline 並啟用 Library Restrictions 進入 Headers Restrictions → 切換至 Blacklist 在 Add header 輸入 algorithm 並送出
預期結果	出現紅色 algorithm 標籤
執行結果	pass

T-F4-008: Functions Blacklist 加入 system

項目	內容
測試編號	T-F4-008
測試項目	Functions Blacklist 加入 system
先前條件	Library Restrictions 開啟
測試方法與流程	展開 Pipeline 並啟用 Library Restrictions 進入 Functions Restrictions → 切換至 Blacklist 在 Add function 輸入 system 並送出
預期結果	出現紅色 system 標籤
執行結果	pass

T-F4-009: Syntax Whitelist/Blacklist 切換後選項不共用

項目	內容
測試編號	T-F4-009
測試項目	Syntax Whitelist/Blacklist 切換後選項不共用
先前條件	Library Restrictions 開啟
測試方法與流程	在 Whitelist 模式下選擇 "while" 切換到 Blacklist 模式 在 Blacklist 模式下選擇 "for" 切回 Whitelist 模式
預期結果	Whitelist 模式下 "while" 為藍色 (btn-info) ；切到 Blacklist 時不應顯示紅色 (btn-error) ；Blacklist 模式下 "for" 為紅色 (btn-error) ；切回 Whitelist 時 "for" 不應顯示藍色 (btn-info) ；各模式選項互不共用
執行結果	pass

T-F4-010: Custom Checker toggle 顯示上傳區

項目	內容
測試編號	T-F4-010
測試項目	Custom Checker toggle 顯示上傳區
先前條件	Pipeline 展開
測試方法與流程	展開 Pipeline 區塊 開啟 Custom Checker toggle 驗證顯示 "Upload Custom_Checker.py" 上傳區且 .py 檔案上傳欄位可見
預期結果	顯示「Upload Custom_Checker.py」區塊，且 .py 檔案上傳欄位可見
執行結果	pass

T-F4-011: 上傳 custom_checker.py

項目	內容
測試編號	T-F4-011
測試項目	上傳 custom_checker.py
先前條件	已啟用 Custom Checker
測試方法與流程	展開 Pipeline 區塊並啟用 Custom Checker (若尚未啟用) 在 "Upload Custom_Checker.py" 的上傳欄位選擇 fixtures/custom_checker.py 驗證上傳後狀態顯示 Uploaded
預期結果	狀態變成 Uploaded
執行結果	pass

T-F4-012: 上傳 scoring_script.py

項目	內容
測試編號	T-F4-012
測試項目	上傳 scoring_script.py
先前條件	打開 Custom Scoring Script
測試方法與流程	展開 Pipeline 區塊並開啟 Custom Scoring Script (若尚未開啟) 在 "Upload Custom_Scorer.py" 的上傳欄位選擇 fixtures/scoring_script.py 驗證上傳後狀態顯示 Uploaded
預期結果	顯示「Upload Custom_Scorer.py」區塊且 .py 欄位可見； 狀態變成 Uploaded
執行結果	pass

T-F4-013: Function Only 上傳 makefile.zip

項目	內容
測試編號	T-F4-013
測試項目	Function Only 上傳 makefile.zip
先前條件	Pipeline 展開
測試方法與流程	展開 Pipeline 區塊 切換 Execution Mode 為 "Function Only" 並確認被選中 在 "Upload Makefile.zip" 上傳欄位選擇 fixtures/makefile.zip 驗證上傳後狀態顯示 Uploaded
預期結果	顯示 Uploaded
執行結果	pass

T-F4-014: Interactive 上傳 teacher.py (<http://teacher.py>).

項目	內容
測試編號	T-F4-014
測試項目	Interactive 上傳 <u>teacher.py</u> (http://teacher.py).
先前條件	Pipeline 展開
測試方法與流程	展開 Pipeline 區塊 切換 Execution Mode 為 "Interactive" 並確認被選中 在 "Upload Teacher Code" 上傳欄位選擇 fixtures/teacher.py 驗證上傳後狀態顯示 Uploaded
預期結果	顯示 Uploaded
執行結果	pass

T-F4-015: Teacher First toggle 可切換

項目	內容
測試編號	T-F4-015
測試項目	Teacher First toggle 可切換
先前條件	Pipeline 展開且在 Interactive
測試方法與流程	展開 Pipeline 區塊並切換 Execution Mode 至 "Interactive" (若尚未) 點擊 "Teacher First" toggle 以切換其狀態 驗證 toggle 狀態於開/關間切換且 UI 有相應反饋
預期結果	狀態可在開/關間切換
執行結果	pass

T-F4-016: Function Only 建題 + 學生提交 AC 路徑

項目	內容
測試編號	T-F4-016
測試項目	Function Only 建題 + 學生提交 AC 路徑
先前條件	foTeacherZip、foStudentZip、testdata.zip
測試方法與流程	進入新題目頁並將 Execution Mode 選為 "Function Only" 上傳教師 Makefile.zip (foTeacherZip) 展開 Set Test Data，上傳 testdata.zip，並將 Score 設為 100 提交以建立題目 (Submit) 進入該題目之 Submit 頁，學生上傳 foStudentZip，選擇語言 C，Submit 提交
預期結果	建題與提交流程完整執行，最終呈現評測結果
執行結果	pass

T-F4-017: 學生 zip 包含 main() 觸發 CE

項目	內容
測試編號	T-F4-017
測試項目	學生 zip 包含 main() 觸發 CE
先前條件	foStudentWithMainZip、foTeacherZip、testdata.zip
測試方法與流程	依照 T-F4-016 完成建題 (Function Only，教師上傳 foTeacherZip，testdata 上傳並 Score=100) 學生在 Submit 頁上傳 foStudentWithMainZip，選擇語言 C，Submit 提交
預期結果	呈現評測結果頁；狀態可能為 CE 或等價錯誤
執行結果	pass

T-F4-018: 學生 main.c 但選 cpp

項目	內容
測試編號	T-F4-018
測試項目	學生 main.c 但選 cpp
先前條件	foStudentZip、foTeacherZip、testdata.zip
測試方法與流程	依照 T-F4-016 完成建題 (Function Only，教師上傳 foTeacherZip、testdata 上傳並 Score=100) 學生在 Submit 頁上傳 foStudentZip，但將語言選為 C++ (與檔案 main.c 不符) 並 Submit
預期結果	呈現評測結果頁；狀態為 CE 或拒絕
執行結果	pass

T-F4-019: 教師資產缺 Makefile

項目	內容
測試編號	T-F4-019
測試項目	教師資產缺 Makefile
先前條件	foTeacherNoMakeZip、testdata.zip
測試方法與流程	進入新題目頁並選擇 Execution Mode 為 "Function Only" 上傳缺 Makefile 的教師 zip (foTeacherNoMakeZip) 展開 Set Test Data，上傳 testdata.zip 並設 Score=100， 然後 Submit 建題
預期結果	建題流程完成；後續評測可能失敗 (缺少編譯檔)
執行結果	pass

T-F4-020: Interactive 正常 AC 流程

項目	內容
測試編號	T-F4-020
測試項目	Interactive 正常 AC 流程
先前條件	itTeacherC、itStudentZip、testdata.zip
測試方法與流程	進入新題目頁並將 Execution Mode 選為 "Interactive" 確保 "Teacher First" 已啟用 (如需) 並上傳 Teacher Code (itTeacherC) 展開 Set Test Data，上傳 testdata.zip 並將 Score 設為 100，然後 Submit 建題 進入該題目之 Submit 頁，學生上傳 itStudentZip，選擇語言 C，Submit 提交
預期結果	建題與提交流程完整執行，最終呈現評測結果
執行結果	pass

T-F4-021: Teacher First 關閉造成互等/TLE

項目	內容
測試編號	T-F4-021
測試項目	Teacher First 關閉造成互等/TLE
先前條件	itTeacherC、itStudentZip、testdata.zip
測試方法與流程	進入新題目頁並將 Execution Mode 選為 "Interactive" 關閉 "Teacher First" toggle (若預設為開) 上傳 Teacher Code (itTeacherC) 展開 Set Test Data，上傳 testdata.zip，並將 Score 設為 100 Submit 建題以建立題目 進入該題目之 Submit 頁，學生上傳 itStudentZip，選擇語言 C，Submit 提交
預期結果	呈現評測結果頁；可觀察互等或 TLE 行為
執行結果	pass

T-F4-022: 未上傳 Teacher Code 或語言設定缺失

項目	內容
測試編號	T-F4-022
測試項目	未上傳 Teacher Code 或語言設定缺失
先前條件	testdata.zip
測試方法與流程	進入新題目頁並將 Execution Mode 選為 "Interactive" 不上傳 Teacher Code (保持未上傳狀態) 展開 Set Test Data · 上傳 testdata.zip · 並將 Score 設為 100 Submit 建題以建立題目 驗證題目編輯頁或建立後頁面仍顯示 Upload Teacher Code · 提示缺失
預期結果	建題流程完成；頁面仍顯示「Upload Teacher Code」 (未上傳) 以提示缺失
執行結果	pass

T-F4-023: 學生 zip 與語言選擇不符

項目	內容
測試編號	T-F4-023
測試項目	學生 zip 與語言選擇不符
先前條件	itTeacherC、itStudentZip、testdata.zip
測試方法與流程	進入新題目頁並將 Execution Mode 選為 "Interactive" 確保 "Teacher First" 已啟用並上傳 Teacher Code (itTeacherC) 展開 Set Test Data，上傳 testdata.zip，並將 Score 設為 100，然後 Submit 建題 進入該題目之 Submit 頁，學生上傳 itStudentZip，選擇語言 C++ (與檔案不符) 並 Submit 提交
預期結果	呈現評測結果頁；狀態為 CE 或拒絕
執行結果	pass

T-F4-024: General 題 Allow Read=On、Allow Write=Off，學生寫檔

項目	內容
測試編號	T-F4-024
測試項目	General 題 Allow Read=On、Allow Write=Off，學生寫檔
先前條件	testdata.zip、aw_write_c.zip
測試方法與流程	進入新題目頁並設定為 General 模式 將 "Allow Read" 打開，將 "Allow Write" 關閉 展開 Set Test Data，上傳 testdata.zip，並將 Score 設為 100 Submit 建題以建立題目 進入該題目之 Submit 頁，學生上傳 aw_write_c.zip，選擇語言 C，Submit 提交
預期結果	呈現評測結果頁；流程完成但結果應為失敗 (不允許寫檔)
執行結果	pass

T-F4-025: General 題 Allow Write 開啟後寫檔成功

項目	內容
測試編號	T-F4-025
測試項目	General 題 Allow Write 開啟後寫檔成功
先前條件	testdata.zip、aw_write_c.zip
測試方法與流程	進入新題目頁並設定為 General 模式 打開 "Allow Read" 與 "Allow Write" 展開 Set Test Data，上傳 testdata.zip，並將 Score 設為 100 Submit 建題以建立題目 進入該題目之 Submit 頁，學生上傳 aw_write_c.zip，選擇語言 C，Submit 提交
預期結果	呈現評測結果頁；流程完成且允許寫檔
執行結果	pass

T-F4-026: Imports Blacklist 阻擋 import os

項目	內容
測試編號	T-F4-026
測試項目	Imports Blacklist 阻擋 import os
先前條件	testdata.zip、py_import_os.zip；Python 啟用
測試方法與流程	進入新題目頁 確保 Python 語言已啟用 展開 Pipeline 並啟用 Library Restrictions 在 Imports 區塊選擇 Blacklist，加入 os 展開 Set Test Data，上傳 testdata.zip 並設定 Score=100 點擊 Submit 建題 進入該題目之 Submit 頁 8. 上傳 py_import_os.zip，選擇語言 Python，點擊 Submit 提交
預期結果	呈現評測結果頁；狀態為 JE 或拒絕（違反黑名單）
執行結果	pass

T-F4-027: Headers Blacklist 阻擋 include <algorithm>

項目	內容
測試編號	T-F4-027
測試項目	Headers Blacklist 阻擋 include <algorithm>
先前條件	testdata.zip、cpp_include_algorithm.zip；C/C++ 啟用
測試方法與流程	進入新題目頁 展開 Pipeline 並啟用 Library Restrictions 在 Headers 區塊選擇 Blacklist，加入 algorithm 展開 Set Test Data，上傳 testdata.zip 並設定 Score=100 點擊 Submit 建題 進入該題目之 Submit 頁 上傳 cpp_include_algorithm.zip，選擇語言 C++，點擊 Submit 提交
預期結果	呈現評測結果頁；狀態為 CE 或 JE（違反黑名單）
執行結果	pass

T-F4-028: Functions Blacklist 阻擋 system()

項目	內容
測試編號	T-F4-028
測試項目	Functions Blacklist 阻擋 system()
先前條件	testdata.zip 、 c_call_system.zip
測試方法與流程	進入新題目頁 展開 Pipeline 並啟用 Library Restrictions 在 Functions 區塊選擇 Blacklist · 加入 system 展開 Set Test Data · 上傳 testdata.zip 並設定 Score=100 點擊 Submit 建題 進入該題目之 Submit 頁 上傳 c_call_system.zip · 選擇語言 C · 點擊 Submit 提交
預期結果	呈現評測結果頁；狀態為 JE 或拒絕（違反黑名單）
執行結果	pass

T-F4-029: 自訂 Checker 建題與提交流程

項目	內容
測試編號	T-F4-029
測試項目	自訂 Checker 建題與提交流程
先前條件	custom_checker.py、sum_cpp.zip、testdata.zip
測試方法與流程	進入新題目頁 展開 Pipeline 並啟用 Custom Checker 上傳 custom_checker.py 至 Custom Checker 區塊，確認為 Uploaded 展開 Set Test Data，上傳 testdata.zip 並設定 Score=100 點擊 Submit 建題 進入該題目之 Submit 頁 上傳 sum_cpp.zip，選擇語言 C++，點擊 Submit 提交
預期結果	建題與提交流程完成；Checker 正常執行
執行結果	pass

T-F4-030: 自訂 Scoring 建題與提交流程

項目	內容
測試編號	T-F4-030
測試項目	自訂 Scoring 建題與提交流程
先前條件	scoring_script.py、testdata.zip
測試方法與流程	展開 Pipeline 並啟用 Custom Scoring Script 在 Upload Custom_Scorer.py 上傳 fixtures/scoring_script.py 展開 Set Test Data，上傳 testdata.zip 並將 Score 設為 100 點擊 Submit 建題
預期結果	題目建立成功；Scoring 開關與上傳狀態正確保存 (不含學生提交)
執行結果	pass

T-F4-031: 截取 /api/problem/manage POST，驗證 payload 內容

項目	內容
測試編號	T-F4-031
測試項目	截取 /api/problem/manage POST，驗證 payload 內容
先前條件	testdata.zip
測試方法與流程	進入新題目頁< 填寫題目名稱 展開 Pipeline 設定 啟用 Allow Read，並確認為勾選 啟用 Library Restrictions，並確認為勾選 在 Syntax Restrictions 選擇 Blacklist 並選中 while， 確認按鈕為紅色 展開 Set Test Data，上傳 testdata.zip 並設定 Score=100 在前端攔截 /api/problem/manage POST，監聽並紀錄 payload 點擊 Submit 建題，檢查攔截到的 payload 是否包含預期欄位
預期結果	HTTP 回應為 200/201；攔截的 payload 包含 allowRead=true、libraryRestrictions.enabled=true
執行結果	pass

T-F4-032: 新增題目頁 Pipeline 預設狀態驗證

項目	內容
測試編號	T-F4-032
測試項目	新增題目頁 Pipeline 預設狀態驗證
先前條件	進入新增題目頁
測試方法與流程	進入新題目頁 展開 Pipeline 檢查 Execution Mode 是否為 General 已選中 檢查 Allow Read 為未勾選狀態 檢查 Allow Write 為未勾選且為 disabled (因 Allow Read 關閉) 檢查 Library Restrictions 未勾選 檢查 Custom Checker 未勾選 檢查 Custom Scoring Script 未勾選
預期結果	所有預設值與 UI 狀態符合規格
執行結果	pass

T-F4-033: 設定 UI，驗證 Pipeline 欄位 UI 狀態正確反映

項目	內容
測試編號	T-F4-033
測試項目	設定 UI，驗證 Pipeline 欄位 UI 狀態正確反映
先前條件	進入新題目頁
測試方法與流程	進入新題目頁 填寫題目名稱 展開 Pipeline< 啟用 Allow Read 啟用 Library Restrictions 在 Syntax Restrictions 選擇 Blacklist 並選中 while 驗證 Allow Read 為勾選狀態 驗證 Library Restrictions 為勾選狀態 驗證 while 按鈕呈紅色 (btn-error)
預期結果	UI 設定正確反映：Allow Read 勾選、Library Restrictions 勾選、Syntax while 按鈕紅色
執行結果	pass

T-F4-034: Library Restrictions 打開後預設選項按鈕載入

項目	內容
測試編號	T-F4-034
測試項目	Library Restrictions 打開後預設選項按鈕載入
先前條件	進入新題目頁
測試方法與流程	進入新題目頁 展開 Pipeline 啟用 Library Restrictions 驗證 Syntax Restrictions 區塊可見 驗證預設快捷按鈕 while、for、recursive 都可見
預期結果	Syntax Restrictions 區塊可見；預設按鈕 (while、for、recursive) 存在且可見
執行結果	pass

T-F4-035: Syntax 快捷按鈕可見且可互動

項目	內容
測試編號	T-F4-035
測試項目	Syntax 快捷按鈕可見且可互動
先前條件	Library Restrictions 開啟
測試方法與流程	進入新題目頁 展開 Pipeline 啟用 Library Restrictions 驗證 while、for、recursive 按鈕可見且可互動 點擊 while 按鈕，驗證顏色變為藍色 (btn-info) 點擊 for 按鈕，驗證顏色變為藍色 (btn-info)
預期結果	while、for、recursive 按鈕可見；點擊後顏色改變為藍色 (btn-info)，表示互動成功
執行結果	pass

Feature 5 題目討論區

T-F5-001: 建立貼文

項目	內容
測試編號	T-F5-001
測試項目	建立貼文
先前條件	討論區已建立且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入First Post 8.在Content輸入This is the first post for Prob1 9.點擊Post
預期結果	每題正確建立貼文，無重複或遺漏。
執行結果	pass

T-F5-002: 瀏覽討論區貼文

項目	內容
測試編號	T-F5-002
測試項目	瀏覽討論區貼文
先前條件	貼文已建立且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion
預期結果	可看到Course中所有貼文，無重複或遺漏。
執行結果	pass

T-F5-003: 程式碼區塊顯示

項目	內容
測試編號	T-F5-003
測試項目	程式碼區塊顯示
先前條件	討論區已建立且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Problems 6.點擊Prob1 7.點擊 7.在title輸入First Code 8.在Content貼上add.cpp 9.點擊Post
預期結果	程式碼區塊與內文其他部分有明顯區隔，且關鍵字有亮色。
執行結果	pass

T-F5-004: 搜尋貼文

項目	內容
測試編號	T-F5-004
測試項目	搜尋貼文
先前條件	已建立數個貼文且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.在Search的輸入框輸入Prob1 3.點擊search
預期結果	Search後可顯示title有關鍵字貼文。
執行結果	pass

T-F5-005: 排序貼文

項目	內容
測試編號	T-F5-005
測試項目	排序貼文
先前條件	已建立數個貼文且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊Hot
預期結果	Sort後可依熱門度顯示貼文。
執行結果	pass

T-F5-006: 排序貼文

項目	內容
測試編號	T-F5-006
測試項目	排序貼文
先前條件	已建立數個貼文且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊New
預期結果	Sort後可依時間順序顯示貼文。
執行結果	pass

T-F5-007: 依problem選擇貼文

項目	內容
測試編號	T-F5-007
測試項目	依problem選擇貼文
先前條件	已建立貼文且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 2.點擊Prob1
預期結果	成功看到所有關於problem 1的貼文。
執行結果	pass

T-F5-008: 按讚貼文

項目	內容
測試編號	T-F5-008
測試項目	按讚貼文
先前條件	已建立貼文且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 5.點擊讚
預期結果	成功幫貼文按讚。
執行結果	pass

T-F5-009: 留言貼文

項目	內容
測試編號	T-F5-009
測試項目	留言貼文
先前條件	已建立貼文且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 3.在留言的輸入框輸入First Message 4.點擊送出
預期結果	成功幫貼文留言。其他使用者看到的與該使用者輸入內容一致。
執行結果	pass

T-F5-010: 截止前禁公開解答

項目	內容
測試編號	T-F5-010
測試項目	截止前禁公開解答
先前條件	存在討論區且學生已登入
測試方法與流程	1.學生點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.選擇某problem(Prob2) 6.貼文內容中含有解答標籤(#Ans)與程式碼區塊 7.建立貼文
預期結果	前端顯示無法建立貼文。
執行結果	pass

T-F5-011: 置頂

項目	內容
測試編號	T-F5-011
測試項目	置頂
先前條件	已存在貼文且老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 5.點擊... 6.點擊置頂 7.點擊確認
預期結果	貼文顯示於最上方。
執行結果	pass

T-F5-012: 關帖

項目	內容
測試編號	T-F5-012
測試項目	關帖
先前條件	已存在貼文且老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入To be closed 8.在Content輸入This is the first post for closing 9.點擊Post 10.點擊... 11.點擊關閉討論 12.點擊確認
預期結果	前端貼文顯示「已關閉」標記。
執行結果	pass

T-F5-013: 標記已解決

項目	內容
測試編號	T-F5-013
測試項目	標記已解決
先前條件	已存在貼文且老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 5.點擊... 6.點擊標記為已解決
預期結果	前端貼文顯示「已解決」標記。
執行結果	pass

T-F5-014: 刪除

項目	內容
測試編號	T-F5-014
測試項目	刪除
先前條件	老師已登入
測試方法與流程	1.老師點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入To be deleted 8.在Content輸入This is the first post for deleting 9.點擊Post 10.點擊... 11.點擊刪除 12.點擊確認
預期結果	該貼文不可見。
執行結果	pass

T-F5-015: 置頂

項目	內容
測試編號	T-F5-015
測試項目	置頂
先前條件	已存在貼文且TA已登入
測試方法與流程	1.TA點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 5.點擊... 6.點擊置頂 7.點擊確認
預期結果	貼文顯示於最上方。
執行結果	pass

T-F5-016: 關帖

項目	內容
測試編號	T-F5-016
測試項目	關帖
先前條件	已存在貼文且TA已登入
測試方法與流程	1.TA點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入To be closed 8.在Content輸入This is the first post for closing 9.點擊Post 10.點擊... 11.點擊關閉討論 12.點擊確認
預期結果	前端貼文顯示「已關閉」標記。
執行結果	pass

T-F5-017: 標記已解決

項目	內容
測試編號	T-F5-017
測試項目	標記已解決
先前條件	已存在貼文且TA已登入
測試方法與流程	1.TA點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 5.點擊... 6.點擊標記為已解決
預期結果	前端貼文顯示「已解決」標記。
執行結果	pass

T-F5-018: 刪除

項目	內容
測試編號	T-F5-018
測試項目	刪除
先前條件	已存在貼文且TA已登入
測試方法與流程	1.TA點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入To be deleted 8.在Content輸入This is the first post for deleting 9.點擊Post 10.點擊... 11.點擊刪除 12.點擊確認
預期結果	該貼文不可見。
執行結果	pass

T-F5-019: 置頂

項目	內容
測試編號	T-F5-019
測試項目	置頂
先前條件	已存在貼文且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文Hi 5.點擊... 6.點擊置頂 7.點擊確認
預期結果	貼文顯示於最上方。
執行結果	pass

T-F5-020: 關帖

項目	內容
測試編號	T-F5-020
測試項目	關帖
先前條件	已存在貼文且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入To be closed 8.在Content輸入This is the first post for closing 9.點擊Post 10.點擊... 11.點擊關閉討論 12.點擊確認
預期結果	前端貼文顯示「已關閉」標記。
執行結果	pass

T-F5-021: 標記已解決

項目	內容
測試編號	T-F5-021
測試項目	標記已解決
先前條件	已存在貼文且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊貼文 5.點擊已解決
預期結果	前端貼文顯示「已解決」標記。
執行結果	pass

T-F5-022: 刪除

項目	內容
測試編號	T-F5-022
測試項目	刪除
先前條件	已存在貼文且Admin已登入
測試方法與流程	1.Admin點擊Course 2.點擊meow (course name) 3.點擊discussion 4.點擊post 5.點擊Select a problem 6.點擊Prob1 7.在Title輸入To be deleted 8.在Content輸入This is the first post for deleting 9.點擊Post 10.點擊... 11.點擊刪除 12.點擊確認
預期結果	該貼文不可見。
執行結果	pass

6. 安全性測試

本系統之安全性測試採用弱點掃描工具進行自動化檢測，詳細掃描結果請參閱 《弱點掃描 V8.pdf》

7. 結論與建議

7.1 測試結論

經過為期 4 週 + 1 日的完整測試週期，NOJ 系統已通過所有 126 項功能測試案例，通過率達 100%。

評估項目	結果
功能完整性	所有需求功能皆已實作並驗證通過
系統穩定性	測試期間無系統崩潰或異常中斷
資料正確性	資料存取與顯示皆正確無誤
權限控制	各角色權限劃分正確
整體品質	符合上線標準

7.2 上線建議

根據本次測試結果，系統可進行正式上線部署。

7.3 後續建議

- 持續監控：上線後持續監控系統運行狀況與使用者回饋
- 定期維護：建立定期安全性掃描與更新機制

8. 附錄

8.1 參考文件

文件名稱	說明
測試計畫書 V2.2	本次測試之依據文件
需求規格書 V2	系統功能需求定義
設計規格書 V1.1	系統架構與設計規格
弱點掃描 V8.pdf	安全性弱點掃描報告

8.2 名詞解釋

名詞	說明
NOJ	Normal Online Judge，本專案之校內線上評測平台
API	Application Programming Interface，應用程式介面
PAT	Personal Access Token，個人存取憑證
E2E	End-to-End Test，端對端測試
CE	Compile Error，編譯錯誤
TLE	Time Limit Exceeded，執行超時
JE	Judge Error，評測錯誤

本報告由 *NOJ* 測試團隊製作

2025 年 12 月 29 日