

# Lab 4 Report

Name:朱峻平

Student ID:107598058

Date:2019/5/15

## 1 Test Plan

### 1.1 Summary/Scope

Lab4 要撰寫測試腳本滿足以下五種 feature 以確保網站行為無誤

(使用 **ISP** 方法進行測試案例設計以達到盡可能的覆蓋)

- Post Features
- Comment Features
- Category Features
- Enquirie Feature
- User Feature

測試案例中每個操作皆需要有 **assertion or wait component visible**

以確保網站正確性

### 1.2 Features to be tested

- Post Features
  - Create post on the Admin UI page
  - Edit post on the Admin UI page
  - Delete post on the Admin UI page
  - Search posts by keyword on the Admin UI page
- Comment Features
  - Create comment on Admin UI page
  - Edit comment on Admin UI page
  - Delete comment on Admin UI page
- Category Features
  - Create category on Admin UI page

- Show posts of the specific category by pressing category name on the "Blog" page
- Enquire Feature
  - Create enquiry on the "Contact" page
  - Delete enquiry on Admin UI page
- User Feature
  - Create a new user on Admin UI page(Name, Email, Phone, and Password must be set when creating the new user)

### 1.3 Test environment and/or infrastructure

1. 安裝 VSCODE
2. 安裝 [python3.6](#)
3. 安裝 selenium library
4. 安裝 [chrome driver](#)
5. 架設 keystoneJS
6. Run testcase(python MainTestsuite.py)

### 1.4 Success criteria

每項操作後之 **assertion or wait component visible** 皆通過  
12 個測試案例全數 pass

### 1.5 Test approaches

使用 ISP 設計 Test Case 需先辨別輸入欄位或下拉選單可能的 partitions 和邊界值，針對 input 設計條件，並設計可能組合以及邊界值，最終根據這些組合，去 設想測試資料並實作出來，最終將所有可能組合 cover 住，確保 testcase 測試完整性。

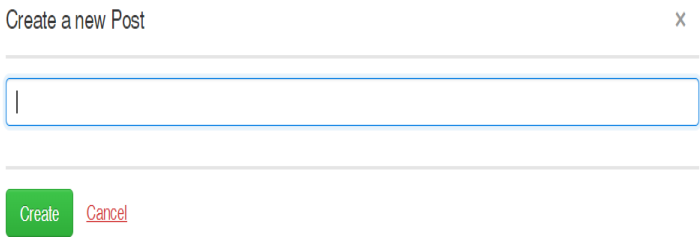
我在這次的 lab 會先設想欄位可以輸入什麼值或下拉選單可以選擇那些選項最後去進行 **assertion or wait component visible** 確保操作符合預期以及網頁運作正確。

### 1.6 Test activities

To implement the proposed strategy, the following activities are planned to perform.

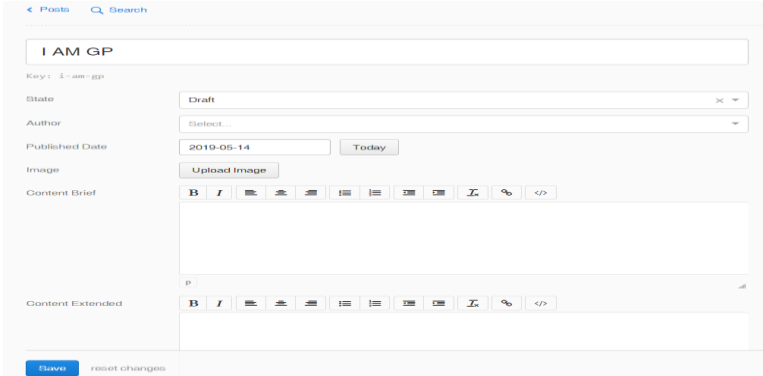
No.	Activity Name	Plan hours	Schedule Date
1	Study selenium library	3	5/7
2	Study python unittest	2	5/7
3	Design test cases for the selected methods	12	5/8
4	Implement test cases	13	5/9-5/12
5	Perform tests	1	5/13
6	Complete Lab4 report	6	5/13-5/15

## 2 Test Design

Use Case Section	Post
Use Case Name	Create post on the Admin UI page (Post.py 的 test_create_post)
Preconditions	Login and go to Posts page
Success Guarantee	創建的 post 皆出現在 Post 頁面上
Main Success Scenario	<ol style="list-style-type: none"> <li>選擇創建 post</li> <li>輸入欲創建之 post name</li> <li>創建並儲存</li> <li>回到 post 頁面</li> <li>確認創建之 post 皆存在於頁面上</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>2.a 可以輸入不同類型 post name               <ol style="list-style-type: none"> <li>web 正常回應</li> <li>繼續進行接下來操作</li> </ol> </li> <li>2.b 輸入不同類型 post name 造成 error               <ol style="list-style-type: none"> <li>web 不正常顯示</li> <li>繼續進行接下來操作</li> </ol> </li> <li>5.a 創建之 post 皆存在於頁面               <ol style="list-style-type: none"> <li>通過 post <b>assertion or wait component visible</b></li> <li>頁面暫停幾秒 (通過 assertion 才會暫停方便人工查看)</li> <li>關閉瀏覽器</li> </ol> </li> <li>5.b 創建之 post 不存在於頁面               <ol style="list-style-type: none"> <li>post <b>assertion or wait component visible</b> 失敗</li> <li>關閉瀏覽器</li> <li>顯現 Fail reason</li> </ol> </li> </ol>
Input Field And Value (ISP)	<p>對 post name 欄位做 ISP 設計</p>  <p>C1: 輸入英文字母 valid: "[A-Z]*" invalid: 非 "[A-Z]*"</p> <p>C2: 輸入數字 valid: "[0-9]*" invalid: 非 "[0-9]*"</p> <p>C3: 只輸入純符號 valid: any character invalid: 不是純符號構成</p>

	<p>Test Requirements: {TFF, FTF, FFT, TTF }</p> <p>Infeasible TRs: { TFT, FTT, TTT, FFF }</p> <p>T1: "I AM GP" covers {TFF}</p> <p>T2: "34567" covers {FTF}</p> <p>T3: "TEST1" covers {TTF}</p> <p>T4: "!@#\$\$%" covers {FFT}</p>
--	--

Use Case Section	Post
Use Case Name	Edit post on the Admin UI page (Post.py 的 test_edit_post)
Preconditions	已存在一個或以上之 Post
Success Guarantee	Edit 的欄位值確實被儲存並顯示
Main Success Scenario	<ol style="list-style-type: none"> <li>1. 選擇欲編輯 post</li> <li>2. 進入 post detail</li> <li>3. 編輯 State 欄位(可選之選項全測)</li> <li>4. 編輯 Author 欄位(可選之選項全測)</li> <li>5. 編輯 Published Date 欄位 (該月日期 1-30 隨機挑選)</li> <li>6. 編輯 Content Brief 欄位</li> <li>7. 編輯 Content Extended 欄位</li> <li>8. 儲存對 post 之修改</li> <li>9. 確認各欄位顯示符合預期結果</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>1.a 選擇 post 成功 <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>1.b 選擇 post 失敗 <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> </li> <li>3.a 編輯 State 欄位成功 <ol style="list-style-type: none"> <li>1. web 正常顯示 State 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>3.b 編輯 State 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 State 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>4.a 編輯 Author 欄位成功 <ol style="list-style-type: none"> <li>1. web 正常顯示 Author 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>4.b 編輯 Author 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 Author 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>5.a 編輯 Published Date <p>該月日期在 1-30 之間 <b>random</b> 挑</p> <ol style="list-style-type: none"> <li>1.a 日期選到 30 號(該月存在 30 號) <ol style="list-style-type: none"> <li>1. web 正常顯示</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>1.b 日期選到 30 號(該月 <b>不</b>存在 30 號) <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> </li> </ol> </li> </ol>

	<p>6.a 編輯 Content Brief 欄位成功</p> <ol style="list-style-type: none"> <li>1. web 正常顯示 Content Brief 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>6.b 編輯 Content Brief 欄位失敗</p> <ol style="list-style-type: none"> <li>1. web 不正常顯示 Content Brief 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>7.a 編輯 Content Extended 欄位成功</p> <ol style="list-style-type: none"> <li>1. web 正常顯示 Content Extended 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>7.b 編輯 Content Extended 欄位失敗</p> <ol style="list-style-type: none"> <li>1. web 不正常顯示 Content Extended 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>9.a Edit 過之 post field value 皆在頁面上顯示</p> <ol style="list-style-type: none"> <li>1. 通過 Edit post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 assertion 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> <p>9.b Edit 過之 post field value 未在頁面上顯示</p> <ol style="list-style-type: none"> <li>1. Edit post <b>assertion or wait component visible</b> 失敗</li> <li>2. 關閉瀏覽器</li> <li>3. 顯現 Fail reason</li> </ol>
<p><b>Input Field And Value (ISP)</b></p>	<p>對 post 各欄位做 ISP 設計</p>  <p>C1: 選擇之 State 欄位值存在 valid: select value in dropdown invalid: select value not in dropdown</p> <p>C2: 輸入之 Author 欄位值存在 valid: select value in dropdown invalid: select value not in dropdown</p> <p>C3: 選擇之 Published Date 欄位值存在 valid: 該月有該日期 (e.g. 2019-05-30) invalid: 該月無該日期 (e.g. 2019-02-30)</p> <hr/> <p>Test Requirements: {TTF, TTT, TFT, TFF } Infeasible TRs: { FTT, FTF, FFT, FFF }</p> <hr/>

	<p>T1: State: "Published" Author: "DEMO User" Published Date: 2019-05-30 covers {TTT}</p> <p>T2: State: "Published" Author: "DEMO User" Published Date: 2019-02-30(not exist) covers {TTF}</p> <p>T3: State: "Published" Author: "TEST"(not exist) Published Date: 2019-05-30 covers {TFT}</p> <p>T4: State: "Published" Author: "TEST"(not exist) Published Date: 2019-02-30(not exist) covers {TFF}</p>
--	---

Use Case Section	Post
Use Case Name	Search posts by keyword on the Admin UI page ( Post.py 的 test_search_post)
Preconditions	Login and go to Posts page
Success Guarantee	搜尋的 post 確實出現在頁面上
Main Success Scenario	<ol style="list-style-type: none"> <li>點選搜尋欄位</li> <li>輸入欲搜尋之 post name (有用 all of name and part of name 去搜尋 e.g name= "TEST" 我也有用 name="TE"去測試 )</li> <li>確認欲搜尋之 post 出現於頁面上</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>2.a 可以輸入不同類型 post name <ol style="list-style-type: none"> <li>web 正常回應</li> <li>繼續進行接下來操作</li> </ol> </li> <li>2.b 輸入不同類型 post name 造成 error <ol style="list-style-type: none"> <li>web 不正常顯示</li> <li>繼續進行接下來操作</li> </ol> </li> <li>3.a 創建之 post 皆存在於頁面 <ol style="list-style-type: none"> <li>通過 post assertion or wait component visible</li> <li>頁面暫停幾秒 (通過 assertion 才會暫停方便人工查看)</li> <li>關閉瀏覽器</li> </ol> </li> <li>3.b 創建之 post 不存在於頁面 <ol style="list-style-type: none"> <li>post assertion or wait component visible 失敗</li> <li>關閉瀏覽器</li> <li>顯現 Fail reason</li> </ol> </li> </ol>

<b>Input Field And Value (ISP)</b>	<p>對搜尋 post 欄位做 ISP 設計</p> <div data-bbox="539 235 1453 342">  </div> <p> C1: 輸入英文字母  valid: "[A-Z]*" invalid: 非 "[A-Z]*"  C2: 輸入數字  valid: "[0-9]*" invalid: 非 "[0-9]*"  C3: 只輸入純符號  valid: any character  invalid: 不是純符號構成 </p> <hr/> <p> Test Requirements: {TFF, FTF, FFT, TTF}  Infeasible TRs: {TFT, FTT, TTT, FFF} </p> <hr/> <p> T1: {"I AM GP", "I", "I AM"} covers {TFF}  T2: {"34567", "345", "3"} covers {FTF}  T3: "TEST1" covers {TTF}  T4: {"!@#\$%", "!@#", "!"} covers {FFT} </p>
------------------------------------	--

Use Case Section	Post
<b>Use Case Name</b>	Deletee post on the Admin UI page (Post.py 的 test_deletete_post)
<b>Preconditions</b>	Login and go to Posts page
<b>Success Guarantee</b>	刪除的 post 確實消失在 Posts 頁面上
<b>Main Success Scenario</b>	1. 選擇欲刪除之 post 2. 刪除該 post 3. 確認刪除之 post 確實消失於 Posts 頁面上
<b>Extensions</b>	2.a 可以依序刪除各 post 1. web 正常回應 2. 繼續進行接下來操作 2.b 不可以依序刪除各 post 5. web 不正常顯示 6. 繼續進行接下來操作 3.a 刪除之 post 確實消失於 Posts 頁面 1. 通過 post <b>assertion or wait component visible</b> 2. 頁面暫停幾秒 (通過 <b>assertion</b> 才會暫停方便人工查看) 3. 關閉瀏覽器 3.b 刪除之 post 依然存在於 Posts 頁面 1. post <b>assertion or wait component visible</b> 失敗 2. 關閉瀏覽器 3. 顯現 Fail reason


<b>Input Field And Value (ISP)</b>	<p>此 testcase 無明顯可設計輸入之欄位 故只有確認所創建之 4 個 Post</p> <ol style="list-style-type: none"> <li>1. I AM GP</li> <li>2. TEST1</li> <li>3. 34567</li> <li>4. !@#\$%</li> </ol> <p>有依序且確實被刪除</p>
------------------------------------	---

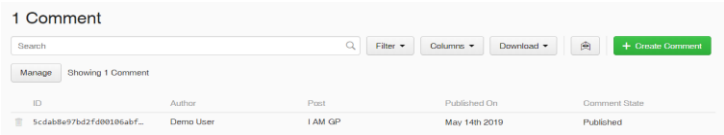
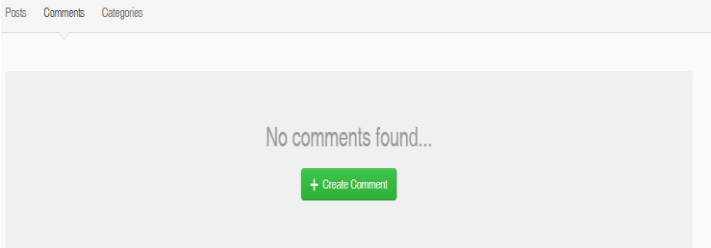
Use Case Section	Comment
<b>Use Case Name</b>	Create comment on Admin UI page ( Comment.py 的 test_create_comment )
<b>Preconditions</b>	至少存在一個或以上的 Post 並已至 Comments 頁面
<b>Success Guarantee</b>	創建的 comment 出現在 Comments 頁面上
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>1. 選擇創建 comment</li> <li>2. 選擇 Author 欄位</li> <li>3. 選擇要回覆之 Post</li> <li>4. 按下創建 comment</li> <li>5. 回到 Comments 頁面</li> <li>6. 確認創建之 comment 存在於頁面上</li> </ol>
<b>Extensions</b>	<p>3.a 輸入要回覆之 post name 存在</p> <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> <p>3.b 輸入要回覆之 post name 不存在造成 error</p> <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> <p>6.a 創建之 comment 存在於頁面</p> <ol style="list-style-type: none"> <li>1. 通過 post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 assertion 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> <p>6.b 創建之 comment 不存在於頁面</p> <ol style="list-style-type: none"> <li>1. post <b>assertion or wait component visible</b> 失敗</li> <li>2. 自動重選一個存在的 Post</li> </ol>

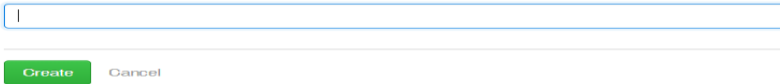


<b>Input Field And Value (ISP)</b>	<p>對 post name 欄位做 ISP 設計</p>  <p>C1: 輸入之 Post 是否存在  valid: "I AM GP"  invalid: "TEST"</p> <p>-----</p> <p>Test Requirements: {T,F }</p> <p>-----</p> <p>T1: "I AM GP" covers {T}  T2: "TEST" covers {F}</p>
------------------------------------	--

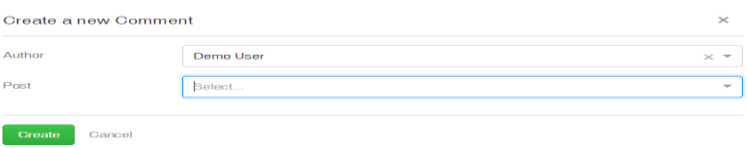
Use Case Section	Comment
<b>Use Case Name</b>	Edit comment on Admin UI page ( Comment.py 的 test_edit_comment )
<b>Preconditions</b>	已存在一個或以上之 comment
<b>Success Guarantee</b>	Edit 的欄位值確實被儲存並顯示
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>1. 選擇欲編輯 comment</li> <li>2. 進入 comment detail</li> <li>3. 編輯 State 欄位(可選之選項全測)</li> <li>4. 編輯 Author 欄位(可選之選項全測)</li> <li>5. 編輯 Content 欄位</li> <li>6. 儲存對 post 之修改</li> <li>7. 確認各欄位顯示符合預期結果</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>1.a 選擇 comment 成功 <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>1.b 選擇 comment 失敗 <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> </li> <li>3.a 編輯 State 欄位成功 <ol style="list-style-type: none"> <li>3. web 正常顯示 State 欄位值</li> <li>4. 繼續進行接下來操作</li> </ol> </li> <li>3.b 編輯 State 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 State 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>4.a 編輯 Author 欄位成功 <ol style="list-style-type: none"> <li>3. web 正常顯示 Author 欄位值</li> <li>4. 繼續進行接下來操作</li> </ol> </li> <li>4.b 編輯 Author 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 Author 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>5.a 編輯 Content 欄位成功 <ol style="list-style-type: none"> <li>1. web 正常顯示 Content Brief 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> </ol>

	<p>5.b 編輯 Content Brief 欄位失敗</p> <ol style="list-style-type: none"> <li>1. web 不正常顯示 Content Brief 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>7.a Edit 過之 post field value 皆在頁面上顯示</p> <ol style="list-style-type: none"> <li>1. 通過 Edit post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 <b>assertion</b> 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> <p>7.b Edit 過之 post field value 未在頁面上顯示</p> <ol style="list-style-type: none"> <li>1. Edit post <b>assertion or wait component visible</b> 失敗</li> <li>2. 關閉瀏覽器</li> <li>3. 顯現 Fail reason</li> </ol>
<p><b>Input Field And Value (ISP)</b></p>	<p>對 comment 各欄位做 ISP 設計</p>  <p>C1: 選擇之 State 欄位值存在 valid: select value in dropdown invalid: select value not in dropdown</p> <p>C2: 輸入之 Author 欄位值存在 valid: select value in dropdown invalid: select value not in dropdown</p> <p>C3: 輸入 Content 欄位值不為英文字母 valid: 輸入數字或符號 invalid: "[A-Z]*"</p> <hr/> <p>Test Requirements: {TTF, TTT, TFT, TFF } Infeasible TRs: { FTT, FTF, FFT, FFF }</p> <hr/> <p>T1: State: "Published" Author: "DEMO User" Content: "12345" covers {TTT}</p> <p>T2: State: "Published" Author: "DEMO User" Content: "I AM GP"(不為數字或符號) covers {TTF}</p> <p>T3: State: "Published" Author: "TEST"(not exist) Content: "!@#\$\$%" covers {TFT}</p> <p>T4: State: "Published" Author: "TEST"(not exist) Content: "I AM GP"(不為數字或符號) covers {TFF}</p>

Use Case Section	Comment
Use Case Name	Delete comment on the Admin UI page (Comment.py 的 test_delete_comment)
Preconditions	Login and go to Comments page
Success Guarantee	刪除的 post 確實消失在 Posts 頁面上
Main Success Scenario	<ol style="list-style-type: none"> <li>選擇欲刪除之 comment</li> <li>刪除該 comment</li> <li>確認刪除之 comment 確實消失於 Comments 頁面上</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>2.a 可以刪除 comment <ol style="list-style-type: none"> <li>web 正常回應</li> <li>繼續進行接下來操作</li> </ol> </li> <li>2.b 不可以刪除 comment <ol style="list-style-type: none"> <li>web 不正常顯示</li> <li>繼續進行接下來操作</li> </ol> </li> <li>3.a 刪除之 comment 確實消失於 Comments 頁面 <ol style="list-style-type: none"> <li>通過 post <b>assertion or wait component visible</b></li> <li>頁面暫停幾秒 (通過 <b>assertion</b> 才會暫停方便人工查看)</li> <li>關閉瀏覽器</li> </ol> </li> <li>3.b 刪除之 comment 依然存在於 Comments 頁面 <ol style="list-style-type: none"> <li>post <b>assertion or wait component visible</b> 失敗</li> <li>關閉瀏覽器</li> <li>顯現 Fail reason</li> </ol> </li> </ol>
Input Field And Value (ISP)	<p>此 testcase 無明顯可設計輸入之欄位 故只有確認所創建之 comment</p>  <p>有確實被刪除</p> 

Use Case Section	Category
Use Case Name	Create category on Admin UI page (Category.py 的 test_create_category)
Preconditions	Login and go to Categories page
Success Guarantee	創建的 Category 皆出現在 Categories 頁面上
Main Success Scenario	<ol style="list-style-type: none"> <li>1. 選擇創建 Category</li> <li>2. 輸入欲創建之 Category name</li> <li>3. 創建並儲存</li> <li>4. 回到 Categories 頁面</li> <li>5. 確認創建之 Category 皆存在於頁面上</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>2.a 可以輸入不同類型 Category name               <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>2.b 輸入不同類型 Category name 造成 error               <ol style="list-style-type: none"> <li>7. web 不正常顯示</li> <li>8. 繼續進行接下來操作</li> </ol> </li> <li>5.a 創建之 Category 皆存在於頁面               <ol style="list-style-type: none"> <li>1. 通過 post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 <b>assertion</b> 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> </li> <li>5.b 創建之 Category 不存在於頁面               <ol style="list-style-type: none"> <li>1. post <b>assertion or wait component visible</b> 失敗</li> <li>2. 關閉瀏覽器</li> <li>3. 顯現 Fail reason</li> </ol> </li> </ol>
Input Field And Value (ISP)	<p>對 Category name 欄位做 ISP 設計</p>  <p>C1: 輸入英文字母 valid: "[A-Z]*" invalid: 非 "[A-Z]*"</p> <p>C2: 輸入數字 valid: "[0-9]*" invalid: 非 "[0-9]*"</p> <p>C3: 只輸入純符號 valid: any character invalid: 不是純符號構成</p> <hr/> <p>Test Requirements: {TFF, FTF, FFT, TTF } Infeasible TRs: { TFT, FTT, TTT, FFF }</p> <hr/> <p>T1: "Test" covers {TFF} T2: "34567" covers {FTF} T3: "Test1" covers {TTF} T4: "!@#\$\$%" covers {FFT}</p>

Use Case Section	Comment
Use Case Name	Show posts of the specific category by pressing category name on the "Blog" page ( Comment.py 的 test_show_post )
Preconditions	Login and go to Posts page
Success Guarantee	創建的 Post 出現在該 Category 頁面下
Main Success Scenario	<ol style="list-style-type: none"> <li>1. 選擇創建 post</li> <li>2. 輸入 Post Name</li> <li>3. 創建 post</li> <li>4. 選擇 state</li> <li>5. 選擇 category</li> <li>6. 進入 blog tab 下</li> <li>7. 點選該 post 所屬之 category</li> <li>8. 確認該 post 正確顯示</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>2.a 輸入要創建之 post name 合乎規則 <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>2.b 輸入要創建之 post name 不合乎規則造成 error <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯示 Fail reason</li> </ol> </li> <li>4.a 選擇 State 欄位成功 <ol style="list-style-type: none"> <li>5. web 正常顯示 State 欄位值</li> <li>6. 繼續進行接下來操作</li> </ol> </li> <li>4.b 選擇 State 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 State 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>5.a 選擇 Category 欄位成功 <ol style="list-style-type: none"> <li>5. web 正常顯示 Author 欄位值</li> <li>6. 繼續進行接下來操作</li> </ol> </li> <li>5.b 選擇 Category 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 Author 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>8.a 創建之 comment 存在於頁面 <ol style="list-style-type: none"> <li>1. 通過 post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 assertion 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> </li> <li>8.b 創建之 comment 不存在於頁面 <ol style="list-style-type: none"> <li>1. post <b>assertion or wait component visible</b> 失敗</li> <li>2. 自動重選一個存在的 Post</li> </ol> </li> </ol>

<b>Input Field And Value (ISP)</b>	<p>對 post name 欄位做 ISP 設計</p>  <p>C1: 輸入之 State 可顯現在 blog valid: "Published" (選擇 published 才可顯現出來) invalid: {"Drafted", "Archived"}</p> <p>C2: 輸入之 Category 存在 valid: "TEST",</p> <p>-----</p> <p>Test Requirements: {TT, TF, FT, FF }</p> <p>-----</p> <p>T1: State: "Published" Category: "Test" covers {TT}</p> <p>T2: State: "Published" Category: "GP"(Not Exist) covers {TF}</p> <p>T3: State: "Draft" Category: "Test" covers {FT}</p> <p>T3: State: "Draft" Category: "GP"(Not Exist) covers {FF}</p>
------------------------------------	---

Use Case Section	Enquirie
<b>Use Case Name</b>	Create enquiry on the "Contact" page (Enquire.py 的 test_create_enquiry)
<b>Preconditions</b>	go to Contact page
<b>Success Guarantee</b>	成功創建 Enquirie
<b>Main Success Scenario</b>	<ol style="list-style-type: none"> <li>輸入 Name 欄位</li> <li>輸入 Email 欄位</li> <li>輸入 Phone 欄位</li> <li>選擇 Regarding</li> <li>輸入 Message</li> <li>提交 Enquirie</li> <li>確認提交成功</li> </ol>
<b>Extensions</b>	<ol style="list-style-type: none"> <li>輸入要創建之 Name 成功             <ol style="list-style-type: none"> <li>web 正常回應</li> <li>繼續進行接下來操作</li> </ol> </li> <li>輸入要創建之 Name 失敗             <ol style="list-style-type: none"> <li>web 不正常顯示 Name 欄位</li> <li>繼續進行接下來操作</li> </ol> </li> </ol>

	<p>2.a 輸入 Email 欄位成功</p> <ol style="list-style-type: none"> <li>1. web 正常顯示 Email 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>2.b 輸入 Email 欄位失敗</p> <ol style="list-style-type: none"> <li>1. web 不正常顯示 Email 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> <p>3.a 輸入 Phone 欄位成功</p> <ol style="list-style-type: none"> <li>1. web 正常顯示 Phone 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>3.b 輸入 Phone 欄位失敗</p> <ol style="list-style-type: none"> <li>1. web 不正常顯示 Phone 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> <p>4.a 選擇 Regarding 欄位成功</p> <ol style="list-style-type: none"> <li>1. web 正常顯示 State 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> <p>4.b 選擇 Regarding 欄位失敗</p> <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> <p>5.a 輸入之 Message 合乎規則</p> <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> <p>5.b 輸入之 Message 不合乎規則造成 error</p> <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> <p>7.a 確認 Enquire 提交成功</p> <ol style="list-style-type: none"> <li>1. 通過 post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 <b>assertion</b> 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> <p>7.b 發現 Enquire 提交失敗</p> <ol style="list-style-type: none"> <li>1. post <b>assertion or wait component visible</b> 失敗</li> <li>2. 關閉瀏覽器</li> <li>3. 顯現 Fail reason</li> </ol>
--	---

## Input Field And Value (ISP)

對 Category name 欄位做 ISP 設計

### Contact

Name	<input type="text" value="First Last"/>
Email	<input type="text" value="name@domain.com"/>
Phone	<input type="text" value="(optional)"/>
Regarding	<input type="text" value="(required)"/>
Message	<input type="text" value="Leave us a message..."/>

Note: others can see your enquiry in the Admin UI

C1: phone 欄位非手機號碼  
valid: "035824384"(市內電話)  
invalid: "0962010830"(手機)

C2: Regarding 都可選  
valid: {"Just leaving a message", "I've got a question"  
"Something else..."}

Invalid: **someone can not be selected**

C3: Message 純英文字

Valid: "Test"

invalid: {"12345", "Test1 ", "!@#%\$%"}  
-----

Test Requirements: {TTT, FTF, FTT, TTF }

Infeasible TRs: { TFT, FFT, TFF, FFF }

(目前還未發現不可點選情形)  
-----

T1: phone: 035824384

Regarding: "Just leaving a message"

Message: "EDIT TEST"

covers {TTT}

T2: phone: **0962010830(手機)**

Regarding: "Just leaving a message"

Message: **"12345"**

covers {FTF}

T3: phone: **0962010830(手機)**

Regarding: "Just leaving a message"

Message: "EDIT TEST"

covers {FTT}

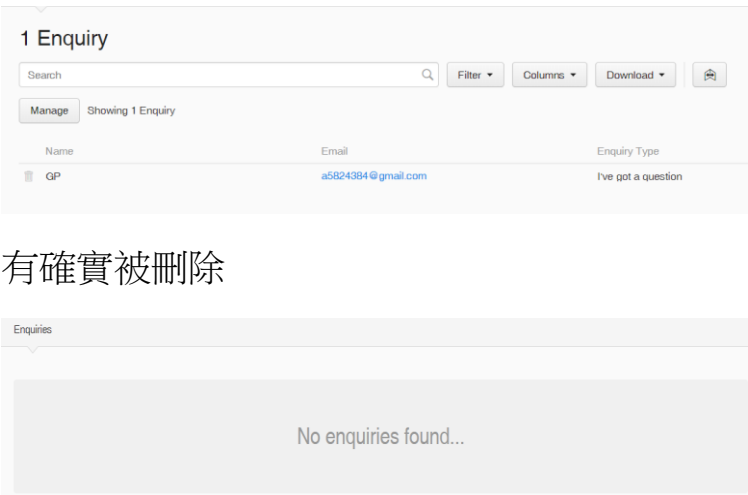
T4: phone: 035824384

Regarding: "Just leaving a message"

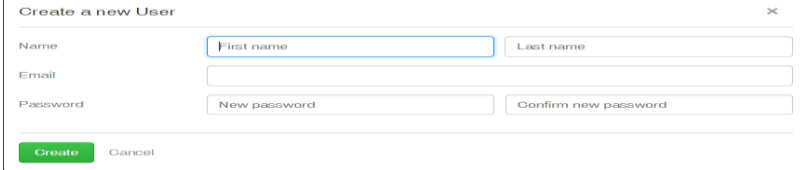
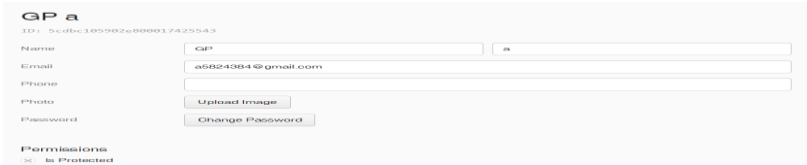
Message: **"12345"**

covers {TTF}



Use Case Section	Enquire
Use Case Name	Delete enquiry on Admin UI page (Enquire.py 的 test_delete_enquiry)
Preconditions	Login and go to Enquiries page
Success Guarantee	刪除的 Enquire 確實消失在 Enquiries 頁面上
Main Success Scenario	<ol style="list-style-type: none"> <li>選擇欲刪除之 Enquire</li> <li>刪除該 Enquire</li> <li>確認刪除之 Enquire 確實消失於 Enquiries 頁面上</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>可以刪除 Enquire <ol style="list-style-type: none"> <li>web 正常回應</li> <li>繼續進行接下來操作</li> </ol> </li> <li>不可以刪除 Enquire <ol style="list-style-type: none"> <li>web 不正常顯示</li> <li>繼續進行接下來操作</li> </ol> </li> <li>刪除之 Enquire 確實消失於 Enquiries 頁面 <ol style="list-style-type: none"> <li>通過 post <b>assertion or wait component visible</b></li> <li>頁面暫停幾秒 (通過 assertion 才會暫停方便人工查看)</li> <li>關閉瀏覽器</li> </ol> </li> <li>刪除之 Enquire 依然存在於 Enquiries 頁面 <ol style="list-style-type: none"> <li>post <b>assertion or wait component visible</b> 失敗</li> <li>關閉瀏覽器</li> <li>顯現 Fail reason</li> </ol> </li> </ol>
Input Field And Value (ISP)	<p>此 testcase 無明顯可設計輸入之欄位 故只有確認所創建之 Enquire</p>  <p>有確實被刪除</p>

Use Case Section	User
Use Case Name	Create a new user on Admin UI page (Name, Email, Phone, and Password must be set when creating the new user) (User.py 的 test_create_user)
Preconditions	go to Users page
Success Guarantee	成功創建 User
Main Success Scenario	<ol style="list-style-type: none"> <li>1. 輸入 First Name and Last Name 欄位</li> <li>2. 輸入 Email 欄位</li> <li>3. 輸入 Password 欄位</li> <li>4. 輸入 Confirm Password 欄位</li> <li>5. 創建 User</li> <li>6. 進入 User detail</li> <li>7. 輸入 Phone 欄位</li> <li>8. 儲存變更</li> <li>9. 回到 Users 頁面</li> <li>10. 確認 User 已創建</li> </ol>
Extensions	<ol style="list-style-type: none"> <li>1.a 輸入要創建之 First Name and Last Name 成功 <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>1.b 輸入要創建之 First Name and Last Name 失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 Name 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>2.a 輸入 Email 欄位成功 <ol style="list-style-type: none"> <li>1. web 正常顯示 Email 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>2.b 輸入 Email 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 Email 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>3.a 輸入 Password 欄位成功 <ol style="list-style-type: none"> <li>1. web 正常顯示 Password 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>3.b 輸入 Password 欄位失敗 <ol style="list-style-type: none"> <li>1. web 不正常顯示 Password 欄位</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>4.a 輸入 Confirm Password 欄位和 Password 一致 <ol style="list-style-type: none"> <li>1. web 正常顯示 State 欄位值</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>4.b 輸入 Confirm Password 欄位和 Password 一致 <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> </li> <li>7.a 輸入之 Phone 合乎規則 <ol style="list-style-type: none"> <li>1. web 正常回應</li> <li>2. 繼續進行接下來操作</li> </ol> </li> <li>7.b 輸入之 Phone 不合乎規則造成 error <ol style="list-style-type: none"> <li>1. 關閉瀏覽器</li> <li>2. 顯現 Fail reason</li> </ol> </li> </ol>

	<p>10.a 確認 User 創建成功</p> <ol style="list-style-type: none"> <li>1. 通過 post <b>assertion or wait component visible</b></li> <li>2. 頁面暫停幾秒 (通過 <b>assertion</b> 才會暫停方便人工查看)</li> <li>3. 關閉瀏覽器</li> </ol> <p>10.b 發現 User 創建失敗</p> <ol style="list-style-type: none"> <li>1. post <b>assertion or wait component visible</b> 失敗</li> <li>2. 關閉瀏覽器</li> <li>3. 顯現 Fail reason</li> </ol>
Input Field And Value (ISP)	<p>對 User 各欄位做 ISP 設計</p>   <p>C1: phone 欄位非手機號碼 valid: "035824384"(市內電話) invalid: "0962010830"(手機)</p> <p>C2: Email 不重複 valid: "iamgp@ntut.org.tw" Invalid: <b>Email already exist</b></p> <p>C3: passwd 只包含英文或數字 Valid: {"12345678", "james0000"} invalid: "!@#\$\$%"</p> <p>-----</p> <p>Test Requirements: {TTT, FTF, FTT, TTF, TFT, FFT, TFF, FFF }</p> <p>-----</p> <p>T1: phone: 035824384 Email: " iamgp@ntut.org.tw" passwd: "james0000" covers {TTT}</p> <p>T2: phone: <b>0962010830(手機)</b> Email: " iamgp@ntut.org.tw" passwd: <b>"!@#\$\$%"</b> covers {FTF}</p> <p>T3: phone: <b>0962010830(手機)</b> Email: " iamgp@ntut.org.tw" passwd: "james0000" covers {FTT}</p> <p>T4: phone: 035824384 Email: " iamgp@ntut.org.tw" passwd: <b>"!@#\$\$%"</b> covers {TTF}</p>

	T5: phone: 035824384 Email: <a href="mailto:a5824384@gmail.com">a5824384@gmail.com</a> (已存在) passwd: "james0000" covers {TFT} T6: phone: 0962010830(手機) Email: <a href="mailto:a5824384@gmail.com">a5824384@gmail.com</a> (已存在) passwd: "james0000" covers {FFT} T7: phone: 035824384 Email: <a href="mailto:a5824384@gmail.com">a5824384@gmail.com</a> (已存在) passwd: "!@#\$\$%" covers {TFF} T8: phone: 0962010830(手機) Email: <a href="mailto:a5824384@gmail.com">a5824384@gmail.com</a> (已存在) passwd: "!@#\$\$%" covers {TFF}
--	--

### 3 Test Implementation

The design of test cases specified in Section 2 was implemented using JUnit

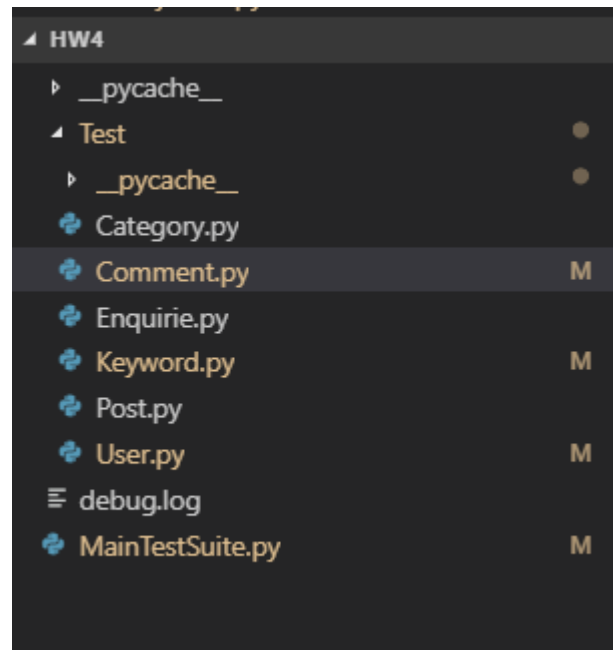
4. The test scripts of 3 selected test cases are given below. [The rest of the test script implementations can be found in the link .](#) < — 請點我

No .	Test method	Source test code
1	test_create_post	<pre> class test_post(unittest.TestCase): # 測試項目 def setUp(self):     self.url="http://127.0.0.1:3000/" # 要執行自動測試的網站  def test_create_post(self):     test=webdriver.Chrome()     test.get( self.url )     test.maximize_window()     # precondition     login( test )     get_web_element( test, '//a[contains(@href,"posts") and contains(@class,"dashboard-group_list-tile")]').click()     # go to create page     create_post( test, 'I AM GP' )     top_tab_go_to_page( test, 'posts' )     create_post( test, 'TEST1' )     top_tab_go_to_page( test, 'posts' )     create_post( test, '!@#\$\$%' )     top_tab_go_to_page( test, 'posts' )     create_post( test, '34567' )     top_tab_go_to_page( test, 'posts' )      exist = is_text_present( test, 'I AM GP' )     self.assertTrue( exist )     exist = is_text_present( test, 'TEST1' )     self.assertTrue( exist )     exist = is_text_present( test, '34567' )     self.assertTrue( exist )     exist = is_text_present( test, '!@#\$\$%' )     self.assertTrue( exist )      time.sleep(2)     test.close() </pre>

2	test_search_post	<pre> def test_search_post(self):     test=webdriver.Chrome()     test.get( self.url )     test.maximize_window()     login( test )     # precondition      get_web_element( test, '//*[contains(@href,"posts") and contains(@class,"dashboard-group__list-tile")]').click()      ExpectText = search_post(test,'!')     self.assertEqual( ExpectText, '!@#%' )     ExpectText = search_post(test,'!@#')     self.assertEqual( ExpectText, '!@#%' )     ExpectText = search_post(test,'!@#%')     self.assertEqual( ExpectText, '!@#%' )      ExpectText = search_post( test,'I' )     self.assertEqual( ExpectText, 'I AM GP' )     ExpectText = search_post( test,'I AM' )     self.assertEqual( ExpectText, 'I AM GP' )     ExpectText = search_post( test,'I AM GP' )     self.assertEqual( ExpectText, 'I AM GP' )      ExpectText = search_post(test,'3')     self.assertEqual( ExpectText, '34567' )     ExpectText = search_post(test,'345' )     self.assertEqual( ExpectText, '34567' )     ExpectText = search_post(test,'34567' )     self.assertEqual( ExpectText, '34567' )      ExpectText = search_post(test,'TEST1' )     self.assertEqual( ExpectText, 'TEST1' )     time.sleep(2)     test.close() </pre>
3	test_create_enquiry	<pre> class test_enquire(unittest.TestCase): # 測試項目     def setUp(self):         self.url="http://127.0.0.1:3000/" # 要執行自動測試的網站      def test_create_enquiry(self):         test=webdriver.Chrome()         test.get( self.url )         test.maximize_window()         go_to_login_page_subtab(test, 'contact' )         # precondition         input_field( test, 'name', "GP" )         input_field( test, 'email', "a5824384@gmail.com" )         input_field( test, 'phone', "0962010830" )         select_contact_dropdown_field( test, 'Just leaving a message' )         exist = is_text_present( test, 'Just leaving a message' )         self.assertTrue( exist )         select_contact_dropdown_field( test, "I've got a question" )         exist = is_text_present( test, "I've got a question" )         self.assertTrue( exist )         select_contact_dropdown_field( test, 'Something else...' )         exist = is_text_present( test, 'Something else...' )         self.assertTrue( exist )          input_text_area( test, 'message', 'EDIT TEST!!!' )         submit( test )          success = get_web_element( test, '//*[normalize-space() = "Success!"]' ).text         self.assertEqual( success, 'Success!' )         time.sleep(2)         test.close() </pre>

## 4 Test Results

### 4.1 Selenium test scripts



### 4.2 execution results

```
PS C:\Users\GP\Desktop\ST\HW4> python .\MainTestSuite.py
test_create_post (Test.Post.test_post) ...
DevTools listening on ws://127.0.0.1:56189/devtools/browser/a5050f64-b10a-4d58-8dc9-3bfeffd98b4f
ok
test_edit_post (Test.Post.test_post) ...
DevTools listening on ws://127.0.0.1:56246/devtools/browser/0d888dbf-98c1-470b-afb2-3978b0d09552
ok
test_search_post (Test.Post.test_post) ...
DevTools listening on ws://127.0.0.1:56289/devtools/browser/691cf8c5-9ab5-4650-9aeb-bd7440b4c081
ok
test_delete_post (Test.Post.test_post) ...
DevTools listening on ws://127.0.0.1:56338/devtools/browser/9690d220-32d9-4741-8010-3e1d17b6cbcd
ok
test_create_comment (Test.Comment.test_comment) ...
DevTools listening on ws://127.0.0.1:56379/devtools/browser/4a383945-bae3-4fdd-865a-8547cced80d7
ok
test_edit_comment (Test.Comment.test_comment) ...
DevTools listening on ws://127.0.0.1:56422/devtools/browser/483a697c-a4a0-4f6d-be99-5a195f9bc7fd
ok
test_delete_comment (Test.Comment.test_comment) ...
DevTools listening on ws://127.0.0.1:56462/devtools/browser/fcb368d5-1b74-44d3-868b-a66ebc2bd3b5
ok
test_create_category (Test.Category.test_category) ...
DevTools listening on ws://127.0.0.1:56528/devtools/browser/dc7fb242-989c-4f9f-a8c1-15f6adc17ad5
ok
test_show_post (Test.Category.test_category) ...
DevTools listening on ws://127.0.0.1:56572/devtools/browser/bc38e95a-e720-4d45-8ec8-585aa91e8d37
ok
test_create_enquiry (Test.Enquirie.test_enquire) ...
DevTools listening on ws://127.0.0.1:56616/devtools/browser/433327cc-fdd9-425e-978f-cd5778c21aa4
ok
test_delete_enquiry (Test.Enquirie.test_enquire) ...
DevTools listening on ws://127.0.0.1:56655/devtools/browser/6f193af2-c78a-4f0b-b6a0-f7b99dcc9d8f
ok
test_create_user (Test.User.test_user) ...
DevTools listening on ws://127.0.0.1:56693/devtools/browser/097ac0b5-f6db-4009-a7c9-5f8e567ea67d
ok

-----
Ran 12 tests in 172.259s

OK
```

## 5 Summary

這次使用 selenium library 撰寫 test scripts，比起使用底層 python 其實我也想推薦使用 robot framework，因為底層自己寫的 python code 不比包好的框架來的實用及穩定，經常性或有 stale element 問題，但之前就有接觸過網頁自動化測試所以整體來說沒有遇到太大困難，我也在 test scripts 中的每次網頁操作插入 **assertion or wait component visible** 確保操作完整，不會有腳本執行太快產生的例外狀況，這次作業我更了解如何抓取 xpath 以及如何寫出高 reuse 的 function，提高使用率不必遇到新的 feature 就寫新的 function 產生過多相似 code，造成維護困難，lab4 整體來說讓我學到很多。