# Cài đặt:

pip install -r requirements.txt

# Cấu trúc:

File-project

|\_\_\_\_\_\_\_ input: chứa các file input

|\_\_\_\_\_\_\_ output: chứa các file output được trả về

|\_\_\_\_\_\_\_ project: chương trình

|\_\_\_\_\_\_\_\_\_ resources

| |\_\_\_\_\_\_\_\_ annotations.xlsx

| |\_\_\_\_\_\_\_\_ templateHTML.html

|

|\_\_\_\_\_\_\_\_\_ src

| |\_\_\_\_\_\_\_\_ myconstants.py

| |\_\_\_\_\_\_\_\_ … code

|

|

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ gen\_flows\_txt.py: (YC1) đọc UESCASES 🡪 FLOWS

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ gen\_testcases\_excel.py: (YC2) đọc FLOWS 🡪 TESTCASE

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ gen\_template\_html.py: (YC3) đọc FLOW 🡪 UI HTML

# Giới thiệu:

## annotation.xlsx

File định nghĩa các quy tắc regex, các annotation, html tương ứng

Gồm có các sheet:

* **Element**: để định nghĩa các đối tượng html

Trong đó:

* annotation: tên viết tắt của element (tên này được sử dụng trong file txt: usecases, flows)
* meaning: ý nghĩa/tên đầy đủ của element (tên này được sử dụng trong file excel: testcases)
* html-open: tag mở trong html
* html-close: tag đóng trong html

(Một số element có html chỉ gồm 1 tag 🡪 chỉ ghi vào html-open)

Graphical user interface, application

Description automatically generated

* **States:** gồm các annotation chỉ trạng thái vd: #empty, #invalid, ..

Dùng để nhận biết các trạng thái khi sinh testcase 🡪 expected output: vd: #empty,..

Table

Description automatically generated

* **Flows:** gồm các kiểu flow

Dùng để đọc file usecase: Khi đọc 1 dòng bắt đầu bằng những từ này 🡪 bắt đầu 1 flow

Table

Description automatically generated

* **Events:** chứa biểu thức chính quy (regex) của các event (hành động trong flow)

Biếu thức chính quy này để phát hiện khi nào 1 event xuất hiện (giúp đọc file usecase)

Trong này em đang để 2 kiểu regex, tuy nhiên chương trình sẽ chỉ sử dụng regex đầu tiên (ở dòng số 2) để nhận diện

Nếu muốn cô có thể thêm/thay thế dòng 2 bằng regex khác

Ví dụ:

^: bắt đầu dòng

\d: các ký tự số

Example: ví dụ về 1 số chuỗi match với regex

Table

Description automatically generated

* **Annotations:** tương tự như events

Bảng này chứa biểu thức chính quy của các annotation trong 1 event

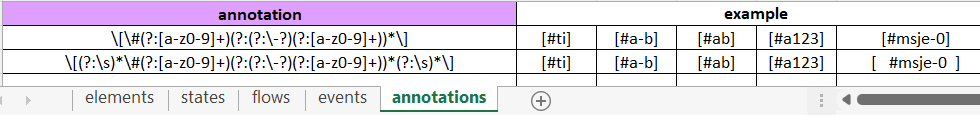
Chương trình sử dụng regex ở dòng đầu tiên để nhận diện các annotation

Ví dụ:

Nếu dùng kiểu 1, các annotation bắt buộc phải có dạng:

**[** + **#** + **chữ hoặc số** + **dấu gạch ngang ( - )** + **chữ hoặc số** + **]**

Nếu dùng kiểu 2, các annotation có thể chứa thêm dấu cách trước hoặc sau dấu ngoặc vuông



## templateHTML.html

file chứa khung html:

A screenshot of a computer

Description automatically generated

Khi thực hiện yêu cầu 3: reading the txt files to generate UI prototype in html files

File txt chứa flow 🡪 đọc các annotation 🡪 chuyển về html tương ứng

Đoạn html tương ứng của các annotation đó sẽ được chèn vào tag: **[#content]** ở file html trả về

Ví dụ:

Text

Description automatically generated

## myconstants.py

lưu các hằng số cố định giúp chương trình đọc các file trong resources:

annotations.xlsx, templateHTML.html

Ví dụ: tên sheet, tên các cột của file annotations.xlsx

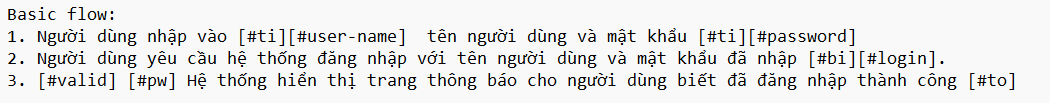
A screenshot of a computer

Description automatically generated with low confidence

## gen\_flows\_txt.py

đọc file usecase (txt) và trả về file flows (txt)

Input:



Output:



## gen\_testcases\_excel.py

đọc file flows (txt) trả về file testcases (excel)

Input:

Text

Description automatically generated

Output:

Graphical user interface, table

Description automatically generated with medium confidence

## gen\_template\_html.py

đọc file flow (txt) trả về UI template html

Input:

File gồm 1 flow (như hình dưới)

Trong trường hợp file có nhiều flow, sẽ chỉ lấy flow đầu tiên để sinh template

Graphical user interface, text, application

Description automatically generated

Output:

Graphical user interface, text, application, email

Description automatically generated

# Hướng dẫn sử dụng:

## Quy tắc (khi đọc usecase):

* Các flow có dạng: “Basic flow…”, “Alternative flow..”,..
* Các event có dạng: “1. “, “12. “
* Các annotation có dạng: [#string] *(string chỉ gồm chữ, số và có thể có dấu gạch nối - )*
* Các annotation được phân loại thành:

+ element: đối tượng html

+ state: các trạng thái (của output)

+ label: trong trường hợp không thuộc 2 nhóm trên

* Thứ tự cho 1 cụm annotation trong 1 event:

State + element + label

Ví dụ: **(ELEMENT + LABEL)**

[#text-output][#DayLaText][#button] 🡪 [#text-output][#DayLaText] , [#button]

1 nút có tên là Close sẽ phải viết thành:

[#button][#Close] 🡪 <button>Close</button>

Thay vì:

[#Close][#button] 🡪 <label>Close</label> <button></button>

## Các bước thực hiện yêu cầu 1:

Chuyển từ file usecase 🡪 file flows:

B1: Cho file usecase cần đọc vào folder:[**input**](#_Cấu_trúc:)

B2: Ở command line:

* Đi đến folder [**project**](#_Cấu_trúc:) bằng lệnh cd
* Gõ câu lệnh:

**py gen\_flows\_txt.py**

+ tên file usecase input (đã được đặt trong folder input) + tên file flows output



B3: sau đó, ở folder [**output**](#_Cấu_trúc:) sẽ xuất hiện file flows.txt

Text

Description automatically generated

Text

Description automatically generated

## Các bước thực hiện yêu cầu 2:

Tương tự như khi thực hiện yêu cầu 1

B1: Cho file flows cần đọc vào folder:[**input**](#_Cấu_trúc:)

**Graphical user interface

Description automatically generated**

B2: Ở command line, gõ lệnh:

**py gen\_testcases\_excel.py**

+ tên file usecase input (đã được đặt trong folder input) + tên file flows output

Text

Description automatically generated

Ví dụ:

file input: flows.txt

File output: testcases.xlsx

B3: Ở folder output sẽ xuất hiện:

Graphical user interface

Description automatically generated

Table, Excel

Description automatically generated

(chú ý: cần kéo giãn các cột thủ công)

## Các bước thực hiện yêu cầu 3:

Tương tự như khi thực hiện yêu cầu 1

B1: Cho file flow cần đọc vào folder:[**input**](#_Cấu_trúc:)

**Graphical user interface, application

Description automatically generated**

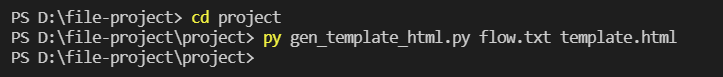
**Graphical user interface, text, application

Description automatically generated**

B2: Ở command line, gõ lệnh:

**py gen\_template\_html.py**

+ tên file usecase input (đã được đặt trong folder input) + tên file flows output



Ví dụ:

file input: flow.txt

File output: template.html

B3: Ở folder output sẽ xuất hiện: template.html

Text

Description automatically generated with medium confidence

Mở file bằng chrome:

Graphical user interface, text, application, email

Description automatically generated