

Chapter 5

System Implementation and Testing

5.1 Implementation and Testing

The implementation of the prototype will be discussed which aims to develop various features to assist in achieving the initial objective of the project. While, the testing for the prototype will be discussed which aims to self-evaluate whether the prototype hit the user requirement and features implemented successfully.

This chapter consists of the following sections :

- Coding Implementation
- Testing
- Software Methodologies
- Test Cases

5.2 Coding Implementation

The prototype is integrated with a few modules listed below:

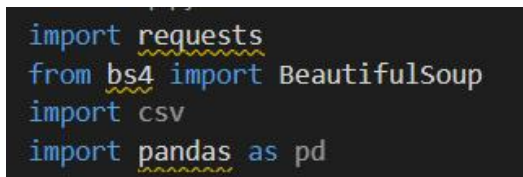
- Data Scraping Module
- Data Gathering Module
- Data Cleaning Module
- Sorting and Filtering Data Module
- User Interface Module

i. Data Scraping Module (DSM)

I was in charge of scraping academic universities staff information and government ministry information. The methods used for web scraping are BeautifulSoup and Selenium.

Part 1: BeautifulSoup Data Scraping (BSDS)

Figure 5.1 shows the installation of the required libraries.



```
import requests
from bs4 import BeautifulSoup
import csv
import pandas as pd
```

Figure 5.1: Library used in BSDS

After importing the libraries, the module will create a variable (URL) to input the link of the web pages that I would like to scrap with, then insert it into the requests library to fetch the page content. Then, it will create a parse Tree object(soup) with the assistance of BeautifulSoup and Python built-in “html” parser.

```
URL = "https://www.kbs.gov.my/pej-menteri-kbs"  
page = requests.get(URL)  
  
soup = BeautifulSoup(page.content, "html.parser")
```

Figure 5.2: Fetching html parser in BSDS

In the website HTML page, I have to find the container’s attribute with all the data needed to be scrapped. On this specific website, the data I needed is in the div container with the value of “jsn-mainbody”. After parsing the div container into the variable (results), I can find the elements I need by mentioning the placement of the data and the class name to get closer to the required data.

```
#find div id  
results = soup.find(id="jsn-mainbody")  
#print(results.prettify())  
  
job_elements = results.find_all("tr", {"class": ["fabrik_row oddRow0", "fabrik_row oddRow1"]})
```

Figure 5.3: HTML container in BSDS

After specifying the direction of the data needed to collect, I created an array variable (gov_list) to store all of the data I will collect soon. Then I create a loop to find all the data with the class name and store all into the each variable (such as name_element, jawatan_element, cawangan_element, tel_element, emel_element) . Before storing into an array (gov), I did some data cleaning by removing and stripping off some blanks spaces and left only words, also inserting the email format “@kbs.gov.my” into every emel_element. Lastly, I will append the gov variable into the gov_list variable we created earlier.

```
gov_list=[]

for job_element in job_elements:
    nama_element = job_element.find("td", class="direktori_staf_kbs__nama_fabrik_element_fabrik_list_1_group_1")
    jawatan_element = job_element.find("td", class="direktori_staf_kbs__jawatan_fabrik_element_fabrik_list_1_group_1")
    cawangan_element = job_element.find("td", class="hidden-phone_direktori_staf_kbs__cawangan_unit_fabrik_element_fabrik_list_1_group_1")
    tel_element = job_element.find("td", class="direktori_staf_kbs__no_telefon_fabrik_element_fabrik_list_1_group_1")
    emel_element = job_element.find("td", class="hidden-phone_direktori_staf_kbs__emel_fabrik_element_fabrik_list_1_group_1")
    nama_element.text.strip()
    jawatan_element.text.strip()
    cawangan_element.text.strip()
    tel_element.text.strip()
    emel_element.text.strip() + '@kbs.gov.my'
    gov = [nama_element, jawatan_element, cawangan_element, str(tel_element), emel_element]
    gov_list.append(gov)
```

Figure 5.4: Storing data scrapped in BSDS

Lastly, all the data in `gov_list` will be output into a csv file (`BStest.csv`) for better visualization and storing purposes.

```
f= open('BStest.csv', 'w')
csv_writer = csv.writer(f)
for i in gov_list:
    csv_writer.writerow(i)

f.close()
```

Figure 5.5: Data output to csv in BSDS

Part 2: Selenium Data Scrapping (SDS)

Figure 5.6 shows the required libraries to be installed.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import pandas as pd
import numpy as np
import csv
```

Figure 5.6: Library used in SDS

After importing the libraries, a master dataframe (`df`) is created before scrapping and `webdriver` will be launching the Chrome in headfull mode controlled by the Python programming while browsing the website inserted in the GET function.

```
#master DataFrame
df=pd.DataFrame(columns=['Name','Position','Department','Tel No.','Email'])

driver = webdriver.Chrome('C:\chromedriver')
driver.get('https://www.kbs.gov.my/pej-menteri-kbs')
```

Figure 5.7: Browsing website in webdriver in SDS

To extract the data from the website, I first inspect the html format in the website, find the elements of the data, and store it with the method of XPath.

```
names = driver.find_elements(By.XPATH, '//td[@class="direktori_staf_kbs__nama_fabrik_element_fabrik_list_1_group_1"]')
positions = driver.find_elements(By.XPATH, '//td[@class="direktori_staf_kbs__jawatan_fabrik_element_fabrik_list_1_group_1"]')
departments = driver.find_elements(By.XPATH, '//td[@class="hidden-phone_direktori_staf_kbs__cawangan_unit_fabrik_element_fabrik_list_1_group_1"]')
tels = driver.find_elements(By.XPATH, '//td[@class="direktori_staf_kbs__no_telefon_fabrik_element_fabrik_list_1_group_1"]')
emails = driver.find_elements(By.XPATH, '//td[@class="hidden-phone_direktori_staf_kbs__emel_fabrik_element_fabrik_list_1_group_1"]')
```

Figure 5.8: XPath method in SDS

Array variable (names_list, positions_list, departments_list, tels_list, emails_list) also created to store all the elements (such as names, positions, department, tels, emails) found. Then the data will be stored into a list to combine all the data.

```
names_list = []
for n in range(len(names)):
    names_list.append(names[n].text)

positions_list = []
for n in range(len(positions)):
    positions_list.append(positions[n].text)

departments_list = []
for n in range(len(departments)):
    departments_list.append(departments[n].text)

tels_list = []
for n in range(len(tels)):
    tels_list.append(tels[n].text)

emails_list = []
for n in range(len(emails)):
    emails_list.append(emails[n].text + '@kbs.gov.my')

#list of each gov names, position, department, contact, email
data_tuples = list(zip(names_list[1:], positions_list[1:], departments_list[1:], tels_list[1:], emails_list[1:]))
```

Figure 5.9: Storing data scrapped in list in SDS

The list will then be converted into a dataframe for easy storing into csv then output into a csv file (Seleniumtest.csv).

```
#create Dataframe of each tuple
temp_df = pd.DataFrame(data_tuples, columns = ['Name','Position','Department','Tel No.','Email'])

#appends to master dataframe
df = df.append(temp_df)
arr = df.to_numpy()
print (arr)

driver.close()

f = open('Seleniumtest.csv', 'w')
csv_writer = csv.writer(f)
for i in arr:
    csv_writer.writerow(i)

f.close()
```

Figure 5.10: Data output to csv in SDS

ii. Data Gathering Module (DGM)

I was in charge in gathering the categorized data from industry projects into one combined data table.

Firstly, I imported the pandas library to read every csv file into the Jupyter Notebook.

```
import pandas as pd
agridata = pd.read_csv('Agriculture.csv')
infodata = pd.read_csv('Info&Tech.csv')
legaldata = pd.read_csv('Legal.csv')
manudata = pd.read_csv('Manufacturing.csv')
marketdata = pd.read_csv('Marketing.csv')
mathdata = pd.read_csv('Math_Sciences.csv')
policydata = pd.read_csv('Policy & Socioeconomic.csv')
professdata = pd.read_csv('Professional Services.csv')
tourismdata = pd.read_csv('Tourism & Hospitality.csv')
```

Figure 5.11: Read csv in DGM

Since the header for some data columns are not understandable, I have to edit the column name into a more understanding name for the convenience of data analysis in the future. Each file had been edited into the same column order and names for the next procedure which is to append and combine all the file into one easily.

```
agridata['Category'] = 'Agriculture'
del agridata['pic']
agridata.rename(columns={'description MyFinB Role': 'MyFinB'}, inplace=True)
agridata.rename(columns={'Full Name for Topic': 'Full Topic'}, inplace=True)
agridata.rename(columns={'Tech': 'Science & Technology Driver'}, inplace=True)
agridata.rename(columns={'Driver': 'Socio-Economic Driver'}, inplace=True)
agridata.head()
```

Figure 5.12: Change column name in DGM

After all the file names are edited, a frame is created to insert all the data into one variable and manipulate the axis and position of the list into dataframe by using concat function.

```
frames = [agridata, infodata, legaldata, manudata, marketdata, mathdata, policydata, professdata, tourismdata]

fulldf = pd.concat(frames)
fulldf.head(30)
```

Figure 5.13: Combine data in DGM

Lastly, the dataframe will be stored into a csv file (FYPPProject.csv) and import to google drive as cloud storage.

```
fulldf.to_csv('FYPPProject.csv', index = False)

from google.colab import drive
drive.mount('/drive')
fulldf.to_csv('/drive/My Drive/FYPPProject.csv', index = False)
```

Figure 5.14: Store csv into drive in DGM

iii. *Data Cleaning Module (DCM)*

I start to clean the data for better analysis results.

I import the pandas library to read the csv file from google drive.

```
import pandas as pd
df = pd.read_csv('FYPPProject.csv')
```

Figure 5.15: Read data in DCM

I started to clean the data by removing the pending project that does not yet exist or progress in the current. Then resetting the index of the data for better looking. I also then remove the useless symbol in the data itself to be convenient when categorizing the data during the analysis phase.

```
# Filter all rows where project still pending
df.drop(df[df['MyFinB'] == 'Pending'].index, inplace = True)

fypdata = df.reset_index(drop=True)
```

Figure 5.16: Remove pending project in DCM

```
for r in ((" ", ""), ("\n", " ", "")):
    fyp1['Socio-Economic Driver'] = fyp1['Socio-Economic Driver'].str.replace(*r)
    fyp1['Science & Technology Driver'] = fyp1['Science & Technology Driver'].str.replace(*r)
```

Figure 5.17: Remove useless symbol in data in DCM

Lastly, I store the data into a csv file (CleanFYP.csv) into google drive.


```
fypdata.to_csv('CleanFYP.csv', index = False)

from google.colab import drive
drive.mount('/drive', force_remount=True)
fypdata.to_csv('/drive/My Drive/CleanFYP.csv', index = False)
```

Figure 5.18: Store data in drive in DCM

iv. *Sort and Filter Data Module (SFDM)*

I also take charge of the Industry Project part for simulating and creating the sorting and function feature.

Pandas library is imported to allow python to read the csv file (CleanFYP.csv)

```
import pandas as pd
df = pd.read_csv('CleanFYP.csv')
```

Figure 5.19: Read csv file in SFDM

Filtering function is created to allow users to input the specific keyword they would like to filter and which column of the data they want to use the filter.

```
#Search specific words
word = input("Enter keyword: ")

Enter keyword: Bioscience

abc = df[df['Science & Technology Driver'].str.contains(word)]
```

Figure 5.20: Filter function in SFDM

While the sort function is created in a way that it will receive the input from the user which column they would like to sort and decide the order they would like to sequence, either ascending or descending.

```
#Sort by words
sortword = input("Enter column to sort: ")
rowseq = input("Ascending or Descending? :(A/D)")

Enter keyword to sort: Topic
Ascending or Descending? :(A/D)a

if (rowseq == 'A' or rowseq == 'a'):
    seq = True
else:
    seq = False

rslt_df = df.sort_values(by = sortword, ascending = seq)
```

Figure 5.21: Sort function in SFDM

v. *User Interface Module (UIM)*

I am in charge for creating the user interface for the university datasets.

The website begins with a simple header with the title of Academic Search and continues with a search function for the universities table.

```
<h1 style="text-align: center; color: black;">Academic Search</h1>

<br>

<p>Key in your input to filter the table:</p>
<input type="text" id="myinput" placeholder="Search..." title="Type in something">
```

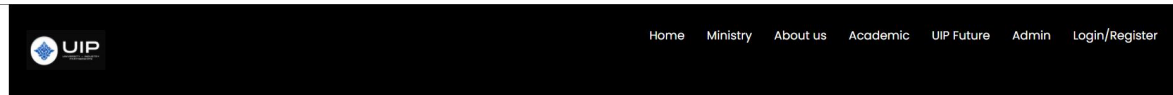
Figure 5.22: Coding of design in UIM

Variable (tableData) are function to simulate the database and created in the javascript which it is also connected with the html.

```
var tableData = [
  (university: 'University Teknologi Malaysia', department: 'Fakulti Alam Bina & Ukur', position: 'Pensyarah Kanan', name: 'Dr. Abdul Rahim Bin Abdul Hamid', email: 'abdul.rahim@utm.my'),
  (university: 'University Teknologi Malaysia', department: 'Fakulti alam bina & ukur', position: 'PENSYARAH KANAN (DS52)', name: 'Sr Dr. Abd Wahid Bin Rasib', email: 'abdwahid@utm.my'),
  (university: 'University Teknologi Malaysia', department: 'Fakulti alam bina & ukur', position: 'PENSYARAH KANAN (DS52)', name: 'Gs. Dr. Abd. Halim Bin Hamzah', email: 'halimhamzah@utm.my'),
  (university: 'University Teknologi Malaysia', department: 'Fakulti alam bina & ukur', position: 'PENSYARAH KANAN (DS51)', name: 'Dr. Abdul Rahim Bin Abdul Hamid', email: 'abdul.rahim@utm.my'),
  (university: 'University Teknologi Malaysia', department: 'Fakulti alam bina & ukur', position: 'PROFESOR MADYA (DS54)', name: 'Prof. Madya Sr. Abdul Wahid Bin Kamarulzaman', email: 'ab_wahid@utm.my'),
  (university: 'University Teknologi Malaysia', department: 'Fakulti alam bina & ukur', position: 'PENSYARAH KANAN (DS52)', name: 'Sr Dr. Abdullah Hisham Bin Omar', email: 'abdullahisham@utm.my'),
]
```

Figure 5.23: Table data backend in UIM

In the user interface itself, the website's appearance to the users is as shown below.



Academic Search

Key in your input to filter the table:

Click on the header of a column to sort the table:

University	Department	Staff Position	Staff Name	Email
University Teknologi Malaysia	Fakulti Alam Bina & Ukur	Pensyarah Kanan	Dr. Abdul Rahim Bin Abdul Hamid	abdul.rahim@utm.my
University Teknologi Malaysia	Fakulti alam bina & ukur	PENSYARAH KANAN (DS52)	Sr Dr. Abd Wahid Bin Rasib	abdwahid@utm.my

Figure 5.24: Frontend design in UIM

When users search the specific keyword, it will activate the filter function, and the particular data will be highlighted to alert the user.

Key in your input to filter the table:

Click on the header of a column to sort the table:

University	Department	Staff Position	Staff Name	Email
University Teknologi Malaysia	Sekolah kejuruteraan elektrik	PROFESOR MADYA (DS54)	Prof. Madya Dr. Choong Weng Wai	cwengwai@utm.my
International Islamic University Malaysia	COUNSELLING & CAREER SERVICES CENTRE	Psychology Officer	Nordinah Binti Mohd Kassim	nordinah@iiium.edu.my

Figure 5.25: Highlight when filter in UIM

However, the user will need to click the table column name to use the sort function in ascending or descending order.

Click on the header of a column to sort the table:

University	Department	Staff Position	Staff Name	Email
International Islamic University Malaysia	DEVELOPMENT DIVISION	Horticulturist	Aries Iskandar Muhammed	ariesaa@iiium.edu.my
Mara University of Technology	ART & DESIGN-Creative Game Design	Lecturer	Fatimah Zahra Ros Azman	fatimahzahra@uitm.edu.my

Figure 5.26: Ascending order when sort in UIM

Click on the header of a column to sort the table:

University	Department	Staff Position	Staff Name	Email
Mara University of Technology	ART & DESIGN-Industrial Ceramic	Senior Lecturer	Zuraidy Abd Rahim	zuraidy@uitm.edu.my
International Islamic University Malaysia	CENTRE FOR POSTGRADUATE STUDIES	Assistant Administrative Officer	Zazura Bt Zainal Abidin	zazura@iiium.edu.my

Figure 5.27: Descending order when sort in UIM

5.3 Software Testing Methodologies

Software Testing is the procedure of evaluating and verifying the particular software product or application does what it is supposed and expected to do and satisfies user requirements (IBM, n.d.). This process is very important and required during every system development as it could prevent bugs issues, reduce excessive development costs, and improve performance in the system before it is published to the public. In system testing, it is categorized as Functional Testing and Non-functional Testing.

Functional Testing involves testing the system application against the business requirements and guaranteeing each part of the software will behave as expected by using several types of test cases (Aebersold, n.d.). For instance, Unit testing, Integration Testing, and System Testing are used to test the functions and incorporate every module and function in the system.

However, Non-functional Testing is a method that incorporates every functional testing but mainly focuses on the operational aspects of a piece of software (Aebersold, n.d.). For exemplification, Performance testing to test the software performance, Security testing aimed to secure the system and data information in it, Usability testing to measure the user-friendliness, and Compatibility testing to gauge the application working in each different environment.

I. Test Cases for Data Scraping Module

Test Case ID: TC_01	Test Designed by: Kong Mun Jun
Test Priority (Low/Medium/High): Low	Test Designed date: 20-10-2021
Module Name: Data Scraping	Test Executed by:
Test Title: Verify the data scraping process	Test Execution date:
Description: To test whether the data crawled are correct in each columns	

Pre-conditions:**Dependencies:**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Enter python file					
2	Upload the excel file	gov.csv	Uploaded and not missing any data			

II. Test Cases for Data Grouping Module

Test Case ID: TC_02	Test Designed by: Kong Mun Jun
Test Priority (Low/Medium/High): Med	Test Designed date: 20-10-2021
Module Name: Data Grouping	Test Executed by:
Test Title: Verify the data grouping process	Test Execution date:
Description: To test whether all the data are combined	

Pre-conditions: connected to Google Drive

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Standardize the columns in every file	“Tech” to “Technology”	Combined data in correct column			
2	Upload the excel file	GroupFYP.csv	Uploaded and not missing any data			

III. Test Cases for Data Cleaning Module

Test Case ID: TC_03

Test Designed by: Kong Mun Jun

Test Priority (Low/Medium/High): Med

Test Designed date: 20-10-2021

Module Name: Data Cleaning

Test Executed by:

Test Title: Verify the data cleaning and process
Test Execution date:

Description: To test whether the data cleaned and processed are accurate

Pre-conditions: connected to Google Drive

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Remove meaningless data	Project- “Pending”	Data removed	Data successfully removed		
2	Remove useless symbol	“-Water & Food\n”	“\n” replace to “ , ”	Symbol successfully removed		
3	Upload the excel file	CleanFYP.csv	Data uploaded and cleaned			

IV. Test Cases for Sort&Filter Function Module

Test Case ID: TC_04	Test Designed by: Kong Mun Jun
Test Priority (Low/Medium/High): Med	Test Designed date: 20-10-2021
Module Name: Data Sort&Filter	Test Executed by:
Test Title: Verify the sorting and filter function	Test Execution date:
Description: To test whether the data able to sort or filter based on conditions	

Pre-conditions: connected to Google Drive

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Input specific columns	MyFinB	Column exist	Column exist		
2	Input ascending or descending	a	Column in ascending order	Column in correct order		
		d	Column in descending order	Column in correct order		
3	Input specific keyword	Bioscience	Rows with "Bioscience" keyword	Row successfully appeared		

V. Test Cases for UI Website Module

Test Case ID: TC_05	Test Designed by: Kong Mun Jun
Test Priority (Low/Medium/High): Med	Test Designed date: 20-10-2021
Module Name: UI test cases	Test Executed by:
Test Title: Verify the UI feature	Test Execution date:
Description: To test whether the UI can provide the correct feature to the users.	

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click into the module	Click from upper tab	Able to direct to Module	Module appeared successfully		
2	Input value to search box	Mara	Rows with "Mara" appear	Rows successfully executed		
3	Filter data in table	Clickable header column	Data reorder in alphabetic order	Data successfully reorder		

5.4 Chapter Summary and Evaluation

Software testing is a vital process and plays an important role in sustaining the quality of the system before it is published to a third party. It is a part of the development cycle of a system project to identify every defects and issues in the system to achieve the user requirement expected by third parties such as clients, stakeholders, developers, and testers.

In a nutshell, the testing will be assigned to assure the prototype has minimum deficiency and achieve further improvisation.