Testing Plan

System to Synchronize External Schedules with Shift Management Platform

Team 7

1. Introduction

1.1. *Project Overview*

The use of external schedules with the Microsoft Shifts platform shows the unsystematic quality of management. The purpose of this project is to make scheduling with Shift management more reliable and more efficient by addressing the mismatch information such as time and location and providing a lookup table of the latest schedules. This is achieved by developing a system to perform data extraction from the course planner website and check the consistency of the Shifts in Microsoft Teams.

1.2. *Scope of the test plan document*

This document covers testing strategy, test execution, testing plan schedule, resource and environment, and risks. Test strategy emphasises the goal of the testing plan and the result of the test. Test execution describes the testing methods for each feature that will be performed in milestone 2. Resources and environment provide the testing tools. Risks include every risk that has been identified.

2. Test Strategy

2.1. *Test scope*

The goal of the testing plan is to verify that the software should conform to its specification (*Appendix A*). Defect testing has been taken during the whole software development. The features of the system are divided into two layers, functionality and performance. Each functionality feature has been tested for unit testing. Integration testing will be testing the whole system. We test for problems that arise from component interactions. The Top-down/Bottom-up/Sandwich strategy is for our integration testing. We perform black-box testing for release testing to test a release of the system that will be delivered to our client. Performance testing and stress testing are not considered in this plan. Acceptance testing will be taken by user testing.

2.2. *Requirements*

- Fetching function to get the Shits details in Microsoft Teams
- Deleting function to erase unmatched schedule to manage the Shifts
- Searching function for user to search specific course details
- Matching function to distinguish between Shift platform and course planner website

2.3. *Testing Report*

See *Appendix B*

2.4. *Test Assumptions*

- The group name in Microsoft Teams Shifts is identical to the course planner
- The course name to search is identical to the course planner
- Any changes made in Microsoft Shifts requires restarting the software to display updated shifts

3. Test Execution

3.1. *Defect testing*

Defect tests have been taken during the whole software development. For example, one defect was the searching function. The test case was to search the course name - Computer Networks & Applications. This explained that the searching function could only handle the course name that did not contain any special characters (‘&’, ‘(‘, ‘)’), and this has been solved by adjusting the web scraper.

3.2. *Unit testing*

Unit testing includes testing the following functions:

- get_teams(token):
  Write a few Python code to run this method. The expected output is a JSON string including the information of teams that the user joined. Manually get user's access token from Microsoft Graph Explorer before testing. The expected JSON string can also be found on Graph Explorer.
  Feature tested: Fetching data from Microsoft Teams.
- get_group_name(token, teamId, groupID):
  Write a few Python code to invoke this function to see if it can return a JSON string of a specific scheduling group. The inputs are access token, group ID, and team ID, which contain the shift queried. We manually input group ID and team ID before testing.
  Feature tested: Fetching data from Microsoft Teams.
- delete(shift, teamID, token):
  Write a few Python code to call this function. Then check the input shift in Microsoft Teams to see if it’s deleted. The inputs are access token, shift JSON, and team ID, which contain the shift queried. We will use Graph Explorer to get the input shift and team ID.
  Feature tested: Managing shifts.
- match(shift):
  Create a list of shifts and manually read course information from the course planner and store it in a Python dictionary. Run this matching method and see the output file. It is expected that in a txt file, all correct shifts are marked “matched”.
  Feature tested: Matching course details.
- search_function(request):
  Generate some JSON shifts and input a course name and run this method. If it can display all the shifts that belong to this course, this method is correct.
  Feature tested: Searching specific course.

3.3. *Integration and release testing*

Integration and release testing are two phases of system testing. Integration testing is to build a system to test for problems that arise from component interactions. For our project, considering client priorities, we will use Bottom-up in this stage.

1. we test receiving all of Microsoft Teams' data and parsing its contents. This integrated get_teams(token), get_group_name(token, teamId, groupID) and data transfer between front-end and back-end. The expected output should be a table displaying all shifts with time, place, and group name. If this test passes, our software reads the data in Microsoft Team correctly
2. Second, we will test the functionality of the user's search. This integrated match(shift), search_function(request) and reading course information on course planner. The testing method is to input a course name, and it shows us whether this course has a matched shift in Microsoft Team. If we pass this test, our software successfully compares the CoursePlanner with Microsoft data.

And for release testing, the goal is to increase our confidence that our system meets the client’s requirements. Using test cases that will reveal defects in the system to test. We will give the prototype of our software to the client. He will use it and check whether it meets his requirements.

3.4. *User acceptance testing*

User acceptance testing will be performed with the client, which is a mock real-world context with real users to get input and advice from users. Our client will test all the functionality and the whole system to decide whether the requirements have been met.

4. Resources and Environment

4.1. Tools

Tool	Version
Microsoft Teams	Version 1.4.00.11161 or later
Terminal	
Microsoft Graph Explorer	
Python	Version 3.9 or later
Django	Version 3.2.3 or later
Chrome	Version 90.0.4430.212 or later
Safari	Version 14.0.3 or later

4.2. Roles

Role	Member(s)	Responsibilities
Client	Ian Knight	1. Giving the requirements for the development of the project. 2. Providing guidance for the project and key information 3. Testing the system by following the user instruction.
Tester	Chun Yee Herman Lai, Fangyu Yuan, Haoyu Wang, Zhendong Shi	1. Understand requirements 2. Coordinate with the client if any issues 3. Write code for the requirement 4. Execute test cases 5. Prepare for the test data 6. Debug

5. Testing Plan Schedule

All the unit tests have been done before 20/05/2021, and we planned to finish the integration and release testing and user acceptance before 27/05/2021. See *Appendix C.*

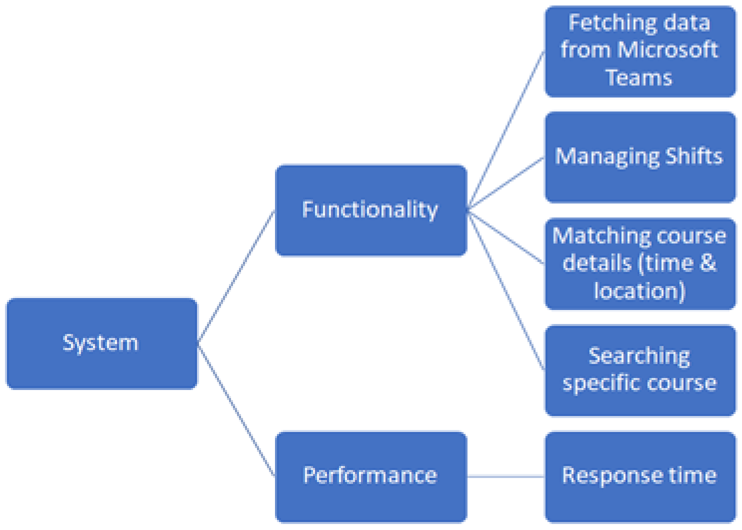
6. Risks

Bug	Severity	Plan	Responsible	Due date
Microsoft Teams does not exist	Critical	Must fix immediately	Haoyu Wang	15 May
No schedule was created	Critical	Must fix immediately	Chun Yee Herman Lai	15 May
The matching module cannot import the web-scraper module	Critical	Must fix immediately	Fangyu Yuan	20 May
Searching function lack performance	Medium	Fix when we have time	Zhendong Shi	25 May

7. Appendixes

Appendix A

Features of the System


System
Functionality
Performance
Fetching data from Microsoft Teams
Managing Shifts
Matching course details (time & location)
Searching specific course
Response time

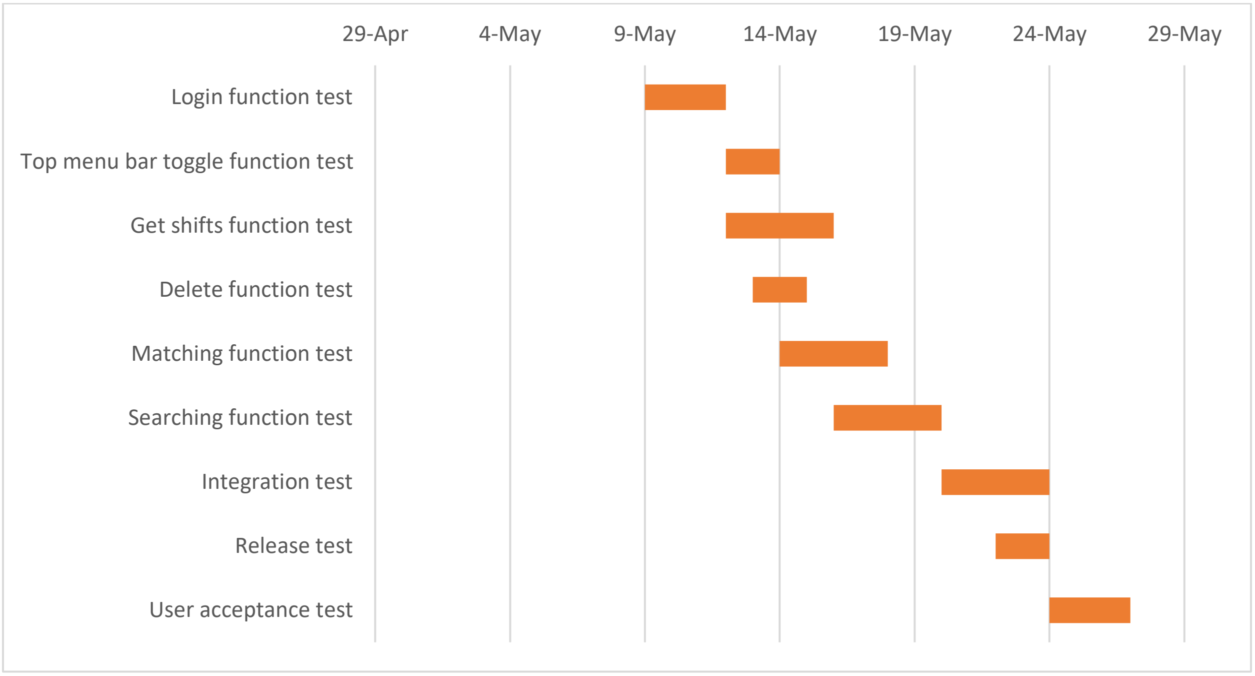
Appendix B

Testing Report

Test name	Test description	Input	Expected output	Current output	Responsible	Reviewer
Login function testing	Users type in their email accounts and passwords to test whether they can log in to the system successfully.	User email account and password.	If the user logs in successfully, the system returns "login successfully". If the user can not log in to the system, the system returns "login failed".	Same as expected	Zhendong Shi	Fangyu Yuan
Top menu bar toggle function testing	After logging in to the system, click the top menu bar icons to test whether corresponding information is displayed on the web page.	Mouse click event	The web page shows correct information if the user clicks the corresponding menu icon.	Same as expected	Zhendong Shi	Fangyu Yuan
Get shifts function testing	Click the menu icon "Shift" to test whether the web page can show all the shifts information from Microsoft Teams Shifts.	Mouse click event	The web page should show all the shifts information from Microsoft Teams Shifts or return "no shifts found".	Same as expected	Chun Yee Herman Lai	Haoyu Wang
Matching function testing	Add course information in Microsoft Teams Shifts to test whether the system can compare the course	Course information	If course information does not match with each other, the system should highlight the different	Same as expected	Fangyu Yuan	Chun Yee Herman Lai

	information from the course planner and Microsoft Teams Shifts automatically.		information. Otherwise, it returns “course information matches”.			
Search function testing	Choose semester, course type, and then type in the course name to test if the search function can show the correct information.	Semester, course type, and course name	The search function should filter the course information from both course planner and Microsoft Teams Shifts correctly or return “no results”.	Same as expected	Haoyu Wang	Chun Yee Herman Lai

Appendix C
Testing Plan Schedule


29-Apr
4-May
9-May
14-May
19-May
24-May
29-May
Login function test
Top menu bar toggle function test
Get shifts function test
Delete function test
Matching function test
Searching function test
Integration test
Release test
User acceptance test