MCI project Testing plan

Team number: 36
Project Title: A Node.JS web application for synchronizing student grades from external sources to MyUni.

1. Introduction

1.1. Project Overview

The traditional way for the lecturers to release students' grades from Web Submission System to MyUni (The Canvas System) is complicated and unfriendly. If the lecturer wants to modify the grades for students, he needs to create a CSV file, make the modifications with the format and upload the changes to the Gradebook. If the lecturer doesn't know how to create a CSV file, he needs to download it from Canvas and edit with the software program such as Excel and Numbers. The students' grade will display after lecturer reupload the same CSV file. The goal of this project is to deliver a web-based application with Node.js that would enable the synchronization of grades from external source to MyUni. The lecturer will simply release the students' grades with one single web page.

1.2. Scope of the test plan document

The document explains the testing plan of our web application for synchronizing student grades. The plan does not include the testing result and debugging solution, as these sections are outside the scope of this plan. This plan focuses on the test strategy which describes the range of testing, the record form of testing results and test assumptions and the test execution that includes the defect testing, unit testing and integration testing. This document will be used in the development of our project.

2. Test strategy

2.1. Test scope: what will be tested and why

To guarantee that our program conforms to its specification and meet user needs, the tests should follow our core functional components shown in the table below. There are three significant components in our project：web performance (User interface display), application structure and functions (in Node.js). Defect test had finished during the first milestone. Unit tests are separated into the project progress. They will be tested when the subtest finished. After the completion of all functions, the system will be tested, and the performance test will be conducted after the rear end functions finished.

The core functional components	Testing contents
Web performance	• Buttons • Response time • Expression accuracy
Application structure	• Data transform • White-box testing • Path Testing
Functions	• Errors • Return data • Asynchronous

2.2. Testing report

The testing report is shown below which contains the tests passed in the project and the future testing report structure will be similar. The example only shows the unit test of our project which passed.

Test description	Input	Expected output	Current output
Transfer data from web submission	CSV from web submission	Json file contain 'Student id' and 'mark'	get the correct Json file
Test type: Get Date: 28/03/2020		Note: Passed	

Test description	Input	Expected output	Current output
Get Json data from MyUni	Json data from MyUni	Json file contain 'id' and 'user_id'	get the correct data
Test type: Get Date: 10/04/2020		Note: Passed	

Test description	Input	Expected output	Current output
Post data to MyUni	'user_id' and 'mark'	changes appear in MyUni grade	get the correct result
Test type: Post Date: 01/05/2020		Note: Passed	

2.3. Test assumptions

All the sources in web submission are correct.
The JSON data in MyUni path is correct.

3. Test execution

3.1. Defect testing

Defect tests have been used throughout the project development process. Once an error occurs, we will conduct appropriate tests to solve the problem. For instance, defect tests were used when we implemented getjson_myuni(). The cause of the problem is that the 'curl-request' template cannot be applied to newer versions, which has affected the progress of our team 's work. We have tried to download the template again, change the Node.JS version and other methods, but none of them can solve the problem. In the case that the above method did not work, we had to change the template and finally passed the defect test.

3.2. Unit testing

Unit tests were done on following parts:

- function getSubmission()

Write some invalid and valid information to the input files named web.csv. Manually write down the expected parsed data before running the function.
Feature tested: Correctly parse and extract valid data

- function getjson_myuni(course_id, assignmnet_id,token)

Find a course as a target and record its course id and assignment id. Use the two id and a token that is always and effectively to run the function. Manually write an expected output based on the information on MyUni. Compare the difference between the expected output and the actual output files after running the function.
Feature tested: Get the personal information of students under any target course

- function postGrade(student_id, mark, token)

Enter the student id and grade of the target student then run the function(We will always use a valid token). Go to the Gradebook page of MyUni to check whether the mark changes as expected.
Feature tested: Can effectively change the scores of any student under the course

3.3. Integration and release testing (what will be tested and how)

Integration tests were done to test the following features:

- Match enrolled students

There may be some invalid student grades in the files obtained from Web Submission. For example, those students didn't enrol in the course before. For marks of the students who didn't register for this course, we will ignore the data submitted by them. We will check the student id of the student who got a mark from Web Submission and will only respond to the request of the official student to change the grade in MyUni.

- Check upload grades

Follow the principle of retaining only the best grade in multiple submissions. When a grade needs to be uploaded to MyUni, it should be better than the one that has been recorded. Otherwise we will ignore this upload request and keep the current version and grade.

- Compatibility

The software will be run on Windows and Linux to test its compatibility.

4. Testing plan schedule

Test description	Date	Reviewer
Testing web submission data transfer	28/03/2020	Junzhe Huang
Testing MyUni get function	10/04/2020	Qianyang Tang
Testing MyUni Post function	01/05/2020	Junzhe Huang
Testing 'user_id' and 'mark' whether match or not.	22/05/2020	Qianyang Tang
Testing web submission get function	26/05/2020	Sicheng Xin
Testing the monitor buttons	29/05/2020	Qianyang Tang
Testing the whole application	01/05/2020	Sicheng Xin

5. Risks

For delivering the project as the requirement and with the full functionality, some project risks also are considered in the testing plan. These potential risks may be identified as testing progresses. Here is the rating of risks and coping strategies.

Probability and Impact Matrix

Probability	Threats					Opportunities				
0.90	0.05	0.09	0.18	0.36	0.72	0.72	0.36	0.18	0.09	0.05
0.70	0.04	0.07	0.14	0.28	0.56	0.56	0.28	0.14	0.07	0.04
0.50	0.03	0.05	0.10	0.20	0.40	0.40	0.20	0.10	0.05	0.03
0.30	0.02	0.03	0.06	0.12	0.24	0.24	0.12	0.06	0.03	0.02
0.10	0.01	0.01	0.02	0.04	0.08	0.08	0.04	0.02	0.01	0.01
	0.05/ Very Low	0.10/ Low	0.20/ Moderate	0.40/ High	0.80/ Very High	0.80/ Very High	0.40/ High	0.20/ Moderate	0.10/ Low	0.05/ Very Low

Impact (numerical scale) on an objective (e.g., cost, time, scope or quality)

Each risk is rated on its probability of occurring and impact on an objective if it does occur. The organization's thresholds for low, moderate or high risks are shown in the matrix and determine whether the risk is scored as high, moderate or low for that objective.

Figure 1 *Probability and impact matrix (Source: PMI 2013)*

Figure1 risk scoring matrix used to digitize the probability of risk occurrence of a project and our project used it to identify the risks, and defined the probability as 0.1.

Id	Description	Likelihood	Rating	Response	trigger	Mitigation
001	Don't get the necessary parameters,the program does not run normal.	Unlikely	very low(0.03)	Avoid	The order of pressing buttons is messed up. Click to "confirm" before filling in the necessary date	Pay more attention to the order of operation. If there is unfilled information,when someone clicks "confirm",the system can remind the operator.

002	Some errors happen due to typing mistakes.	Unlikely	very low(0.02)	Avoid	Because there are methods or variables with similar prefixes,when we use the automatic completion spelling system, it matches a name which is not what we want.	More careful typing, especially using the automatic completion spelling system. Double check the spelling and test the program when we finish every function.
003	Some errors don't throw out.	Unlikely	very low(0.01)	Avoid	Some errors happened and we don't concern it and do not throw out.	throw out all possible errors.
004	APIs don't work due to version changes	Unlikely	very low(0.01)	Accept	Some API developer might change their code and it may not suit for our project	We will use another API with similar features.