ICES4HU
Software Test Report

1. Introduction

1.1 Purpose and Scope

The purpose of this Software Test Report is to document the methods and results of the tests we performed on the backend system of our academic evaluation project. This report is important for the successful development and functioning of our application.

Our team has dedicated considerable effort to the development of the backend for our project, prioritizing various features. To facilitate the testing of our APIs, we have utilized the Postman platform extensively. This report aims to provide a comprehensive overview of the functionalities including user login and logout, account management, course enrollment and management, survey creation, editing, and management, as well as instructor and student evaluations we have successfully implemented thus. It gives us a clear understanding of the current state of our project and what we need to do in the future.

2. Descriptions of Test Activities

2.1 Document Security

The security and privacy of our Software Test Report (STR) are paramount due to the sensitive and potentially confidential nature of the data contained within the document. We adhere to strict protocols to ensure that the STR document is adequately protected. Access to the document is restricted and is granted only to authorized team members, project stakeholders, and relevant management personnel.

Security measures in place include robust authentication systems, data encryption, and usage of secure network environments. Also, any prints of the document are kept secure, and disposal of these copies is conducted responsibly. Furthermore, all team members are trained on the significance of confidentiality and have signed non-disclosure agreements, ensuring the non-publication of sensitive data.

2.2 Data Recording, Reduction and Analysis

Data recording, reduction, and analysis procedures play a significant role in our testing activities. We use a combination of manual, automatic, and semi-automatic techniques for capturing, managing, and examining test results.

In our API testing process, we rely on Postman and Thunder Client in VSCode as our primary tools for recording and tracking test results. Postman automatically captures the responses generated by our API calls, encompassing crucial information such as HTTP response codes, headers, and the content of the response body. These recorded results are subsequently subjected to manual review, where we compare them against the expected outputs outlined in our Test Case Definitions document. This meticulous review process ensures the accuracy and alignment of our API functionalities with the defined test cases.

As part of the data reduction process, we filter out any unnecessary information and focus on the results that are most relevant to our test objectives. We categorize the test results based on their associated test cases and isolate any results that indicate errors or unexpected behavior.

For data analysis, we use spreadsheet software to organize and interpret the test results. Each test case is assigned a separate sheet where results are tabulated and analyzed for patterns, anomalies, or potential problems. The team members involved in testing evaluate the results and provide their observations and recommendations.

In addition to Postman, Thunder Client and spreadsheets, we use Jira for defect tracking and project management. Whenever a test case fails, a bug is logged in Jira with all the necessary details such as the test case ID, description of the error, steps to reproduce the issue, and the severity of the bug.

By employing these strategies, we can systematically track and analyze our test results, fostering more efficient communication and collaboration across our team and ultimately leading to the development of higher-quality software.

3. Test Preparation

3.1 Hardware Preparation

We made sure our computers met the hardware requirements to run our project by providing sufficient processing power, RAM and hard disk space. It is also verified for stable internet connection, smooth communication and access to online resources. We did not make any extra preparations other than such simple things. Our own hardware systems were sufficient for us to run these tests.

3.2 Software Preparation

Software preparation is equally critical to facilitate a seamless test execution process. Our application is containerized using Docker, allowing us to manage and deploy software reliably across different environments.

The application under test, along with the necessary data, is stored in AWS S3 buckets. Clear instructions for loading and initializing the software are documented in our operations manual. Steps involve pulling the latest Docker image from our DockerHub repository, initiating the Docker containers, and validating successful deployment by checking the application's response.

Common software initializations that apply to multiple test scripts include setting up the necessary environment variables, initiating the application under test, and confirming the database connectivity and availability. Detailed information is available in the software setup section of the ITP.

3.3 Other Pre-test Preparations

Beyond hardware and software setup, we undertake some other pre-test activities to ensure a smooth testing process. These include briefing all test personnel on their roles and responsibilities, ensuring all test case definitions and scripts are updated, and checking that necessary tools and resources are available.

4. Test Results

Use Case Number	**Use Case Name**	**Test Case Description**	**File Name**	**Retest?**	**Number of Activity Paths Tested**	**Number of Non-working Activity Paths**	**Time Spent**
1	Login	Verify that a user can successfully login for an account	Login.js	No	3	0	2 hours
2	Logout	Verify that a user can successfully logout from an account	Logout.js	No	1	0	1 hour
3	Manage Account	Test user's ability to manage their own account details	ManageAccount.js	Yes	5	1	3 hours

Use Case Number	Use Case Name	Test Case Description	File Name	Retest?	Number of Activity Paths Tested	Number of Non-working Activity Paths	Time Spent
4	Account Recovery	Testing the functionality of account password recovery process	AccountRecovery.js	No	4	0	2 hours
5	Merge Student Emails	Test Name: Merge student emails	MergeStudentEmails.js	No	4	0	2 hours
6	Prepare Evaluation Forms	Testing the ability of the admin to delete inappropriate questions from surveys	PrepareEvaluationForms.js	No	4	0	2 hours
7	Delete Accounts	Admin may remove dismissed students or resigned	DeleteAccounts.js	No	3	0	2 hours
8	Start Semester	Verify that the admin can successfully start a new semester in the system	StartSemester.js	No	1	0	1 hour
9	Add Courses	Test the functionality of adding courses to the database in the admin module	AddCourses.js	Yes	3	1	2 hours
10	Enroll in the System	To verify that students can successfully enroll in the system	EnrollSystem.js	No	4	0	2 hours
11	Display all Courses and Instructors	Students can display all courses and instructors	DisplayCoursesInstructors.js	No	3	0	2 hours
12	Evaluate Courses and Instructors	Students can save a form and continue later	EvaluateCoursesInstructors.js	No	4	0	2 hours
13	Save, Continue Editing and Submit Evaluation Forms	Students can save a form and continue later	SaveContinueEditSubmitForms.js	No	6	0	3 hours

Use Case Number	Use Case Name	Test Case Description	File Name	Retest?	Number of Activity Paths Tested	Number of Non-working Activity Paths	Time Spent
14	Create, Edit and Submit Evaluation Forms	Test evaluation form creation and submission functionality for instructors	CreateEditSubmitForms.js	Yes	4	1	2 hours
15	Display and Download Evaluation Results	Instructors can create, save, continue editing and submit evaluation forms	DisplayDownloadResults.js	No	6	0	3 hours
16	Display Surveys and Results	Verify if the department manager can view the surveys and results	DisplaySurveysResults.js	No	2	0	1 hour
17	Assign Instructors to Courses	The selected instructor can be assigned to the selected course	AssignInstructorsCourses.js	No	4	0	2 hours
	-	-	-TOTAL	-	60	2	34 hours

4.1 Criteria for Evaluating Results

a. Output Variation: The result of a test must be as close as possible to what we expected. If it's not, there might be a problem.

b. Input-Output Combinations: We aim to test a wide array of combinations, covering all potential scenarios. An acceptable test result should include a minimum set of combinations, which ensures that both the nominal and edge cases have been addressed.

c. Test Duration: Tests should finish in a reasonable time. If a test takes too long, it might mean something is wrong.

d. System Breaks: The occurrence of interrupts, halts, or breaks during testing should be minimal. Any such occurrence will require an investigation into the root cause.

e. Processing Errors: The occurrence of processing errors should be minimal. Any such error might signify an underlying problem with the system, which requires immediate attention.

f. Retesting Conditions: If a test doesn't work or gives strange results, we will do it again. Also, if we change the backend code, we will retest to make sure everything still works.

g. Test Data Irregularities: If the outputs indicate a potential issue with the test data or the test procedures, an investigation into these aspects should be conducted.

h. Test Control and Status: The system should tell us clearly how each test is doing and when it's ready for the next one.

i. Additional Criteria: Depending on the specific requirements of the project, additional evaluation criteria might be defined. These would be treated with the same level of importance as the above-mentioned criteria.

5. References

Project is mostly based on our past knowledge and application of that knowledge, we don't have any specific references to cite. The project's development and testing are guided by general principles of software engineering taught in our classes. Various online resources such as StackOverflow, Mozilla Developer Network, and Postman's official documentation were consulted to solve particular issues or understand specific concepts.