Software Testing Plan

Student Code Online Review and Evaluation

A terminal program and web-application for use in Florida Tech's CSE department to facilitate the submission of code for professor created assignments.

Team Members:

Michael Komar - mkomar2021@my.fit.edu
Charlie Collins - ccollins2021@my.fit.edu
Logan Klaproth - lklaproth2021@my.fit.edu
Thomas Gingerelli - tgingerelli2021@my.fit.edu

Faculty Advisor / Client:

Dr. Raghuveer Mohan - rmohan@fit.edu
09/20/2024

Table of Contents

1. Introduction

   This document describes the specific test cases that will be used to ensure the application meets all of the specified requirements. To ensure we meet this goal, the test cases cover using both the shell client as well as the web application. Additionally, we cover the two different types of users, students and professors

2. Testing Plan

   The test cases outlined in this document describe procedures that will test specific parts of the application. Included in the document are both functional and user test cases. Functional test cases are procedures that test specific functions, and there should be at least one functional test case for each requirement specified in the SRS. A user test case describes functions that a tester should interact with to ensure the application is user friendly.

3. Functional Test

   3.a. Assignment Creation

   - Assignment name: Professor can add an assignment name
     - Test 1: On the web app, as a professor user, add a valid name in the assignment name field. The new assignment, once created, should have the inputted name.
     - Test 2: On the shell client, as a professor user, add a valid name when prompted for the assignment name. The new assignment, once created, should have the inputted name.
   - Assignment description: Professor can upload a description for a new assignment
     - Test 1: On the web app, as a professor, navigate to the assignment creation page, select the upload assignment description, and supply a .doc file. The file should be accepted.
     - Test 2: On the web app, as a professor, navigate to the assignment creation page, select the upload assignment description, and supply a .docx file. The file should be accepted.
     - Test 3: On the web app, as a professor, navigate to the assignment creation page, select the upload assignment description, and supply a .pdf file. The file should be accepted.
     - Test 4: On the web app, as a professor, navigate to the assignment creation page, select the upload assignment description, and supply an unaccepted file type. The file should be rejected and the application should display an “Unsupported File Type” error.
     - Test 5: In the shell client, as a professor,using the assignment creation command, select the upload assignment description, and supply a .doc file. The file should be accepted.
     - Test 6: In the shell client, as a professor,using the assignment creation command, select the upload assignment description, and supply a .docx file. The file should be accepted.

    - Test 7: In the shell client, as a professor,using the assignment creation command, select the upload assignment description, and supply a .pdf file. The file should be accepted.
    - Test 8: In the shell client, as a professor,using the assignment creation command, select the upload assignment description, and supply an unaccepted file type. The file should be rejected and the application should display an “Unsupported File Type” error.
- Number of Allowed Attempts: Professor can select the number of attempts for a specific assignment
    - Test 1: On the web app, as a professor, navigate to the assignment creation page, enter a positive integer into the Number of Attempts field. The application should accept this input.
    - Test 2: On the web app, as a professor, navigate to the assignment creation page, enter a non-positive integer into the Number of Attempts field. The application should reject this input, revert back to the default unlimited option, and display an “Invalid Input” error message.
    - Test 3: On the shell client, as a professor, using the assignment creation command, enter a positive integer into the Number of Attempts field. The application should accept this input.
    - Test 4: On the shell client, as a professor, using the assignment creation command, enter a non-positive integer into the Number of Attempts field. The application should reject this input, revert back to the default unlimited option, and display an “Invalid Input” error message.
- Assignment Due Date: Professor can set the due date of the assignment
    - Test 1: On the web app, as a professor, navigate to the assignment creation page, and enter a valid date into the due date field. The application should accept this input.
    - Test 2: On the web app, as a professor, navigate to the assignment creation page, and enter an invalid date into the due date field. The application should reject input.
    - Test 3: On the client shell, as a professor, using the assignment creation command, enter a valid date into the due date field. The application should accept this input.
    - Test 4: On the client shell, as a professor, using the assignment creation command, enter an invalid date into the due date field. The application should reject input.
- Assignment Test Cases: Professor can create test cases
    - Test 1: On the web app, as a professor, navigate to the assignment creation page, and click create new test case. In the input field, enter text into the text box, in the output field, enter text into the text box, in the feedback field, enter text into the text box, and set the visibility to **visible**. The test case should successfully be created, and the test case should be visible by a student user.

    - Test 2: On the web app, as a professor, navigate to the assignment creation page, and click create new test case. In the input field, upload a text file, in the output field, upload a text file, in the feedback field, enter text into the text box, and set the visibility to **hidden**. The test case should successfully be created, and the test case should not be visible by a student user.
    - Test 1: In the shell client, as a professor, using the assignment creation page, select create new test case. In the input field, enter text into the text box, in the output field, enter text into the text box, in the feedback field, enter text into the text box, and set the visibility to **visible**. The test case should successfully be created, and the test case should be visible by a student user.
    - Test 2: In the shell client, as a professor, using the assignment creation page, select create new test case. In the input field, upload a text file, in the output field, upload a text file, in the feedback field, enter text into the text box, and set the visibility to **hidden**. The test case should successfully be created, and the test case should not be visible by a student user.

3.b Assignment Submission

- Student can submit file(s) to a valid assignment
    - Test 1: On the web app, as a student, navigate to a valid assignment from the student dashboard, and click the submit button. The assignment submission page should open.
    - Test 2: On the web app, as a student, navigate to the assignment submission page. Upload a single python file and click submit. The application should successfully submit the assignment.
    - Test 3: On the web app, as a student, navigate to the assignment submission page. Upload multiple python files and click submit. The application should successfully submit the assignment.
    - Test 4: On the web app, as a student, navigate to the assignment submission page. Upload a single java file and click submit. The application should successfully submit the assignment.
    - Test 5: On the web app, as a student, navigate to the assignment submission page. Upload multiple java files and click submit. The application should successfully submit the assignment.
    - Test 6: On the web app, as a student, navigate to the assignment submission page. Upload a single C/C++ file and click submit. The application should successfully submit the assignment.
    - Test 7: On the web app, as a student, navigate to the assignment submission page. Upload multiple C/C++ files and click submit. The application should successfully submit the assignment.
    - Test 8: On the web app, as a student, navigate to the assignment submission page. Upload an unsupported file type. The application should reject the file and display an “Unsupported file type” error message.

    - Test 9: On the web app, as a student, navigate to the assignment submission page. Upload a folder or compressed archive. The application should reject the file and display an “Unsupported file type” error message.
    - Test 10: On the web app, as a student, navigate to the assignment submission page. Upload as many submissions as the assignment allows. The submit button should no longer be clickable.

3.c Assignment Deletion

- Professor can delete an assignment
    - Test 1: On the web app, as a professor, navigate to the assignment dashboard, click on an assignment, then select delete. A delete confirmation window should appear.
    - Test 2: On the web app, as a professor, on the delete assignment confirmation window, select confirm. The assignment should now be deleted and no longer show up on the professor or student dashboard.
    - Test 3: On the web app, as a professor, on the delete assignment confirmation window, select cancel. The assignment should not be deleted and remain on both the professor and student dashboard.

3.d Auto Test

- Assignment submissions are graded upon being received.
    - Test 1: On the web app, as a student, navigate to a valid assignment with an associated auto test, and submit a valid python file. The application should run the auto test and display the results.
    - Test 2: On the web app, as a student, navigate to a valid assignment with an associated auto test, and submit a valid java file. The application should run the auto test and display the results.
    - Test 3: On the web app, as a student, navigate to a valid assignment with an associated auto test, and submit a valid C/C++ file. The application should run the auto test and display the results.
    - Test 4: In the client shell, as a student, using the submit assignment command, navigate to a valid assignment with an associated auto test, and submit a valid python file. The application should run the auto test and display the results.
    - Test 5: In the client shell, as a student, using the submit assignment command, navigate to a valid assignment with an associated auto test, and submit a valid java file. The application should run the auto test and display the results.
    - Test 6: In the client shell, as a student, using the submit assignment command, navigate to a valid assignment with an associated auto test, and submit a valid C/C++ file. The application should run the auto test and display the results.

3.e Immediate Feedback

- Upon failing a test case, the student should receive the associated feedback.
    - Test 1: On the web app, as a student, navigate to a valid assignment that has an auto test with feedback. Submit code that fails a test case. The application should display the associated feedback for that test case.
    - Test 2: On the web app, as a student, navigate to a valid assignment that has an auto test with feedback. Submit code that passes all test cases. The application should not display any professor feedback.

3.f MOSS

- The professor should see submissions flagged for similarity.
    - Test 1: From multiple student accounts, submit different files to the same assignment. Let the assignment due date pass. In the web app, as a professor, the application should not display that any of the submissions have been flagged.
    - Test 2: From multiple student accounts, submit the same file to the same assignment. Let the assignment due date pass. In the web app, as a professor, the application should underline and change the font to red for each submission and display a warning flag next to them.

3.g Grading Portal

- After the assignment due date has passed the grading portal should display the grades of the student submissions.
    - Test 1: From multiple student accounts, submit to the same valid assignment. Let the assignment due date pass. As a professor, navigate to the grading portal of the assignment. The application should display each submission, along with the associated scores and grades.
    - Test 2: As a professor, navigate to the grading portal of an assignment. In the grade field, click on the current grade. Enter a new grade. The assignment should now display that as the grade.

3.h Canvas Integration

- Grades should get uploaded to Canvas
    - Test 1: As a professor, navigate to the grading portal of a valid assignment. Click the Upload to Canvas button. The assignment page on Canvas should now be populated with matching grades from the grading portal.

4. User Test

4.a Shell Client

- Test 1 (Professor): A professor should use the application's shell client to interact with the following features: creating an assignment, deleting an assignment, viewing the grading portal, and pushing the grades to Canvas. The tester will then be asked if they were able to accomplish each task, and to rank the ease of completing each task.

- Test 2 (Student): A student should use the application's shell client to interact with the following features: submit an assignment, view auto test score, and view submission feedback. The tester will then be asked if they were able to accomplish each task and to rank the ease of completing each task.

4.b Web App

- Test 1 (Professor): A professor should use the application's web app interface to interact with the following features: creating an assignment, deleting an assignment, viewing the grading portal, and pushing the grades to Canvas. The tester will then be asked if they were able to accomplish each task, and to rank the ease of completing each task.
- Task 2 (Student): A student should use the application's web app interface to interact with the following features: submit an assignment, view auto test score, and view submission feedback. The tester will then be asked if they were able to accomplish each task and to rank the ease of completing each task.