**TEST PLAN**

**Automated Grading and Feedback Tool for Java**

**Team C – Street Coders**

Contents

[1.0 INTRODUCTION 3](#_Toc486238005)

[1.1 Objectives 3](#_Toc486238006)

[1.2 Team Members 3](#_Toc486238007)

[2.SCOPE 4](#_Toc486238008)

[2.1 Strategy 4](#_Toc486238009)

[2.2 Test Scenarios 5](#_Toc486238010)

[3. TEST APPROACH 8](#_Toc486238011)

[3.1 Test Automation 8](#_Toc486238012)

**Team C- Street Coders**

**Test Plan**

# 1.0 INTRODUCTION

The Test Plan has been created to communicate the test approach to team members while developing the Automated Grading and Feedback Tool for Java. It includes the objectives, scope, schedule, risks, and approach during the development. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

## Objectives

We will be using Bugzilla for logging bugs in code which are found during the testing process. Every team member will be maintaining their own version of test suite whose details will be uploaded to Bugzilla once a day to track defects and the test results.

Testing will be done on a daily basis whenever the code is updated. The core feature testing will carry the highest priority for testing. Since every team member will be coding in this project the developer will choose one of the remaining five team members to test their code and log it in the bug tracker

## 1.2 Team Members

|  |  |
| --- | --- |
| **Resource Name** | **Role** |
| Siva Reddy Mekapothula | Primary contact |
| Harish Babu Achanta | Client management |
| Vamshi Krishna Girikala | Requirements management |
| Madanamohan Reddy Govindu | Data management |
| Venkatesh Katragadda | Issues management |
| Sunil Kumar Sangaraju | Quality and testing management |
| Prasanthi Rani Bhogaraju | Communications and documentation management |

# 2.SCOPE

## 2.1 Strategy

The initial phase will include all ‘must have’ requirements. These and any other requirements that get included must all be tested. A tester must be able to:

1. Create a manual test with as many steps as necessary
2. Save it
3. Retrieve it and can view it when running the test
4. Enter results and appropriate comments
5. View results

The above-mentioned tester duties will be managed with the following control procedures

**Control Procedures:**

The team members work on different tasks and integrate the resulting code during milestones mostly working individually and collaborating at milestones.

**Reviews:**

We individually review others work and comment them for better enhancement.

**Review meetings:**

Review meetings every week to discuss, review and fix any errors occurred

**Change Request:**

We need to check with Client Requirements regularly whether we are developing the app accordingly or not, if not change them as per the client request.

**Defect Reporting:**

While using the app, if any defect occurs, Tester reports it in the bug tracking tool and assigns it to the concerned developer to fix it.

## 2.2 Test Scenarios

|  |  |  |
| --- | --- | --- |
| Requirements ID | Description | Test Case |
| R 1.1 | Verifying the login details | * If the student doesn’t enter any login credentials and tries to login   **Expected output**  **Error Message: Please enter the Username and Password.**   * If student enters wrong credentials and tries to login   **Expected output**  **Error Message: Your username and password doesn’t match**   * If student tries with username only   **Expected output**  **Error Message: Please enter password**   * If student tries with password only   **Expected output**  **Error Message: Please enter Username** |
| R 1.2 | Verifying student login details with validations | * If student tries with less than 8 characters in password column   **Expected output**  **Error Message: Password should not meet our requirements** |
| R 1.3 | Verifying that Student page consists of File upload option to upload the java assignment file. | * If student uploads empty zip file   **Expected output**  **Error Message: We don’t find any java files in your zip file** |
| R 1.4 | Verifying Student should have an option to view the grades that are given by the instructor. | * If student tries to view   **Expected output**  **Error Message: Student should not able to see grades of assignments he/she not enrolled in or grades of any other student** |
| R 1.5 | Verifying that Student can also view the Feedback from the instructor. | * If student tries to see the other assignments that have not been graded   **Expected output**  **Error Message: Student should not able to see feedback of his/her another assignments or feedback of any other student** |
| R 1.6 | Verifying that Student Feedback consists of detailed view of the errors along with the marks deducted | * If student tries to see the other assignments that have not been graded   **Expected output**  **Error Message: Student should not able to see another assignment of detailed error report** |
| R 2.1 | Verifying that instructor logs into the application to view all the student’s assignments | * If instructor enters wrong credentials and tries to login   **Expected output**  **Error Message: Your username and password doesn’t match** |
| R 2.2 | Verifying that instructor needs to upload various test cases that need to be executed against the student's written java code. | * If instructor uploads wrong test case document   **Expected output**  **Error Message: All the Students have unexpected program code, please check your test case document** |
| R 2.3 | Verifying that instructor also includes the expected output for each test case in the test cases documents in a specific format. | * If instructor uploads wrong expected output document   **Expected output**  **Error Message:** \*Not yet Decided |
| R 2.4 | Verifying that instructor should be able to select a student’s assignment that needs to be graded and click on Grade button. | \*Not yet Decided |
| R 2.5 | Verifying that what if the instructor clicks the grade button. | **Expected output**  **Error Message: Sorry you are not selected any zip file** |
| R 2.6 | Verifying that zip file gets extracted to a folder automatically or not. | * If zip file is empty   **Expected output**  **Error Message: Selected zip file have no files** |
| R 2.7 | Verifying that input to the program is fetched from the file that has all the test cases uploaded. | * If instructor uploads empty test case file   **Expected output**  **Error Message: Sorry your test case file is empty** |
| R 2.8 | Verifying that program is compiled and executed for each of the test cases. | \*Not yet Decided |
| R 3.1 | Verifying that 50% of marks will be deducted from total if the program fails to compile | * If java file has no code   **Expected output**  **Error Message: Total marks are deducted** |
| R 3.2 | Verifying that Java source code needs to be validated with the rules written in the validation files for proper function names, variable names and comments for each method. | * If student not follow certain validations like proper function names, variable names, comments in their written program   **Expected output**  **Error Message: You are not follow certain validations** |
| R 3.3 | Verifying that If there are mismatches or errors in the source code written by the student, the marks should be automatically deducted based on the error. | * If student deducted marks for methods   **Expected output**  **Error Message: You deducted marks for this method** |
| R 3.4 | Verifying that Rubric for deduction of the marks is also included in the Validation file. | * If rubric is not provided by instructor   **Expected output**  **Error Message: Student will not get marks for that assignment.** |
| R 3.5 | Verifying that data containing errors along with the deducted marks is recorded into an excel sheet which will be given as a feedback to the student. | * If student get full marks without any errors   **Expected output**  **Error Message: You are good** |
| R 4.1 | Verifying that program compiles and executes successfully. | * If compiler is failed   **Expected output**  **Error Message:** \*Not yet Decided |
| R 4.2 | Verifying that output is compared with the expected output. | * If application is failed to compare with expected output   **Expected output**  **Error Message:** \*Not yet Decided |
| R 4.3 | Verifying that in case, any output mismatches the marks are deducted based on the rubric present in the validation file uploaded by the instructor. | \*Not yet Decided |
| R 4.4 | Verifying that program is again scanned for proper program structure. | * If application failed to scan the code for proper program structure   **Expected output**  **Error Message:** \*Not yet Decided |
| R 4.5 | Verifying that program is scanned for the presence of comments for each method and proper indentation. | * If application failed to scan each method and proper indication   **Expected output**  **Error Message:** \*Not yet Decided |
| R 4.6 | Verifying that Java source code is also scanned in accordance with the rules written in the validation file. | * If application didn’t find java source code file   **Expected output**  **Error Message:** \*Not yet Decided |
| R 4.7 | Verifying that validation file contains all the required method names, variable names, method arguments, constant names. | * If validation file contains only some validations like method names, constant names   **Expected output**  **Error Message:** \*Not yet Decided |
| R 4.8 | Verifying that if the student's source code doesn’t comply with validation rules written by the instructor, the score is deducted automatically based on the rules specified in the validation file | \*Not yet Decided |
| R 4.9 | Verifying that data containing errors along with the deducted marks is recorded into an excel sheet which will be given as a feedback to the student. | \*Not yet Decided |
| R 5.1 | Verifying that grade is given to the student based on the rubric that is formulated in the validation file. | * If student didn’t get grade   **Expected output**  **Error Message:** \*Not yet Decided |
| R 5.2 | Verifying that grade will be picked from the instructor view and automatically reflected in the student view. | \*Not yet Decided |
| R 5.3 | Verifying that feedback is recorded in the Excel sheet will be attached in the student’s view for each and every student. | \*Not yet Decided |

# **3.** TEST APPROACH

The project is using an agile approach, with weekly iterations. At the end of each week, the requirements identified for that iteration will be delivered to the team and will be tested.

Exploratory testing will play a large part of the testing as the team has never used this type of tool and will be learning as they go.

## 3.1 Test Automation

No scenarios for testing automation have been identified