Software Quality Assurance Plan

Version 1.1

April 14, 2022

RECORD OF CHANGES

VERSION NUMBER	DATE	TITLE OR BRIEF DESCRIPTION
1.0	28/3/22	Draft of Software Quality Assurance Plan submitted to be approved by the instructor.
1.1	14/4/22	Second draft of Software Quality Assurance Plan submitted to be approved by the instructor.
1.2	18/4/22	Third draft of Software Quality Assurance Plan submitted to be approved by the instructor.

Section 1. Purpose

The purpose of this plan is to define the Software Quality Assurance (SQA) policy for the Learning Management System project developed by students of Software Engineering (L2) Spring 2022, Habib University. The document describes the quality metrics as well as the tools, techniques and methodology via which quality will be evaluated and assured throughout the course of this project.

Section 2. Management

2.1 Organization

Following are the functional groups that influence and control software quality:

2.1.1 Project Manager (Instructor of the class) is responsible for the following items:

i. Reviewing and approving the Learning Management System SQA Plan.
ii. Follow-up on the implementation of SQA throughout the project.
iii. Assign a specific time duration at the end of each sprint within which all groups must evaluate the assigned code and submit feedback as per the SQA plan.
iv. After the deadline of the submission of feedback as given in (iii), assign a specific time duration after which all groups must revise their code as per the feedback.
v. Identifying and informing all groups about the consequences of not submitting relevant feedback as per the SQA Plan.

2.1.2. Software Engineering groups (All the groups for the project in the class) are responsible for:

i. Reviewing, understanding and having consensus on the SQA Plan.
ii. Implementing the software quality in accordance with the SQA Plan.
iii. Evaluating the assigned code for quality assurance within the time duration given by Project Manager.
iv. Submitting the feedback on quality of the assigned code (in the form of a filled copy of the check list given in Appendix A).

2.2. Resources

2.2.1. Software Engineering groups shall have access to SQA Plan approved by Project manager to perform SQA functions as per the SQA plan.
2.2.2. Project Manager shall be familiar with the SQA Plan developed.

Section 3. Task: Evaluate Software Implementation/Development Process

Software implementation or coding is the point in the software development cycle at which the design is finally implemented. By the end of each sprint, following SQA activities shall be implemented:

i. Evaluate whether features implemented in the code are as per the tickets created on JIRA for that particular sprint.

ii. Evaluate whether the code is as per the quality metrics mentioned SMART formula.

iii. Evaluate the quality of the code.

iv. Verify that the code generates desired output as per the test cases.

Software Engineering groups shall use the checklist in Appendix A for conducting the evaluation. Each group will evaluate the testcases and user stories of another group. The filled checklist shall be submitted to project manager and the group whose code is being evaluated. Each group will validate their code with verified testcases and will have a certain time duration (as per communicated by project manager) to upload the updated code after correcting it based on the feedback received from the filled checklist.

Section 4. Tools, Techniques, and Methodologies

4.1 Tools:

4.1.1. Checklist (for code evaluation as given in Appendix A)

4.1.2. Github

4.1.3. User Stories for the particular sprint of relevant/pair group.

4.1.4. List of tasks for the particular sprint of relevant/pair group.

4.1.5. List of test cases for the particular sprint of relevant/pair group.

4.2. Techniques:

4.2.1. Evaluation of the code based on a checklist (Appendix A)
4.2.2. Submission of checklist as a means to give feedback.
4.2.3. Incorporating changes in the code based on feedback.

4.3 Methodologies: After the completion of coding, the project management will assign each group another group's code. All Software Engineering groups will have access to their pair group's Github repository, list of testcases and user stories through google doc.

https://docs.google.com/document/d/1s6lbU0NykhT5dNdwTyibEwOSfLBCUla0lU3NYD-mgls/edit?usp=sharing

The link to google doc given above is to be used for sharing test cases and user stories.

Pair groups are responsible for verification of the testcases. Validation of the testcases shall be done by the team itself. All groups shall communicate with their pair group as per their convenience.

All groups would be provided a checklist (refer to Appendix A) which uses the SMART formula. The SMART formula stands for:

1. Specific
2. Measurable
3. Appropriate
4. Realistic
5. Timely

The checklist evaluates whether the user stories and testcases are created in conformation with the requirements as per the list of tasks (tickets) assigned in the particular sprint. Another quality metric included in the checklist is the style with which a code is written. The code must be readable, modular and properly formatted.

Each group shall fill the checklist and submit it to Project Manager and the group whose code is evaluated via canvas discussion forum. The deadline to submit the checklist would be decided and communicated by the Project Manager. After the submission of the checklist, each group will have a time duration to update their code. This time duration would be decided and communicated by Project Manager. Thus, on violation of quality metrics (as per Appendix A checklist), the team shall get a chance to revise the particular piece of code and their grades shall not be impacted by it.

Section 5. Code Control

Code control includes the following items:

5.1. Distributing copies of the code: Learning Management System uses GitHub for code control. The Github repository of each group shall be the only official channel to share each sprint's code.

5.2. **Identifying the documentation that is affected by a change:** Upon the introduction of any change, all relevant documentation shall be updated as per the change management policy.

5.3. Establishing a new version: Establishing a new version of the code or any documentation shall be as per the change management policy.

Appendix A

S. No.	**Aspect to evaluate**	**Check / Uncheck**	**Comments (if any)**
1	Testcases and user stories are in accordance with the required tasks (based on the tickets).		
2	**Code Quality:** Code is readable, modular and properly formatted.		