Software Quality Assurance (SQA) Plan

By Inspiration

Date: 21 February 2022

Signature Page

Prepared by: Inspiration **Date: 21/02/2022**

Reviewed by 1: Hui Zhan **Date: 20/02/2022**

Reviewed by 2: Clarence Hong **Date:20/02/2022**

Approved by : Manav **Date: 20/02/2022**

Document Change Record

Revision	Description of Change	Approved by	Date
0.10	Initial Template	Jovan Huang	7/02/22
0.20	Inserting details for Sections 1 to 14	Clarence Hong	12/02/22
0.30	Edit details for sections 1 to 14	Hui Zhan	18/02/22
1.0.0	Approval by Project Manager	Manav	20/02/22

Contents

1. Purpose and Scope

1.1. Purpose

The purpose of this Software Quality Assurance (SQA) Plan is to establish the goals, processes, and responsibilities required to implement effective quality assurance functions for the **Friendstagram** project.

The Software Quality Assurance Plan provides the framework necessary to ensure a consistent approach to software quality assurance throughout the project life cycle. It defines the approach that will be used by the Quality Assurance Manager (QAM) and Software Quality (SQ) personnel to monitor and assess software development processes and products to provide objective insight into the maturity and quality of the software. The systematic monitoring of products, processes, and services will be evaluated to ensure they meet requirements and comply with policies, standards, and procedures, as well as applicable Institute of Electrical and Electronic Engineers (IEEE) and International Organization for Standardization (ISO) standards.

1.2. Scope

The purpose of SQA is to ensure that the software developed does not deviate from the original intended product. SQA is also concerned with identifying any errors, omissions, inconsistencies, and alternatives, enhancements or improvements that can be made at any stage of development.

Friendstagram is a web application that recommends new potential friends to users. This web application shall allow users to register for a personal account, fill in their personal interests and hall accommodation information before allowing them to find new potential friends based on this information. Users may also view their history of matched friends. The web application shall be mobile responsive to facilitate mobile users.

Friendstagram shall be built with React and Redux on the frontend and Django on the backend. React and Redux are chosen as they are component-based, efficient and have an easy learning curve. As a component-based library, components built with React are reusable, i.e. modular in nature, this also helps the project team in splitting up work. Redux provides a global, top-level storage for easier state management. Django is chosen as it is simple, flexible, reliable and scalable. It is also powerful and efficient in terms of API calls and supports relational databases.

Django extensions are also used to provide user authentication, allowing the customization of user properties as well as a quick extension point for future integration of industry standard OAuth2 login and logins from other social network providers.

2. Reference Documents

- IEEE STD 730-2002, IEEE Standard for Software Quality Assurance Plans (http://standards.ieee.org/reading/ieee/std_public/description/se/730-2002_desc.html)
- ISO IEC 90003:2004 Software Standard (http://praxiom.com/iso-90003.htm)
- Project Proposal
- System Requirement Specifications

3. Management

This section describes the management organizational structure, its roles and responsibilities, and the software quality tasks to be performed.

3.1. Management Organisation

The implementation of the quality assurance system is the responsibility of the Quality Assurance Manager (QAM).

3.1.1. Project Management

The Project Manager will be responsible for approving:

- The system requirement specification document
- The overall timeline for the project
- The choice of system development life cycle
- The choice of software development tools and techniques utilized
- The selection of project teams
- The training of project teams

3.1.2. Assurance Management

The QAM provides Project Management with visibility into the processes being used by the software development teams and the quality of the products being built. The QAM maintains a level of independence from the project and the software developers.

In support of software quality assurance activities, the QAM has assigned and secured Software Quality personnel from the pool of available SQ trainees to coordinate and conduct the SQ activities for the project and report back results and issues.

3.2. Tasks

This section summarizes the tasks (product and process assessments) to be performed during the development of software. These tasks are selected based on the developer's Project Plan and planned deliverables, and identified reviews.

3.2.1. Product Assessments

The following product assessments will be conducted by SQ personnel:

- Registering for account(s) successfully
- Logging into an account with existing credentials successfully
- Unable to log into an account with *wrong* credentials
- Setting an account's profile successfully
- Editing a personal profile component successfully
- Retrieving recommendation of friends successfully
- Viewing a recommended friend's profile successfully
- Viewing personal profile component

3.2.2. Process Assessments

The following process assessments will be conducted by SQ personnel:

- Requirement Management Process
- Risk Management Process
- Change Management Process
- Release Process
- Testing Process

3.3.Roles and Responsibilities

This section describes the roles and responsibilities for each assurance person assigned to the Project.

3.3.1. QAM

Responsibilities include, but are not limited to:

- Secure and manage SQ personnel resource levels
- Ensure that SQ personnel have office space and the appropriate tools to conduct SQ activities

- Provide general guidance and direction to the SQ personnel responsible for conducting software quality activities and assessments
- Assist SQ personnel in the resolution of any issues/concerns and/or risks identified as a result of software quality activities
- Escalate any issues/concerns/risks to project management

3.3.2. Software Quality Personnel

Responsibilities include, but are not limited to:

- Develop and maintain the project software quality assurance plan
- Generate and maintain a schedule of software quality assurance activities
- Conduct process and product assessments, as described within this plan
- Identify/report findings, observations, and risks from all software assurance related activities to the QAM

4. Documents

4.1. Purpose

This section identifies the minimum documentation governing the requirements, development, verification, validation, and maintenance of software that falls within the scope of this software quality plan. Each document below shall be assessed (reviewed) by SQ personnel.

4.2. Minimum Document Requirements

- Project Proposal
- Use Case Model
- System Requirement Specifications
- Project Plan
- Quality Plan
- Risk Management Plan
- Design for Software Maintainability Report
- Configuration Management Plan
- Change Management Plan
- Release Management Plan

- IEEE829 Test Plan
- Test Cases and Test Coverage Report

5. Standards, Practices, Conventions and Metrics

5.1. Purpose

This section highlights the standards, practices, quality requirements, and metrics to be applied to ensure a successful software quality program.

5.2. Software Quality Programme

These practices and conventions are tools used to ensure a consistent approach to software quality for all programs/projects.

The four most important qualities for Friendstagram are Efficiency, Portability, Usability and Reliability.

- **Efficiency**: The system must have a maximum loading time of 2s when navigating between different pages. The web application must be able to handle concurrent requests without slowing down.
- **Portability**: The system must be able to run on different user agents across different O.S. platforms including mobile browsers such as Safari and Google Chrome.
- **Usability**: The web application must be user-friendly. This includes designing the user interface with simplicity and consistency in mind. In simplicity, the user interface must not be cluttered with excess information and navigation between different pages must be intuitive and uncomplicated. The user interface design must also adhere to consistency in terms of size of components, font-sizes, colors, etc. The recommendation process must also be simple to use and intuitive.
- **Availability**: The web application must have an availability of "5 nines" (i.e. 99.999% of up time, or 5.26 minutes of downtime yearly) and needs to be able to maintain its level of performance at all times, especially when there are many users using the platform concurrently.

5.2.1. Standard Metrics

The following standard metrics are the minimum planned metrics that will be collected, reported, and maintained in the area of software quality assurance:

- **Length of code**: This measures the size of the program. Long code length generally implies that it is more error-prone and more complex. It is a good metric which helps in predicting error-proneness. It encourages programmers to reduce code length as much as they can to avoid errors.

- **Length of identifiers**: This measures the average length of unique identifiers (i.e. variable/function/module names). Long identifiers are more likely to be meaningful and increase understandability of the code. This metric encourages developers to be more careful during development by giving more meaningful and intuitive identifiers.

- **Cyclomatic complexity**: This measures the control complexity of a program by counting the number of branches in a method. The lower the cyclomatic complexity, the more understandable the code is. This metric helps ensure that programmers use minimal branches in their codes, hence increasing understandability.

- **Fog index**: This measures the average length of words and sentences in the documents for the web application. The higher the average length, the higher the fog index, and the more difficult it is to understand. This metric will encourage programmers to write shorter sentences and words to increase the comprehensibility and understandability of the web application's documents.

- **Fan in/Fan out**: This encourages programmers to minimize the number of functions that call some other function (Example: X). It also encourages them to minimize the number of functions that are called by function X. This reduces the complexity of X so that if there are any changes to X, there will be less extensive knock-on effects.

- **Depth of conditional nesting**: This encourages programmers to minimize the depth of nesting of if-statements so as to increase understandability and lower error-proneness.

6. Software Reviews

6.1. Purpose

This section identifies the number and type of system/subsystem reviews and engineering peer reviews that will be supported by the SQ Personnel. The project milestone chart, and the SQ Personnel resource levels determine the reviews that are supported.

6.2. Minimum Software Reviews

For each review, SQ personnel will assess the review products to assure that review packages are being developed according to the specified criteria, the review content is complete, accurate, and of sufficient detail, and Requests for Action are captured, reviewed, and tracked to closure. In addition, SQ personnel will also assess the processes used to conduct the reviews to determine if appropriate personnel are in attendance, correct information is presented, entry and exit criteria are met, and appropriate documents are identified for update.

The following software reviews will be assessed by SQ:

- Project Plan Review
- Requirements Analysis Review
- Software Design Review
- Test Plan Review
- Acceptance Review

7. Test

SQ personnel will assure that the test management processes and products are being implemented per Test Plan. This includes all types of testing of software system components as described in the test plan, specifically during integration testing (verification) and acceptance testing (validation). SQ personnel will monitor testing efforts to assure that test schedules are adhered to and maintained to reflect an accurate progression of the testing activities. SQ will assure that tests are conducted using approved test procedures and appropriate test tools, and that test anomalies are identified, documented, addressed, and tracked to closure. In addition, SQ will assure that assumptions, constraints, and test results are accurately recorded to substantiate the requirements verification/validation status. SQ personnel will review post-test execution related artifacts including test reports, test results, problem reports, updated requirements verification matrices, etc.

8. Problem Reporting and Corrective Action

SQ personnel generate, track, and trend assessment findings and observations in a centralized Reporting and Corrective Action System. The project team shall use Github for version control. With Github, any source code changes can be tracked easily. A CI/CD pipeline shall also be set up to ensure any code changes can be reviewed by the Lead Developer before being approved and merged into the release branch. Additionally, Project Manager, QAM and SQ personnel can also review the repository at all times to ensure the quality of the system. If any problems are detected by the project manager, QAM, or SQ personnel, a new *Issue* can be created in the Github repository and a new item shall be created in Trello and the developers will take on these issues to rectify them. Trello is a KanBan-style online collaboration tool which shall be used to track development progress. If necessary, manpower resources shall be diverted over the developer team to assist in rectifying these issues.

9. Tools, Techniques and Methodologies

SQ personnel will require access to the following:

9.1. Software Quality Tools

- Microsoft Office tools (i.e., Word, Excel, and PowerPoint)
- Google Drive
- Github
- Trello
- Figma

10. Media Control

SQ deliverables shall be documented in one of the following Microsoft software applications: Word, Excel, or PowerPoint. Deliverables shall be in soft copy. See Section 11 for additional details on the collection and retention of key records. SQ personnel shall request space on the project's secured server for SQ records. This server is password protected and backed up nightly.

The Project Manager, QAM and SQ personnel shall ensure that the latest documentation is uploaded timely onto our team's shared Google Drive, Wiki and SVN. The Lead Developer and SQ personnel will also make sure that all codes on Github and SVN are of the latest version. Google Drive, Wiki, SVN and Github are all only accessible by SQ personnel and team members.

11. Record Collection, Maintenance, and Retention

SQ personnel will maintain records that document assessments performed on the project. Maintaining these records will provide objective evidence and traceability of assessments performed throughout the project's life cycle. There are two types of records that will be maintained: Hardcopy and Electronic. SQ personnel will maintain electronic or hard copies of all assessment reports and findings. SQ Project folders will contain hardcopies of the assessment work products such as completed checklists, supporting objective evidence, and notes.

The table below identifies the record types that will be collected, as well as the Record Custodian and Retention period

Record Title	**Record Custodian**	**Record Retention**
SQA Assessments	SQ Personnel	One Year
SQA Checklists	SQ Personnel	One Year
Deliverable Defects	SQ Personnel	One Year

12. Training

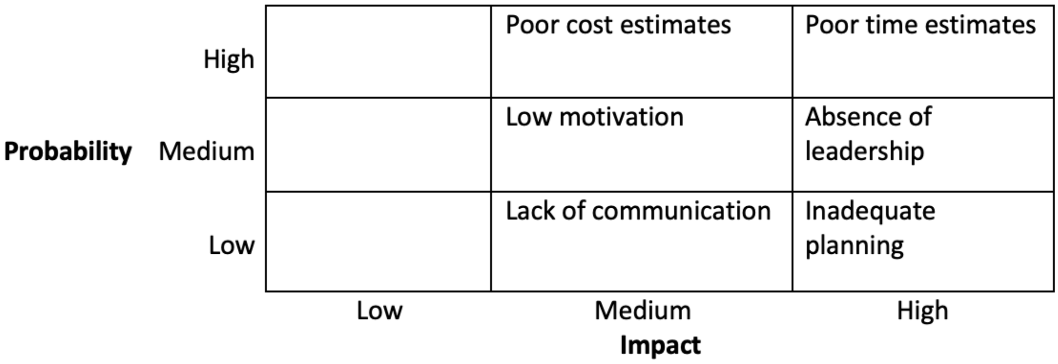
SQ personnel have fundamental knowledge in the following areas through prior experience, training, or certification in methodologies, processes, and standards:

- Audits and Reviews (Assessments)
- Risk Management
- Software Assurance
- Configuration Management
- Software Engineering
- ISO 9001, ISO 9000-3
- CMMI
- Verification and Validation

13. Risk Management

SQ personnel will assess the project's risk management process and participate in monthly risk management meetings and report any software risks to the QAM and the project manager.

SQ personnel will first use SWOT analysis (strengths, weaknesses, opportunities, and threats) to identify the broad positive and negative risks. After risk identification, we will perform qualitative risk analysis using a probability/impact matrix. For each of the risks identified in our SWOT analysis, we will label them as low, medium or high in terms of its probability of occurrence and its impact if it did occur. Figure 1 below shows a sample of how a probability/impact matrix will look like.



	High		Poor cost estimates	Poor time estimates
Probability	Medium		Low motivation	Absence of leadership
	Low		Lack of communication	Inadequate planning
		Low	Medium	High
			Impact	

Fig 1. A sample probability/impact matrix

14. SQA Plan Change Procedure and History

SQ personnel are responsible for the maintenance of this plan. It is expected that this plan will be updated throughout the life cycle to reflect any changes in support levels and SQ activities. Proposed changes shall be submitted to the Quality Assurance Manager (QAM), along with supportive material justifying the proposed change.