Software Quality Assurance (SQA) Plan
By Team Eagles

Date: February 2021

Version: 1.5 Approved

Dandapath Soham – Project Manager, Back-end Developer

Gupta Jay – Lead Developer, Release Engineer

Kanodia Ritwik – Front-end Developer, Release Engineer

Mundhra Divyesh – Front-end Lead Developer

Bhatia Ritik – Back-end Lead Developer

Somani Palak – QA Manager, QA Engineer

Bansal Aditya – QA Engineer

School of Computer Science & Engineering

Nanyang Technological University, Singapore

Signature Page

Reviewed By 1:  **Date:** 5 February 2021

Somani Palak
(Quality Manager)

Reviewed By 1:  **Date:** 10 February 2021

Bansal Aditya
(Quality Engineer)

Reviewed by 2:  **Date:** 15 February 2021

Bhatia Ritik
(Backend Lead)

Approved by:  **Date:** 15 February 2021

(Bansal Aditya)

Document Change Record

Revision	Description of Change	Approved by	Date
0.1	**Initial Template**	**Bhatia Ritik**	**02/02/21**
1.0	**Initial Draft**	**Somani Palak**	**06/02/21**
1.1	**Use Case Diagram & Protype added**	**Dandapath Soham**	**07/02/21**
1.2	**Functional Requirements added**	**Bansal Aditya**	**09/02/21**
1.3	**Overall Formatting**	**Somani Palak**	**09/02/21**
1.4	**Proofread and minor changes**	**Bhatia Ritik**	**11/02/21**
1.5	**Final Approval & Signatures**	**Bansal Aditya**	**14/02/21**

Table of 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 and monitoring functions for the **NTUCollab** project.

Software quality is defined as how well the software meets its established requirements and stakeholder wants, needs, and expectations. It is one of the key attributes that a software must have to be a successful product and function effectively, as intended. Maintaining high standards of software quality should be of high priority and this calls for the need of a Software Quality Assurance Plan, to properly document the steps and formalize the process for the same.

The Software Quality Assurance Plan provides the framework necessary to ensure a consistent approach to software quality assurance throughout the project life cycle. It details 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 ISO standards. These standards help provide a baseline metric against which current processes adopted for software development can be evaluated and rectified, if needed.

1.2. Scope

The purpose of SQA is to ensure that the software developed does not deviate from the original requirements or intended product. SQA is also concerned to identify any errors, omissions, inconsistencies, and alternatives, enhancements or improvements that can be made at any stage of development. Quality is proactive, consisting of activities to ensure that quality is built into the product. More formally, SQA is a cascading series of activities that lead to a justified statement of confidence that a software development project yields a product that fulfils the purposes that the stakeholders intended for them.

NTUCollab is a cross-platform mobile application which will allow students to find groups among their peers based on similarities of interests and hobbies. Based on the preferences of the users, they will have the ability to join groups from broadly 3 categories: *Modules*, *Clubs,* and *Interest Groups*. These groups are intended to facilitate easy interaction between peers who want to connect with like-minded people and collaborate on projects, sports, competitions etc.

NTUCollab's team consists of seven members who have been selectively assigned roles based on their expertise, to oversee the success of their respective department. Some member handle multiple roles at the same time, depending on how integrated the roles are and the responsibilities for the same. There is a single Project Manager (PM), to oversee the entire project and is responsible for tasks like scheduling

meetings, ensuring project deadlines are met and track overall progress of the project through frequent meetings. We have one Lead Developer who oversees all the aspects of the project, namely, frontend, backend, and Quality Assurance (QA). Two members are in the backend team, with one member solely dedicated for its development, two members are in the QA team and two members are in the frontend team with one member solely dedicated for its development. Finally, we have one Release Engineer who oversees the different released versions of NTUCollab.

NTUCollab will be developed using the Flutter mobile development framework and Dart programming language. Further, it will utilize Firebase APIs and Firestore database for achieving other functionalities. The SQA Plan intends to cover this software, its source, and verify its authenticity. The scope of the SQA Plan is meant to be as rigorous as possible, to ensure the highest quality of software development process and components and to avoid future compromises in case the software quality is not up to mark. In this spirit, this plan will cover all the software items like Flutter, Firebase and Firestore, User Interface prototyping tools like Figma, versioning system (e.g., GitHub, BitBucket etc.) and the Integrated Development Environment (IDE) used (e.g., VSCode, Android Studio etc.) for the development of NTUCollab.

Regular testing will be conducted for all the components of the system, namely the User Interface, database, and internal logic (that is, recommendations made and matching of users with modules, clubs and interest groups based on the preferences). Thorough and rigorous tests are meant to find any bugs in the software and resolve them before declaring the code to be release-ready.

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 (https://entuedu.sharepoint.com/teams/ASEProject/Shared%20Documents/General/Lab%201/FINAL/Project_Proposal.pdf)
- System Requirement Specifications (SRS) (https://entuedu.sharepoint.com/teams/ASEProject/_layouts/15/Doc.aspx?OR=teams&action=edit&sourcedoc={7365657A-6FEC-4B34-9B6E-33E7AB32FD8F})
- GitHub documentation (https://help.github.com/en/github)
- Flutter documentation (https://flutter.dev/docs)
- Firebase documentation (https://firebase.google.com/docs)
- Kanban board (https://www.atlassian.com/agile/kanban/boards)
- 829-2008 - IEEE Standard for Software and System Test Documentation (https://ieeexplore.ieee.org/document/4578383)

- 1008-1987-IEEE Standard for Software Unit Testing (https://standards.ieee.org/standard/1008-1987.html)
- Android Studio Documentation (https://developer.android.com/docs)
- Learning Firebase and Flutter (https://codelabs.developers.google.com/codelabs/flutter-firebase#0)

3. Management

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

3.1. Management Organisation

The implementation of quality assurance system and subsequently ensuring its application 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 time scale for the project
- The choice of system development life cycle
- The choice of software development tools and techniques utilised
- The selection of project teams
- The training of project teams

In addition to the above approvals, the role of the Project Manager includes:

- Ensure that the team remains on schedule through regular follow ups and deliverable submissions
- Boost the team morale and help resolve conflicts as and when the need arises
- Promote healthy discussion, evaluate alternatives, and make informed decisions
- Remind all team members of maintaining highest quality in software development processes, as laid out in the Quality Assurance Plan

3.1.2. Assurance Management

A QAM helps develop and safeguard quality standards in a production company. 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 and hence makes all evaluations independently without the influence of other members.

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. The QAM then analyses the reports and findings of other SQ personnel and after deliberations, will compile all the findings. Like any other project, NTUCollab will have to go through stringent Quality Assurance processes before it can reach the users.

3.2. Tasks

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

3.2.1. Product Assessments

To ensure high quality of NTUCollab as a product, the following product assessments will be conducted by SQ personnel:

- **User Interface (UI)**

  The UI of NTUCollab will be assessed to ensure it is simple yet engaging and pleasing to look at. The interface should be self-explanatory, and a user must take minimal time to get familiar with the application interface and must find it easy to navigate its different functionalities.

- **Register and Login**

  NTUCollab must be able to support simple social media login and registration and should be able to seamlessly communicate with the Firestore database to store user information for future logins. During assessment, these two functionalities must have close to zero errors.

- **Database Communication**

  This is a very important aspect of NTUCollab as almost all operations rely on database communication and integrity. SQ personnel need to thoroughly assess that database communication at all levels, from taking user preferences to reading data, should happen efficiently and with zero errors. Ensuring a seamless database integration will guarantee NTUCollab's success.

- **STARS Integration**

  STARS is the course registration system provided by NTU and stands for Student Automated Registration System. It is used by each and every student, every semester and is hence an integral part of the student life at NTU. Hence, NTUCollab should have error-free and efficient integration with STARS and the quality of this integration should be assessed by SQ personnel. Tasks like communication with the STARS system, retrieving values from STARS, viewing courses etc. should take

place with minimum latency and maximum efficiency and hence the SQ personnel must devise relevant load tests to gauge the quality of the same.

- **Recommendation System**

  NTUCollab's main section is the recommendation of modules, clubs, and groups, based on user preferences. It is necessary to assess the complete functionality of this section with minimal errors. This includes capturing user preferences, storing it in database and when requested, display relevant modules, clubs, and groups, by retrieving the same from the database.

- **Posts**

  NTUCollab aims to develop a community of users and the *Post* feature is a necessary aspect for the same. SQ personnel should assess whether users can easily create and read posts and that their feed is always updated, with minimal downtime. Quality interaction with others hinges on quality development of this functionality and hence it should be assessed carefully by the SQ team.

- **Filtering and Searching**

  To improve the scalability of NTUCollab, the filter and searching of modules should work properly. This is because as the database and the number of users grow, so will the number of modules, clubs, and interest groups. To keep users satisfied, it is necessary to ensure they can efficiently search for their interests and filter them according to their preferences. SQ personnel need to assess the efficiency and latency of using the *filter* and *search* functionalities.

3.2.2. Process Assessments

Process is as important as the product itself. Hence, the following process assessments will be conducted by SQ personnel:

- **Requirement Management Process**

  The process of requirement elicitation and analysis is covered under the scope of Quality Assurance and Management, especially since it is the most important part of the project. SQ personnel must draft and enforce requirement elicitation procedures to maximize coverage of functional requirements of NTUCollab. This should be followed by further discussions and meetings to formally finalize the requirements of NTUCollab as an application.

- **Change Management Process:**

  A significant change in the team and its processes might be needed to achieve certain business outcomes. SQ personnel must assess readiness and ease of changing processes, techniques, and people to achieve the expected outcomes.

- **Maintainability Management Process**:

  NTUCollab must be maintainable, which means that it should be easy to correct faults, improve performance and change as per changing requirements. The SQ team must formalize a set of baseline expectations for maintainability and the steps to achieve the same. During development, this management process must be assessed by the SQ team to ensure high process quality.

- **Risk Management Process**

  SQ personnel are accountable to ensure that all risk management measures are put in place and adopted, to mitigate risks and minimize the impact, if any. This includes drafting and relaying a formal risk management plan and enforcing the adoption of the same. All edge cases and potential risks should be flagged, and the team should prepare for such contingency scenarios in advance. Possible risk scenarios include running behind schedule, lack of resources for implementation of some features, lack of developer experience regarding some part of the application etc.

3.3. Roles and Responsibilities

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

3.3.1. QAM (Quality Assurance Manager)

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
- Developer quality control policies and standard for the rest of the team to follow, after discussions with the SQ personnel

- Assist SQ personnel in the resolution of any issues/concerns and/or risks identified because of software quality activities
- Assess operational data and identify quality problems
- 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
- Hold regular meetings with the QAM to align quality expectations and work to relay the information to Project Management, as and when needed

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 an SQ personnel. Any errors, clarifications and abnormalities are to be reported to the QAM to decide on further action to rectify the same.

4.2. Minimum Document Requirements

- System Requirement Specifications
- Project Proposal
- Project Plan
- Risk Management Plan
- Test Plan
- Test Cases and Test Coverage Report
- Release Plan
- Change Management Plan
- Configuration Management Plan

- Design report on software maintainability

5. Standards, Practices, Conventions and Metrics

5.1. Purpose

The purpose of this section is to highlight the practices, standards, conventions, and quality metrics to ensure that the software produces if of the highest quality. It is important to lay down a set protocol regarding the standard, practices and metrics that forms the baseline and can be referred to by anyone within the team. This section aims to achieve uniformity by acting as the de facto reference point for various conventions, metrics and practices as mentioned earlier. The purpose of following a comprehensive set of software tests is to ensure that the user encounters minimal errors, and to ensure smooth integration of different parts of the code when any single part is changed or updated. This aids in the proper deployment of the code as well and in the process, minimizes the downtime.

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 4 most important qualities required in the software, as per the ISO 9126 convention are:

- **Functionality**

  The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions. It is important that NTUCollab is a fully functional mobile application and works as expected by the user. It should satisfy all functional requirements and should be traceable if any error arises. Further, it is compulsory that NTUCollab has cross-platform support, that is, available to users of both Android and iOS, to cater to the needs of a wider audience.

- **Usability**

  The capability of the software product to be understood, learnt, used and attractive to the user when used under specified conditions. It is imperative that NTUCollab be usable by having a simple user interface, so that the user is not overwhelmed. The users of our mobile application will accept it only if they find it easy to navigate and use.

- **Reliability**

  The capability of the software product to maintain its level of performance under stated conditions for a stated period. NTUCollab should be a reliable

software. Its code design should not include tight coupling or singletons and should make use of more reusable code. This will help to improve its performance and minimize downtime.

- **Maintainability**

  The capability of the software product to be modified. Modifications may include corrections, improvements, or adaptions of the software to changes in the requirements. The coding practices used for developing NTUCollab should be such that they ensure the code is maintainable. Developers should maintain maintenance logs to refer to similar scenarios in the future. The software should be designed such that it is easy to modify and change as per changing requirements. Further, errors and bugs should be easy and use minimal steps to troubleshoot and resolve.

5.2.1. Standard Metrics

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

- **Mean Time to Failure (MTTF)**

  It is the average time between failures. It provides an idea of how frequent errors might occur in the software. With regards to the project NTUCollab, the SQ personnel will track the MTTF in various scenarios of the application like excessive load, parallel access of resources used by the application etc.

- **Defect Density**

  It measures the number of defects relative to the software size expressed as lines of code or function point. Simply put, it measures the code quality per unit. The SQ personnel are responsible for collecting and tracking the number of defects detected per 250 lines of code in NTUCollab and suggest amends to minimize the same.

- **Length of Code**

  The length of code is a very simple yet common metric. It is based on the notion that longer the code, the more difficult it is to understand it. The length of code can be minimized by improving reusability of the code through object-oriented practices like using functions to modularize the code, inheritance, interfaces, abstract classes etc. The SQ personnel must ensure that they frequently monitor the length of the code of NTUCollab software, detect if there is unnecessary code that can be removed and suggest ways to further keep the length of the code to a minimum. The standard unit used for the measurement of the length of code for NTUCollab is KSLOC (Thousand Source Lines of Code)

- **Problems per User-Month in UAT**

  User Acceptance Testing (UAT) is an important metric since it is used to simulate the user experience of using the application. If the number of problems detected during UAT exceeds a particular threshold, then it is a cause of worry and it is important to immediately resolve such issues as users might not adopt the application otherwise. During the UAT phase of NTUCollab, the SQ personnel must take note of all errors encountered and track the average number of errors per month. If this number is high, they should immediately notify the Project Management team, along with a detailed log of the issues faced, for effective troubleshooting.

- **Cyclomatic Complexity**

  It is an important software metric and is used to indicate the complexity of a program, by looking at the source code. It is a quantitative measure of the number of linearly independent paths in the program's source code. SQ personnel must monitor NTUCollab's source code and ensure the cyclomatic complexity is low, that is, not many parts of the program are interdependent on each other. If this complexity is high, the Project Management must be notified of the same and asked to modularize the code into independent functional sections.

- **Percent delinquent fixes**

  It is a software maintainability metric and is simply the percentage of all fixes in a time interval that are defective. It is imperative to keep this at a minimum through thorough testing of a fix. The SQ personnel are responsible for tracking the fixes in NTUCollab and relay any defective fixes to the Project Management team, as a high number of defective fixes points to a larger issue in the codebase, highlighting to many other potential defects.

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.

Software review is an important part of "Software Development Life Cycle (SDLC)" that assists software engineers in validating the quality, functionality, and other vital features and components of the software. It is a complete process that involves testing the software product and ensuring that it meets the requirements stated by the client.

Software reviews are also essential for assuring code submitted by members within the team meet a coding standard - for example have appropriate commenting, low coupling, high cohesion etc.

Following are some of our objects of carrying out software review:

- Improving the productivity of the development team
- Making the process of testing time & cost effective, as more time is spent on testing the software during the initial development of the product.
- Ensuring fewer defects are found in the final software, which helps reduce the cost of the whole process.
- Gaining sufficient reviews since the reviews provided at this stage are found to be cost effective, as they are identified at the earlier stage, as the cost of rectifying a defect in the later stages would be much more than doing it in the initial stages.
- Eliminating inadequacies that encourage defects. Elimination of defects or errors can benefit the software to a great extent. Frequent check of samples of work and identification of small-time errors can lead to low error rate.

6.2.Minimum Software Reviews

For each review, SQ 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 will 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
    - Software Specification Review
        - Ensuring the SRS performance requirements are feasible, complete, and consistent with the higher-level specification requirements
        - Ensuring all derived requirements have been identified and documented
        - Ensuring the requirements as stated are testable and measurable
        - Ensuring there are complete verifiable requirements for all performance requirements
        - Evaluating reserve capacity requirements and scenarios / procedures for measurement
        - Evaluating agreements on interfaces and boundaries
        - Evaluating results of functional analyses
        - Evaluating requirements allocation decisions
        - Evaluating identified software risks and proposed mitigation methods
        - Evaluating applicable design constraints
        - Examining the proposed software development processes

        - Examining baseline control and configuration management processes
    - Estimation, Master Schedule, and Project Plan Review

- Requirements Analysis Review
    - Reviewing the software requirement development
    - Determining whether the stated requirements are clear, complete, unduplicated, concise, valid, consistent, and unambiguous, and resolving any apparent conflicts.

- Software Design Review
    - Preliminary Design Review
        - Ensuring that the software requirements are reflected in the software architecture
        - Specifying whether effective modularity is achieved
        - Defining interfaces for modules and external system elements
        - Ensuring that the data structure is consistent with the information domain
        - Ensuring that maintainability has been considered
        - Assessing the quality factors

    - Critical Design Review
        - Assuring that there are no defects in the technical and conceptual designs
        - Verifying that the design being reviewed satisfies the design requirements established in the architectural design specifications
        - Assessing the functionality and maturity of the design critically
        - Justifying the design to the outsiders so that the technical design is more clear, effective, and easy to understand

    - Program Design Review
        - Assuring the feasibility of the detailed design
        - Assuring that the interface is consistent with the architectural design
        - Specifying whether the design is compatible to implementation language
        - Ensuring that structured programming constructs are used throughout
        - Ensuring that the implementation team is able to understand the proposed design

- Peer Reviews (EPR)
    - Code Walkthrough
    - Design Review

- Test Plan Review
- Acceptance Review

- Release Review
    - Process Audit: Final Release
- Project Closing Review
    - Ensuring all requirements from the project scope document have been met
    - External review after final delivery
    - Adding all documents together, including finalizing all project reports, then organizing and archiving them as historical data to be used for future reference.

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 and updated requirements verification matrices.

On top of this, unit testing will be used, primarily for the backend of the application. Test Driven Development (TDD) will be followed - building a test suite, and then implementing code to pass the tests. SQ personnel will be responsible for running tests and tracking what is passing, failing, and yet to be implemented.

To summarize, testing will be done in three primary ways:

1. Unit Testing

All code will be unit tested to ensure that the individual unit (class) performs the required functions and outputs the proper results and data. Proper results are determined by using the design limits of the calling (client) function as specified in the design specification defining the called (server) function. Unit testing is typically white box testing and may require the use of software stubs and symbolic debuggers. This testing helps ensure proper operation of a module because tests are generated with knowledge of the internal workings of the module.
To summarize, individual classes will be tested to ensure reliability and functionality within a unit level. Furthermore, testing module will be created before the tested code to ensure that the code is testable.

2. Integration Testing

There are two levels of integration testing. One level is the process of testing a software capability e.g., being able to send a message via a DMA port. or the ability to acquire a row of CCD data. During this level, each module is treated as a black box, while conflicts between functions or classes and between software and appropriate hardware are resolved.
A second level of integration testing occurs when sufficient modules have been integrated to demonstrate a scenario e.g., type of science mode or the ability to queue and receive commands. During this phase, composite builds, or baselines, of the software are married to the engineering versions of the hardware to evaluate the combined hardware/software performance for each operational function.
Both hardware and software documentation are reworked a s necessary.
To summarize, several classes will be tested together to ensure sufficient execution and compliance with the requirements after integration. Integration testing will be performed before beta release.

3. System Testing

The purpose of system (stress) testing is to identify the operational envelope of the instrument. The whole system shall be used for system testing to ensure all requirements is satisfied, and reliability will be included in the testing to measure successful rate of message delivery. System testing will be performed before beta release.

8. Problem Reporting and Corrective Action

SQ team collects and analyses the QA activity metrics as well as assigns, tracks, and verifies corrective actions resulting from audits, evaluation and monitoring in a centralized Reporting and Corrective Action System implemented on a Kanban board system using GitHub Projects Boards. Conditions identified during QC reviews or QA audits as needing corrective action should be addressed promptly.

The lists set up are:

1. Problems/Issues

- Issues unassigned to any SQ personnel
- Detailed descriptions of issue with a tracking label or tag
- Priority of the issue

2. In progress

- Current tasks by different SQ personnel are displayed so the others can monitor and be aware of the progress in various parts of the project.

3. Review

- Issues ready to be reviewed by other SQ personnel to check on quality.
- If issue is resolved, it would be moved to “Completed” list, if it needs more work, it would be moved to “Progress” list

4. Completed

- Resolved issues.

Furthermore, minimum documentation for any problem should include:

- The location of the failure
- The date and time of the failure
- The operation being performed
- The failure symptom
- The name and number of the reporting individual(s)
- The environment under which the failure occurred
- The impact of the failure on hardware
- For each test-related failure, a "Stress statement" on whether or not the failure induced any stress in the hardware.

A means should also be available for tabulating the failure information for determining trends and the mean-time-between-failure of the equipment

The general approach for defining corrective action requirements would involve:

- Identifying corrective action needs and causes
- Establishing appropriate corrective action responses
- Verifying the timely implementation and effectiveness of the corrective action taken.

9. Tools, Techniques and Methodologies

SQA will assure that all purchased or developed tools that directly affect the contents of the software that resides in the DPA are uniquely identified, tested/evaluated.

SQ personnel will require access to the following:

9.1. Software Quality Tools

The purpose of this section is to highlight the practices, standards, conventions, and quality.

- Microsoft Office tools (i.e., Teams, Word, Excel, and PowerPoint)
- GitHub Project Boards
- Flutter Test (flutter_test) Library - Frontend testing
- Firebase Emulator Suite - Backend testing
- Google Docs

10. Media Control

The SQ deliverables will be documented in the following software applications and forms:

1. **Microsoft Software Application – Word (MS Teams)**

   SQ deliverables will be documented in the following Microsoft software applications: Word (MS Teams). Deliverables will be in soft copy, with the exception of completed checklists from process and product assessments. Microsoft Word (part of MS Teams) tracks the document history and version changes.

2. **MediaWiki**

   Software Quality personnel will also request space on the project's MediaWiki for SQ records. This MediaWiki is password protected and backed up nightly, to ensure any disaster management any discrepancies in the documents.

3. **Project SVN**
   Further, all the SQ documents will also be uploaded and tracked on the project SVN to ensure record of the history of all the changes being made to the documents.

All these applications are password protected and allow a single source of truth for the quality control and management documents. Upon project completion, a zip folder containing the entire project document, prototype, and final product is created.

11. Supplier Control

Supplier control is not applicable in the capacity of this project, involving no suppliers associated with the delivery of the software application.

12. 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. Example records include the process and product assessments reports, completed checklists, the SQ Activity Schedule, metrics, weekly status reports, etc.

Further, 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. In

case of the electronically stored documents, internationally accepted security standards will be met.

All the standard and internationally accepted frameworks and guidelines will be followed in the preparation, collection, maintenance, and retention of the above-mentioned documents. Through version control systems, and dedicated software applications the modification history will be kept intact and ensure traceability and cross validation of all the quality procedures.

All the QA management documents will be available and accessible until 6 months after the Project completion, unless stricter rules or regulation do not state a later date. Representatives of relevant authorities and authorised personnel will be entitled to examine the project, all relevant documentation, and accounts of the project also after its closure. 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	6 months
SQA Checklists	SQ Personnel	6 months
Deliverable Defects	SQ Personnel	6 months

13. Training

All the SQ personal are highly qualified and appropriately trained in the software quality management practices and standards. The 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

14. Risk Management

SQ personnel will assess the project's risk management process and participate in weekly risk management meetings and report any software risks to the QAM and the project manager. A comprehensive and robust risk management procedure will be

followed which is clearly documented in a separate risk management plan. The salient features are briefly described here.

- **Risk Identification and analysis**

Risk will be actively identified, considering potential risks to and from project team, stake holders along with extrinsic factors, like environmental, cultural, and political factors capable of adversely impacting the project. The Project proposal, deliverables, constraints, schedule, and budget will be scrutinized to mitigate risks. Thereafter risks will be analysed in two broad categories of quantitative and qualitative risks. The likelihood and the probability if each identified risk will be calculated, into categories of low, medium, and high. Further, quantitative metrics to provide an objective numerical rating to the risk will be estimated. All these will be clearly documented in the risk management plan.

- **Risk Response Planning, Monitoring & Control**

The analysed risk will be categorized into varying level of criticality of red, yellow, and green zones. The Red and yellow risks will be assigned to individual personnel, to maintain and ensure each risk is carefully mitigated. Through framework of avoid, mitigate, accept and transfer will be used to address the above risks. The risks prioritized according to their criticality will be tracked, monitored, and reported throughout the project lifecycle. The risk impact on key project processes will be analysed and the management will be notified of any important changes to the risk assessment.

15. 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. The QAM will then review and approve any necessary changes. All these will be duly recorded, documented, and tracked. A version history will be maintained with all the changes verifiable and reverted in case of any errors or discrepancies in the records at a later stage. All the documents are enabled with password protection and stored in secured software applications, as well as version control systems.

16. Glossary

Abbreviation	Full Form
API	Application Programming Interface
CMMI	Capability Maturity Model Integration
EPR	External Peer Review
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Organization for Standardization
LOC	Lines of Code
MTTF	Mean Time to Failure
SDLC	Software Development Lifecycle
SQA	Software Quality Assurance
TDD	Test Driven Development
QAM	Quality Assurance Manager
UAT	User Acceptance Testing
UI	User Interface