Software Quality Assurance (SQA) Plan

By Team Strawhats

Date: 23rd February 2022

Signature Page

Prepared by: Acharya Atul **Date: 23rd February 2022**

Reviewed by 1: Arora Srishti **Date: 23rd February 2022**

Reviewed by 2: Tayal Aks **Date: 23rd February 2022**

Approved by: Arora Srishti **Date: 23rd February 2022**

Document Change Record

Revision	Description of Change	Approved by	Date
1	**Preliminary Version**	**QA Manager & Project Manager**	**23/02/22**

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 *RoomieLah* 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 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.

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.

RoomieLah's primary purpose is to find you your soul-roommate. Someone who shares your interests and lifestyle, would not be just a roommate but your best friend as you step into your university life.

RoomieLah is a cross platform mobile application wherein university students could create their account and fill in their roommate preferences and the traits they would be comfortable with, after which the application would dynamically suggest to them other students who they might be compatible with. They would then have an option to chat with the matched user and if needed, connect with them on social media. In the bigger picture, we want to create a one-stop solution for university students to find roommates they are compatible with and thereby improve overall living experience.

The software would be available on the Google Play Store and iOS App Store for download.

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 Plan
- System Requirement Specifications
- Software Reviews, *GeeksForGeeks*, (https://www.geeksforgeeks.org/software-engineering-software-review/)
- ISO/IEC 25010 (https://iso25000.com/index.php/en/iso-25000-standards/iso_-25010)
- Risk Management in an agile lifecycle, *Agilealliance.Org,* (https://www.agilealliance.org/wp-content/uploads/2016/01/Agile-Risk-Management-Agile-2012.pdf)

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 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 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

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 our project, RoomieLah. 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 RoomieLah as a product, the following product assessments will be conducted by SQ personnel:

- **User Interface (UI):** The UI of our application RoomieLah will be examined to make sure that the design is not overly complicated. The design should be easy to understand, and users must be able to navigate the application easily, get familiar with the interface and take advantage of all the functionalities without the need of specific usage instructions.

- **Register and Login:** The application will be assessed to gauge the ease of login for existing users, and registration and account creation for new users. The application must also only allow university students to register as users, and must be able to seamlessly communicate with the user account database tables. The process must also be completely error-free.

- **Database Communication:** The assessment of the application must also examine the communication process between the application and the database. Seamless and error-free integration, along with minimal latency are key requirements for the successful functioning of all features of the application, and SQ personnel must assess the database communication at all levels of operation to ensure integrity and thus guarantee RoomieLah's success.

- **Recommendation algorithm:** The main feature of RoomieLah's functioning is dependent on the matching algorithm of the application, which takes the users' profiles and preferences as inputs and generates potential matches for the user to go through. The SQ assessment must measure the capability of RoomieLah to be able to successfully retrieve user information from the database, capture the semantic information accurately, and generate and display profile matches for the user on demand.

- **Matching with users:** RoomieLah aims to provide the users the ability to match with other users sharing a mutual interest in each other's profiles. This aim will be fulfilled by the "swiping" functionality, where users will swipe left or right on users depending on their interest in matching with each other. The product assessment with regards to this functionality must ensure that when two users swipe right on each other's profiles, the application updates their relation to reflect a match in the database and must also check that the chat feature is subsequently enabled for these two users, with minimal processing time.

- **Chat:** The chat feature of RoomieLah is essential to facilitate hassle-free communication between matched users. SQ personnel must assess all aspects of this feature thoroughly, including the latency in message delivery, the error rate in communication between the devices, and the security of the message information and contents via encryption in the database, to ensure complete privacy of the users.

- **Scalability:** To improve the scalability of RoomieLah, the application must not use region-specific or university-specific information during profile creation or matching, unless necessary. The assessment must measure the scalability of the application in terms of the UI i.e. checking that no major modification would be

required to the screens in case of expansion to new regions, and also in terms of performance i.e. if the app has high latency during operation for the current number of users, then its use might not be feasible with an increase in the number of users or the number of features to be processed during recommendation and matching.

3.2.2. Process Assessments

To ensure high quality of RoomieLah as a product, the following product 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 RoomieLah. This should be followed by further discussions and meetings to formally finalize the requirements of RoomieLah 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:** RoomieLah 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 the project RoomieLah.

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 because 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

- System Requirement Specifications
- Project Proposal
- Project Plan
- Quality Management Plan
- Risk Management Plan
- Test Plan
- Test Cases and Test Coverage Report
- Release Plan
- Quality Management Plan
- Quality Assurance Plan
- Configuration Management Plan
- Design Report on Software Maintainability

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 product quality model defined in ISO/IEC 25010 is adopted for this project. With reference to this model, the four qualities that are deemed to be most important to *RoomieLah* are:

- Functional Suitability

This is used to measure the degree to which the application meets the explicitly stated as well as the implicit requirements. It consists of the following sub-qualities:

- Functional Completeness
- Functional Correctness
- Functional Appropriateness

- Usability

This is used to measure the degree to which the application can be used to achieve the specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. It consists of the following sub-qualities:

- Appropriateness recognizability
- Learnability
- Operability
- User error protection
- User interface aesthetics
- Accessibility

- Maintainability

This is used to measure the degree of effectiveness and efficiency with which the application can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. It consists of the following sub-qualities:

- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

- Portability

This is used to measure the degree of effectiveness and efficiency with which an application can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics:

- Adaptability
- Replaceability

Standard Metrics

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

- Number of error messages
- Fan-in/Fan-out
- Length of Code
- Cyclomatic complexity
- Defect density
- Functionality check based on specific use cases

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 the systematic inspection of a software by a team that works to find and mitigate the defects in the software during the early stages of Software Development Life Cycle (SDLC). Software review is an essential part of the SDLC that helps software engineers in validating the quality, functionality and other vital features and components of the software. It is a whole process

that includes testing of the software product and committing to meet requirements stated by the stakeholders.

Another reason why software reviews prove to be essential is that they coerce discussions on code practices within the team, for example having appropriate commenting, low coupling, and meeting software design rules.

The main objectives behind carrying out a software review with respect to our product are listed as:

1. It bolsters team productivity.

2. It makes the process of testing time & cost effective, as more time is spent on testing the software during the initial development of the product.

3. Fewer defects are found in the final software, which helps reduce the cost of the whole process.

4. The process of reviewing done early turn out to be cost effective as the cost of rectifying a defect in the later stages would be much more than doing it in the initial stages.

5. 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.

These software reviews will be assessed during software quality checks:

- Project Plan Review
    - Software Specification Review
        - Ensuring all requirements have been identified and well-documented.
        - Ensuring the stated requirements are testable and measurable.
        - Ensuring there are legitimate and verifiable requirements for all performance requirement specifications.
        - Ensuring the SRS performance requirements are feasible, complete, and consistent with the higher-level specification requirements.
        - Evaluating results of functional analyses
        - Evaluating applicable design constraints
        - Examining the proposed software development 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
        - Ensuring that the data structure is consistent with the information domain
    - 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
    - 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.

    - Peer Reviews (EPR)
        - Code Walkthrough within teammates working on the shared codebase.
        - 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.

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.

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.

In essence, 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. 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.

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.

To summarize, several classes will be tested together to ensure sufficient execution and compliance with the requirements after integration.

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 are satisfied, and reliability will be included in the testing to measure successful rate of message delivery.

4. Load Testing

The objective of Load Testing is to check how much load or maximum workload a system can handle without any performance degradation. Load helps to find the maximum capacity of the system under specific load and any issues that cause software performance degradation. Load testing shall be performed using tools like JMeter, LoadRunner, WebLoad, Silk performer, etc.

8. Problem Reporting and Corrective Action

The SQ team is responsible to collect and analyse the QA activity metrics. The team audits, assesses and monitors these metrics and assigns, tracks, and verifies corrective action which result from them. A centralized Reporting and Corrective Action System has been implemented on a Kanban Board hosted on the Github Projects Boards such that communication to all relevant team members (including the QAM and Project Manager) is taken care of. Problems identified during such reviews or audits should be addressed promptly by the team.

The following pipeline has been set up for efficient problem reporting and corrective action:

1. Defining the Problem

a. After identification of the problem, it must be clearly defined, and a detailed description should be attached to avoid confusion in the future.

b. Any unassigned problem should be assigned to one or more SQ personnel.

c. The urgency/priority of the identified issue should also be explicitly mentioned at the beginning.

2. Tracking the Solution

a. Assigned problems/tasks are displayed in one place for the team to monitor.

b. The entire process should be transparent, and the assigned personnel should keep updating this list whenever progress has been made.

3. Reviewing the Solution

a. The assigned SQ personnel should inform the team when the solution is ready and ready to be reviewed by the team.

b. The team, upon reviewing, then discusses and deliberates on the corrective action.

c. If resolved, the problem is moved to the Completed list.

d. If not, it stays in the Tracking the Solution list with necessary updates.

4. Completed

a. This contains a list of issues that have been dealt with already and necessary corrective action has been taken by the team.

A table like structure should also be implemented to keep track of the failures as it would help in identifying trends and statistics like ‘mean time between failures’ which would help assure better quality of the software.

9. Tools, Techniques and Methodologies

SQA should assure that tools, either purchased or developed, which could directly affect the software or its contents, are uniquely identified and tested.

SQ personnel will require access to the following:

9.1. Software Quality Tools

The purpose of this section is to list all the tools/software used by team Strawhats to maintain good practice, uphold good standards and to deliver a product of high quality.

- Microsoft Office tools (i.e., Word, Excel, and PowerPoint)
- MediaWiki
- Github
- Google Docs
- Draw.io - UML Diagrams
- Figma - Application Prototypes
- Flutter test (flutter_test) Library - Frontend testing
- Firebase Emulator Suite - Backend testing

10. Media Control

SQ deliverables will be documented in the following software applications:

1. Microsoft Software Application – WORD (MS TEAMS)

SQ Deliverables will be documented in the following Microsoft applications: Word (MS Teams). Deliverables will be in soft copy except for completed checklists from process and product assessments. Microsoft Word (part of MS Teams) tracks the version history and all changes to the document.

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 every night to ensure disaster management and discrepancies in the documents.

3. Project SVN

All SQ documents will also be uploaded and tracked on the project SVN to ensure record the history of all changes being made to the documents.

11. Supplier Control

Supplier Control is not applicable for the project.

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. 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 electronic documents, internationally accepted security standards will be met. All the standard and internationally accepted frameworks and guidelines will be followed for the preparation, collection and maintenance and retention of the above-mentioned documents. Using version control systems and dedicated software applications, the modification history will be visible to ensure traceability and cross-validation of all quality procedures. All QA management documents will be available and accessible until 1 year after the completion of the project unless stricter rules or regulations do not state a later date. Representatives of relevant authorities and authorized personnel will be entitled to examine the project, documents, and accounts of the project 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	One Year
SQA Checklists	SQ Personnel	One Year
Deliverable Defects	SQ Personnel	One Year

13. 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

14. 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. Additionally, a risk register will also be maintained which will entail the following details:

Description of risk — Summary description of the risk—easy to understand.

Recognition Date — Date on which stakeholders identify and acknowledge the risk.

Probability of occurrence — Estimate of probability that this risk will materialize (%).

Severity — The intensity of undesirable impact to the project—if the risk materializes.

Owner — This person monitors the risk and acts if necessary.

Action — The contingent response if the risk materializes.

Status — current team view of the risk: potential, monitoring, occurring, or eliminated.

Loss Size — Given in hours or days, this is a measure of the negative impact to the project.

Risk Exposure — Given in hours or days, this is a product of probability and loss size.

Priority (optional) — This is either an independent ranking, or the product of probability and severity. Typically, a higher-severity risk with high probability has higher relative priority.

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. The SQ personnel will also maintain a version history to track and document the changes.