`

Foodify

Software Quality Assurance (SQA) Plan

By Runtime Terror

Date: 22/09/2022

Signature Page

Prepared by: Kolady Anamika Martin	**Date: September 17 2022**
Prepared by: Verma Nandini	**Date: September 17 2022**
Reviewed by 1: Chua Jim Sean	**Date: September 22 2022**
Reviewed by 2: Duong Ngoc Yen	**Date: September 22 2022**
Approved by : Anandarajan Sindini	**Date: September 22 2022**

Document Change Record

Revision	Description of Change	Approved by	Date
0.10	**Initial Draft**	**Anandarajan Sindini**	**22/09/22**
0.20	**Final draft**	**Verma Nandini**	**21/10/2022**

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 Foodify 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 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. 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 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. 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 fulfills the purposes that the stakeholders intended for them.

Foodify is a web application that allows its users to track the expiry of their food articles to reduce food wastage. Registration is simple and hassle-free using an email address. A registered user can add a food item to his/her account. The expiry date of the item will be required for registration as well. When a food item has been consumed fully, the user can update to delete that food item from his/her account. While the food item is unconsumed and unexpired, the user will receive email notifications and suggestions on how to cook and consume it. Alternatively, the user can sell a food item in the web application's Marketplace page to other users who are interested in purchasing it for a negotiable price. Once the food item has expired, it will be deleted from the user's account. Using the different features of our Foodify web application, users can better monitor and save their food from being wasted and contribute towards sustainability.

Foodify's team consists of five members who have been selectively assigned roles based on their expertise, to oversee the success of their respective department. Some members 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 holds proficiency and demonstrated experience in frontend development while overseeing other aspects of the project like backend. The Backend Developer is dedicated for the application development, one Quality Assurance manager to oversee the different released versions of Foodify. The QA Manager also assumes the role of Release Engineer. Lastly, we have one Quality Assurance Engineer.

Foodify will be developed using React JS library and VSCode editor. 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 React JS, Firebase and Firestore, User Interface prototyping tools like Figma, versioning system (e.g., GitHub, GitHub Actions etc.) and the Integrated Development Environment (IDE) used (e.g., VSCode etc.) for the development of Foodify.

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://drive.google.com/file/d/1UrlvrUhPzOXA7V5Qu6Zv1wWvM_Yy9YXz/view?usp=sharing)
- System Requirement Specifications (https://drive.google.com/file/d/1Cg2jFAuaWHIAe_pdZJs2pdI-yOfPpJ-n/view?usp=sharing)
- Project Plan (https://drive.google.com/file/d/12HsSjnkHaIpRkSfJX_9foEPu21dU4xM2/view?usp=sharing)
- 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)

3. Management

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

3.1.Management Organization

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 time scale 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

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

- Ensuring 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

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. 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 analyzes the reports and findings of other SQ personnel and after deliberations, will compile all the findings. Like any other project, Foodify 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 Foodify. 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 Foodify as a product, the following product assessments will be conducted by SQ personnel:

- **User Interface**

  The UI of Foodify will be assessed to ensure it is simple yet engaging and pleasing to look at. The interface should be self explanatory, intuitive 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**

  Foodify must be able to support simple 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 Foodify 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 Foodify's success.

- **Posts**

  Foodify 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 for selling/purchasing food items on the Marketplace page 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.

3.2.2. Process Assessments

The following process assessments will be conducted by SQ personnel:

- Requirement Management
- Project Management
- Project Monitoring and Control
- Risk Management
- Test Management

- Release Management
- Software Development

The processes listed above will be assessed by SQ personnel during scheduled internal audits made known beforehand to parties involved as well as software reviews attended by the SQ personnel. The processes will be assessed using the IEEE STD 730-2002, IEEE Standard for Software Quality Assurance Plans. The assessments will then be reviewed by the QA Manager and QA team.
An issue will be logged if there is a problem without a visible plan for resolution. Once a list of issues has been compiled, it will be reviewed with the project manager to see if any new or additional information might mitigate or eliminate any of them. Issues identified as well as the recommended corrective actions to be taken will be handled as specified in section 8 of this document titled "Problem Reporting and Corrective Action".

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
- Assess operational data and identify quality problems
- Develop quality control policies and standard for the rest of the team to follow, after discussions with the SQ personnel

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

This section highlights the standards, practices, quality requirements, and metrics to be applied to ensure a successful software quality program. It highlights the practices, standards, conventions, and quality metrics to ensure that the software produced is 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 Foodify is a fully functional web application and works as expected by the user. It should satisfy all functional requirements and should be traceable if any error arises.

• **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 Foodify has a simple and intuitive user interface, so that the user is not overwhelmed. The users of our web application will use 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. Foodify 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 adaptations of the software to changes in the requirements. The coding practices used for developing Foodify 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 the area of software quality assurance:

- **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 Foodify 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 the 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 Foodify 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 Foodify, 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.

- **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 Foodify, 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.

- **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 Foodify'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 Foodify 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**

o 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

o Estimation, Master Schedule, and Project Plan Review

• Requirements Analysis Review

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

• Software Design Review

o 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

o 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

o 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)

o Code Walkthrough

o Design Review

• Test Plan Review

• Acceptance Review

• Release Review

o Process Audit: Final Release

• Project Closing Review

o Ensuring all requirements from the project scope document have been met o External review after final delivery

o 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, updated requirements verification matrices, etc.

8. Problem Reporting and Corrective Action

Some foreseeable problems which may arise in the course of the software development are as follows:

1. Documentation
    - 1.1. Incompleteness/ Lacking essential content of documentation
    - 1.2. Incorrect information in documentation
    - 1.3. Error in documentation
    - 1.4. Outdated documentation
    - 1.5. Formatting issues
2. Development
    - 2.1. Incorrect functionality
    - 2.2. Lack of functionality
    - 2.3. Inconsistency with planned design

SQ personnel generate, track, and trend assessment findings and observations in a centralized Reporting and Corrective Action System which is maintained in an Excel sheet located in the shared Google Drive project folder.

In general the following approaches will be taken for corrective action:

1. SQ personnel will conduct regular and scheduled checks and reviews to discover and identify potential problems.
2. Upon discovering a problem with the project, SQ personnel or members will be required to inform the Project Manager and QA Manager.
3. Project Manager and QA Manager will review the problem and determine the severity of the problem.

4. Depending on the severity of the problem, relevant members in the project group will be notified of the problem, and meetings will be held if necessary.
5. Corrective action will be determined through discussion.
6. The problem and corrective action to be taken will be documented down in the Problem Reporting and Corrective Action Document Excel Sheet located in the shared Google Drive project folder.
7. Team members will be appointed to implement the corrective action by a given deadline.
8. Assessment data and corrective action status will be communicated to the QA Manager during the weekly regular updates or the monthly risk management meetings.
9. Once corrective actions have been implemented, SQ personnel/ SQ Manager will review the corrective actions. If it is not implemented correctly or not up to standard, the team will revert back to Step 3 for review again.
10. If corrective actions are satisfactory, the problem case will be closed and logged down in the Excel Sheet in the shared Google Drive project folder.

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 (For collaboration)
- Visual Paradigm (Use Case Diagrams)
- Visual Studio Code (Appropriate IDE)
- Firebase (Backend Service)
- Figma (UI Prototyping)
- Github (Code Collaboration Platform for Version Control)
- Chromium Browser (Modern Web Browser)

10. Media Control

SQ deliverables will be documented in one of the following Microsoft software applications: Word, Excel, or PowerPoint. Deliverables will be in soft copy, with the exception of completed checklists from process and product assessments. See Section 12 for additional details on the collection and retention of key records. Software Quality personnel will request space on the project's secured server for SQ records. This server is password protected and backed up nightly.

The following services are used for this project for media control

- MediaWiki
- Git and GitHub
- Zoom
- When2meet

- Telegram
- Google Drive

MediaWiki is a free open-source service which is easy to use. There are many online resources available to teach users the various functions available to create information in different styles. It also allows for collaboration and concurrent editing of information by different users.

Git is an open-sources version control system first developed in 2005. It is installed on your local system and also allows for branching. GitHub is a cloud-based Git repository hosting service, which allows sharing of codes, thus allowing for collaboration and making revisions and edits.

Zoom is a cloud-based video conferencing tool that lets you host virtual one-on-one or team meetings easily. With powerful audio, video and collaboration features, this remote communication tool connects remote team members with each other.

Telegram is a globally accessible freemium, cross-platform, cloud-based instant messaging service. The service also provides optional end-to-end encrypted chats and video calling, VoIP, file sharing and several other features. It was used as the primary mode of communication for updating the team, sharing suggestions and clarifying queries.

When2meet helps in finding the best time for a group to meet. It is compact, so that events can be created and accessed quickly. It is a free service, anyone can create and participate in availability surveys at no cost. We used it for scheduling meetings in a hassle-free fashion.

Google Drive is used for collaboration for all the documentations, as it has a built-in office suite (Google Docs, Google Sheets, etc), which also allows for real-time collaboration and version control. It is also used for file storage and file sharing.

11. Record Collection, Maintenance, and Retention

Throughout the Foodify software development life cycle, SQ personnel will maintain records that document assessments performed on the project. These documents will be constantly maintained and updated by the SQ personnel while checked and approved by the QA manager.

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 weekly risk management meetings and report any software risks to the QAM and the project manager.

With regards to this Software Quality Plan, the QA team has identified some possible risks, the probability, seriousness, consequences and the risk management strategy for the Foodify Project:

Risk Type	**Risk**	**Probability**	**Seriousness**	**Impact**	**Strategy**
Resource	Unable to meet Project Deadline	Moderate	High	Reduced functionalities or delay the deployment date.	Have regular meetings and updates from the developer team to check on their progress.
Hardware	Hardware failure e.g., Hard Drive corrupted	Low	High	Unable to meet deadlines as codes cannot be written or run.	Maintain backups via version control software, e.g., git to minimize the impact.

Software	Feature creep	Moderate	High	Will delay the project and may not be able to meet the set deadlines.	Maintain traceability between documentation and code to not go beyond the specified requirements.
Communication	Updates to code not communicated to every group member	High	Moderate	Overlapping of codes which may lead to errors while running the program, and documentation may be inaccurate.	Ensure transparency between group members when it comes to updating of work done, which will lead to less miscommunication and conflicting information.
People	Time mismanagement (Due to underestimation of the complexity of the project or overconfidence of capabilities.)	High	High	Unable to meet deadlines as codes are not completed or not	Conduct regular meetings to update on improvement in work and to ensure timeliness.

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.