**Smart-glass based**

**Remote Guidance System**

**SOFTWARE QUALITY ASSURANCE PLAN**

Group 21

Lyndon Prado - 9740783

Tingcong Jimmy Li - 100017000

Keagan Foster - 101609822

Ayub Khan - 100667654

Dineth Gunawardena - 100862158

Kosala Edirisinghe - 101265981

Krishna Adhikari - 4953193

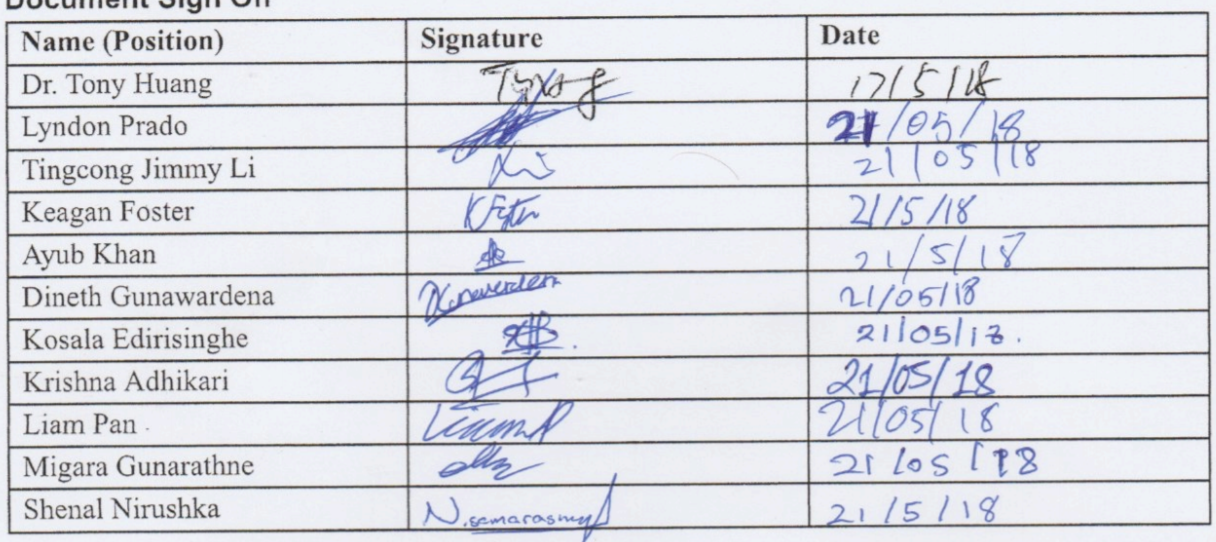
Liam Pan -101106174

Migara Gunarathne - 101002269

Shenal Nirushka – 101054998

**Table 1. Document Change Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** |  | **Date** | **Author** | **Comments** |
| 1.00 |  | 21/03/2018 | All Authors | Created initial draft document. |
| 1.10 |  | 13/04/2018 | All Authors | Updated based on feedback from external reviews. |
| 1.20 |  | 23/04/2018 | K. Adhikari | Updated based on internal review and Supervisor feedback. |
| 1.21 |  | 13/05/2018 | Liam Pan | Document Review |
| 1.30 |  | 24/05/2018 | Krishna Adhikari | Formatted, reviewed and edited document to make it ready for submission |

**Table 2. Document Sign Off**

**Acronyms/Abbreviations**

**ASAP** As Soon as Possible

**COB** Close of Business (5 pm)

**SGBRG**: Smart Glass Based Remote Guidance

[1. Introduction 5](#_Toc514948853)

[1.1. Author List/Roles 5](#_Toc514948854)

[1.2 Purpose 5](#_Toc514948855)

[2. Reference Documents 6](#_Toc514948856)

[3. Management 7](#_Toc514948857)

[3.1. Organisation/Roles 7](#_Toc514948858)

[3.1.1. Meeting Roles 7](#_Toc514948859)

[3.1.2. Formal Review Meeting Roles 7](#_Toc514948860)

[3.1.3) Champion Roles 8](#_Toc514948861)

[3.1.4. Communication Roles 9](#_Toc514948862)

[3.2. Tasks and Responsibilities 9](#_Toc514948863)

[3.2.1. General Team Member Responsibilities 9](#_Toc514948864)

[4. Documentation 12](#_Toc514948865)

[4.1. Software Documents 12](#_Toc514948866)

[4.1.1. SQAP 12](#_Toc514948867)

[4.1.2. SRS 12](#_Toc514948868)

[4.1.3. Project Plan 12](#_Toc514948869)

[4.1.4 SDLC 12](#_Toc514948870)

[4.1.5. Module Plan 13](#_Toc514948871)

[4.1.6. Self-assessment reports 13](#_Toc514948872)

[4.1.7. Audit Report 13](#_Toc514948873)

[4.2. Management Documents 14](#_Toc514948874)

[4.2.1. Meeting Agendas 14](#_Toc514948875)

[4.2.2) Meeting Minutes 14](#_Toc514948876)

[5. Standards, practices, conventions and metrics 15](#_Toc514948877)

[5.1. Purpose 15](#_Toc514948878)

[5.2. Standards 15](#_Toc514948879)

[5.2.1. Coding Standard 15](#_Toc514948880)

[5.2.2. Documentation Formatting Standard 15](#_Toc514948881)

[5.2.2.1. Document Structure 15](#_Toc514948882)

[5.2.2.2. Title Page 16](#_Toc514948883)

[5.2.2.3. Table of Contents 16](#_Toc514948884)

[5.2.2.4. Normal Text 16](#_Toc514948885)

[5.2.2.5. Titles, Headings and Subheadings 16](#_Toc514948886)

[5.2.2.6. Tables 17](#_Toc514948887)

[5.2.2.7. Images 17](#_Toc514948888)

[5.2.2.8. Abbreviations and Acronyms 17](#_Toc514948889)

[5.2.3. Filename/Location standards 17](#_Toc514948890)

[5.2.4 Git Standards: 18](#_Toc514948891)

[5.2.5. Document Releases 18](#_Toc514948892)

[5.3 Practices 18](#_Toc514948893)

[5.3.1. Communication Practices 19](#_Toc514948894)

[5.3.1.1. Client 19](#_Toc514948895)

[5.3.1.2. Team 19](#_Toc514948896)

[5.3.1.3. Supervisor 19](#_Toc514948897)

[5.3.2. Meetings 20](#_Toc514948898)

[5.3.3. Worklogs 20](#_Toc514948899)

[5.3.4. Git 20](#_Toc514948900)

[5.3.5. Coding practices 20](#_Toc514948901)

[6. Reviews and Audits 22](#_Toc514948902)

[6.1. Purpose 22](#_Toc514948903)

[6.2. Review/Audit list 22](#_Toc514948904)

[6.2.1. Reviews 22](#_Toc514948905)

[6.2.1.1. Formal Review Process 22](#_Toc514948906)

[6.2.1.2) Informal Review Processes 23](#_Toc514948907)

[6.2.2. Audits 24](#_Toc514948908)

[7. Testing 25](#_Toc514948909)

[7.1. Requirements 26](#_Toc514948910)

[7.2. Use Case Generation 26](#_Toc514948911)

[7.3. Installation and User Documentation Generation 26](#_Toc514948912)

[8. Problem reporting and corrective action 27](#_Toc514948913)

[8.1. Personnel 27](#_Toc514948914)

[8.2. Work 27](#_Toc514948915)

[8.2.1. Project major timeline 27](#_Toc514948916)

[8.2.2. Stage-dependent tasks 27](#_Toc514948917)

[8.2.3). Crossed-states tasks 27](#_Toc514948918)

[8.2.4. Task creation 28](#_Toc514948919)

[8.2.5 Task assignment 28](#_Toc514948920)

[8.2.6 Task life 28](#_Toc514948921)

[8.2.7. Issue Categories 28](#_Toc514948922)

[9. Tools and methodologies 30](#_Toc514948923)

[9.1. Tools 30](#_Toc514948924)

[9.1.1. Google Docs 30](#_Toc514948925)

[9.1.2. Slack 30](#_Toc514948926)

[9.1.3. Issues tracking (Jira) 31](#_Toc514948927)

[9.1.4. GitHub 33](#_Toc514948928)

[9.1.4. Skype 33](#_Toc514948929)

[9.1.5. Qt 33](#_Toc514948930)

[9.2. Design Methodology 34](#_Toc514948931)

[10. Records collection, maintenance and retention 35](#_Toc514948932)

[11. Risk Management 36](#_Toc514948933)

[11.1. Purpose 36](#_Toc514948934)

[11.2. Categorization 36](#_Toc514948935)

[11.2.1) Risks with respect to the work to be done 36](#_Toc514948936)

[11.2.2. Risks with respect to the management 37](#_Toc514948937)

[11.2.3. Risks with respect to the client 38](#_Toc514948938)

# 1. Introduction

## 1.1. Author List/Roles

**Table 3. Team Roles**

|  |  |  |
| --- | --- | --- |
| **Author** | **Student ID** | **Role Semester 1** |
| Lyndon Prado | 9740783 | Team Leader / Client Liaison |
| Keagan Foster | 101609822 | Team Leader |
| Krishna Adhikari | 4953193 | Git/ Jira Champion/ Sergeant At Arms |
| Tingcong Jimmy Li | 100017000 | Business Analyst/Team Leader |
| Ayub Khan | 100667654 | Testing Champion |
| Liam Pan | 101106174 | Documentation Champion |
| Shenal Nirushka | 101054998 | Usability Champion |
| Migara Gunarathne | 101002269 | Coding Champion |
| Kosala Edirisinghe | 101265981 | Coding Champion |
| Dineth Gunawardena | 100862158 | Coding Champion |

## 1.2 Purpose

This document outlines the policies and procedures that members of Team 21 will follow to achieve an overall high standard of quality for Smart Glass Based Remote Guidance(SGBRG) for Dr. Tony Huang. All team members are expected to adhere to the processes outlined in this document.

# 2. Reference Documents

* Institute of Electrical and Electronics Engineers (IEEE) Std 730-1998, IEEE Standard for Software Quality Assurance Plans
* IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
* Swinburne Java Coding Standard - SwinBrain

# 3. Management

## 3.1. Organisation/Roles

The following list contains currently identified Roles:

### 3.1.1. Meeting Roles

**Chair**

Performed by the Team Leader and is responsible for running/controlling the meeting as well as determining and distributing the agenda for the meeting. Apologies and items to be added to the agenda from other team members are to be sent to the Chair prior to meeting.

**Sergeant at Arms**

Keeps an eye on proceedings and time spent on each item. Responsible for keeping the meeting on topic and on time. Their conduct must ensure that team members still can voice their opinions and ideas. Additional time can be utilized for discussion outside of the arranged meeting time.

**Scribe**

Records the meeting minutes and is responsible for circulation of meeting minutes. This role will be rotated through team members.

### 3.1.2. Formal Review Meeting Roles

**Moderator**

Plans the review and coordinates the review process

**Scribe**

Documents any issues or problems found during the review

**Inspector**

Examines the document/product for defects

**Author**

The creator of the work being reviewed

**Reader**

Reads out documents for the inspectors.

### 3.1.3) Champion Roles

Champion is a role that is directly responsible for ensuring the quality of a particular area assigned to them, as well as the compliance of standards and procedures of their section’s of the project. They are not however responsible for completing majority of the works. They are to delegate work, assist where appropriate and are the single point of contact for issues.

**Team Leader** is responsible ensuring the team works effectively together to achieve successful completion of the project. The Team Leader typically chairs team meetings and is the supervisor liaison. They should be notified of any political issues within the team. The team leader is also responsible for monitoring the completion of work logs to ensure members are recording time spent and contributing. They are also responsible for writing status reports where applicable and attending team leader meetings.

**Documentation Champion** in in charge of making sure all documentation is consistent, complete and to a high standard. This includes Latex formatting and filenames.

**Git Champion** is responsible for ensuring that the repository is used to its full potential and enforcing the Git standards. They are also responsible for maintaining directory structure and resolving any issues with the repository.

**Jira Champion** is responsible for maintaining the Jira bug tracker and ensuring tasks are closed out once completed. They should encourage members to complete their allocated tasks. They work with the business analyst to create stories and tasks as per the requirement. They also work closely with the project manager to track the progress of tasks. They enforce certain workflow and standard that everyone in the team must follow.

**Usability Champion** needs to ensure the product meets usability related non-functional requirements. They should understand the Client Requirements and ensure that any products are built with the client use in mind. They should assist with testing to ensure the GUIs are usable and intuitive.

**Code Champion** is responsible for quality and on schedule delivery of code. They are to ensure other members are building and uploading code that compiles and is up to standard. They are responsible for ensuring delegated works are completed in a timely manner.

**Testing Champion** is responsible for running and reporting on test results. This includes unit and functional testing. They should work with the Usability Champion to ensure that exceptions and errors are understandable and informative. They are to ensure tests carried out by other members are captured and recorded. Testing champion will encourage other team members to create unit tests for their own sections. Testing Champion is responsible for ensuring the following tests are done correctly tests:

1. **Unit testing/module testing/component testing**
   1. verifies that the component functions properly with the types of input expected from studying the component’s design.
2. **Integration Testing**
   1. the process of verifying that the system components work together as described in the system and program design specifications.
3. **Function testing** 
   1. does the integrated system perform as promised by the requirements specification?
4. **Performance testing** 
   1. are the non-functional requirements met?
5. **Acceptance test** 
   1. is the system what the customer expects?
6. **Installation test**
   1. Does the developed app work on the smart glasses as expected?

### 3.1.4. Communication Roles

**Client**

The Client Liaison acts as the single point of contact between the team and the client. This allows coordination of all incoming and outgoing correspondence with the client and distribution to all team members. They are also responsible for setting up regular Client Meetings.

**Supervisor**

The Supervisor Liaison allows the University and Supervisor to have a single point of contact for the team. However, other team members may contact the Supervisor for issues themselves. The role is typically filled by the Team Leader.

## 3.2. Tasks and Responsibilities

### 3.2.1. General Team Member Responsibilities

* If a team member is selected for a task they will complete the task by the allocated time. If unable to complete task in time, member is to raise an issue prior to deadline with the team leader.
* Meeting Actions are binding unless changed at a later meeting.
* Team members are responsible for the logging of their own time sheets.
* Members are to conduct themselves in an appropriate manner facilitating a healthy work environment.
* Members are required to maintain communication with team.
* Members are required to follow all processes as described in the SQAP.
* Members must make their best effort attend all allocated meetings/workshops and are to submit an apology if they are unable to attend.
* Members are to follow all directives from champions.
* Members are to actively partake in group discussion and provide input to the product and the process.

3.2.2) Champions

**Team Leader**

* Responsible for the running of weekly team meetings.
* Responsible for the booking of weekly meeting room.
* Responsible for maintaining administrative documentation.
* Responsible for motivating and tracking team progress.
* Responsible for monitoring work logs and time spent on project.
* Responsible for being a point of contact for issues/resolution.
* Responsible for liaising with the supervisor

**Client Liaison**

* This champion is directly responsible for all communications with the client.
* Correspondence from team members must be relayed to/from client in a timely manner via client liaison.
* Minor/non-urgent communications are to be collated to avoid bombarding client.
* Any compiled versions that need to be tested will be sent via the liaison.
* Liaison is responsible for ensuring client receives all relevant information for a test, as well as distribution of the test results supplied by the client to the team (bidirectional communication).

**Usability and Graphical User Interface (GUI)**

* Responsible for ensuring team members take usability into account during development.
* Responsible for ensuring consistency of GUI
* Responsible for ensuring that performance does not negatively impact usability.

**Documentation**

* Responsible for creating and maintaining document templates.
* Maintaining quality and standards of documents.
* Responsible for helping with documentation issues.
* Responsible for maintaining documentation tools.

**Code**

* Responsible for quality control of code artifacts.
* Responsible for tracking code progress
* In charge of organising developer meetings to discuss progress and address any difficulties or concerns.
* In charge of ensuring appropriate workload is assigned for each developer.
* In charge of ensuring standards and best practices are met and followed, respectively, during the development process.

**GIT**

* Creating and maintaining repository.
* Monitoring commit messages.
* Maintaining file structure and location standards.
* Helping with branching and merging.

**Jira**

* Creating and maintaining Jira tickets
* Closing out resolved issue
* Monitoring progress of members with issues/tasks
* Create and maintain standards

**Testing**

* Completing or delegating running of tests
* Reporting results of tests to team and on Jira
* Building test documents
* Encouraging testing within the team and compile results

# 4. Documentation

## 4.1. Software Documents

### 4.1.1. SQAP

The SQAP is a plan written before any development that outlines all standard practices and procedures to ensure a quality process therefore help produce a high-quality product.

### 4.1.2. SRS

A Software Requirements Specification will be developed to describe the behavior of the proposed SGBRG as derived from client requirements. The SRS will be based on the IEEE 830 standard but will be modified to make it appropriate for our project.

A general outline of the document is as follows:

1. Introduction
   1. Purpose - Outline of the SRS
   2. Scope - What the DMS is, what it will do and its application
   3. Definitions, acronyms, and abbreviations
   4. References
   5. Overview
2. Overall description
   1. Product perspective - How the product works with other tools, i.e dataflow
   2. Product functions
   3. User characteristics - Who will use the product, what training will they get
   4. Constraints
   5. Assumptions and dependencies
3. Specific requirements

### 4.1.3. Project Plan

A document to guide the building of the product. It will include a brief description of the project and why it should be built, what needs to be done to build the software and a timeline for when modules should be complete.

As more modules are mapped and more details are known about each item, the project plan is to be updated. It should also contain milestones where applicable and deliverable dates.

### 4.1.4 SDLC

A document with the reasoning behind why Scrum was chosen as preferred methodology. It talks about decision process, application of this methodologies along with the advantages, disadvantages and risks associated to it.

### 4.1.5. Module Plan

Module plan is to be developed for each sprint. There will be a sprint planning meeting before starting each sprint.

Module plan contains:

* Module scope
* Design solution for module
* Work division
* Timeline of artifacts and testing

### 4.1.6. Self-assessment reports

A self-assessment report is to be completed by each team member each as per the unit outline that will provide evidence of work completed and self-reflection. It will document knowledge and experience that has been gained during the process.

This document should contain the following sections

* Summary
* Work completed
* Mistakes made
* Knowledge gained
* Evidence

### 4.1.7. Audit Report

Whenever an audit is carried out a document must be produced that indicates the outcome; anything that does not follow the processes outlined in this document SQAP, whether the process is followed and corrective actions.

Audits can be carried out internally and externally.

## 4.2. Management Documents

### 4.2.1. Meeting Agendas

* This document will be of Google Doc/Word document type and will be prepared by the Team Leader (Or Meeting Chair) prior to each meeting.
* All team members are expected to contribute to the agenda by requesting topics of their choice be added to the agenda.
* Topics shall be owned by the team member who submitted them, unless otherwise stated.
* Owners shall attend the meeting prepared to introduce and discuss their topic.
* Submissions will be accepted by Close of Business (5:00 PM) (COB) the day prior to the meeting.
* The author of the agenda will upload to the Google Drive by COB the day before the meeting. A reminder email will also be sent to the required attendees with a link to the google doc file of the agenda.

### 4.2.2) Meeting Minutes

* Will be collected at every meeting.
* Must follow the minutes template as outlined in the meetings folder on the google drive.
* A copy of the template will be filled in during the meeting and will be exported to word document for formatting.
* Formatted minutes will be released on the Google Drive no later than the following day’s COB.
* A PDF copy of the minutes can also be emailed if requested, however, members are expected to find the minutes on the Google Drive and complete their actions independently.

# 5. Standards, practices, conventions and metrics

## 5.1. Purpose

The following standards will be maintained to ensure the software quality. The following section will cover the standards used to govern the development and management of the project.

This section includes the guidelines that the development team will follow. The quality of the project's deliverables will be measured and ensured using the following standards and practices.

## 5.2. Standards

The standards in this document will be used to control the quality of the project. They will be followed always through the duration of the software life cycle.

### 5.2.1. Coding Standard

The following language-specific standards are used:

* Swinburne Java Standards - <https://swinbrain.ict.swin.edu.au/wiki/Swinburne_Java_Coding_Standard>
* Swinburne C# and .Net standards -

### 5.2.2. Documentation Formatting Standard

#### 5.2.2.1. Document Structure

Each document shall contain:

* Title Page
* Document Change Control
* Document Sign Off
* Table of Contents
* Report Body

As most documents will be stored on Google Drive, each document should be created using Google Docs, and all text settings will refer to what is used in this setting. All documents shall be reformatted accordingly if they have been created outside of Google Docs for any reason.

#### 5.2.2.2. Title Page

The title page shall contain:

* Unit Number
* Unit Name
* Latest Revision Date
* Project Name
* Document Title

#### 5.2.2.3. Table of Contents

The table of contents shall refer to every heading and subheading of the document without fail.

* Font: Times New Roman
* Font Size: 12

#### 5.2.2.4. Normal Text

Under most circumstances, Normal Text shall be formatted as follows:

* Font: Arial
* Font Size: 11
* Font Colour: Black
* Line Spacing: 1.15

#### 5.2.2.5. Titles, Headings and Subheadings

Document titles shall be formatted as follows:

* Categorised under “Title” in Google Docs.
* Font: Times New Roman
* Font Size: 30
* Font Colour: Black
* Bolded

Document Subtitles (if present) shall be formatted as follows:

* Font: Times New Roman
* Font Size: 20
* Font Colour: Black

All Headings of the first layer (e.g. Section 3) shall be formatted as follows:

* Categorised under “Heading 1” in Google Docs.
* Font: Times New Roman
* Font Size: 20
* Font Colour: Black
* Bolded

All Subheadings of the second layer (e.g. Section 3.1) shall be formatted as follows:

* Categorised under “Heading 2” in Google Docs.
* Font: Times New Roman
* Font Size: 16
* Font Colour: Black

All Subheadings of the third layer (e.g. Section 3.1.1) shall be formatted as follows:

* Categorised under “Heading 3” in Google Docs.
* Font: Times New Roman
* Font Size: 14
* Font Colour: Grey

Subheadings of the fourth layer onwards shall be formatted as follows:

* Categorised under “Heading 4” in Google Docs
* Font: Times New Roman
* Font Size: 12
* Font Colour: Light Grey

#### 5.2.2.6. Tables

Tables shall be formatted as follows:

* Every table shall be assigned a heading, which consists of a table number and a label that clearly describes the contents. This heading shall be bolded but is otherwise formatted as the following.
* The heading of each column of the table shall be bolded.
* Font: Times New Roman
* Font Size: 11
* Font Colour: Black

#### 5.2.2.7. Images

If required, image sets shall span the width of the page so as to prevent text from wrapping around it. Each image shall be labeled with a number (e.g. Figure 1) and a short description.

#### 5.2.2.8. Abbreviations and Acronyms

Abbreviations should be avoided under most circumstances. If needed, they can be used, but it is not preferable.

On the first reference of an acronym, it shall be spelled out fully such that any future reference of the acronym can be expected to be understood by the reader

### 5.2.3. Filename/Location standards

* All file and folder names will be in camel case with the first letter of every word in uppercase
* There will be no white spaces in the file names
* In the case that the filename consists of more than one word “\_” will be used in between each word.
* Management/Administration files will be named in the following format “filename yyyy mm dd”.



### 5.2.4 Git Standards:

* Always pull from either develop or the branch you are in before you start coding.
* Always create a new branch when you are working on a feature. One branch should only contain codes related to one feature.
* Always branch out from develop to ensure that you get the latest copy of another members work. This reduces merge conflict significantly.
* Always create a pull request to merge back to develop once you have completed development and add other members for code review.
* Use git diff or git add -p to make sure you are not pushing anything unnecessary
* Keep the branch names and commit message meaningful. krishna\_branch might not provide a lot of information about the branch is but feature/hand-gesture-swipe-right does.
* Commit more often and small section of codes. Only one issue needs to be addressed in one commit.
* Never push anything to develop or master or somebody else’s branch unless the need.
* Do not work on multiple branches/feature at once. This will create merge conflict.
* If there is a merge conflict, try to solve it locally. If unsure, do not push it to the cloud.
* While reviewing, check the code difference in the GitHub, pull the repo locally and run the code and verify everything is working as intended. Check if any unit test is failing if everything is all good, then only approve the pull request.
* A pull request can be merged only if all reviewer have approved and any pull request must have at least 2 reviewers.

### 5.2.5. Document Releases

If a document is released to an outside party; be it submission to the university or the client, it must be:

* Converted into a static format such as a pdf
* Appropriately renamed with a revision number
* Moved to a release folder within its current folder

Each release will be named as follows: filename rxxx, where xxx is a number. The first release will be 100 and each additional release will be incremented by 10 i.e. filename r100 will be followed by filename r110.

## 5.3 Practices

The practices listed below will be used for quality control purposes. All the listed practices shall be followed throughout the whole software development life cycle. Practices will be audited to ensure that they are being followed appropriately.

### 5.3.1. Communication Practices

#### 5.3.1.1. Client

* The client shall only be contacted through the client liaison
* If the Client liaison is not available a replacement will be selected in advance and approved by the Supervisor.
* Contact will be primarily made through email.
* Meetings with the client shall be regular, taking place approximately every two weeks. The client will be contacted more regularly through email after supervisor approval.
* Every available member should attend the Client meeting, if unable to attend the meeting inform the Team Leader 24 hours in advance.
* Client meeting (with the client’s consent) will be recorded and minutes taken

#### 5.3.1.2. Team

* The primary mode of communication when not able to communicate face to face will be through slack.
* Messages sent to the group on slack should be acknowledged once read.
* Should respond to messages on slack within 24 hours.
* Polls made on slack will be considered over 24 hours after initiated.
* The ‘announcements’ channel on slack will be used to communicate group wide notices, the ‘announcements’ channel should not be used to hold any discussions.
* General discussions should only be conducted on the ‘general’ channel on slack.
* Meeting times and locations should be put up on the ‘room-booking’ channel on slack 24 hours prior to the meeting.
* Polls will be held in the ‘polls’ channel on slack.
* All other discussions and announcements shall take place in the slack channels assigned to that topic.
* Student email addresses are to be used in all email communications.
* The use of Skype is permitted in situations where face to face meetings cannot be organized
* Email communication will be kept to a professional standard.
* Emails can be used to confirm verbal contracts.

.

#### 5.3.1.3. Supervisor

* The Team Leader will be conduct all the formal communications with the supervisor.
* There will be one supervisor meeting approximately each week or when required.
* The team leader will discuss the location and time of each meeting, with the supervisor
* Communication with supervisor will be done primarily through email.
* Agreements with supervisor will be confirmed via email or verbally during the previous meeting.
* All emails to the supervisor should CC the entire team unless they are of a personal nature.

### 5.3.2. Meetings

* Team meeting will be held every week
* Team meeting will have a standard length of 1 hour.
* All team members must be present.
* If a team member cannot be present, the team leader should be informed as soon as possible preferably 24 hours prior to the meeting.
* All meetings require minutes to be taken. An informal meeting outside of the weekly team meeting does not need all members present but notes are required.
* notes/minutes will be stored on the Git/Google drive

### 5.3.3. Worklogs

* Weekly update of Worklogs on the Git/Google Drive
* The Team Leader/selected member will monitor Worklogs.
* The Scrum Master/Project manager /Team leader is to monitor and maintain the project hours summary sheet

### 5.3.4. Git

* Temporary/intermediate files are not to be committed
* Standard commit message to be used
* Frequent commit needed to be done
* Each commit to address only one issue
* Pull request to be created after pushing
* Branches must be reviewed before merging

### 5.3.5. Coding practices

**General guidelines**

* Strictly follow Java standards as outlined in 5.2.1
* Keep the code simple, avoid using unnecessary “clever” code.
* Functions should be kept short and to the point.
* All methods, fields and properties must have comments that follows the official standard in 5.2.1

**Guidelines on project structure**

This will be updated once the platform, sdk and language have been finalised.

**Guideline on components design**

Each component design must be justified by thorough analysis into quality requirements of the component, applied design patterns or tactics.

It is recommended that to the very least, a component design should incorporate and separate the following component into directories in the library:

• Interface: include all abstract classes and interfaces

• Enumeration: all enumerations

• Structures: all data structures if any

**Guideline on naming and namespaces**

This will be updated once the platform, sdk and language have been finalised.

# 6. Reviews and Audits

## 6.1. Purpose

This section of the SQAP defines a set of procedures used to validate project deliverables and to verify team processes with respect to defined requirements and standards.

The purpose of validation is to ensure that the correct deliverables are being produced with respect to the client requirements and team standards. This is done through internal and external reviews.

The purpose of verification is the ensure that processes outlined in the SQAP are followed to ensure product quality. This is done through internal and external audits.

The standards, procedures and practices can be found in chapter 5, Standards and Practices.

## 6.2. Review/Audit list

### 6.2.1. Reviews

Reviews are held during all phases of the project’s life-cycle.

#### 6.2.1.1. Formal Review Process

All formal review meetings must use the following process, a formal review is to be declared on a case by case basis:

1. A review committee is selected, and the specified roles are filled.
2. The Moderator identifies and/or confirms the review’s objectives.
3. The Moderator ensures that all members of the committee understand the objectives and the review process.
   1. Individual: the review committee will prepare to review the work by examining it carefully for potential defects.
   2. Team: the review committee meets at a planned time to pool the results of their preparation activity and arrive at a consensus regarding the status of the document or standard being reviewed
4. Author of the work makes the required changes as specified by the review committee.
5. Moderator verifies that the actions required by the Author have taken place.

#### 6.2.1.2) Informal Review Processes

**Code**

Code quality is to be ensured through regular reviews as listed below. If code is found to be unsatisfactory the results will be communicated to the relevant team member and raised as an Issue.

1. Peer review: Code commits shall be reviewed by a peer developer, assigned in the team meeting, as recommended by the Code champion, for the following aspects. Weekly inspection will be carried out on all commits by the assigned peer prior to the next meeting.
   1. Coding Standard
   2. Task Completion
   3. Verified against initial specifications, from each stage’s detail design.
   4. Agreement upon any changes to specifications
2. Client review: Every 2 weeks, all working branches are merged and sent to client for testing and review against the following: (The method of transfer will not require meeting with the client face to face and is distinctly different from client meetings.)
   1. Deliverable timeline
   2. Verified against specifications
   3. Validate task completion

**Meeting**

Meeting quality will primarily be maintained through audits of the correct process, but all meeting related documents will also be reviewed for quality.

Meeting minutes will be reviewed following the first meeting of each secretary against the standards. This is done alongside the formal acceptance of minutes at the end of each meeting.

Agenda will be accepted prior to each meeting and formally reviewed prior to the following meeting.

Any documents found to be unsatisfactory will have results communicated to the secretary and raised as an Issue.

**Management Document**

Management documents will be reviewed against document standards prior to being finalised and released. If the document is found to be unsatisfactory, a list of improvements will be generated and raised as an issue. One example of this is the feedback sheets provided by the supervisor.

### 6.2.2. Audits

Audits should be held regularly during all phases of the project’s life-cycle to ensure processes put in place are being adhered to.

**Coding Practices Audit**

Coding practices will be audited by code champion on a case by case basis (normally because of consecutive unsatisfactory peer reviews). Failure to meet defined coding processes will result in a list of improvements being generated and communicated to responsible team member/s.

**Communication Audit**

Communications will be audited monthly by Team Leader. Failure to meet defined communication processes will result in a list of improvements being generated and communicated to responsible team member/s.

**Git Practices Audit**

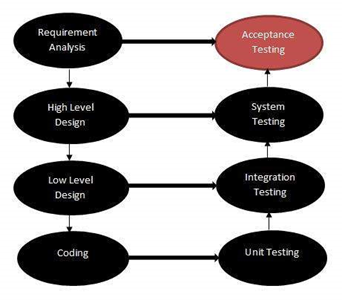
Git Practices will be audited as part of the routine maintenance by the Git champion. Failure to meet the prescribed practices will be communicated to relevant team members with recommendations for improvement.

# 7. Testing

Testing for the Smart Glass Project will mainly be done in order to:

1. Detect any Inconsistencies or lag in the video stream being transmitted between the operator and the worker.
2. Detect any failure or defects in the backend of system (OS, Network, Code) in a controlled environment.
3. Detect if the final implementation meets the standard of the client’s requirements.
4. Detect if any hand gesture made by the operator is clearly visible to the worker in live time.
5. Detect if there are any problems between beginning and end of transmission.
6. Detect irregularities or failures in any of the feature like object recognition, video recording etc.

To achieve this the system will be tested using a four-level testing strategy. These levels will mainly consist of unit testing, integration testing, system testing and acceptance testing. The aim of the team will be to follow and perform these tests as often as possible before handing over the final prototype/product to the client. The following sections outline these testing levels.



**Unit Tests**

Several unit tests will be generated to test specific classes and pieces of code to ensure they perform their individual functions. This will help detect defects, bugs in the code early on saving both time and reduce risk. It also helps in verifying the system design and architecture. Unit testing is typically white box testing and these tests are ideally created by the developer.

**Integration Testing**

Integration testing will be used to test various modules of the system together to evaluate how the system works. Integration tests should ideally be written and run by the testing team or the developers. This will ensure and test whether different sections of the system worked on by different members of the team work effectively when put together.

**System Testing**

System Testing will be used to eventually test the completed product which is integrated with all features that were to be implemented. This will help evaluate whether the system meets and satisfies the specified requirements. System testing falls under black-box testing where no knowledge of the backend code or logic is required. The system test must be carried out by members who have not been involved in the development process of the project.

**Acceptance Testing**

Acceptance testing is the last level of testing that the system must go through to be deemed as complete. The Acceptance test will be conducted in front of the client where 2 members acting as the Operator and Worker will attempt to perform a series of tasks to test the system’s functionality. This will be conducted in a controlled environment and according to a case scenario stated by the client. The System will be given a pass only if accepted by the client.

## 7.1. Requirements

Requirements shall be documented by consulting the client during the meetings. They may change over the course of the project if the client decides to change his mind. The requirements documented by the team will be verified by the client just to be clear. Achieving these requirements will be the top priority of the team over the development of the project.

## 7.2. Use Case Generation

The user case will be generated according to the case scenario given to us by the client. The documented user case will be verified by the client and the project will be mainly focused around the scenario specified by the client.

## 7.3. Installation and User Documentation Generation

There will be an android apk file installed on the Smart Glass. The user will only be required to put on the glasses and carry on with their tasks. The user will be provided with basic instruction on using the glasses along with the tasks he/she must carry out.

The client will also be provided with the instructions along with appropriate design and code documents. On approval of the product the client will be given the final product, in case of failure the product will be given back to the development team to be fixed.

# 8. Problem reporting and corrective action

## 8.1. Personnel

If an issue arises with personnel, the Team Leader is to be notified. The Team Leader will follow up on the issue to gather all the facts. Once known the Team Leader will suggest a corrective action. Corrective Action can include but is not limited to: counselling, team reorganization, protocol changes.

## 8.2. Work

### 8.2.1. Project major timeline

There is one parent project, which is “Smart-Glass Based Remote Guidance System” (SGBRG). SGBRG shall contain multiple phases, which are the major modules that will be developed: Data Communication, Object Recognition, Meat Processing, and Image Processing.

SGBRG will have have multiple versions (SGBRG (Major version).(Minor version).(Revision number)), each shall be a major milestone for development.

Each sub-project will also have multiple version, with each major version is one iteration.

To implement spiral process model, the following postfix is included into each sub-project version to indicate the stage

* “dsg” indicating the design stage.
* “dev” indicating the development stage.
* “tst” indicating the testing stage.

Release date of each version is the due date, scope is to be planned according to this due date.

### 8.2.2. Stage-dependent tasks

Stage-dependent issues and tasks must indicate which sub-project they belong to, as well as which version. (e.g.Data Transmission v1.0 dsg indicating this issue belongs to the design phase of data transmission first release.)

### 8.2.3). Crossed-states tasks

Crossed-states tasks and issues must indicate parent’s version. For example, documentation tasks may fall under SGBRG v1.0.

### 8.2.4. Task creation

Minute taker is to convert meeting’s actions to task and assign to appropriate developer. Developer is responsible to divide the task logically into smaller tasks if necessary.

Team member would also create task as appropriate: for bug reporting or planning. Task creator must check for existing issue prior to creating task.  
All the tasks are created in Jira.

### 8.2.5 Task assignment

When a task cannot be assigned upon creation, the respective champion of the task must perform assignment within 24h of task creation.

### 8.2.6 Task life

Assignee (who is assigned to the task), must response within 12h if the assignment is deemed inappropriate.

Resolver is responsible to ensure solutions are checked against the appropriate standards and practices prior to marking the issue as “Resolved”.

It is stressed that the resolver must entered the time spent on the task into the time-spent box before confirming as ’resolved’

After an issue is marked “Resolved”, respective champion is responsible to formally/informally review the task (exception for trivial tasks), then mark the issue as “closed”

### 8.2.7. Issue Categories

Categories can be updated to adapt to the project’s development, the following are most up to date:

* Administration
* Audit - External
* Audit - Internal
* Client Liaison
* Coding - Prototype
* Documentation - General
* Documentation - PP
* Documentation - SDLC
* Documentation - SQAP
* Documentation - SRS
* Lecture
* Meeting - Client
* Meeting - Supervisor
* Meeting - Team
* Presentation Preparation
* Research - Coding
* Research - Documentation
* Review - External
* Review - Internal
* Git Management
* Requirements - Module 1
* Design - Module 1
* Coding - Module 1
* Testing - Module 1
* Requirements - Module 3
* Design - Module 3
* Coding - Module 3
* Testing - Module 3

# 9. Tools and methodologies

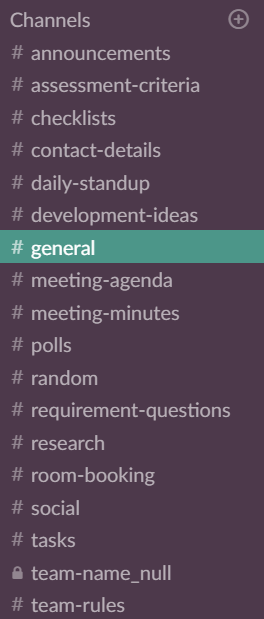
## 9.1. Tools

### 9.1.1. Google Docs

Google Docs is the ideal documentation tool as its similarity to MS Word makes it easy for team members to use. Its online functionality makes it convenient for all members of the team to work simultaneously on the same document.

### 9.1.2. Slack

Slack is a team collaboration tool which we selected as our primary communication tool due to its simplicity and accessibility.



Channels

Multiple channels have been created for discussion of different topics to avoid spam and to ensure all teammates are up to date with the most important information.

#Announcements

This is the channel where the announcements of conclusions to decisions are pinned. It is categorized as a must-read channel where all members of the slack group must read every 24 hours.

#Checklists

This channel has different checklists for specific topics pinned up. For example: The checklist for all the content needed to be included in the Requirements document or all the assessments that are due.

#Contact-Details

Includes all the contact details of the team members so that communication can be made easier.

#Daily-Standup

In this channel, members of the team will log what they worked on throughout the day every day.

#Development-ideas

Discussion of design and development aspects of the Vuzix Smart Glass.

#General

The most active channel in the group where team members can send messages regardless of topic as it isn’t necessary to stick to any specific topic.

#Meeting-Agenda

Meeting agenda documents are uploaded onto this channel so that everyone can easily view them.

#Meeting-minutes

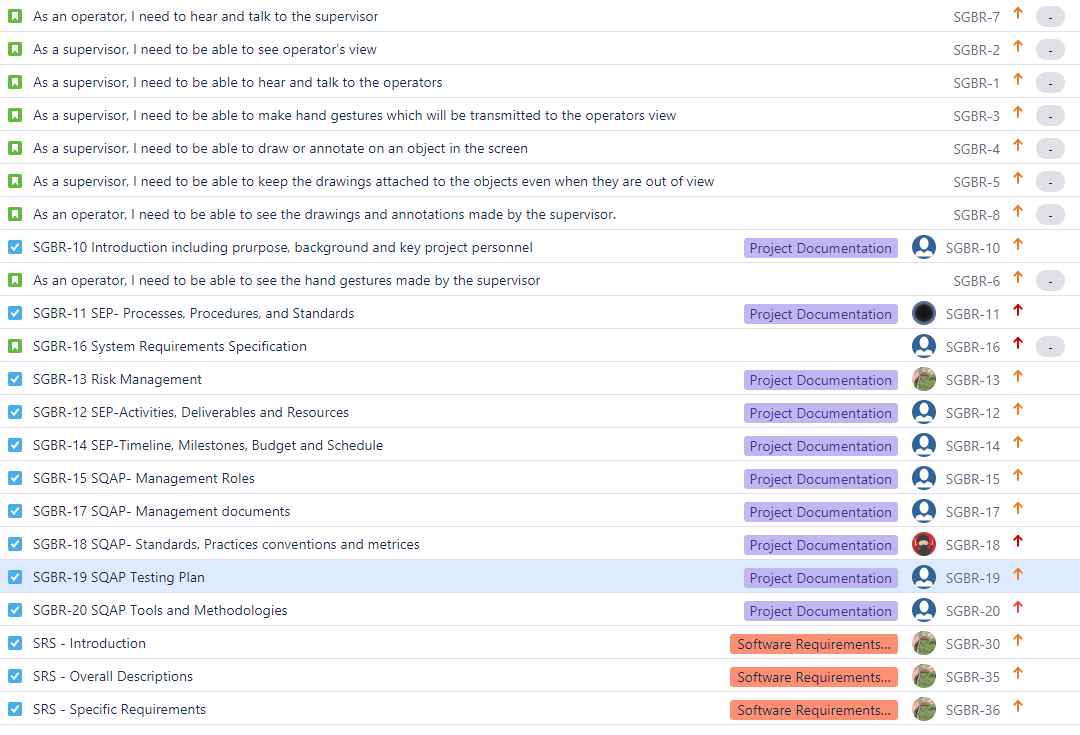
Meeting minutes are uploaded onto this channel.

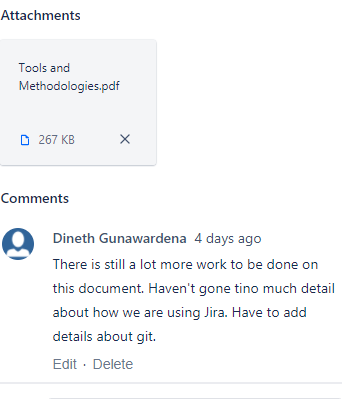
#Polls

Polls are posted on this channel. This is a must-read channel since all members are required to participate in a poll before making a final decision.

### 9.1.3. Issues tracking (Jira)

Jira is a tool which can be used for group management and issue tracking. Currently Jira is being used to assign tasks to members of the team. The tasks being, sections of the project plan and quality assurance plan. Jira is a tool which can be used for group management and issue tracking. As shown below, issues are assigned to each member, and categorized under their respective section and priority.



****

Jira was selected over other issue tracking tools like Trello because of its advanced features-

* Each task can have a sub task.
* Hours spent on each task can be logged with a description of what was done during that time.
* Documents can be attached onto the task making it more convenient as everything is available in one place.
* Comments can be left after updating your documents informing the team about what was done and what is still left to work on.

### 9.1.4. GitHub

GitHub is a web hosting service that hosts projects that uses Git VCS. Unlike GitHub, Bit Bucket supports Mercurial VCS. However, we wouldn’t find that useful as we are using Git VCS, as most of us have more experience with it.

GitHub has features such as:

* Support for over 100 programming languages.
* Data transmitted over SSL or SSH.
* API integration which allows easy integration of third party tools.
* Allows Branch comparison views.

### 9.1.4. Skype

Skype is being used whenever a group member is unable to attend a meeting, group members can still skype call and still give their input in the group meetings. Skype is selected over other tools since Skype is accessible and everyone is experienced with the tool.

### 9.1.5. Qt

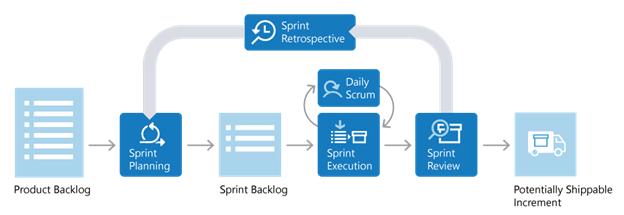
Qt is a cross platform framework, which provides binding for several programming languages. Applications are written in C++ and enhanced with extensions. This framework is popular due to its ease of use.

Differences between Android and Qt:

* Qt is a cross platform framework, which provides binding for several programming languages. Applications are written in C++ and enhanced with extensions. This framework is popular due to its ease of use.
* Android is based on java powered by Linux-based Android OS whereas Qt is a cross platform framework which is powered by operating systems such as Symbian and MeeGo. This limits Android apps to the Android OS which is mainly for smartphones whereas Qt can work across both mobile and desktop platforms.

## 9.2. Design Methodology

Scrum Methodology is a flexible framework to guide teams since it has simple concepts which are easy to learn. It follows the agile principles which focuses on incremental delivery, team collaboration, continual planning and continual learning.



The scrum process works in sprints. Each sprint is 2-4 weeks long. At the start of the sprint team members select tasks they would wish to do from the product backlog. In daily standups, during the sprint team members report their progress. At the end of each sprint an increment (product of a sprint) is demonstrated to the stakeholders in a sprint review. A sprint Retrospective also takes place allowing the team to discuss what issues they faced during the sprint and how they would fix them.

# 10. Records collection, maintenance and retention

Minutes, Agendas and Notes from meetings are added to project team’s Git Repository as well as google drive. Minutes and Notes will be added following approval by meeting participants.

All documentation will be retained in repository for the duration of the project.

# 11. Risk Management

## 11.1. Purpose

Risk management is undertaken to facilitate the creation of a product that is high quality, on time and delivers the scope specified by the client.

## 11.2. Categorization

For this project three major categories of risks have been identified:

1. Risks with respect to the work to be done.
2. Risks with respect to the management.
3. Risks with respect to the client.

In the following sections each of these categories have their major risks identified. For each risk, a description, a probability to occur, its impact and the preventative/(reductive) action associated are given.

Both the probability of a risk occurring and the impact of a risk if it does occur have been quantified as being low, moderate or high. Actions have been categorized as preventative and reductive; preventative actions aim to reduce the likelihood of risks occurring and reductive actions reduce the impact of risks if they do occur.

### 11.2.1) Risks with respect to the work to be done

* Corruption of repository
  + Probability: Low.
  + Impact: High resulting in loss of work.
  + Reductive Action: Weekly backups plus local checkouts reduce impact significantly.
* Design Errors
  + Probability: High.
  + Impact: High, design errors would potentially increase production time and/or produce a deliverable not valid to client requirements.
  + Preventative Action: Rigorous design methodology prior to development.
* Time Shortage
  + Probability: High.
  + Impact: High, resulting in a loss of product quality, loss of functionality or delivered past deadline.
  + Preventative Action: Rigorous design methodology prior to development including work distribution and conservative timelines.
* Illness or absence of team members
  + Probability: High.
  + Impact: Variable impact dependent on time in schedule.
  + Reductive Action: Shared understanding of work allows load to be distributed.
* Software non-deployable
  + Probability: Moderate.
  + Impact: High, will be unable to provide client with the SGBRG.
  + Preventative Action: Regular contact with client with minor releases to ensure that they can be deployed on the system.

### 11.2.2. Risks with respect to the management

* Illness or sudden absence of team leader
  + Probability: Moderate.
  + Impact: Variable impact dependent on time in schedule.
  + Reductive Action: Emergency meeting to be organised by Team Leader to elect temporary team leader.
* SQAP not suitable for our purposes
  + Probability: Low.
  + Impact: High, failure to follow SQAP would reduce product quality.
  + Preventative Action: SQAP to be produced with full team input and cleared with team supervisor.
* Team member leaves the team
  + Probability: Low.
  + Impact: High, as their roles and responsibilities are no longer being fulfilled, and the amount of manhours the team can provide in a given time is diminished.
  + Preventative Action: None. However, roles and responsibilities will need to be redistributed to the remaining team members.

### 11.2.3. Risks with respect to the client

* Changing client requirements
  + Probability: Moderate.
  + Impact: Moderate, increased workload and timeline issues.
  + Preventative Action: Rigorous discussion of requirements plus official SRS document early in project timeline.
* Client unavailable
  + Probability: Moderate.
  + Impact: Low, unavailability for questions and software releases
  + Reductive Action: Vital questions for client to be communicated before they become critical.
* Client abandons project
  + Probability: Low.
  + Impact: High, no more work modules for the project to complete.
  + Reductive Action: Get as many modules and requirements off the client as possible.