SOFTWARE DESIGN AND IMPLEMENTATION: LABELLING APPLICATION FINAL REPORT

Rishi Singh 2018 (N0834113), Maleesha Dedigama (N0803016),
Dayeeta Das (N0830182), Joshua Ssendagire (N0834492)
Lab Tutor: David Adama

Group 9

Abstract

Background

A labelling application software for annotating data sets offers the ability to be used in hot research/innovation topics with the aid of Convulsion Neural Networks (CNN). By combining a labelling annotation program and CNN, machines are able to replicate some human capabilities however, before the use CNN and Machine Learning we must create a training dataset.

The labelling tool, we have created as a group can be used to provide a training dataset which can later be implemented with CNN and Machine Learning techniques.

Aim

Our aim was to develop a labelling application using C++ to import a number of images where the user is able to draw a number of specified shapes with the ability to control the type of shape, dimensions, naming the object in the shape and a few extra features. We aimed to at least use a variety of data structures, sorting/searching algorithms, exception handling where required and object-oriented programming techniques using classes with public, private and protected data members.

Method

The team has used Asana and Slack as communication tools to distribute tasks appropriately depending on individual capabilities and considering individual external commitments. For the designing of the application we have used Visual Paradigm to represent the program in terms of diagrams. Our method to achieving these aims was to use Qt Creator as it consists of a wide variety of implemented classes which have been used throughout the implementation stages of the assignment. In regards to testing the application, we have used a QtTest module to support us in the testing stages of the Graphics User Interface (GUI) and the algorithms used to support the functionality of the program.

Conclusion

We have developed a robust labelling application which meets the aims that were set and have been identified throughout the documentation. We tried to use appropriate data structures where possible and included sorting algorithms using linked lists.

Revision History

Version	Issue Date	Stage	Changes	Author
Version 1	10/01/20	Project Plan	Creation of document.	Rishi Singh
Version 1.1	19/01/20	Project Plan	Compilation of requirements, Gantt chart, mitigation table and communication tools	Rishi Singh
Deliverable 2.1	03/02/20	Project Design	UML diagrams created – use case diagram, class diagram and sequence diagram.	Maleesha Dedigama
Deliverable 2.2	06/02/20	Project Design	UML diagrams added – sequence diagram, state diagram and component diagram.	Maleesha Dedigama
Deliverable 2.3	08/02/20	Project Design	GUI mock up, library tools and frameworks added.	Maleesha Dedigama
Deliverable 3		Project Development	Coding style Added	Dayeeta Das
Deliverable 3.1		Project Development	Contribution guide and Reference manual added.	Dayeeta Das
Deliverable 4		Project Testing	Test plan – introduction, product analysis and Test strategy created.	Josh Ssendagire
Deliverable 4.1		Project Testing	Test objectives criteria, Resource planning, test environment and schedule added.	Josh Ssendagire
Deliverable 5		Final Document	Abstract and Background research added.	Rishi Singh
Deliverable 5.1		Final Document	Requirements and tasks changed.	Rishi Singh
Deliverable 5.2		Final Document	UML Diagrams updated and checked	Maleesha Dedigama & Rishi Singh
Deliverable 5.3		Final Document	Adopted coding style added	Dayeeta Das
Deliverable 5.4		Final Document	Linked list image added to pseudocode	Maleesha Dedigama
Deliverable 5.5		Final Document	Test scenarios and results added	Josh Ssendagire
Deliverable 5.6		Final Document	Conclusion, future work, overall reflection of work done and Appendix	Rishi Singh

Contents

Table of Contents

List of Tables

List of Images

Introduction

For the Software Design & Implementation module we have been assigned to implement a labelling application (software) using C++ to produce annotating datasets which can be used by CNN (Convolutional Neural Networks) in the future.

Project Definition

The project the team has be assigned was to develop a labelling application which allows the user browse through a target folder and import compatible image into the image pane in the application. The application should allow the user to choose an image of their choice, once chosen they have the ability to import the image into the application and draw a number of different shapes such as triangles, rectangles, squares, and polygons from up to eight sides. The user also has the ability to sort the image names in alphabetical order or by file date which both having the options to sort in ascending or descending order.

Once a shape is drawn the user is able to create a 'class'. A class refers to the object inside the shape drawn for example if a shape is drawn on dog then a class with the name dog is created. Users are able to add and remove classes along with sorting the class names in alphabetical order. The user should also be able to create an annotation file which contains details of the images being annotated on.

Team

The team consists of four members the **Project Manager** (Rishi Singh), **Software Designer** (Maleesha Dedigama), **Software Developer** (Dayeeta Das) and the **Software Tester** (Josh Ssendagire). The roles of the team were chosen at the beginning of the project with individual preferences considered when the roles were allocated.

Communication tools

The communication tool the team has decided to use is Slack; we have decided to use Slack because we were able to create a Gantt chart using Asana and Instagantt which we are able to link to Slack giving each member notifications and emails regarding the tasks they need to complete. Slack is good communication tool as it allows users to make different channels (new chats) depending on the topic of conversation. It is very easy to search for history in slack and integrating/share files is also very easy.

Background Research

Background Research strategy

To complete the background research for this project as a team we had decided to begin a general research around the project identifying the needs of the project and tools could be helpful for the team. From the research we came up with a number of tools which we could use and made a list of pros and cons which would help the team move forward. This would also give the team a few back up options if things were not to go as planned later on in the project.

In this section we have identified and categorised the appropriate research which is required for each role as this can help the team find points of reference when a member is having trouble with a specific area of the project. In the table below are tools listed which the team has found from their research with an explanation.

Table 1: Background Research tools for Project Manager

Project Manager	
Tools used for project planning	Explanation
Slack	Slack is communication and monitoring tool which can be used to monitor and communicate with members of the group. Slack allows the user to create different channels and share work. Slack also a very good tool to use as it has many additional tools such as Asana which help in managing tasks.
Asana	Asana is a very helpful which can be used to allocate tasks to different members in the group. Asana and Slack together give instant notifications to the group when tasks are changed, added or completed. Asana is a good tool to have to organise the work load, distribute task along with monitoring work.
Instagantt	Instagantt is a tool which can be used with Slack and Asana. Instagantt allows the user to create a Gantt chart and manage the project. With Instagantt and Asana emails are sent to the members in the team reminding them about their task to be completed and which ones are overdue.

Table 2: Background Research tools for Software Architect

Software Architect	
Tools used for Project Design	Explanation
Visual paradigm	Visual Paradigm is a tool which can be used to model different systems and create UML diagrams which will be very useful for this project. Using Visual Paradigm also allows the Software Architect to generate source code from the diagrams which can be used by the Software Developer.
Umlet	Umlet is another system modelling tool which can be used to create UML diagrams and represent aspects of the project in diagrams. Umlet also allows the Architect to generate source code from the UML diagrams produced.
QT Designer	Qt Designer can be used to produce GUI mock ups

Table 3: Background Research tools for Software Developer

Software developer	
Tools, Libraries and Frameworks	Explanation
QT Creator	QT is an open-source widget toolkit used to develop graphical user interfaces (GUIs), as well as cross-platform applications that runs on different platforms. Our group decided to use QT for the development of the application as it supports various compilers, includes QML that allows the use of JavaScript and especially because it includes the feature JSON parsing.
QPainter	QPainter is a Qt class which can be used to paint on widgets and provides highly optimised functions help in drawing GUI projects.
OpenCV	OpenCV is a library of programming functions. It runs on number of desktops operating systems, including macOS, Windows, Linux and mobile operating systems, including iOS, Android, BlackBerry and Maemo.
OpenGL	OpenGL is an API for rendering 3D graphics and drawing 2D shapes. There is a Qt OpenGL module already available in Qt which can be very helpful in drawing shapes.

Table 4: Background Research tools for Software tester

Tools for testing	Explanations
Jenkins	Jenkins is a free and open source automation server. Our group decided to use Jenkins as it supports Git and it also helps to program the non-human part of the software development process.
Boost	Boost is a unit test framework which can be used to produce unit tests in the testing phase and can be integrated with Qt Creator.
Qt Test	Qt Test is a framework which can be used for unit testing if Qt based applications providing common functionality other unit testing frameworks and also includes extensions for testing graphical user interfaces.

Design

List of requirements

The following table lists the requirements and tasks the project requires. Using MoSCoW analysis the requirements have been categorised by the type of roles each member has and subcategorised into **must** haves, **should** have and **could** have.
The table divided in columns of requirement number, requirement description requirement implication and tasks. The task numbers correlate to the tasks on the Gantt chart which can be seen later on in this document.

Table 5: List of Requirements

Requirement Number	Requirement Description	Requirement Implication	Task(s)
Software Architect			
1	UML diagrams **must** be created.	To visualise a software program which can help in the development of the software.	T1 – Create UML Diagrams
2	**Must** create a Use Case diagram.	To produce an efficient system the team must specify the behaviour of the system and design the system from the user's perspective.	T1.1 – Create a Use Case diagram.
3	**Must** create a Class diagram.	The project will require a structure by showing attributes and operations of the system.	T1.2 – Create a Class diagram.
4	**Must** create a state diagram.	To represent the dynamic behaviour of the system.	T1.3 – Create a State diagram.
5	**Must** create a sequence diagram.	To show how elements of the system interact over time.	T1.4 – Create a Sequence diagram.
6	**Must** create a deployment diagram.	To represent the structure of the run-time system.	T1.5 – Create a Deployment diagram.
7	**Must** use a GUI design tool and graphical libraries/frameworks.	For the project to be completed smoothly, it will be beneficial to use frameworks and libraries which already exist.	T2 – Selection of graphical libraries and frameworks. T2.1 – Research into which GUI design tool should be used. T2.2 - Research into which frameworks/libraries should be used. T2.3 – Practice using Qt creator. T2.4 - Practice using Qt creator with OpenCV library.

8	**Could** use artificial intelligence libraries to add identifying objects functionality to program.	To make the project identify classes in an image by itself, libraries can be used to make this easier.	T2.5 – Basic research should be done into TensorFlow.
9	A GUI mock-up **must** be presented to before the development of code.	This will help the designer and developer minimise chances of errors during the integration.	T2.6 – Create a mock-up GUI using Qt creator.
Software Developer			
10	The software developer **must** produce a contribution guide.	To understand the roles of each team member, during the implementation of the project.	T3 – Contribution guide
11	The software developer **must** document and present to the team how to use git repository and how many times work should be committed.	Members of the team should have a clear idea of how to upload their work so the team can see their contribution. Members of the team should be working on the project thoroughly, so milestones and deadlines are not missed.	T3.1 – Document how members of the team should commit their work to GitHub and how many times they should be committing work.
12	The application **MUST** include a simple GUI.	A simple GUI will allow the team to focus more on the main functionality of the application. A simple GUI will also benefit the user as it would be an easy to use application.	T4 – Redo(update) a simple GUI mock-up must be created with the help from **the Software Architect**.
13	GUI **must** include buttons and text fields.	For the user to be able to communicate with the program buttons and text fields are required	T4.1 – Use buttons and text fields as components of the design.
14	The application **must** include a class selector.	The class selector will enable the user to navigate through folders and select class files.	T4.2 Design a class selector interface (pane).
15	All the classes **should** be listed in the classes pane.	The class pane should allow the user to select a class.	T4.3 – Classes pane should display a list of classes.
16	The application **must** display a pane that shows all the compatible image files in that folder along with options to sort the image by file name and date.	Allows the user to select an appropriate image file to execute the functions on it.	T4.4 - Check GUI has a pane which lists all compatible images. If not add pane to design and implement the functionality of pane.

17	The application **must** include a class selector with a browse button where the user can select the number of classes already saved.	The class selector will enable the user to navigate through folders and select class files.	T4.5 – Check GUI design for a class selector and a browse button. T5 – Implement the functionality of the class selector and browse button.
18	The application **should** be able to add/remove classes.	The application should allow the user to add or remove classes (add to the list of classes saved and remove from the list).	T5.1 – Implement the functionality of adding and removing classes.
19	**Sorting algorithms must** be used- (at least one proposed in lectures).	The user should be allowed to sort file names and dates of creation.	T6 - Allow the user to sort the file in ascending/descending order based on the file name or date. T6.1 – Use sorting algorithms to sort file name in ascending and descending order. T6.2 – User sorting algorithm to sort file date in ascending and descending order.
20	Application **must** be able to select and use several shapes to draw on the image.	Application must enable the user to have only four shape options to choose from: triangle, square/rectangle, trapezium and polygon (with up to 8 points).	T7 – Implementation of shape designer. T7.1 – Implement the ability to draw a square T7.2 – Implement the ability to draw a triangle T7.3 – Implement the ability to draw a square/rectangle T7.4 – Implement the ability to draw a polygon (with up to 8 points).
21	Annotations **should** be displayed on the image.	Annotations should be able to be displayed on top of the image.	T7.5 – Draw shapes on images. (Implement the functionality of annotating on images).
22	**Should** be able to perform a few shape operations using the mouse.	These shape operations allow the user to annotate the image accurately.	T7.6 – Implement shape operations using the mouse. T7.6.1 – Increase/decrease the size of the shape. T7.6.2- Move vertices of a polygon. T7.6.3 – Delete a shape T7.6.4 – Copy & paste shape. T7.6.5 – Visualise name of classes on top of the shape.

23	File handling - Annotations **must** be saved in an annotations file.	The application should automatically save the annotation file every minute.	T8 – Implement file handling. T8.1 – Save file. T8.2 – Open file. T8.3 - Load file. T8.4 – Append to file. T8.5 – Warning should be displayed to the user to confirm the ability to overwrite.
24	File handling- **Must** autosave.	Application should automatically save the annotation file every minute.	T8.6 – Autosave files, threads must be used.
25	Data structures **must** be used for storing data in memory.	To save data appropriately in the memory.	T9 – Use data structures to implement the saving of data in memory. T9.1 – Use linked list as a starting data structure however if a better data structure is available; **discuss** with **Software Developer**.
26	Data **must** be stored in annotation files accurately.	By including a specific number of data values in the annotation files, it is easier to open and load files in the image pane.	T9.2 – When saving annotation file-specific data should be saved. T9.2.1 – Store number of annotated files. T9.2.2 – Store image file name. T9.2.3 – Store number of shapes per image. T9.2.4 – Store shape type T9.2.5 – Store Coordinate of points.
27	The application **must** include exception handling.	If a user was to do a monkey test on the application the software should break hence the project requires exception handling to avoid these problems.	T10 – Thorough exception handling must be implemented in the project.
28	**Could** include help buttons to navigate around the application.	To help the user have a clear idea of the purpose of the program.	T11 – Implement a help button.
29	Could **make** the GUI more appealing to the user.	An appealing GUI will attract the user to use the application more.	T12- Develop GUI to be more appealing.
Software Tester			
30	A test plan **must** be produced.	The software must not have any bugs in the functionality, and this should be	T13 – Software tester must produce a test plan. T13.1 – Design Unit testing.

		thoroughly checked before the deadline.	T.13.2 – Design Integration testing. T.13.3 – Design System testing. T.13.4 – Design Acceptance testing.
31	**Must** complete all tests planned.	The software must not have any bugs hence testing must be conducted.	T14 – Complete all tests planned. T14.1 – Complete Unit testing T14.1.1 – Test Linked List T14.1.2 – Test Sorting Algorithm T14.2 – Complete Integration testing T14.3 – Complete Acceptance
32	**Must** correct any errors found in code refer to the **Software developer**.	Any errors found in the testing must be corrected, the software tester and software developer **must** work together.	T15 – Fix any errors found.

List of tasks

The following table shows a list of tasks numbered and grouped in the roles of each team member. A brief description of the task is also given to help in understanding what is required from the task.

The tasks in the following table are correlated to the tasks on the Gantt chart and can be identified using the task numbers.

Table 6: List of Tasks

Tasks number	Task	Task description
Software Architect		
T1	Create UML Diagrams	Creation of UML diagrams are required to help the software developer in the implementation stage.
T1.1	Create a Use Case diagram.	Create use case scenarios and represent the interaction between the user and the system.
T1.2	Create a Class diagram.	Create a class diagram which can be used in the implementation stages to structure code.
T1.3	Create a State diagram.	The state diagram should represent the behaviour of the diagram.
T1.4	Create a Sequence diagram.	Create a sequence diagram to capture the interaction of objects and classes.
T1.5	Create a Deployment diagram.	Create a deployment diagram to show the interaction between hardware and software.
T2	Selection of graphical libraries and frameworks.	Select appropriate libraries and frameworks which can be used.
T2.1	Research into which GUI design tool should be used.	Look into the GUI and which components are required to form a good GUI using QT.
T2.2	Research into which frameworks/libraries should be used.	Research into other frameworks and libraries for shapes.

T2.3	Practice using Qt creator.	All members should practice using QT creator so they can help in development
T2.4	Practice using Qt creator with OpenCV library.	Try using OpenCV with QT.
T2.5	Basic research should be done into QPainter.	Try using QPainter with QT.
T2.6	Create a mock-up GUI using Qt creator.	A mock GUI must be present.
Software Developer		
T3	Contribution guide	Document shows how members of the team should commit work and comment.
T3.1	Document how members of the team should commit their work to GitHub and how many times they should be committing work.	Document should also help in solving merges which the software developer is responsible for.
T4	Redo(update) a simple GUI mock-up must be created with the help from **the Software Architect**.	GUI must be reviewed. **Software developer** and **architect** must be in **communication**.
T4.1	Use buttons and text fields as components of the design.	Use of buttons and appropriate user interface tools should be used.
T4.2	Design a class selector interface (pane).	Class selector should be on UI.
T4.3	Classes pane should display a list of classes.	List of classes created must be seen on the UI.
T4.4	Check GUI has a pane which lists all compatible images. If not add pane to design and implement the functionality of pane.	Certain image files should only be compatible such as jpg and png.
T4.5	Check GUI design for a class selector and a browse button.	Class selector and browse buttons are required in the GUI.
T5	Implement the functionality of the class selector and browse button.	User should be able to browse for classes.
T5.1	Implement the functionality of adding and removing classes.	User should be able to add and remove classes in the application.

T6	Allow the user to sort the file in ascending/descending order based on the file name or date.	Application must be able to sort file name or date in ascending or descending order. **Sorting algorithm** from **lectures** are recommended to have a look at.
T6.1	Use sorting algorithms to sort file name in ascending and descending order.	File name should be sorted in ascending and descending order.
T6.2	User sorting algorithm to sort file date in ascending and descending order.	File date should be sorted in ascending or descending order.
T7	Implementation of shape designer.	Shapes should be drawn on an image.
T7.1	Implement the ability to draw a square and rectangle	User should be able to draw a square and rectangle using QPainter.
T7.2	Implement the ability to draw a triangle	User should be able to draw a triangle using QPainter.
T7.3	Implement the ability to draw a trapezium.	User should be able to draw a trapezium using QPainter.
T7.4	Implement the ability to draw a polygon (with up to 8 points).	User should be able to draw a polygon from 5 sides to 8 using QPainter.
T7.5	Draw shapes on images. (Implement the functionality of annotating on images).	Shapes should be drawn onto of images.
T7.6	Implement shape operations using the mouse.	Shapes should be drawn using a mouse.
T7.6.1	Increase/decrease the size of the shape.	User should be able to change the size of the shape.
T7.6.2	Move vertices of a polygon.	User should be able to move verticies.
T7.6.3	Delete a shape	Shape can be deleted.
T7.6.4	Copy & paste shape.	Option to copy and paste.
T7.6.5	Visualise name of classes on top of the shape.	Class names should be visualised on shape when drawn.
T8	Implement file handling.	File handling should be shown.

T8.1	Save file.	User should be able to save image.
T8.2	Open file.	Images should be opened.
T8.3	Load file.	Files should be loaded.
T8.4	Append to file.	Shapes should be appended to file.
T8.5	Warning should be displayed to the user to confirm the ability to overwrite.	Warning should be shown to overwrite.
T8.6	Autosave files, threads must be used.	The application should autosave.
T9	Use data structures to implement the saving of data in memory.	Key data structures should be used when saving data in memory such as linked lists, queues and stacks.
T9.1	Use linked list as a starting data structure however if a better data structure is available; **discuss** with **Software Developer**.	One data structure is a **must** in the application.
T9.2	When saving annotation file-specific data should be saved.	Annotations should be saved in a file in Json format.
T9.2.1	Store number of annotated files.	Number of annotated files should be stored.
T9.2.2	Store image file name.	File name should be stored
T9.2.3	Store number of shapes per image.	Shapes per image should be stored
T9.2.4	Store shape type	Shape type should be stored.
T9.2.5	Store Coordinate of points.	Coordinates of the shape should be stored.
T10	Thorough exception handling must be implemented in the project.	Exception handling throughout the project is required to limit errors.
T11	Implement a help button.	A help button for the user interface is required.
T12	Develop GUI to be more appealing.	GUI must look appealing and professional.
Software Tester		

T13	Software tester must produce a test plan.	A test plan must be produced for the preparation of testing.
T13.1	Design Unit testing.	Unit test must be designed and implemented to check aspects of the program.
T.13.2	Design Integration testing.	Integration testing must be designed.
T.13.3	– Design System testing.	System test must be designed and implemented.
T.13.4	– Design Acceptance testing.	Acceptance testing must be designed an implemented.
T14	– Complete all tests planned.	All tests must be planed before implementation.
T14.1	Complete Unit testing	Unit tests should be implemented specifically for linked lists and sorting algorithm.
T14.1.1	Test Linked List	Linked list must be tested.
T14.1.2	Test Sorting Algorithm	Sorting algorithms must be tested.
T14.2	– Complete Integration testing	Integration testing must be completed.
T14.3	– Complete Acceptance	Acceptance testing must be completed.
T15	– Fix any errors found	Any errors should be shown to the **developer** and corrected.

Time plans

Initial Time plan

The Gantt chart displays the project plan with accurate dates and times of when specific tasks which should be completed. As a team, we have adjusted some dates, for our convenience considering other module deadlines. The project design can only start once the project plan is completed and the implementation stage can only be started when the project design is completed. This is the same for the testing phase, which can only begin once the implementation stage is completed. However, during the planning and designing of the project, members of the team will also start planning tests for the testing phase.

The Gantt chart contains tasks which are identifiable as 'T1'; these tasks correlate to the requirements table. Milestones are represented by 'M1' which are kept every week or every two weeks depending on the tasks. Deliverables are represented by 'D1'.

The due dates of the deliverables are not accurate to the submission dates, as the team would like to give more time to the coding aspect of the project. Hence, we have decided to finish the deliverables before the submission due date. We have also given enough time before the submission due date, so we can revise the deliverables to a high standard. Some deliverables are accurate with dates due to more time being allocated to the deliverables as members of the team do have the option to extend dates up till the deliverable deadlines.

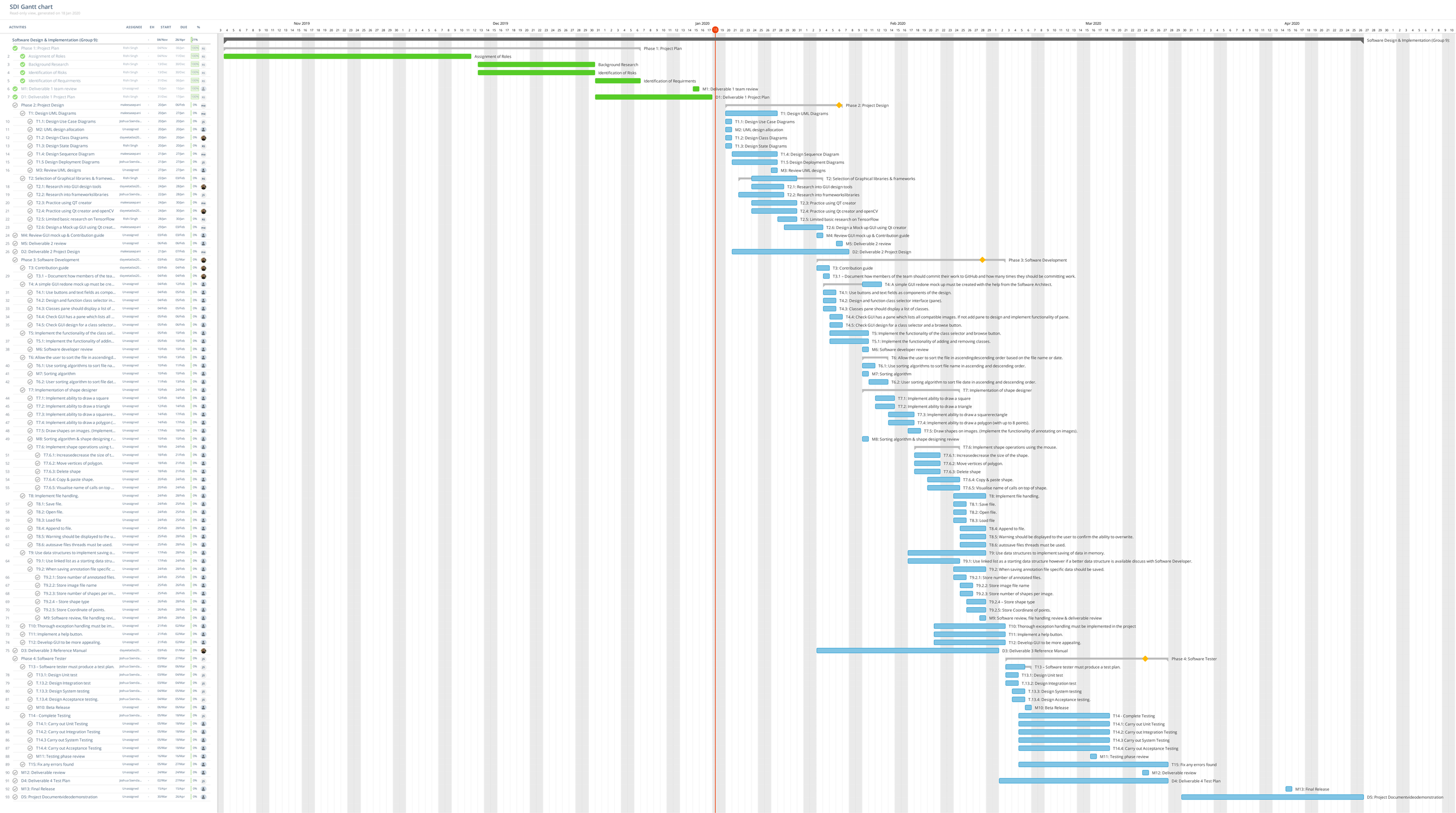
Submission dates of deliverables:

Deliverable 1: Project Plan	19/01/2020
Deliverable 2: Project Design	09/02/2020
Deliverable 3: Reference Manual	01/03/2020
Deliverable 4: Test Plan	29/03/2020
Project Document/Video/Demonstration/source code	26/04/2020

Some tasks are unassigned due to the contribution guide which is not complete yet; once the guide is complete the Gantt chart will be updated and will include the tasks that have been assigned to each group member.

SDI Gantt chart

Read-only view, generated on 18 Jan 2020



	ACTIVITIES	ASSIGNEE	EH	START	DUE	%
	Software Design & Implementation (Group 9):		-	04/Nov	26/Apr	21%
	Phase 1: Project Plan	Rishi Singh	-	04/Nov	06/Jan	100%
2	Assignment of Roles	Rishi Singh	-	04/Nov	11/Dec	100%
3	Background Research	Rishi Singh	-	13/Dec	30/Dec	100%
4	Identification of Risks	Rishi Singh	-	13/Dec	30/Dec	100%
5	Identification of Requirments	Rishi Singh	-	31/Dec	06/Jan	100%
6	M1: Deliverable 1 team review	Unassigned	-	15/Jan	15/Jan	100%
7	D1: Deliverable 1 Project Plan	Rishi Singh	-	31/Dec	17/Jan	100%
	Phase 2: Project Design	maleesasepani	-	20/Jan	06/Feb	0%
	T1: Design UML Diagrams	maleesasepani	-	20/Jan	27/Jan	0%
10	T1.1: Design Use Case Diagrams	Joshua Sisenda...	-	20/Jan	20/Jan	0%
11	M2: UML design allocation	Unassigned	-	20/Jan	20/Jan	0%
12	T1.2: Design Class Diagrams	dayeetadas20...	-	20/Jan	20/Jan	0%
13	T1.3: Design State Diagrams	Rishi Singh	-	20/Jan	20/Jan	0%
14	T1.4: Design Sequence Diagram	maleesasepani	-	21/Jan	27/Jan	0%
15	T1.5 Design Deployment Diagrams	Joshua Sisenda...	-	21/Jan	27/Jan	0%
16	M3: Review UML designs	Unassigned	-	27/Jan	27/Jan	0%
	T2: Selection of Graphical libraries & framewo...	Rishi Singh	-	22/Jan	03/Feb	0%
18	T2.1: Research into GUI design tools	dayeetadas20...	-	24/Jan	28/Jan	0%
19	T2.2: Research into frameworkslibraries	Joshua Sisenda...	-	22/Jan	28/Jan	0%
20	T2.3: Practice using QT creator	maleesasepani	-	24/Jan	30/Jan	0%
21	T2.4: Practice using Qt creator and openCV	dayeetadas20...	-	24/Jan	30/Jan	0%
22	T2.5: Limited basic research on TensorFlow	Rishi Singh	-	28/Jan	30/Jan	0%
23	T2.6: Design a Mock up GUI using Qt creat...	maleesasepani	-	29/Jan	03/Feb	0%
24	M4: Review GUI mock up & Contribution guide	Unassigned	-	03/Feb	03/Feb	0%
25	M5: Deliverable 2 review	Unassigned	-	06/Feb	06/Feb	0%
26	D2: Deliverable 2 Project Design	maleesasepani	-	21/Jan	07/Feb	0%
	Phase 3: Software Development	dayeetadas20...	-	03/Feb	02/Mar	0%
	T3: Contribution guide	dayeetadas20...	-	03/Feb	04/Feb	0%
29	T3.1 – Document how members of the tea...	dayeetadas20...	-	04/Feb	04/Feb	0%
	T4: A simple GUI redone mock up must be cre...	Unassigned	-	04/Feb	12/Feb	0%
31	T4.1: Use buttons and text fields as compo...	Unassigned	-	04/Feb	05/Feb	0%
32	T4.2: Design and function class selector in...	Unassigned	-	04/Feb	05/Feb	0%
33	T4.3: Classes pane should display a list of ...	Unassigned	-	04/Feb	05/Feb	0%
34	T4.4: Check GUI has a pane which lists all ...	Unassigned	-	05/Feb	06/Feb	0%
35	T4.5: Check GUI design for a class selector...	Unassigned	-	05/Feb	06/Feb	0%
	T5: Implement the functionality of the class sel...	Unassigned	-	05/Feb	10/Feb	0%
37	T5.1: Implement the functionality of addin...	Unassigned	-	05/Feb	10/Feb	0%
38	M6: Software developer review	Unassigned	-	10/Feb	10/Feb	0%
	T6: Allow the user to sort the file in ascendingd...	Unassigned	-	10/Feb	13/Feb	0%
40	T6.1: Use sorting algorithms to sort file na...	Unassigned	-	10/Feb	11/Feb	0%
41	M7: Sorting algorithm	Unassigned	-	10/Feb	10/Feb	0%
42	T6.2: User sorting algorithm to sort file dat...	Unassigned	-	11/Feb	13/Feb	0%
	T7: Implementation of shape designer	Unassigned	-	10/Feb	24/Feb	0%
44	T7.1: Implement ability to draw a square	Unassigned	-	12/Feb	14/Feb	0%
45	T7.2: Implement ability to draw a triangle	Unassigned	-	12/Feb	14/Feb	0%
46	T7.3: Implement ability to draw a squarere...	Unassigned	-	14/Feb	17/Feb	0%
47	T7.4: Implement ability to draw a polygon (...	Unassigned	-	14/Feb	17/Feb	0%
48	T7.5: Draw shapes on images. (Implement...	Unassigned	-	17/Feb	18/Feb	0%
49	M8: Sorting algorithm & shape designing r...	Unassigned	-	10/Feb	10/Feb	0%
	T7.6: Implement shape operations using t...	Unassigned	-	18/Feb	24/Feb	0%
51	T7.6.1: Increasedecrease the size of t...	Unassigned	-	18/Feb	21/Feb	0%
52	T7.6.2: Move vertices of polygon.	Unassigned	-	18/Feb	21/Feb	0%
53	T7.6.3: Delete shape	Unassigned	-	18/Feb	21/Feb	0%
54	T7.6.4: Copy & paste shape.	Unassigned	-	20/Feb	24/Feb	0%
55	T7.6.5: Visualise name of calls on top ...	Unassigned	-	20/Feb	24/Feb	0%
	T8: Implement file handling.	Unassigned	-	24/Feb	28/Feb	0%
57	T8.1: Save file.	Unassigned	-	24/Feb	25/Feb	0%
58	T8.2: Open file.	Unassigned	-	24/Feb	25/Feb	0%
59	T8.3: Load file	Unassigned	-	24/Feb	25/Feb	0%
60	T8.4: Append to file.	Unassigned	-	25/Feb	28/Feb	0%
61	T8.5: Warning should be displayed to the u...	Unassigned	-	25/Feb	28/Feb	0%
62	T8.6: autosave files threads must be used.	Unassigned	-	25/Feb	28/Feb	0%
	T9: Use data structures to implement saving o...	Unassigned	-	17/Feb	28/Feb	0%
64	T9.1: Use linked list as a starting data stru...	Unassigned	-	17/Feb	24/Feb	0%
	T9.2: When saving annotation file specific ...	Unassigned	-	24/Feb	28/Feb	0%
66	T9.2.1: Store number of annotated files.	Unassigned	-	24/Feb	25/Feb	0%
67	T9.2.2: Store image file name	Unassigned	-	25/Feb	26/Feb	0%
68	T9.2.3: Store number of shapes per im...	Unassigned	-	25/Feb	26/Feb	0%
69	T9.2.4 – Store shape type	Unassigned	-	26/Feb	28/Feb	0%
70	T9.2.5: Store Coordinate of points.	Unassigned	-	26/Feb	28/Feb	0%
71	M9: Software review, file handling revi...	Unassigned	-	28/Feb	28/Feb	0%
72	T10: Thorough exception handling must be im...	Unassigned	-	21/Feb	02/Mar	0%
73	T11: Implement a help button.	Unassigned	-	21/Feb	02/Mar	0%
74	T12: Develop GUI to be more appealing.	Unassigned	-	21/Feb	02/Mar	0%
75	D3: Deliverable 3 Reference Manual	dayeetadas20...	-	03/Feb	01/Mar	0%
	Phase 4: Software Tester	Joshua Sisenda...	-	03/Mar	27/Mar	0%
	T13 – Software tester must produce a test plan.	Joshua Sisenda...	-	03/Mar	06/Mar	0%
78	T13.1: Design Unit test	Joshua Sisenda...	-	03/Mar	04/Mar	0%
79	T.13.2: Design Integration test	Joshua Sisenda...	-	03/Mar	04/Mar	0%
80	T.13.3: Design System testing	Joshua Sisenda...	-	04/Mar	05/Mar	0%
81	T.13.4: Design Acceptance testing.	Joshua Sisenda...	-	04/Mar	05/Mar	0%
82	M10: Beta Release	Unassigned	-	06/Mar	06/Mar	0%
	T14 - Complete Testing	Joshua Sisenda...	-	05/Mar	18/Mar	0%
84	T14.1: Carry out Unit Testing	Unassigned	-	05/Mar	18/Mar	0%
85	T14.2: Carry out Integration Testing	Unassigned	-	05/Mar	18/Mar	0%
86	T14.3 Carry out System Testing	Unassigned	-	05/Mar	18/Mar	0%
87	T14.4: Carry out Acceptance Testing	Unassigned	-	05/Mar	18/Mar	0%
88	M11: Testing phase review	Unassigned	-	16/Mar	16/Mar	0%
89	T15: Fix any errors found	Unassigned	-	05/Mar	27/Mar	0%
90	M12: Deliverable review	Unassigned	-	24/Mar	24/Mar	0%
91	D4: Deliverable 4 Test Plan	Joshua Sisenda...	-	02/Mar	27/Mar	0%
92	M13: Final Release	Unassigned	-	15/Apr	15/Apr	0%
93	D5: Project Documentvideodemonstration	Unassigned	-	30/Mar	26/Apr	0%

Changes made to the time plan

All the tasks have been assigned to members of the group and the tasks assigned are corelated to the roles each member has in regards to the project.

Due to the unexpected coronavirus incident and internal issues the group had due to challenges in the development of software some task was delayed by two weeks and some were not completed which are highlighted in red in the Gantt Chart shown in figure X. The two weeks which had delayed the development and caused issues were from the 16th-28th March as team members had to relocate and other deadlines were due for at that point. Therefore, one extra week was given for all tasks which were not included in that time period which can be seen in figure X.

The tasks in the Gantt chart correlate to the tasks in the requirements and tasks table presented earlier in the document. All the tasks were assigned to members of the group and tasks which were not completed were reallocated to different member of the or left incomplete.

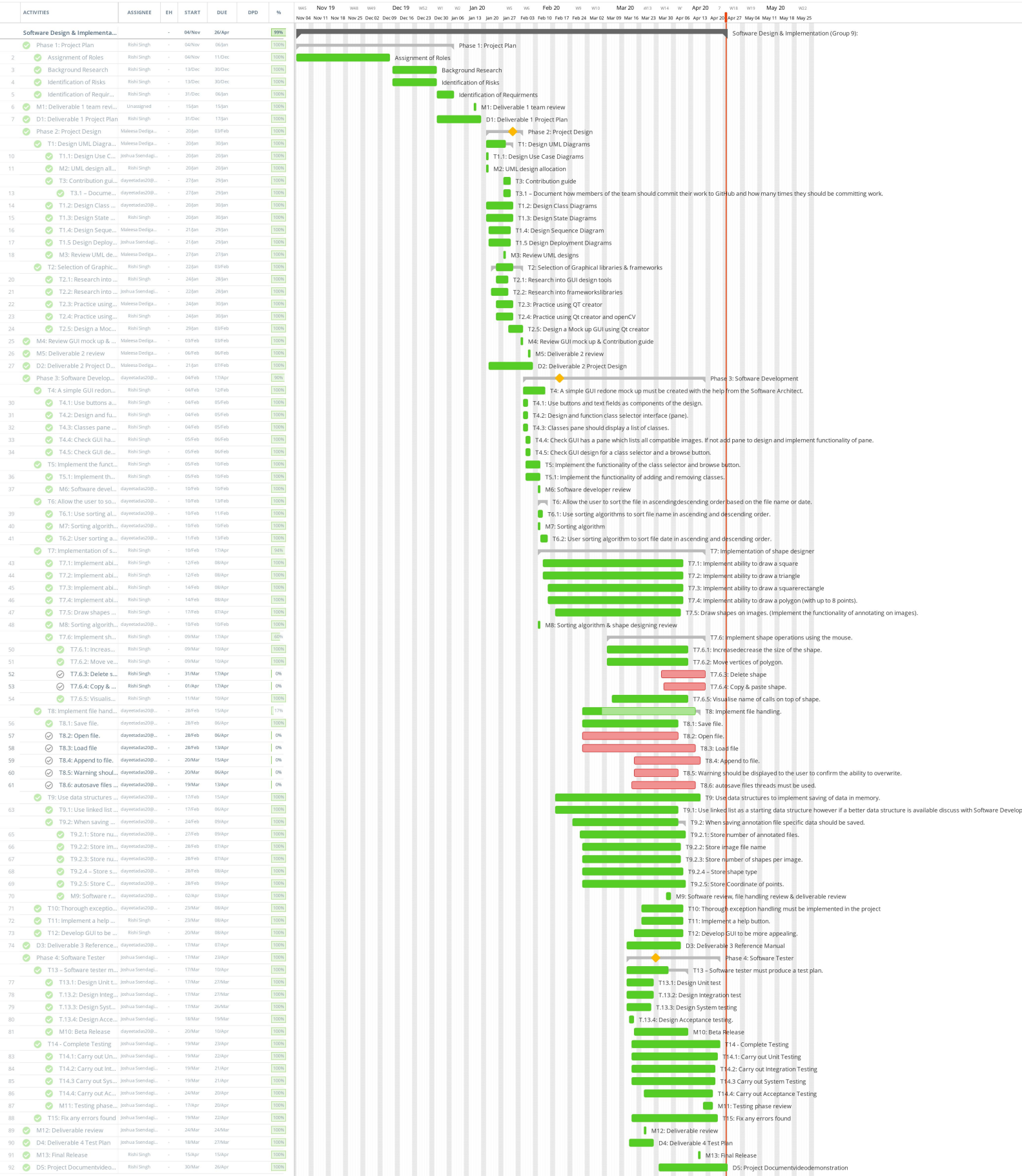
The submission dates of deliverables stayed the same.

Submission dates of deliverables:

Deliverable 2: Project Design	09/02/2020
Deliverable 3: Reference Manual	01/03/2020
Deliverable 4: Test Plan	29/03/2020
Project Document/Video/Demonstration/source code	26/04/2020

SDI Gantt chart

Read-only view, generated on 26 Apr 2020



	ACTIVITIES	ASSIGNEE	EH	START	DUE	DPD	%
	Software Design & Implementa...		-	04/Nov	26/Apr		99%
	Phase 1: Project Plan	Rishi Singh	-	04/Nov	06/Jan		100%
2	Assignment of Roles	Rishi Singh	-	04/Nov	11/Dec		100%
3	Background Research	Rishi Singh	-	13/Dec	30/Dec		100%
4	Identification of Risks	Rishi Singh	-	13/Dec	30/Dec		100%
5	Identification of Requir...	Rishi Singh	-	31/Dec	06/Jan		100%
6	M1: Deliverable 1 team revi...	Unassigned	-	15/Jan	15/Jan		100%
7	D1: Deliverable 1 Project Plan	Rishi Singh	-	31/Dec	17/Jan		100%
	Phase 2: Project Design	Maleesa Dediga...	-	20/Jan	03/Feb		100%
	T1: Design UML Diagra...	Maleesa Dediga...	-	20/Jan	30/Jan		100%
10	T1.1: Design Use C...	Joshua Ssendagi...	-	20/Jan	20/Jan		100%
11	M2: UML design all...	Rishi Singh	-	20/Jan	20/Jan		100%
	T3: Contribution gui...	dayeetadas20@...	-	27/Jan	29/Jan		100%
13	T3.1 – Docume...	dayeetadas20@...	-	27/Jan	29/Jan		100%
14	T1.2: Design Class ...	dayeetadas20@...	-	20/Jan	30/Jan		100%
15	T1.3: Design State ...	Rishi Singh	-	20/Jan	30/Jan		100%
16	T1.4: Design Seque...	Maleesa Dediga...	-	21/Jan	29/Jan		100%
17	T1.5 Design Deploy...	Joshua Ssendagi...	-	21/Jan	29/Jan		100%
18	M3: Review UML de...	Maleesa Dediga...	-	27/Jan	27/Jan		100%
	T2: Selection of Graphic...	Rishi Singh	-	22/Jan	03/Feb		100%
20	T2.1: Research into ...	Rishi Singh	-	24/Jan	28/Jan		100%
21	T2.2: Research into ...	Joshua Ssendagi...	-	22/Jan	28/Jan		100%
22	T2.3: Practice using...	Maleesa Dediga...	-	24/Jan	30/Jan		100%
23	T2.4: Practice using...	Rishi Singh	-	24/Jan	30/Jan		100%
24	T2.5: Design a Moc...	Rishi Singh	-	29/Jan	03/Feb		100%
25	M4: Review GUI mock up & ...	Maleesa Dediga...	-	03/Feb	03/Feb		100%
26	M5: Deliverable 2 review	Maleesa Dediga...	-	06/Feb	06/Feb		100%
27	D2: Deliverable 2 Project D...	Maleesa Dediga...	-	21/Jan	07/Feb		100%
	Phase 3: Software Develop...	dayeetadas20@...	-	04/Feb	17/Apr		90%
	T4: A simple GUI redon...	Rishi Singh	-	04/Feb	12/Feb		100%
30	T4.1: Use buttons a...	Rishi Singh	-	04/Feb	05/Feb		100%
31	T4.2: Design and fu...	Rishi Singh	-	04/Feb	05/Feb		100%
32	T4.3: Classes pane ...	Rishi Singh	-	04/Feb	05/Feb		100%
33	T4.4: Check GUI ha...	Rishi Singh	-	05/Feb	06/Feb		100%
34	T4.5: Check GUI de...	Rishi Singh	-	05/Feb	06/Feb		100%
	T5: Implement the funct...	Rishi Singh	-	05/Feb	10/Feb		100%
36	T5.1: Implement th...	Rishi Singh	-	05/Feb	10/Feb		100%
37	M6: Software devel...	dayeetadas20@...	-	10/Feb	10/Feb		100%
	T6: Allow the user to so...	dayeetadas20@...	-	10/Feb	13/Feb		100%
39	T6.1: Use sorting al...	dayeetadas20@...	-	10/Feb	11/Feb		100%
40	M7: Sorting algorith...	dayeetadas20@...	-	10/Feb	10/Feb		100%
41	T6.2: User sorting a...	dayeetadas20@...	-	11/Feb	13/Feb		100%
	T7: Implementation of s...	Rishi Singh	-	10/Feb	17/Apr		94%
43	T7.1: Implement abi...	Rishi Singh	-	12/Feb	08/Apr		100%
44	T7.2: Implement abi...	Rishi Singh	-	12/Feb	08/Apr		100%
45	T7.3: Implement abi...	Rishi Singh	-	14/Feb	08/Apr		100%
46	T7.4: Implement abi...	Rishi Singh	-	14/Feb	08/Apr		100%
47	T7.5: Draw shapes ...	Rishi Singh	-	17/Feb	07/Apr		100%
48	M8: Sorting algorith...	dayeetadas20@...	-	10/Feb	10/Feb		100%
	T7.6: Implement sh...	Rishi Singh	-	09/Mar	17/Apr		60%
50	T7.6.1: Increas...	Rishi Singh	-	09/Mar	10/Apr		100%
51	T7.6.2: Move ve...	Rishi Singh	-	09/Mar	10/Apr		100%
52	T7.6.3: Delete s...	Rishi Singh	-	31/Mar	17/Apr		0%
53	T7.6.4: Copy & ...	Rishi Singh	-	01/Apr	17/Apr		0%
54	T7.6.5: Visualis...	Rishi Singh	-	11/Mar	10/Apr		100%
	T8: Implement file hand...	dayeetadas20@...	-	28/Feb	15/Apr		17%
56	T8.1: Save file.	dayeetadas20@...	-	28/Feb	06/Apr		100%
57	T8.2: Open file.	dayeetadas20@...	-	28/Feb	06/Apr		0%
58	T8.3: Load file	dayeetadas20@...	-	28/Feb	13/Apr		0%
59	T8.4: Append to file.	dayeetadas20@...	-	20/Mar	15/Apr		0%
60	T8.5: Warning shoul...	dayeetadas20@...	-	20/Mar	06/Apr		0%
61	T8.6: autosave files ...	dayeetadas20@...	-	19/Mar	13/Apr		0%
	T9: Use data structures ...	dayeetadas20@...	-	17/Feb	15/Apr		100%
63	T9.1: Use linked list ...	dayeetadas20@...	-	17/Feb	06/Apr		100%
	T9.2: When saving ...	dayeetadas20@...	-	24/Feb	09/Apr		100%
65	T9.2.1: Store nu...	dayeetadas20@...	-	27/Feb	09/Apr		100%
66	T9.2.2: Store im...	dayeetadas20@...	-	28/Feb	07/Apr		100%
67	T9.2.3: Store nu...	dayeetadas20@...	-	28/Feb	07/Apr		100%
68	T9.2.4 – Store s...	dayeetadas20@...	-	28/Feb	08/Apr		100%
69	T9.2.5: Store C...	dayeetadas20@...	-	28/Feb	09/Apr		100%
70	M9: Software r...	dayeetadas20@...	-	02/Apr	03/Apr		100%
71	T10: Thorough exceptio...	dayeetadas20@...	-	23/Mar	08/Apr		100%
72	T11: Implement a help ...	Rishi Singh	-	23/Mar	08/Apr		100%
73	T12: Develop GUI to be ...	Rishi Singh	-	20/Mar	08/Apr		100%
74	D3: Deliverable 3 Reference...	dayeetadas20@...	-	17/Mar	07/Apr		100%
	Phase 4: Software Tester	Joshua Ssendagi...	-	17/Mar	23/Apr		100%
	T13 – Software tester m...	Joshua Ssendagi...	-	17/Mar	10/Apr		100%
77	T13.1: Design Unit t...	Joshua Ssendagi...	-	17/Mar	27/Mar		100%
78	T.13.2: Design Integ...	Joshua Ssendagi...	-	17/Mar	27/Mar		100%
79	T.13.3: Design Syst...	Joshua Ssendagi...	-	17/Mar	26/Mar		100%
80	T.13.4: Design Acce...	Joshua Ssendagi...	-	18/Mar	19/Mar		100%
81	M10: Beta Release	dayeetadas20@...	-	20/Mar	10/Apr		100%
	T14 - Complete Testing	Joshua Ssendagi...	-	19/Mar	23/Apr		100%
83	T14.1: Carry out Un...	Joshua Ssendagi...	-	19/Mar	22/Apr		100%
84	T14.2: Carry out Int...	Joshua Ssendagi...	-	19/Mar	21/Apr		100%
85	T14.3 Carry out Sys...	Joshua Ssendagi...	-	19/Mar	21/Apr		100%
86	T14.4: Carry out Ac...	Joshua Ssendagi...	-	24/Mar	20/Apr		100%
87	M11: Testing phase...	Joshua Ssendagi...	-	17/Apr	20/Apr		100%
88	T15: Fix any errors found	Joshua Ssendagi...	-	19/Mar	22/Apr		100%
89	M12: Deliverable review	Joshua Ssendagi...	-	24/Mar	24/Mar		100%
90	D4: Deliverable 4 Test Plan	Joshua Ssendagi...	-	18/Mar	27/Mar		100%
91	M13: Final Release	Rishi Singh	-	15/Apr	15/Apr		100%
92	D5: Project Documentvideo...	Rishi Singh	-	30/Mar	26/Apr		100%

Assumptions

Assumptions the team have made are that the user has appropriate physical hardware and software required to run the application. Some of the assumptions include the user having a monitor, mouse and keyboard to use the application and interreact with the program. We have also assumed the user has their own images they would like to annotate over which they can import into the application.

Adopted coding standards

Background

The programming language that we have chosen to use for our project is C++. C++ is a robust, versatile, object-oriented programming language. It presents a number of dynamic libraries to choose from that makes the code reliable, reusable and most importantly manageable.
The main goal of this document is to present to the user a description of the constraints used by the programmers for coding the project using C++. These constraints are meant to make the code base manageable. However, the coder is free to use the productive features of the language.
It has been assumed that the reader is familiar with the programming language used here, that is C++.

Goals of the Style Guide

There are a few goals that we want to achieve through this guide. These guide states all the fundamental rules that one must follow while contributing to the particular project and why we have opted these styles. The goals that we have strived to achieve through this guide are as stated below:

- The style guide is large enough to justify asking of all engineers to remember it.
- This coding style is mainly optimized for the reader than the writer. For a person who wants to contribute towards this project, the style guide should make it transparent for him/her to understand the format in which the code has been written and will enable him/her to make sense from the existing code.
- The style guide enables to maintain consistency throughout the code. This will in turn make debugging easier and will also allow automation. Maintaining the same style throughout will enable the developers to get back to working on a particular part of the project easily specially in case of sprints.
- The guide encourages one to avoid surprising or dangerous constructs. There is a high bar for style guide waivers on such restrictions, because waiving such rules often directly risks compromising program correctness.
- The guide encourages one to avoid constructs that average C++ programmers will find hard to maintain.
- The guide enables one to concede to optimization when necessary. Performance optimizations can sometimes be necessary and appropriate, even when they conflict with the other principles of this document.

C++ Version

The code should target the current version of C++ which is C++17 at the moment. It should be noted that the targeted version by the guide might advance over time.

Header Files

For each .cpp file, there should be a .h file except for the unit tests and for small .cpp files that contains just a main() function. Correct use of header files can enhance the readability, size and overall performance of the code.

Self-Contained Headers

Header files should be able to compile on their own and end in .h. A header file should have header guards and all the other headers it needs so that the users or refactoring tools won't require special conditions to include the header.
The definitions for the templates and functions should be placed in the same file as their declarations. Their definitions should be included into every .cpp file that uses them else the program might fail to link in some build configurations.

The #define Guard

All header files should have #define guards to prevent multiple inclusion. The format of the symbol name should be <PROJECT>_<PATH>_<FILE>_H_. To guarantee uniqueness, they should be based on the full path in a project's source tree.
Figure 1: #define Guard example from code snippet:

```
1  /// defining the header files
2
3  #ifndef ANNOTATIONS_H
4  #define ANNOTATIONS_H
5
```

Forward Declarations

Try to reduce the use of forward declarations and try using #include to include the headers wherever possible.

Figure 2: Forward Declarations

```
36
37  private slots:
38      void on_but_openImage_clicked();
39      void on_but_CreateClass_clicked();
40      void on_but_RemoveClass_clicked();
41      void on_checkBox_Date_clicked(bool checked);
42      void on_checkBox_File_clicked(bool checked);
43      void on_sort_Ascending_clicked();
44      void on_sort_Descending_clicked();
45      void on_but_sortClass_Ascending_clicked();
46      void on_but_sortClass_Descending_clicked();
47      void on_comboBoxImages_activated(const QString &arg1);
48      void on_comboBox_ChooseShape_activated(const QString &arg1);
49      QString on_classBrowse_clicked();
50      void on_comboBox_SetColour_activated(const QString &arg1);
51      void on_spinBoxPenWidth_valueChanged(int arg1);
52      void on_spinBox_RectHeight_valueChanged(int arg1);
53      void on_spinBox_RectWidth_valueChanged(int arg1);
54      void on_spinBox_Square_valueChanged(int arg1);
55      void on_pushButton_Help_clicked();
56      void on_pushButton_HelpSquare_clicked();
57      void on_spinBoxVerticies_valueChanged(int arg1);
58      void on_CreateClassFile_clicked();
59      void on_AnnotatedFiles_clicked();
60      void on_SaveImageFile_clicked();
61      void on_pushButton_addtext_clicked();
```

Definition:

In computer programming a forward declaration is a declaration of an identifier for which the programmer hasn't yet given a complete definition.

Advantages:

- Forward declarations save compile time. The complier has to open more files and process more inputs when using header files.
- Forward declarations prevent unnecessary compilations. Header files may force the code to get compiled more than necessary in case any changes are made to them.

Disadvantages:

- Forward declarations might prevent the users from making necessary compilations when changes are made in the headers.
- Forward declarations can prevent the header owners from making compatible changes to their APIs.
- Forward declaring symbols from namespace std:: yields undefined behaviour.
- Sometimes replacing a #include with the forward declaration can change the meaning of the piece of code.

- In case multiple symbols need forward declaration, it is best to switch that with an #include header file.
- Structuring code to enable forward declarations can make the code slower and more complex.

Inline Functions

Only small functions are defined inline.
Definition:
The inline functions are a C++ enhancement feature to increase the execution time of a program. Functions can be instructed to compiler to make them inline so that compiler can replace those function definitions wherever those are being called.
Advantages:
Inline functions can generate more efficient object code.
Disadvantages:
Overuse of inlining can make the program to run slower. Inlining small functions will decrease the code size but inlining a large function will increase the code size. This in turn will make a modern processor to run slower than usual due to the instruction cache.

Names and Order of Includes

The headers should be included in the following order:

- Related header
- C++ standard library headers
- Other libraries' headers (if necessary)
- The project's headers

All of a project's header files should be listed as descendants of the project's source directory without use of UNIX directory aliases. (the current directory) or `..` (the parent directory).
For each section, the includes must be ordered alphabetically.
Include the headers that define the symbols that are used in the program except in the unusual cases of forward declaration.
Sometimes, system-specific code needs conditional includes. Such code can put conditional includes after other includes. The system-specific code should be small and localized.

Figure 3: Name and order of includes

```
4   #define MAINWINDOW_H
5
6   /// Necessary header files gets included
7
8   #include <QMainWindow>
9   #include <QListWidget>
10  #include <QVector>
11  #include <QMessageBox>
12  #include <QPainter>
13  #include "shapes.h"
14
```

Scoping

Namespaces

Considering a few exceptions, try to place the code in a namespace. Try to give a unique name to the namespace based on the name of the project and its path. Avoid using directives or inline namespaces.

Definition:

Namespaces subdivide the global scope into distinct, named scopes, and so are useful for preventing name collisions in the global scope.

Advantages:

- Namespaces prevent names conflict in larger programs and allow the code to use reasonable short names.
- Inline namespaces automatically place their names in the enclosing scope.

Disadvantages:

- Namespaces can be confusing, because they complicate the mechanics of figuring out what definition a name refers to.
- Inline namespaces, in particular, can be confusing because names aren't actually restricted to the namespace where they are declared. They are only useful as part of some larger versioning policy.
- In some contexts, it's necessary to repeatedly refer to symbols by their fully-qualified names. For deeply-nested namespaces, this can add a lot of clutter.

Unnamed Namespaces and Static Variables

If definitions in a C++ file don’t need external referencing then place them in an unnamed namespace or declare them static and do not use them in the header files.

Definition:

Static is a keyword in C++ used to give special characteristics to an element. Static elements are allocated storage only once in a program lifetime in static storage area. And they have a scope till the program lifetime. Static Keyword can be used with following, Static variable in functions.

Nonmember, Static Member, and Global Functions

Place non-member functions in a namespace and avoid using global functions. A class should not be used to simply group static functions. Static methods of a class should generally be closely related to instances of the class or the class's static data.

Advantages:

Non-member and static member functions can be useful in some situations. Putting non-member functions in a namespace avoids polluting the global namespace.

Disadvantages:

Non-member and static member functions may make more sense as members of a new class, especially if they access external resources or have significant dependencies.

Local Variables

The variables used in a function should be placed in the narrowest scope and they should be initialized with declarations.

C++ enables the programmer to declare the variables anywhere. However, for convenience of the readers, it should be declared closer to where it has been used and within a local scope.
Variables needed for the branch statements and the loops should be declared within them so that they are confined to their scopes.
If the variable is an object, its constructor is invoked every time it enters scope and is created, and its destructor is invoked every time it goes out of scope.
Figure 4: Local Variables

```
67  void MainWindow::SortImages_Ascending()
68  {
69      /** check to see if the ui->listWidget_images field is empty */
70      if(ui->listWidget_images->count() == 0) return QMessageBox::about(this
71      try /// try-catch block for handling exceptions
72      {
73          ui->listWidget_images->clear();
74          int size;
75          size = mylist_Image.count();
76          /// sorting image file names using bubble sort
77          for(int i=0; i< size; i++)
78              {
79                  for(int j=0; j< size-1; j++)
80                  {
81                      if(mylist_Image[j]>mylist_Image[j+1])
82                      {
```

Static and Global Variables

Static variables are only declared when they are absolutely necessary for the program.
Definition:
Static is a keyword in C++ used to give special characteristics to an element. Static elements are allocated storage only once in a program lifetime in static storage area. And they have a scope till the program lifetime. Static Keyword can be used with following.

- Static variable in functions
- Static Class Objects
- Static member Variable in class
- Static Methods in class

Static variables when used inside function are initialized only once, and then they hold their value even through function calls. These static variables are stored on static storage area , not in stack.

Static keyword works in the same way for class objects too. Objects declared static are allocated storage in static storage area, and have scope till the end of program.

Static objects are also initialized using constructors like other normal objects. Assignment to zero, on using static keyword is only for primitive datatypes, not for user defined datatypes.

Static data members of class are those members which are shared by all the objects. Static data member has a single piece of storage, and is not available as separate copy with each object, like other non-static data members.

Static member variables (data members) are not initialized using constructor, because these are not dependent on object initialization.

Also, it must be initialized explicitly, always outside the class. If not initialized, Linker will give error.

Static member functions can also be declared which work for a class as whole rather for a particular object of the class.

Advantages:

Global and static variables are very useful for a large number of applications: named constants, auxiliary data structures internal to some translation unit, command-line flags, logging, registration mechanisms, background infrastructure, etc.

Disadvantages:

Global and static variables having dynamic initialization or having non-trivial destructors create complexity that makes it hard to find bugs.

Decision on Destructors

When destructors are trivial, their execution is not subject to ordering at all otherwise we are exposed to the risk of accessing objects after the end of their lifetime. Therefore, we only allow objects with static storage duration if they are trivially destructible. Fundamental types (like pointers and int) are trivially destructible, as are arrays of trivially destructible types.

Decision on initialization

Constant initialization is always allowed. Any non-local static storage duration variable that is not so marked should be presumed to have dynamic initialization, and reviewed very carefully.

Dynamic initialization of nonlocal variables is discouraged, and in general it is forbidden. However, we do permit it if no aspect of the program depends on the sequencing of this initialization with respect to all other initializations. Under those restrictions, the ordering of the initialization does not make an observable difference.

thread_local Variables

thread_local variables that don't have a declaration inside a function should be initialized with a true compile-time constant.

Definition:

Thread-local storage can be created using the thread_local keyword. A variable declared with the thread_local specifier is said to have thread storage duration**.**

- Each thread in a program has its own copy of each thread-local variable.
- A thread-local variable with function (local) scope will be initialized the first time control passes through its definition. Such a variable is implicitly static, unless declared extern.
- A thread-local variable with namespace or class (non-local) scope will be initialized as part of thread start up.

- Thread-local variables are destroyed upon thread termination.
- A member of a class can only be thread-local if it is static. There will therefore be one copy of that variable per thread, rather than one copy per (thread, instance) pair.

Advantages:

- Only one thread can access a thread-local data. This makes it useful for concurrent programming.
- thread_local is the only formal way of creating a thread-local data.

Disadvantages:

- Accessing a thread_local variable may trigger execution of an unpredictable and uncontrollable amount of other code.
- thread_local variables are effectively global variables, and have all the drawbacks of global variables other than lack of thread-safety.
- The memory consumed by a thread_local variable scales with the number of running threads (in the worst case), which can be quite large in a program.
- An ordinary class member cannot be thread_local.
- thread_local may not be as efficient as certain compiler essentials.

Classes

Classes are the basic unit of code in C++ and are used widely throughout the project. This section lists the main constraints one must follow while working with the classes.

Doing Work in Constructors

Virtual method calls should be avoided in a constructor and any such initialization must be avoided that may cause a signal error.

Definition:

Arbitrary initialization can be performed in the body of the constructor.

Advantages:

- One does not have to be worried even if the class has not been initialized.
- Objects that are fully initialized by constructor call can be const and may also be easier to use with standard containers or algorithms.

Disadvantages:

- If the work calls virtual functions, these calls will not get dispatched to the subclass implementations. Future modification to your class can quietly introduce this problem even if your class is not currently subclassed, causing much confusion.
- There is no easy way for constructors to signal errors, short of crashing the program (not always appropriate) or using exceptions (which are forbidden).

- If the work fails, we now have an object whose initialization code failed, so it may be an unusual state requiring a bool IsValid() state checking mechanism (or similar) which is easy to forget to call.
- You cannot take the address of a constructor, so whatever work is done in the constructor cannot easily be handed off to, for example, another thread.

Figure 5: Constructors

```
14  /** function for setting up the ui */
15  MainWindow::MainWindow(QWidget *parent)
16      : QMainWindow(parent)
17      , ui(new Ui::MainWindow)
18  {
19      ui->setupUi(this);
20  }
```

Decision:

Constructors should never call virtual functions. If appropriate for your code , terminating the program may be an appropriate error handling response. Otherwise, consider a factory function or Init() method. Avoid Init() methods on objects with no other states that affect which public methods may be called (semi-constructed objects of this form are particularly hard to work with correctly).

Implicit Conversions

Do not define implicit conversions. Use the explicit keyword for conversion operators and single-argument constructors.

Definition:

Implicit conversions allow an object of one type (called the source type) to be used where a different type (called the destination type) is expected, such as when passing an int argument to a function that takes a double parameter.

In addition to the implicit conversions defined by the language, users can define their own, by adding appropriate members to the class definition of the source or destination type. An implicit conversion in the source type is defined by a type conversion operator named after the destination type (e.g. operator bool()). An implicit conversion in the destination type is defined by a constructor that can take the source type as its only argument (or only argument with no default value).

The explicit keyword can be applied to a constructor or (since C++11) a conversion operator, to ensure that it can only be used when the destination type is explicit at the point of use, e.g. with a cast.

Advantages:

- Implicit conversions can make a type more usable and expressive by eliminating the need to explicitly name a type when it's obvious.
- Implicit conversions can be a simpler alternative to overloading, such as when a single function with a string_view parameter takes the place of separate overloads for std::string and const char*.
- List initialization syntax is a concise and expressive way of initializing objects.

Disadvantages:

- Implicit conversions can hide type-mismatch bugs, where the destination type does not match the user's expectation, or the user is unaware that any conversion will take place.
- Implicit conversions can make code harder to read, particularly in the presence of overloading, by making it less obvious what code is actually getting called.
- Constructors that take a single argument may accidentally be usable as implicit type conversions, even if they are not intended to do so.
- When a single-argument constructor is not marked explicit, there's no reliable way to tell whether it's intended to define an implicit conversion, or the author simply forgot to mark it.
- It's not always clear which type should provide the conversion, and if they both do, the code becomes ambiguous.
- List initialization can suffer from the same problems if the destination type is implicit, particularly if the list has only a single element.

Decision:
Type conversion operators, and constructors that are callable with a single argument, must be marked explicit in the class definition. As an exception, copy and move constructors should not be explicit, since they do not perform type conversion. Implicit conversions can sometimes be necessary and appropriate for types that are designed to transparently wrap other types. In that case, contact your project leads to request a waiver of this rule. Constructors that cannot be called with a single argument may omit explicit. Constructors that take a single std::initializer_list parameter should also omit explicit, in order to support copy-initialization.

Inheritance

When using inheritance, make it public.
Definition:
When a sub-class inherits from a base class, it includes the definitions of all the data and operations that the base class defines. "Interface inheritance" is inheritance from a pure abstract base class (one with no state or defined methods); all other inheritance is "implementation inheritance".
Advantages:

- Implementation inheritance reduces code size by re-using the base class code as it specializes an existing type.
- Inheritance is a compile-time declaration, you and the compiler can understand the operation and detect errors.

Disadvantages:

- For implementation inheritance, because the code implementing a sub-class is spread between the base and the sub-class, it can be more difficult to understand an implementation.

Figure 6: Access Control

```
24  class shapes : public QLabel
25  {
26  public:
27      explicit shapes(QWidget * parent = 0);
28      int mPX,mPY,mP1X,mP1Y,mP2X, mP2Y,mP3X,mP3Y,mP4X,mP4Y,mP5X,r
29      int rectHeight = 100; /// size of rect
30      int rectWidth = 200;
31      int squareDimension = 120;
32      QPointF pointsTriangle[3],pointsTrap[4],pointsPent[5],poin
33      QColor lineColour;
34      int penWidth = 4; /// width of pen
35      bool mFirstClick,mSecondClick,mThirdClick,mFourthClick,mFi
36      bool mPaintFlag; /// boolean for draw
37      int shapeIndex; /// shape type
38      int verticies; /// number of vertices
39      QString class_text;
40
41  protected:
42      /// shapes();
43      void mousePressEvent(QMouseEvent *event);
44      void paintEvent(QPaintEvent *event);
45      QString pointName(int points);
46
47  };
48
```

Declaration Order

Similar declarations should be grouped under the same access specifiers. The class definition should usually start with the declaration of the public members followed by the private or protected members. Large method definitions should not be put as inline in the class definitions.

Functions

Functions are used throughout the code in order to represent and implement a modular and object-oriented approach. Different types of functions are used based on the requirements of the project.

The coder must follow certain rules while using functions in the code:

1. The function name should match their purpose.
2. Functions should be declared in the headers before being used in the source file.
3. It is better to return values rather than using output parameters.
4. Functions should not be too lengthy.
5. Try to keep the functions intact to only a particular functionality.
6. Try to use different functions for implementing functionality of different parts of the UI.
7. In case one function has to implement several operations for a particular task, use function overloading.

Using small functions helps in easy debugging of the program.

Figure 7: Functions

```
587  /** function for displaying the text when ui->pushButton_HelpSquare is clicked */
588  void MainWindow::on_pushButton_HelpSquare_clicked()
589  {
590      try /// try-catch for handling exceptions
591      {
592          QMessageBox message;
593          message.setText("If you are unabe to see a square on the image please choo
594          message.exec();
595      } catch (...) {
596          qDebug()<<"Handling exceptions not caught in slot";
597      }
598  }
599
```

Output Parameters

The output of a C++ function is naturally provided via a return value and sometimes via output parameters.

Prefer using return values over output parameters: they improve readability, and often provide the same or better performance.

Parameters are either input to the function, output from the function, or both.

Write Short Functions

Prefer small and focused functions. We recognize that long functions are sometimes appropriate, so no hard limit is placed on functions length. Even if your long function works perfectly now, someone modifying it in a few months may add new behaviour. This could result in bugs that are hard to find. Keeping your functions short and simple makes it easier for other people to read and modify your code. Small functions are also easier to test.

You could find long and complicated functions when working with some code. Do not be intimidated by modifying existing code: if working with such a function proves to be difficult, you find that errors are hard to debug, or you want to use a piece of it in several different contexts, consider breaking up the function into smaller and more manageable pieces.

Exceptions

We have used try-catch block to implement exception handling in our code. We have also used if statements in order to put checks on certain aspect of our project.

The try-catch block is used to handle any generic exceptions arising in a particular slot in any class.

The if statements are used to put checks on specific aspect of that particular slot. This means that the if statement could put a check within a try catch block itself.

Advantages:

- Exceptions allow higher levels of an application to decide how to handle "can't happen" failures in deeply nested functions, without the obscuring and error-prone bookkeeping of error codes.
- Exceptions are used by most other modern languages. Using them in C++ would make it more consistent.
- Exceptions are really handy in testing frameworks.

Disadvantages:

- When you add a throw statement to an existing function, you must examine all of its transitive callers. Either they must make at least the basic exception safety guarantee, or they must never catch the exception and be happy with the program terminating as a result.
- More generally, exceptions make the control flow of programs difficult to evaluate by looking at code: functions may return in places you don't expect. This causes maintainability and debugging difficulties.
- Turning on exceptions adds data to each binary produced, increasing compile time (probably slightly) and possibly increasing address space pressure.
- The availability of exceptions may encourage developers to throw them when they are not appropriate or recover from them when it's not safe to do so.

Decision:
On their face, the benefits of using exceptions outweigh the costs, especially in new projects. However, for existing code, the introduction of exceptions has implications on all dependent code. If exceptions can be propagated beyond a new project, it also becomes problematic to integrate the new project into existing exception-free code.

Naming and Naming Style

General Naming Rules

The names used for the objects in the code should be relevant to their purpose. This enables the coder as well as a contributor to easily identify an object based on their need. IN the long run when the code eventually gets more complex and lengthier, it helps in easy identification of the objects that are required to make any necessary changes. Naming should be given priority by all the contributors and coders. The name should reflect the purpose of an object used.

File Names

Filenames should be all lowercase and can include underscores (_) or dashes (-). Follow the convention that your project uses. If there is no consistent local pattern to follow, prefer "_".

Type Names

Type names start with a capital letter and have a capital letter for each new word.
The names of all types — classes, structs, type aliases, and type template parameters — have the same naming convention. Type names should start with a capital letter and have a capital letter for each new word.

Variable Names

The names of variables (including function parameters) and data members are all lowercase, with or without underscores between words. Data members of classes (but not structs) additionally have trailing underscores.

Constant Names

Variables declared constexpr or const, and whose value is fixed for the duration of the program, are named with mixed case. Underscores can be used as separators in the rare cases where capitalization cannot be used for separation. All such variables with static storage duration. This convention is optional for variables of other storage classes, e.g. automatic variables, otherwise the usual variable naming rules apply.

Function Names

Regular functions have mixed case; accessors and mutators may be named like variables. Ordinarily, functions should start with a capital letter and have a capital letter for each new word with or without an underscore between them.
Accessors and mutators (get and set functions) may be named like variables. These often correspond to actual member variables, but this is not required.

Figure 8: Function Names

```
73      void add_to_combobox(QString fileName);
74      void add_to_listWidget(QString fileName);
75      void displayImage(QImage image);
76      void removeDuplicates();
77      void SortImages_Ascending();
78      void SortImages_Descending();
79      void SortDates_Ascending();
80      void SortDates_Descending();
81      void SortClasses_Ascending();
82      void SortClasses_Descending();
83      void addShapes_To_comboBox();
84  };
```

Namespace Names

Namespace names are all lower-case. Top-level namespace names are based on the project name. Avoid collisions between nested namespaces and well-known top-level namespaces. The name of a top-level namespace should usually be the name of the project or team whose code is contained in that namespace. The code in that namespace should usually be in a directory whose base name matches the namespace name (or in subdirectories thereof).

Comments Style

Just like naming, comments are also very essential towards making our code look more readable. Comments especially help a person who is looking at our code for the first time or one who will be contributing to our project.
/* */ This is used in the code when our comment exceeds multiple lines or for describing the purpose of a particular piece of code. While // this is used when we are commenting out a small operation or pointing out an identifier.

Comments Style adopted for DOXYGEN

Comments Used Before Declarations:
/// statements are used for commenting before class or function declarations in the header or source files throughout the project.

Figure 9: Comments Style

```
1   /// Defining the header files
2
3   #ifndef MAINWINDOW_H
4   #define MAINWINDOW_H
5
6   /// Necessary header files gets included
7
8   #include <QMainWindow>
9   #include <QListWidget>
10  #include <QVector>
11  #include <QMessageBox>
12  #include <QPainter>
13  #include "shapes.h"
```

Descriptive Comments:

/** … */ statements are used for describing the purpose of a function or a class or for comments that give a detailed description of a particular snippet of code and might range up to multiple lines.

Figure10: Descriptive comments

```
19  /** class MainWindow is declared
20   *  all the member functions,
21   *  all the slots and
22   *  all the variables
23   *  used for the functioning of the ui
24   */
25
26  class MainWindow : public QMainWindow
27  {
28      Q_OBJECT
```

Short Comments:

/// … are used for stating a brief description of a particular statement within the program.

Figure 11: Short Comments

```
589 {
590     try /// try-catch for handling exceptions
591     {
592         QMessageBox message;
```

Formatting

General

- Each line in the code should not be too long.
- Whitespaces must be used wherever necessary in order to make the code neater and easily readable.
- Whitespaces should be used before an operator and after an operator to prevent a statement from looking clumsy.

Functions

- Choose good parameter names.

- A parameter name may be omitted only if the parameter is not used in the function's definition.
- If you cannot fit the return type and the function name on a single line, break between them.
- If you break after the return type of a function declaration or definition, do not indent.
- The open parenthesis is always on the same line as the function name.
- There is never a space between the function name and the open parenthesis.
- There is never a space between the parentheses and the parameters.
- The open curly brace is always on the end of the last line of the function declaration, not the start of the next line.
- The close curly brace is either on the last line by itself or on the same line as the open curly brace.
- There should be a space between the close parenthesis and the open curly brace.
- All parameters should be aligned if possible.
- Default indentation is 2 spaces.
- Wrapped parameters have a 4-space indent.

Conditionals and Exceptions

- In case of conditionals like if-else or switch-case, the code must be written within curly braces in case it extends upto multiple lines. This makes the code cleaner and easier to look at for a new reader.
- In case of try-catch or exceptions, the code must be written within curly braces because it represents functionality over a block of code. Even if the block is just a line, the usage of curly braces is encouraged for maintaining consistency.

Comments

- Declarative comments /* */ should be placed before the start of a piece of code.
- Identifying comments // should be placed beside the particular line of code.
- This helps to maintain the readability of the code.
- There should be space between a literal and an operator.

Though these rules of formatting might be hampered in the test file. It is best to try abiding these rules to maintain a consistency in the format of the entire piece of code. This enhances the readability of the code.

Parting words

I would encourage anyone who is contributing to this project to take some time to go through this Coding Style guide or examine the code around the files in the projects and maintain consistency while making changes to the code. In this way, the readability of the code will not be hampered. The discontinuity in the format of the code can cause distraction to other contributors or the developers when they are making changes to the code.

Other relevant information

Risk Assessment

This risk assessment will highlight key aspects of the project which might cause the team problems during the development of the application. The table contains a number of problems which we believe as a team might hinder the performance of the team during any stage of the application. The table includes a risk, probability scale of a risk occurring, impact scale of risk, the impact on the project and prevention method.

Probability Scale: 1 – Very Unlikely, 2 – Moderately Unlikely, 3 - Likely, 4 – Moderately highly likely, 5 – Very likely.

Impact Scale: 1 – Low impact, 2 – Moderately low impact, 3 - Likely, 4 – Moderately highly impact, 5 – High impact.

Table 7: Risk Assessment

Risk	Probability (1-Very Unlikely, 5-Very Likely)	Impact (1-Low Impact, 5-High Impact)	Impact on project	Prevention method
Power failure or failure in saving.	3	5	Although sometimes this risk can be unavoidable, technical risks such as power cuts etc could potentially lead to unsaved work and progress being lost. Can potentially be a high risk, if work isn't regularly saved and backed up.	Power failure cannot be avoided. However, a loss of data can. Team members should be saving their work every 20 minutes. This should minimise the chances of loss of work.
Lack of experience.	4	3	Having a lack of experience in design the GUI and creating the software could lead to the application being completed to an unsatisfactory standard.	A lack of experience can be avoided by thorough background research and reading of lectures. Also, other resources such as the internet can be used but have to be referenced if extracts of code are taken.
Poor time management.	3	5	Poor time management could lead to the project not being completed and the project would fail to receive the timely and sufficient feedback before the project deadline.	Creating and implementing a project plan with hard deadlines will minimise the chances of poor time management.

Monitoring contribution of team members.	3	5	Continuous collaboration of each individual group member is required to deliver a successful project outcome.	All members of the group are required to contribute to the project. Contribution can be monitored by monitoring tools such as Slack.
Availability of software (resources)	4	4	Software that is not available on every operating system such as visual studios. As some members use their own PC with the adapted OS, this might be an issue when working as a group.	The team has chosen the QT creator software, which is a cross-platform open-source software. By using QT creator all the team members have access to the software from the university and at home.
Poor communication.	4	4	Lack of communication between members and lack of regular group meetings will lead to disjoined work and lack of cohesion. The meeting minutes must be kept as evidence to track the progress of the project.	Poor communication can be solved by using a communication tool such as Facebook messenger, Slack, Microsoft tool or other social media applications. Group members contribution can be monitored by monitoring tools such as Slack and Microsoft teams.
Poor project design.	3	4	A project that is based on a bad, over-complicated, design, and is wrongly produced will itself be over-complicated and will not meet certain standards. Individual tasks should be kept on track with diagrams created accordingly.	Poor project processes can be avoided by having clear project requirements and a clear project design. Having clear project requirements will allow team members to not overthink about the requirements. This should ideally minimise the number of errors in the project.
Significant changes in user requirements / introduction of new requirements.	4	5	User requirements should be fully investigated and agreed before specification. Although this could be changed over time, it's vital to have a basic UI to start the project and all the members should be informed about the changes.	The team will need to discuss any new requirements which are introduced. The team will need to choose an agile methodology which allows the smooth introduction of new requirements.
Overcomplicating the problem /	3	3	By overcomplicating the problem, the team might	The project manager should overlook all parts of the

adding too many unnecessary features.			go through a period of stagnation where they will not be able to proceed on to the next task.	development stage and stress to the developers not to overcomplicate any components.
Other module deadlines might get in the way of completing tasks.	4	3	Other module deadlines might need more attention from team members which might slow down the development of the project.	Module deadlines will need to be considered by team members, so work can be reallocated. Members of the team who might have to prioritise other deadlines will need to inform the project manager so work can be reallocated.
Breakdown of task specification.	3	3	During the initial phases of integration and coding, task specifications might be unclear. Moreover, developers may also find it to be incomplete.	Struggling to complete a task can be helped by organising group meetings. If developers are having problems solving tasks, then they should organise group meetings, so the team can go through the problem together.
Poor test plan	3	4	A poor test plan can cause there to bugs in the program which could have a poor effect on the final project.	A poor test plan can cause problems, a generous amount of time should be given to create a good test plan. By creating a test plan in the early development stages there will be fewer chances of poor testing as more focus will be given to the testing stage.
Bugs in code	4	5	Bugs in the code can cause the problems for team members which would prevent the clear requirements from being achieved.	A good test plan should be produced to avoid any bugs from escaping the attention of team members. Testing should be done incrementally of small sections of code produced.
Poor version control/ loss of software.	4	5	Code can be accidentally overwritten which could mean code might need to be rewritten leading to time being wasted. Code can also be lost which can have a poor impact on the project.	Git should be used and the software developer must document and indicate, to members of the team how they should commit to Git and how many times they are required to commit their work.

Loss of group member.	3	4	Loss of a group member can cause the project to slow down and tasks might be left incomplete.	The project manager must let their lab tutor and module leader aware of the issue. The project manager should keep on emailing the individual for evidence. Work should then be reallocated evenly between members of the team.
Illness or absences during the project development without notice.	4	4	If team members are ill or absent for a period without any notice then work will be reallocated and more pressure will be added on other members of the team.	Team members should let the project manager and their tutor know when they will be missing supervised meetings due to illness or any other reason. Team members should let the project manager know if there are to miss any casual meetings and should try to complete the work if any absences are planned. Work should be reallocated if no notice is given from members of the team.
Software conflicts when members are committing changes to the same file.	3	4	Software conflicts can cause the code becoming messy and code being lost.	Members of the team should sort out any branches they get when committing work. Otherwise, members should commit work at different times.
Low motivation for members.	2	4	Members having low motivation can cause the development stage of the project from stagnating.	Regular meetings should be held to keep members of the team on track and discuss any problems which are found.
Work balance	2	3	Members of the should not overstress about the work as they must have a healthy work balance.	If members of the team are unable to complete any work allocated due to, too much work. The project manager must reallocate the work.
Member conflicts.	2	4	During the project members may have conflicts with each other over certain aspects of the project.	The project manager will have the final decision however this should be the final resort solution. Group members should try their best to convince each other their view is right and a

				group conscience should be met.
Loss of project manager	2	5	If for some reason the project manager leaves or is not in touch with the team, the project can be derailed and other members of the team will become confused.	If the project manager leaves the project one of the group members should step up and take the responsibility. The team should document that the team has lost their project manager during the project.
Pandemic events such as the coronavirus	5	4	Due to the spread of coronavirus in the world; communication between the group can be dampened and the flow of work can be disrupted.	The team must have more than one communication tools which they can use to make voice calls with the group. The project manager must also allow extra time for tasks which might be affected by this incident.

Implementation

UML Diagrams

For the development process, our group, decided to adopt a Decorator pattern. As it allows the changes to be added to an individual object, either statically or dynamically, without affecting the other objects from the same class. For UML diagrams, class diagram and component diagrams are used to reason the structure of the Image application software. While sequence diagram, collaboration diagram, statechart diagram and activity diagram are used to specify the behaviour of this software. At the edge of the system's software and hardware, deployment diagram is used to elaborate the topology of processors and devices in which the image application software executes.

Use Case Diagram

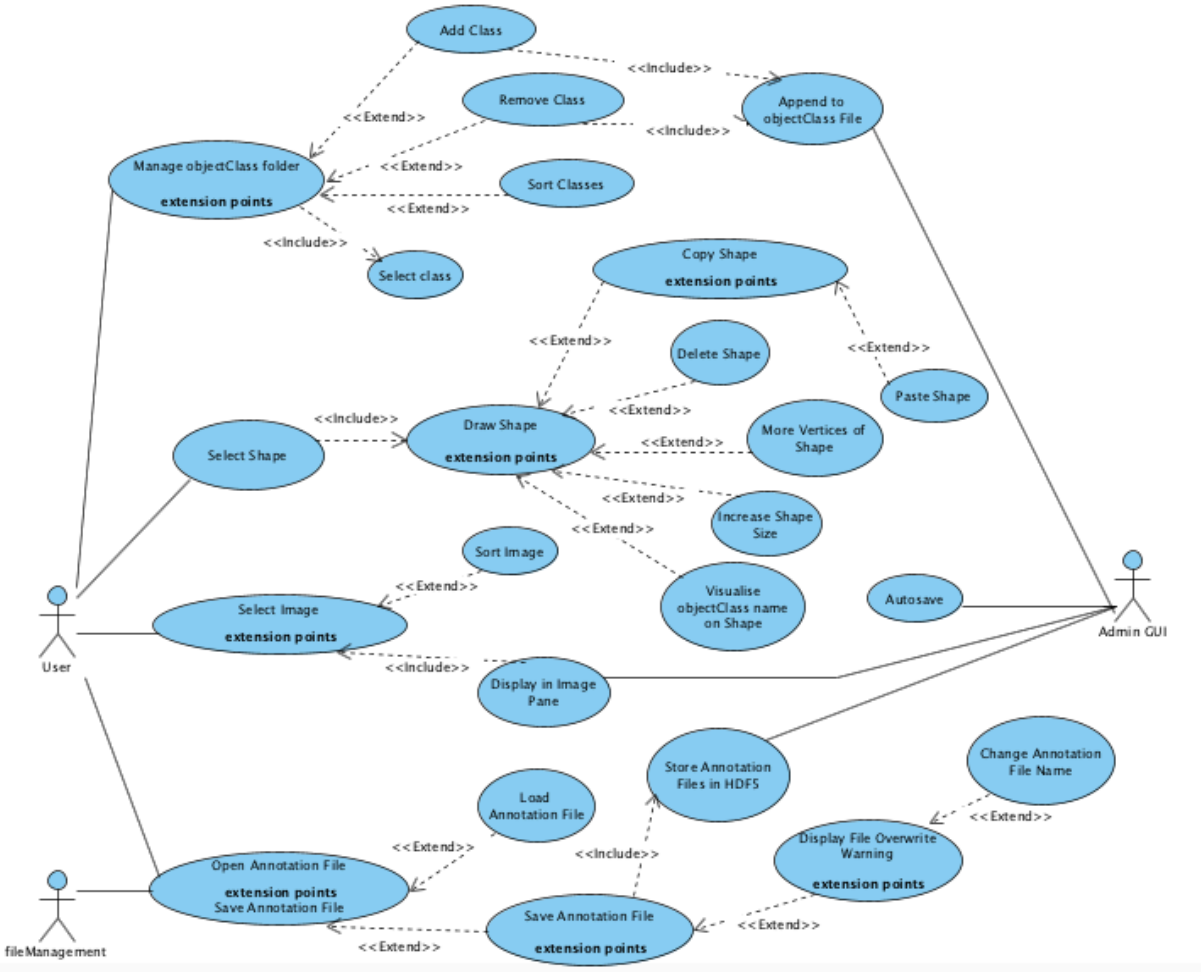


Figure 12: Use Case Diagram

The Use case diagram is a UML diagram which annotates the functionality of a system using actors and use cases. This use case was modelled for the 'Image Labelling Application'. The three actors present in this use case are: User, File Management and Admin (GUI). The User is the actor who will be using the application, File Management interacts with annotation file management and Admin (GUI) is the actual application itself. The relationship between the User and the use cases are based on the actions that the User will be able to perform

while using the Image Labelling application. In addition to this, the relationship between the Admin (GUI) and the uses cases are also based on the actions that the system will perform while the User is using the application. File Management will only interact with use cases which relate to the management of files within the system. These use cases were determined by closely following the requirements of the project. There are set requirements which explain what the User, File Management and Admin (GUI) must do.

Table 8: Use Cases

Actor(s)	Use Case	Relationship	Relationship Description
User	Manage Class Folder	Association	The User participates in Manage Class Folder, Select Shape, Select Image and Open Annotation File use cases shown through an Association relationship. These are the actions the User will perform in the application.
	Select Shape		
	Select Image		
	Open Annotation File		
Admin (GUI)	Autosave	Association	The Admin (GUI) participates in Autosave, Display Image in Image Pane, Store Annotation Files in HDF5 and Append to objectClass File shown through an Association relationship. These are the actions the Admin (GUI) will perform in the application.
	Display Image in Image Pane		
	Store Annotation Files in HDF5		
	Append to objectClass File		
File Management	Open Annotation File	Association	File Management participates in Open Annotation File use case. These are the actions File Management will perform in the application.

Use Case (1)	Relationship to	Use Case (2)	Relationship to	Use Case (3)	Relationship Description
Manage objectClass Folders	Extend	Add Class	Include	Append to objectClass File	Add Class, Remove Class and Sort Classes are extending use cases to the base use case, Manage Class Folder. While these extending use cases are optional, the base use case also includes the Select Class use case which is a compulsory action within the system. Add Class and Remove Class include the Append to objectClass File use case. The system must append changes to the objectClass file after adding or removing classes. This is also a compulsory action within the system.
	Extend	Remove Class			
	Extend	Sort Classes			
	Include	Select Class			

Use Case (1)	Relationship to	Use Case (2)	Relationship to	Use Case (3)	Relationship to	Use Case (4)
Select Shape	Include	Draw Shape	Extend	Copy Shape	Extend	Paste Shape
			Extend	Delete Shape		
			Extend	Move Vertices of Shape		
			Extend	Increase Shape Size		
			Extend	Visualise Objectlass Name on Shape		

Relationship Description
Select Shape includes the Draw Shape use case. The system must allow the user to select and draw a shape. Use cases (3) extends to the base use case, Draw Shape. These actions are options that are available when drawing a shape. Paste Shape extends to Copy Shape. For a shape to be pasted, it must be copied first.

Use Case (1)	Relationship to	Use Case (2)	Relationship Description
Select Image	Include	Draw Shape	Select Image includes the Draw Shape and Display Image in Image Pane use cases. The system must allow the user to select an image, then display that image in the image pane and allow the user to draw a shape on top of that image. Without the Select Image use case, the user will not be able to draw a shape and the image won't be displayed in the image pane.
	Extend	Sort Image	
	Include	Display Image in Image Pane	

Use Case (1)	Relationship to	Use Case (2)	Relationship to	Use Case (3)	Relationship to	Use Case (4)
Open Annotation File	Extend	Load Annotation File				
		Save Annotation File	Extend	Display File Overwrite Warning	Extend	Change Annotation File Name

Relationship Description

Load Annotation File and Save Annotation File extends to the base use case, Open Annotation File. The system will allow the user to open, load and save their annotations in an annotation file. Display File Overwrite extends to the base use case, Save Annotation File. If the user tries to save an annotation file with a file name of an annotation file that already exists, the system will display a file overwrite warning to alert the user. Change Annotation File Name extends to the base use case, Display File Overwrite Warning. Once the user has received the file overwrite warning, they will be asked to overwrite the annotation file or change the file name.

Class Diagram

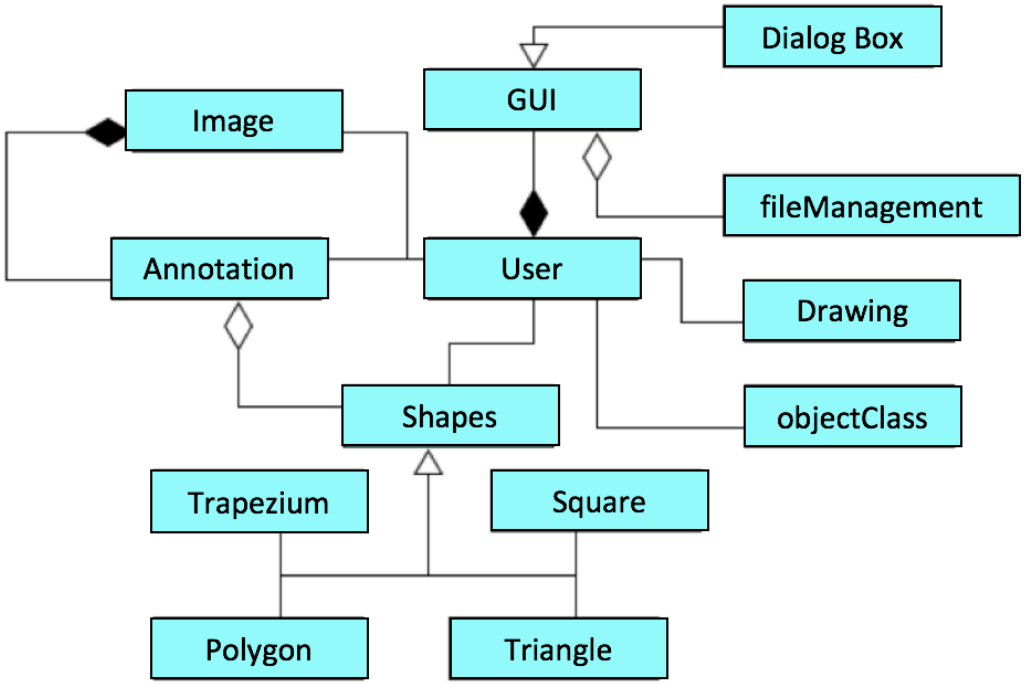
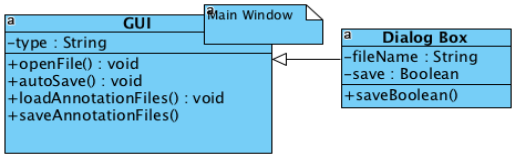
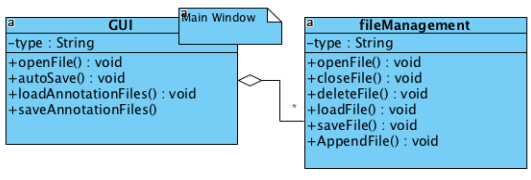


Figure 13.0: Simple Class Diagram

This simple diagram demonstrates the class diagram, which is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations and the relationship among the objects. The classes are GUI, User, fileManagement, Dialog Box, Drawing, objectClass, Shape, Circle, Square, Polygon, Triangle, Image and Annotation. User is the superclass and the main classes are User and GUI. The User have associations to the classes Image, Annotation, Shape, objectClass and Drawing. GUI has side note to specify it's the Main Window. objectClass have the side note to clarify that classes are the pictures inside the Shapes.

Table 9: Class diagram Explanation

	***Figure 13.01 interaction between the GUI and Dialog Box*** This is linked with an inheritance as Dialog Box is a child class given it inherits the same functionalities of the class GUI.
	***Figure 13.02 interaction between the GUI and FileManagement*** The formation of the class GUI as a result of fileManagement being aggregated. It is not dependent on the lifecycle of the container.

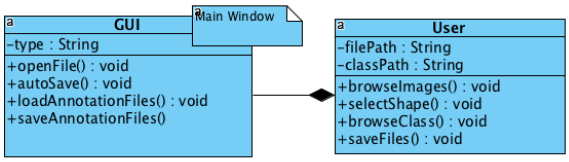


Figure 13.03 interaction between the GUI and the User

This composition relationship emphasises the dependence of the class GUI to the life cycle of the container class, User.

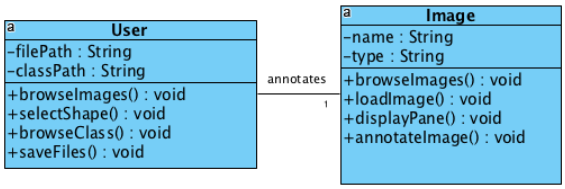


Figure 13.04 interaction between the User and Image

An association relation is established between the classes User and Image, as the User annotates the Images. The multiplicity, 1, shows there can be only one image displayed on Pane.

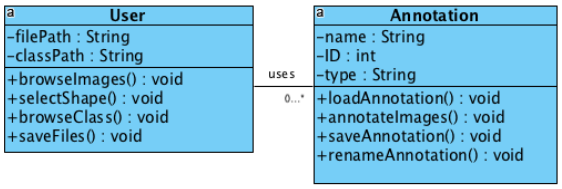


Figure 13.05 interaction between User and Annotation

An association relation is established between the classes User and Annotation, as the User uses the annotations to the given purposes. The multiplicity, 0...*, shows there can zero or more annotations saved on storage.

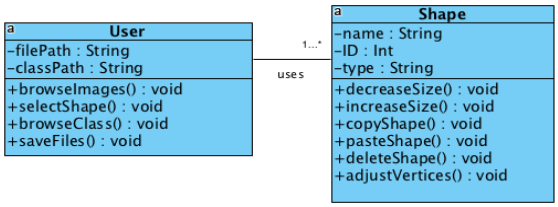


Figure 13.06 interaction between the User and Shape

An association relation is established between the classes User and Shape, as the User uses the Shapes to annotate the images. The multiplicity, 1...*, shows there can one or more shapes available to draw.

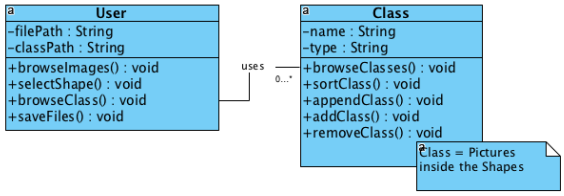


Figure 13.07 interaction between the User and Class

An association relation is established between the classes User and Class, as the User uses the Class, refers to the pictures inside the Shapes. The multiplicity, 0...*, shows there can zero or more classes.

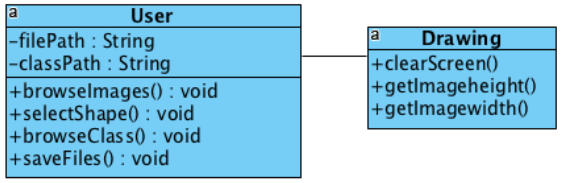


Figure 13.08 interaction between User and Drawing

An association relation is established between the classes User and Drawing, as the User requires Drawing in the process of annotating the image.

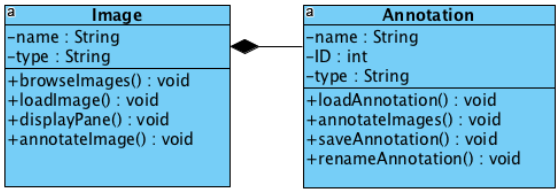


Figure 13.09 the interaction between Image and Annotation

The composition relationship emphasizes the dependence of the contained class, Annotation to the life cycle of the container class, Image. Annotation will be obliterated when the Image is destroyed.

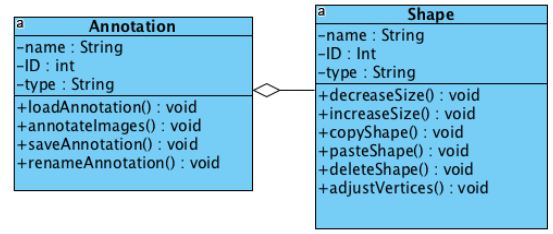


Figure 13.10 the interaction between Annotation and Shape

Aggregation shows the relationship Annotation is part of Shape. As Annotation is formed as a collection of the shapes. Objects of each of these classes have separate lifetimes.

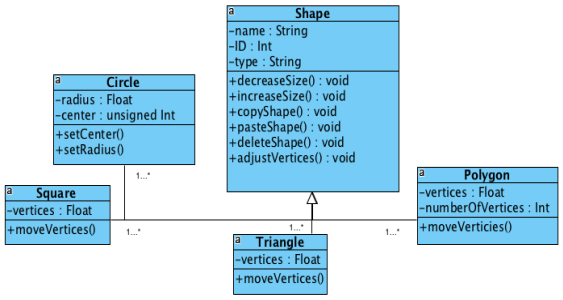


Figure 13.11 the interaction between the class Shape and subclasses Circles, Polygon, Triangle and Square

This represents an "is-a" and taxonomic relationship between Shape and its subclasses. Thus, Shapes inherits the features of the general classifiers. The multiplicity, 1...*, on each subclass shows there can one or more of the specific shape available for the User to draw.

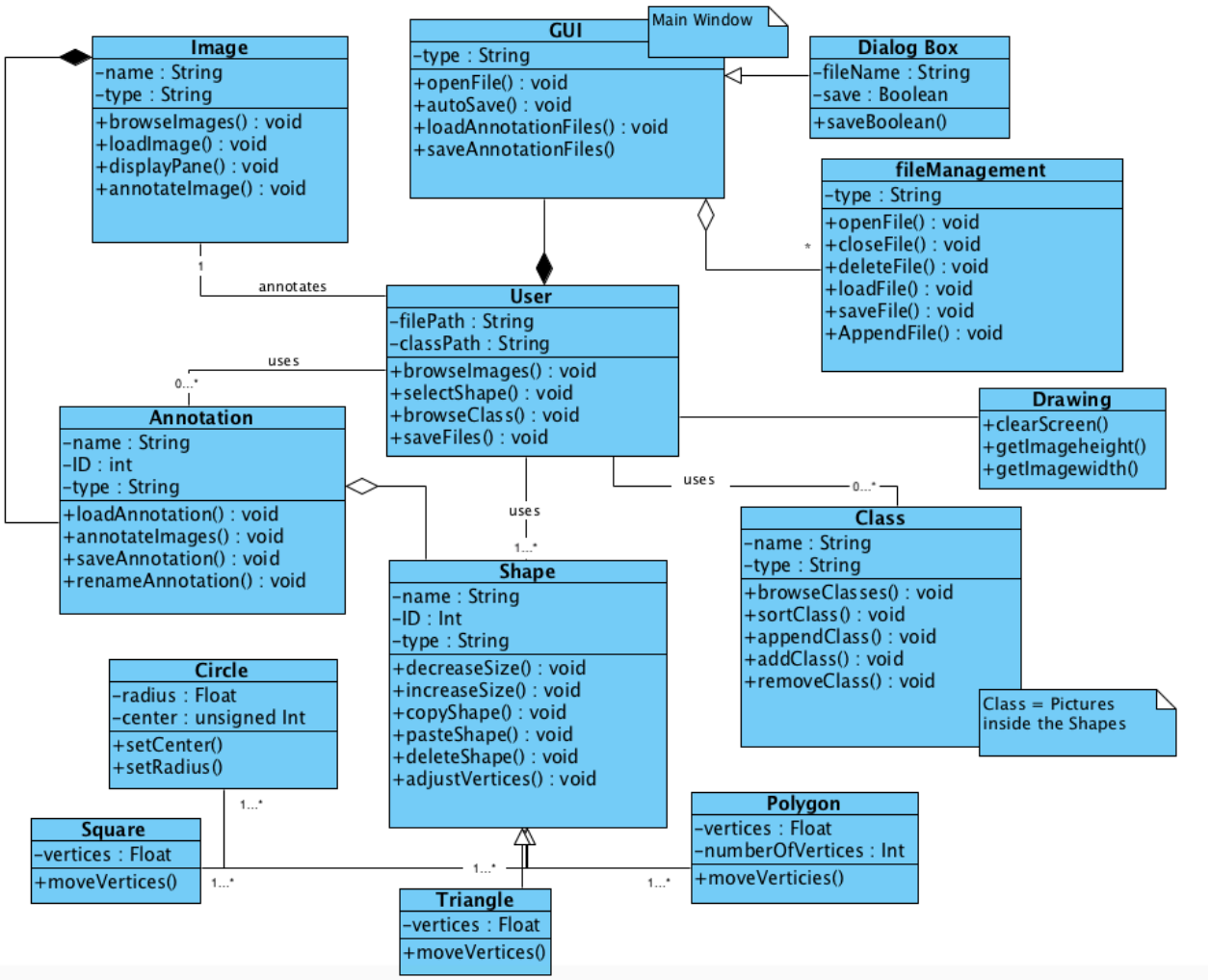
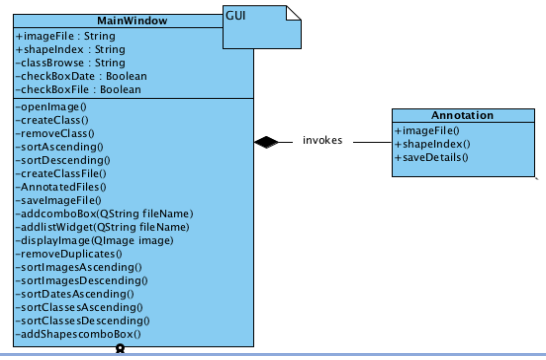
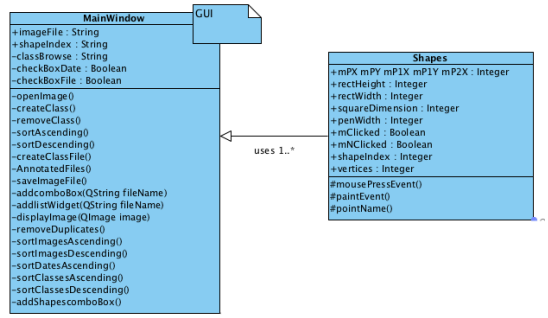
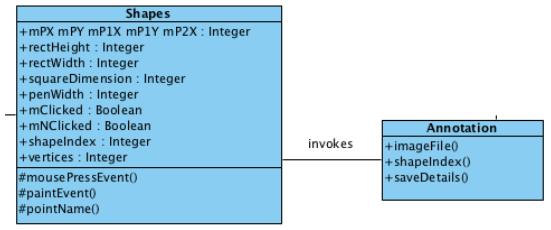
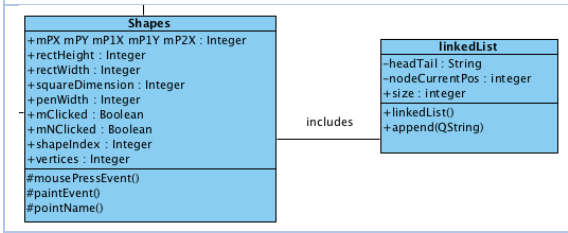


Figure 13.12 (complex) Class Diagram

Final class diagram

After continuous improvements to the code and discussion with the software developer, the final class diagram is designed as shown below. It contains three classes and the super class is the MainWindow, also addressed as the GUI.

Table 10: Final Class diagram

	***Figure 13.13 the interaction between main window and Annotation*** The composition relationship emphasizes the dependence of the contained class, Annotation to the life cycle of the container class, MainWindow. The super class MainWindow invokes the function saveDetails () from the class Annotation. While Annotation invokes the functions imageFile () and shapeIndex () form the class MainWindow
	***Figure 13.14 the interaction between MainWindow and Shapes*** This is linked with an inheritance as Shapes is a child class given it inherits the same functionalities of the class MainWindow. And MainWindow invokes coordinates mP2Y, mP3X, mP3Y, mP4X, mP4Y, mP5X, mP5Y, mP6X, mP6Y, mP7X, mP7Y, mP8X, mP8Y from class Shapes.
	***Figure 13.15 the interaction between Shapes and Annotation*** An association relation is established between the Shapes and Annotation, as Annotation invokes the function imageFile () from Shapes.
	***Figure 13.16 the interaction between Shapes and LinkedList*** An association links these two classes, as Shapes include LinkedList.

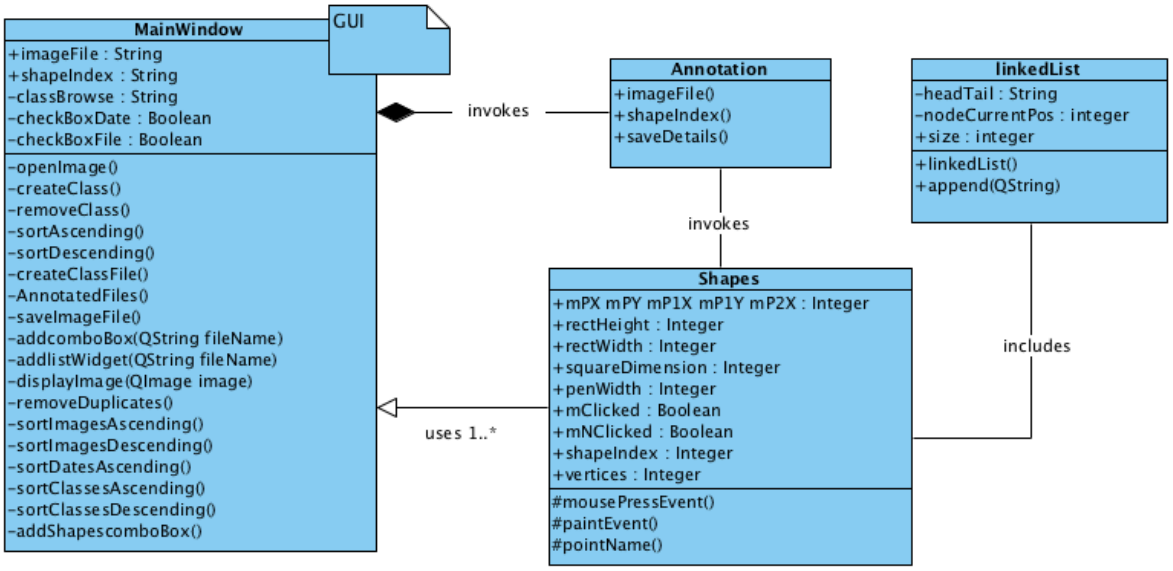


Figure 13.17 Final class diagram

Pseudocode for final class diagram

Figures 13.7 to .18 pseudocodes displays rough draft of the application program. With the help of this, the developer focused on the algorithm part of the code development. MainWindow have 2 public attributes and 3 private attributes along with listed operations. While class Shapes, 10 public attributes along with 3 protected operations. Annotation have 3 public operations.

Figure 13.18 pseudocode for class MainWindow

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

class MainWindow {

public:
        String imageFile;
        String shapeIndex;
private:
        String classBrowse;
        Boolean checkBoxDate;
        Boolean checkBoxFile;

        void openImage();

        void createClass();

        void removeClass();

        void sortAscending();

        void sortDescending();

        void createClassFile();

        void AnnotatedFiles();

        void saveImageFile();

        void addcomboBox(int QString_fileName);

        void addlistWidget(int QString_fileName);

        void displayImage(int QImage_image);

        void removeDuplicates();

        void sortImagesAscending();

        void sortImagesDescending();

        void sortDatesAscending();

        void sortClassesAscending();

        void sortClassesDescending();

        void addShapescomboBox();
};

#endif
```

Figure 13.19 pseudocode for class LinkedList.

```
#ifndef LINKEDLIST_H
#define LINKEDLIST_H

class linkedList {

private:
        String headTail;
        integer nodeCurrentPos;
public:
        integer size;

        linkedList($);

        void append(int QString);
};

#endif
```

Figure 13.20 pseudocode for class Shapes.

```
#ifndef SHAPES_H
#define SHAPES_H

class Shapes : MainWindow {

public:
        Integer mPX_mPY_mP1X_mP1Y_mP2X;
        Integer rectHeight;
        Integer rectWidth;
        Integer squareDimension;
        Integer penWidth;
        Boolean mClicked;
        Boolean mNClicked;
        Integer shapeIndex;
        Integer vertices;
        Integer mPX_mPY_mP1X_mP1Y_mP2X;

protected:
        void mousePressEvent();

        void paintEvent();

        void pointName();
};

#endif
```

Figure 13.21 pseudocode for class annotation

```
#ifndef ANNOTATION_H
#define ANNOTATION_H

class Annotation {


public:
        void imageFile();

        void shapeIndex();

        void saveDetails();
};

#endif
```

Sequence Diagram

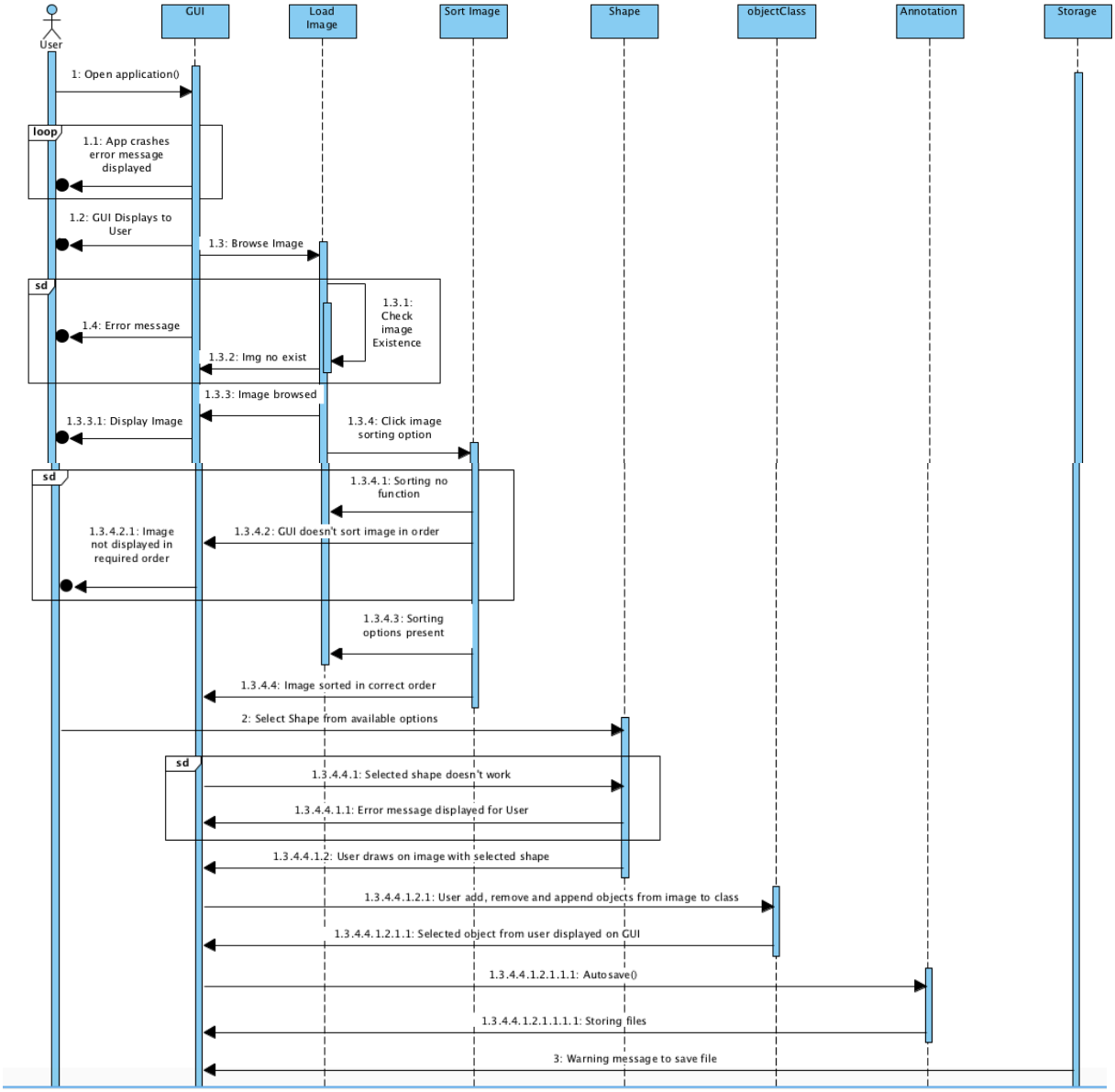
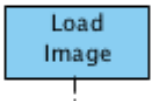
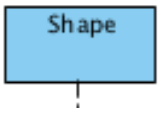
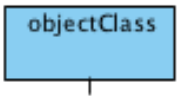
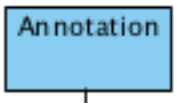


Figure 14.0 Sequence Diagram

The above figure shows the sequence diagram; it portrays the interaction of the objects in a chronological order. The only actor in here is the User, which represents the interaction with the GUI application and its objects. When the user opens the first thing displayed is the GUI.

Table 11: Sequence diagram explanation

	The lifeline GUI is the main source here, where rest of the other lifelines depend on. Therefore, it's on the all the time the application is open. If the user cannot open the GUI, it will pop up with an error message to demonstrate the issue. With the given guidelines' user can use the image application appropriately using the GUI.
	The lifeline Load Image will unable user to browse through the Image folder and choose the compatible image to be displayed on the pane. If the image doesn't exist an error message will be displayed on the GUI.
	The lifeline Sort Image will aid the user to sort the images in their desired order, either ascending or descending. The frame shows what will be displayed in GUI, if this function doesn't work.
	The lifeline Shape represents the shapes drawn on the image. After browsing through the given shapes, user can choose the one of their choice to draw on the image, then copy, paste, delete, increase and decrease the size of the shape to fit to the objects on the image. If an error occurs during this process a message will pop in the GUI.
	The lifeline objectClass will enable the user to add, remove and append class as appropriately.
	The lifelines Annotation is given to autosave annotation files for the user.
	The lifeline Storage is for saving the files. GUI will auto save the annotation files every minute, hence this lifeline is always on. In case the User closes the GUI, accidentally, a warning sign will pop up to remind the user to save their files. Lastly, the User has the chance of saving their annotation files in different file formats, such as VOC, YOLO and COCO.

State Diagram

The State Diagram for the given scenario identifies the states of the object under consideration which is the **Image Status** in this case. The diagram has been provided with two complex states along with several states in them that manifests the metamorphosis in the state of the object. The states have been labelled with the events that trigger these changes in the states. The **Shapes Status** state is linked with a state of the **Image File Status**.

Figure 15.0 State Diagram

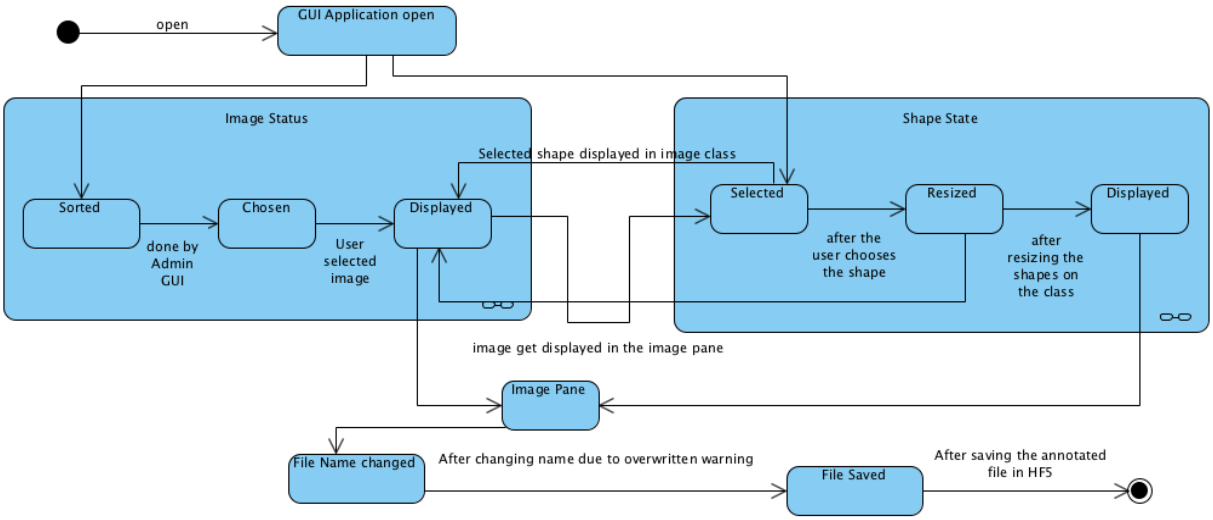
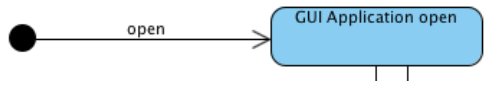
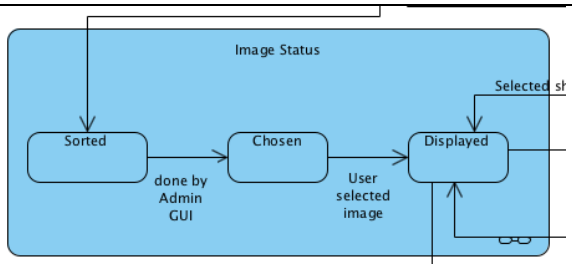


Table 9: State Diagram Explanation

	***Figure 15.1 Open to GUI*** The start marker opens the state GUI application.
	***Figure 15.2 the submachine state Image state*** This demonstrates the process of browsing an image. This complex state has three primary states under it. - **sorted:** the image file is sorted in ascending/ descending order based on its name. - **selected:** refers to the status of the image file chosen by the user. - The selected file is finally **Displayed** on the Image Pane.

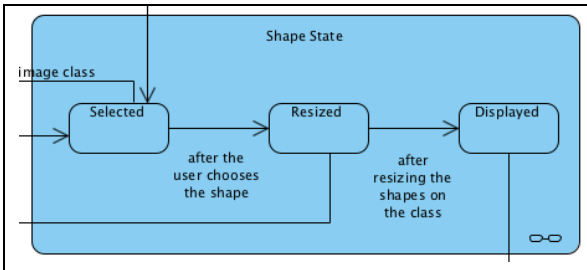


Figure 15.3 the submachine state Shape state
This figure demonstrates the 3 states under the Shape status.

- **selected:** refers to the status of the shape selected by the user.
- **resized:** the shape is adjusted according to user's requirements.
- The resized shape is **displayed** on a class of the image file with the class name on it.

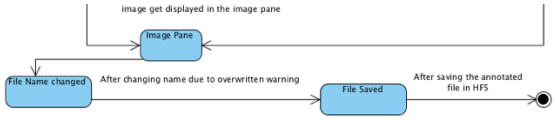


Figure 15.4 saving states
The selected image and shape will be displayed in the, state, Image pane. The transition arrow to next state, File name changed, image file needs to be changed due to overwritten warning. The final state is File saved; the annotated file will be saved. Leading to the terminator, final state.

Component diagram

The component diagram for the labelling application displays different components of the system and how they interact with each other. Components, ports and user interface interactions are shown in the diagram. The diagram provides a large system labelled as the **Labelling Application** which consist of a number of components such as GUI, image, annotations, class, draw shape and file management.

Figure 16.0 Component Diagram

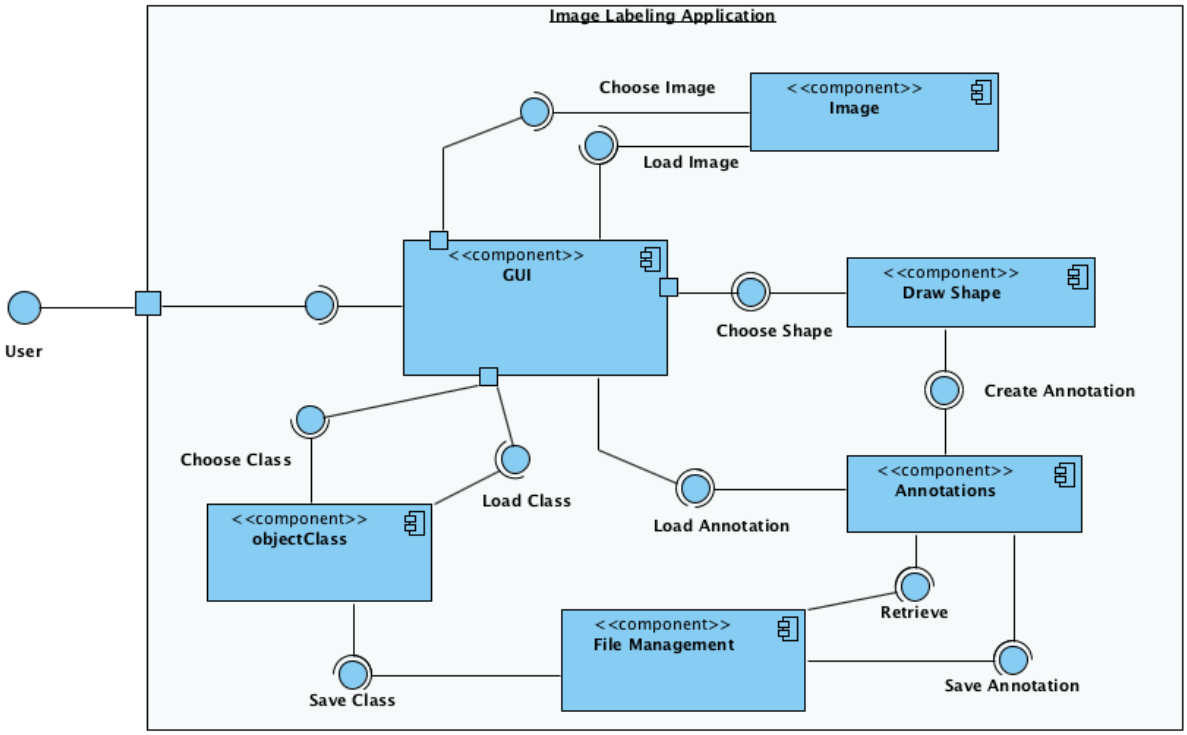


Table 12: Component Diagram explanation

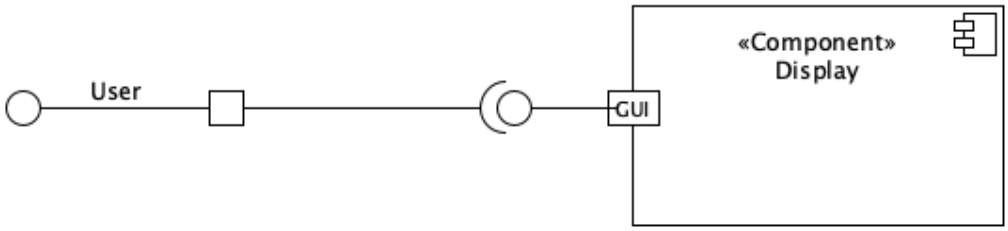


Figure 16.1 User and Display component

User and Display component

A user is required a user interface which shows the design of the application to be displayed. The GUI is an important component of the system as it interacts with all the other components.

- The user makes all the decisions via the GUI which then interacts correctly with other components depending on the decision.

- The relationship between the user and the GUI is a simple association as one user is able to function a simple GUI.

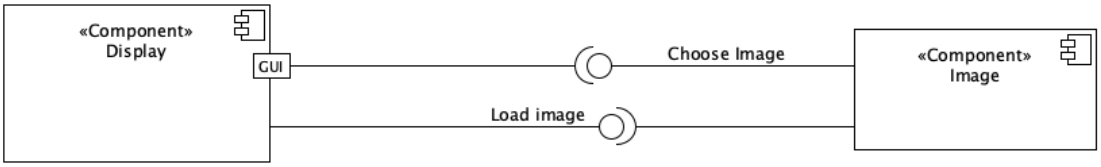


Figure 16.2 Image component

Image component

The user is able to select an image to be displayed in the GUI.

- In this interaction a user interface is provided where the user searches through a directory of their choice and chooses an image.
- Once the image is chosen, the image is loaded into the GUI for the user to see and choose other functionalities.

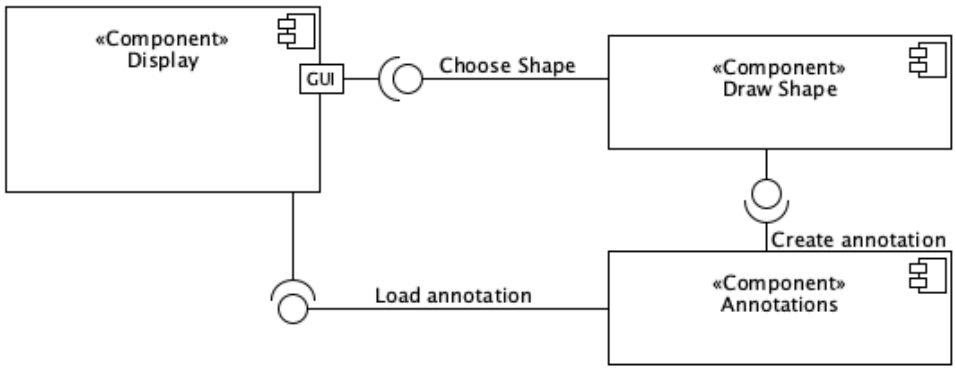


Figure 16.3 shape and annotations components

Draw shape and Annotations component

The user is able to draw shapes on an image selected in the GUI.

- From the display the user can draw a shape in order to do this a user interface allows the user to choose a shape.
- The user then draws the shape on the image which is an annotation.
- This annotation is then loaded into the display for the user see.

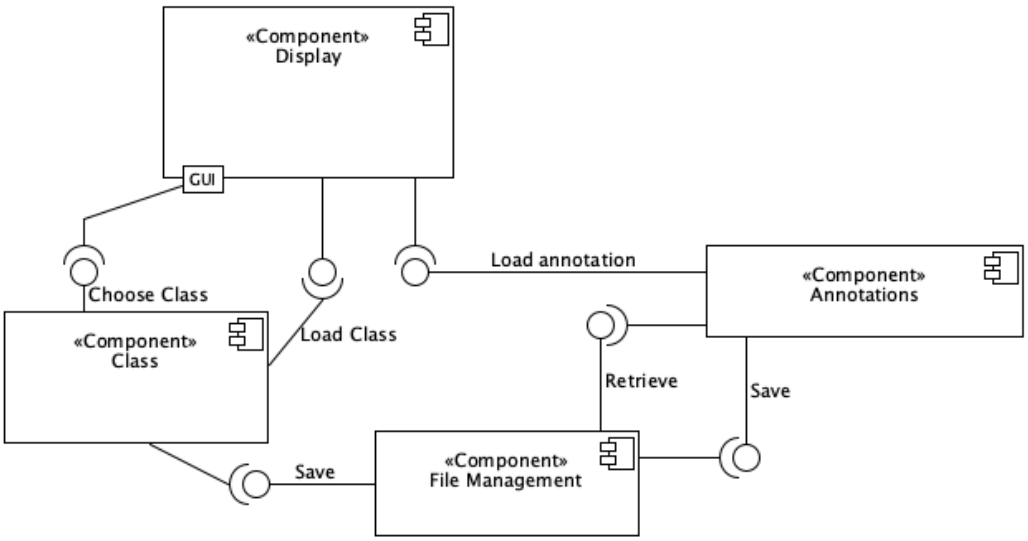
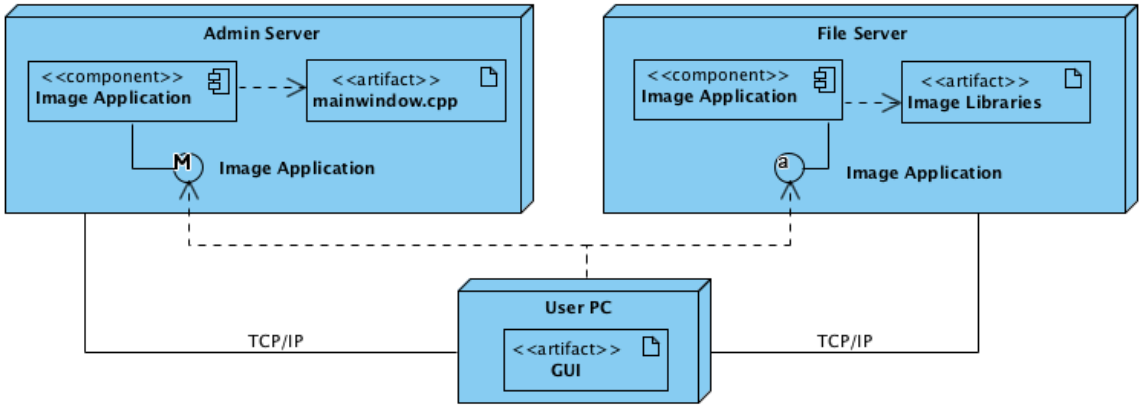


Figure 16.4 class, annotations and file management components

Deployment diagram

In the Image Labelling application, the deployment diagram is used to demonstrate the static aspect of the physical software and hardware nodes and their association and to stipulate their details of construction. It uses TCP/IP network communication protocol for connecting between the Admin Server and File server for the GUI. Both Admin and File Servers contains the following elements; component, artifact and interface. The component Image Application is the code module that indicates a software element. The artifact mainwindow.cpp is the source file, product, of the software development process. The interface Image application indicates a contractual relationship.

Figure 17.0 Deployment Diagram



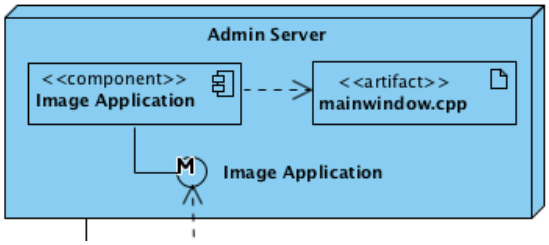


Figure 17.1 Admin Server

Admin Server
The node Admin server contains the elements component, artifact and interface. Admin Server has the overall control of the application, oversees the performance and conditions of File Server and User PC.

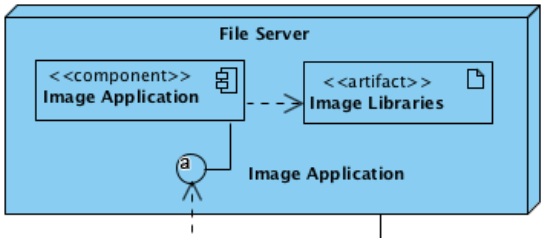


Figure 17.2 File Server

File Server
The node File server contains the elements component, artifact and interface. The File Server carries the role storing client's workstations, controlling access to separately stored files, as part of a system. It doesn't perform any computation tasks or run the application behalf of the Admin server.

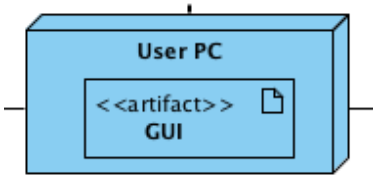


Figure 17.3 User PC

User PC
This node demonstrates the software console GUI, that will be displayed on the User's PC. This will be the main interaction of User that will be used to perform the given tasks.

Activity Diagram

This diagram is used to describe the dynamic aspects of the image application system. The initial node brings the user to the activity of opening the GUI. Browsing through the image, checking selected image's existence brings the system to the decision node. The Fork node shows if the image doesn't exist, user can browse through for another one, this image, or else error message, will be displayed in the Pane. These two activities are connected by the Join node, as shown. User selects the shape to draw on the image, fork node shows if the action cannot be completed error message will be displayed else the selected shape displayed to be drawn on the image. Join node connects these actions to save the object by entering the class name. if the application is closed, the warning message will be displayed on the GUI, saving the annotated file bring all to the activity final node.

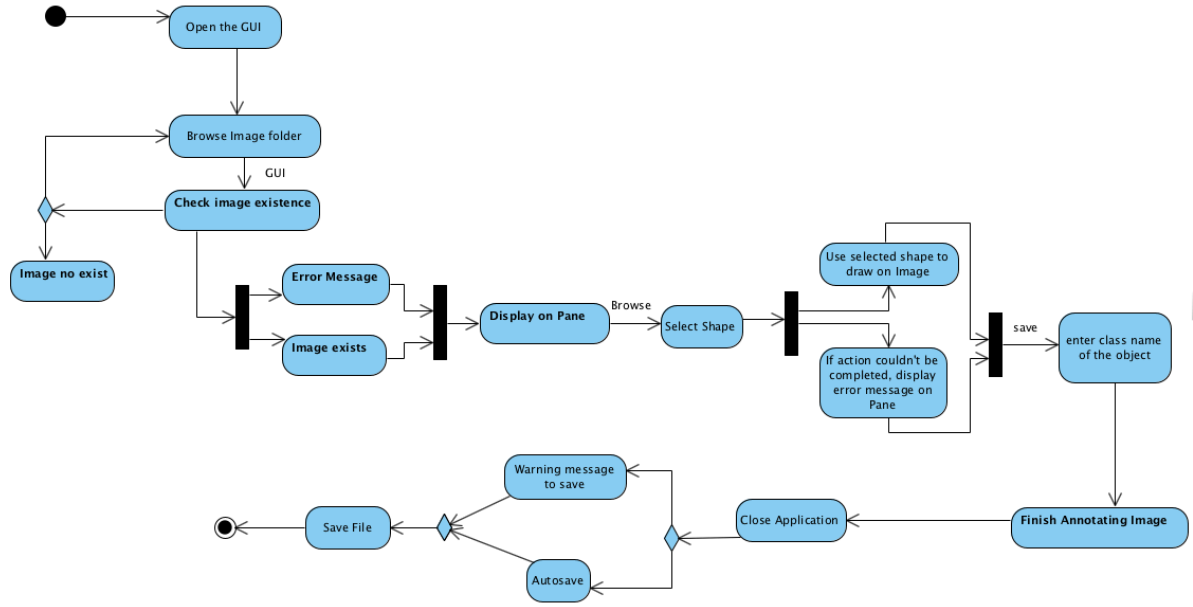


Figure 18.0 Activity Diagram

Model View Controller Architecture

For this application we adapted the MVC architecture as it has good benefits for the development process. Development becomes fast as multiple developers can work and collaborate simultaneously on the model, controller and views. It's easier to debug as multiple levels properly written in the application. Other benefits of MVC includes high cohesion, low coupling and ease of modification.

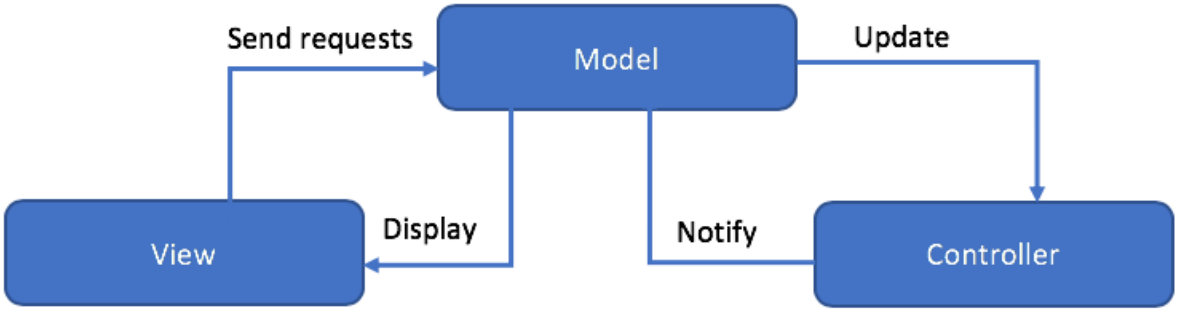


Figure 19.0 MVC design pattern

Responsibility 1 – Model – QT

Model provides the central functionality of the image application, aware of each of its dependent view and controller components. It maintains the data, allowed operations on that data and the rules that govern access to and update of the information in the application.

Responsibility 2 – View – GUI

The GUI retrieves data from the QT and updates its presentation when information is changed in one of the other views. GUI corresponds to a particular style and format of presentation of data to the user.

Responsibility 3 – Controller – System Admin

The system admin accepts user input in the form of events on each view and translates these to trigger the execution of operations within the model, QT.

Architecture Trade-off Analysis - ATAM

Software architecture, in image application, represent trade-off made by the designers. The figure 9.0, utility tree, is designed to provide a fundamental way to highlight the software architecture's approach to satisfy the following quality attributes; performance, modifiability, availability and security. These attributes are broken down into subcategories to enhance the requirements of each one. The letters H, M and L stands for High, Medium, Low. first letter characterized for the importance for the success of the system and second prioritised for the difficulty to achieve (architect's assessment)

1. Present the ATAM – the assembled project representative.
2. Present Business Drivers – these are the functional, quality attribute and critical requirements; a simple GUI interface that allows users to interact with the software system, a pane listing the compatible images files with ascending and descending orders, browse button classes selector, use given shapes to annotate and annotation file and filename selector
3. Present architecture – operating system can be either Windows or iOS. The architect has decided to use the Decoder pattern to meet the requirements.
4. Identify architectural approaches – the predominant architectural approach selected is the Model-view-Controller pattern, MVC. This pattern was chosen as it divides the way internally represented information from the way's information is presented to and acknowledged from, the user. It also separates components and prioritise code reuse.
5. Generate Quality attribute Utility tree – performance, modifiability, security and availability the selected are the most important quality goals.

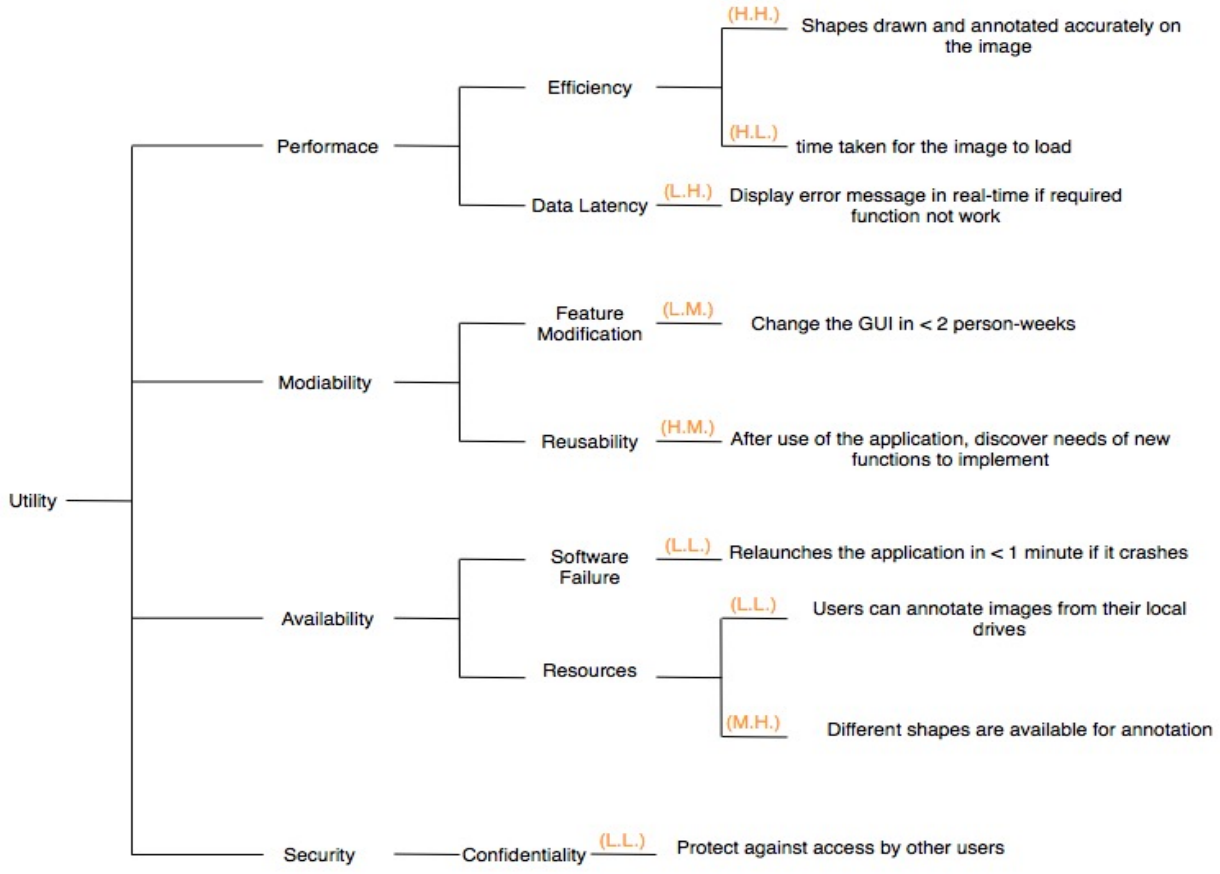


Figure 20.0 Utility tree for ATAM diagram

Internal Data Structures Used

Linked List

Reason for the choice:

- Linked Lists can grow or shrink during the execution of a program.
- Insertion is not a time-consuming operation as shifting is not necessary to insert a new element.
- Deletion is also not a time-consuming operation as left shifting is not necessary to delete elements.
- Memory can be physically released using the free function for latter allocation of nodes.
- The chances of memory wastage are minimum because memories are allocated dynamically as the requirement of the user.
- The concept of overflowing of values cannot arise due to linked list.

Usage in Project:

- Linked List has been used in the project to store the shapes.
- Whenever, a user selects a particular shape from the comboBox_ChooseShape, it gets added to the linked list and its size increases.

List (QT StringList)

Reason for the choice:

- Makes it easier to handle data with the similar data type.
- QT StringList provides in-built functions like insert(), append() and so on for adding items to the list directly without the need for a loop.
- Manipulating on the data values becomes way easier and faster.
- Accessing an element from the list becomes easier with the use of in-built indexOf() function.
- Makes it easier to implement sort and search algorithms.

Usage in Project:

- StringList has been used for storing the file names, file dates, file paths and so on.
- It has also been used for sorting the file names and dates along with the classes.

Sort Algorithm Used

Choice Made:

Bubble Sort

Reason for Choice:

- It is easy to implement bubble sort.
- Bubble sort works very efficiently for small data sets.
- Simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.
- It is very memory efficient. It requires only O(1) space, as the only extra space that it requires is that of the temporary value created when swapping adjacent elements.
- It is stable, that is the order of the identical items does not change when the list is being sorted.

Usage in Project:

Bubble sort has been used in the project for:

- Sorting the file names.
- Sorting the file dates
- Sorting the classes

In both ascending and descending order.

Explanation of working algorithm (as used in the project):

For sorting the file names, dates and classes in ascending order:

- The file names or class names or dates are stored in a String List. E.g.: StringList mylist_Image
- A temporary variable of data type QString named temp is initialized. E.g.: QString temp
- Adjacent elements of the list are compared. E.g.: if(mylist_Image[j]>mylist_Image[j+1])
- If an element is greater than the next element, then its value is stored in a temporary variable named QString temp. E.g.: temp = mylist_Image[j]
- Then the element under consideration is assigned the value of the next element. E.g.: mylist_Image[j] = mylist_Image[j+1]
- The next element now is assigned the value of temp. E.g.: mylist_Image[j+1] = temp
- In other words, the values of the 2 adjacent elements gets swapped.
- The element with a greater value is placed behind the element with a smaller value.
- If the next element in the list is greater in value than the element before it, it is skipped.
- This goes on till the entire list gets sorted in ascending order.

For sorting the file names, dates and classes in descending order:

- The reverse happens when sorting in descending. E.g.: if(mylist_Image[j]<mylist_Image[j+1])
- If an element is smaller than the next element, then its value is stored in a temporary variable named temp.
- The value of the next element is assigned to the element under consideration.

- The next element is assigned the value of temp.
- If the next element in the list is smaller than the element under consideration, then it is skipped.
- This goes on till the entire list is sorted in descending order.

Test Plan

A Test Plan is the documentation required to describe the strategy of testing within a software development project. The test plan will help to validate the development of the labelling application as well as informing all members outside of the test team the details of testing.

Product Analysis

The product that is being developed is a labelling application which allows the user to browse through a target folder, import any compatible images into an image pane and annotate the imported image. The image pane will list all compatible image files and depending on the user's preference, the application will allow the sorting of these images in descending/ascending order by filename or file date. The application allows the user to select and draw one of the following shapes for annotating the imported images: triangle, square/rectangle, trapezium and polygon (with up to 8 points). These annotations made by the user will be displayed on the image. In addition to this, while using the application, the user will be able to save and load their annotation in a dedicated folder for a later use. A class selector pane with a class selector button will be available, where the user can select existing classes or add/remove classes. The main audience of the application are users who need an image labelling application to annotate specific images. The product will use OpenCV for image processing as well as graphical libraries and frameworks like MVC through Qt creator to create the graphical user interface. Additionally, C++ programming will also be implemented for exception/error handling and data structuring as well as sorting and/or searching algorithms for image pane sorting functionality. Windows and MacOS machines will be used to create and use the product.

Test Strategy

The test strategy will highlight the project's testing objectives as well as how the team will meet these objectives. Testing efforts and cost will also be recognised.

Scope of Testing

Scope of testing will define the in scope and out of scope testing needed to test this product. The scope of the project will provide the relevant information of the testing and highlight what should be tested and what shouldn't. In scope refers to the components of the system that needs to be tested such as hardware, software and middleware. While, out of scope refers to the components of the system that will not be tested.

Table 13: Scope of Testing

In Scope Test Number	In Scope Test	In Scope Test Description
		Basic functionality Testing
1	Buttons	All buttons in the application **must** be tested to ensure their function is executed when clicked. 1.1 - The target folder which contains all compatible images will have buttons for browsing through the folder. 1.2 - The class selector pane will have a browse button for navigating through folders to select class files. 1.3 - An add and remove button will be implemented for adding and removing classes. 1.4 - An ascend and descend button will be implemented for sorting compatible images files by filename and file date. 1.5 – Open button will be implemented for opening annotation files. 1.6 – Save button will be implemented for saving annotation files. 1.7 – Confirm button will be implemented to confirm user requests. For example, button to confirm the overwriting of a file if it already exists. 1.8 – Cancel button will be implemented to cancel user requests. 1.9 – Help button could be implemented to help the user navigate the application.
2	Text Fields	All text fields in the application **must** be tested to ensure text can be entered into each field without any problems. 2.1 – Text fields will be implemented for naming annotation files.
		Code Testing
3	Appending file changes	All code relating to appending file changes **must** be tested to ensure file changes are successful. 3.1 – The addition/removal of classes will be appended to the classes file. 3.2 – The saving of annotation files will be appended to annotation folder.
4	Sorting/ Searching Algorithms	All code relating to sorting/searching options **must** be tested to ensure their function is executed when used. 4.1 – Sorting/searching algorithms will be implemented to sort ascending/descending compatible images by filename and file date. 4.2 -Sorting/searching algorithms will be implemented to sort ascending/descending all classes listed in classes pane.

5	AutoSave using threads	All code relating to the auto save function **must** be tested to ensure the application is automatically saving annotation files. 5.1 – Auto save will be implemented to automatically save annotation files every minute using threads.
6	Shape Operations	All code relating to shape operations using the mouse **must** be tested to ensure each shape operation can be performed. 6.1 – The ability to Increase shape sizes will be implemented to increase the shape size 6.2 – The ability to move the vertices of polygons will be implemented to allow the user to modify the shape of polygons. 6.3 – The ability to delete shapes will be implemented to remove shapes which aren't needed. 6.4 – The ability to copy shapes will be implemented to duplicate shapes. 6.5 – The ability to paste shapes will be implemented to paste duplicated images. 6.6 – The ability to visualise the name of the class on top of the shape will be implemented to recognise the class name of each shape.
7	Modifying Annotation Files	All code related to the modifying annotation files **must** be tested to ensure their function is executed successfully. 7.1 - The user will be able to change the name of an existing file.
8	Data Structures	All code related to data structures **must** be tested to ensure their functionality is correct 8.1 – Data structures will be used to store data in memory through linked lists.
9	Error and Exception Handling	All code related to error and exception handling **must** be tested to ensure their functionality is correct. 9.1 – If a file already exists, a warning will be displayed to the user to ask the user to confirm the overwrite of the file.
		Functionality Testing
10	Shape Options	Shape options **must** be tested to ensure all shape options provided are available to be selected and used. 10.1 – Shape option (Triangle) will be implemented for the drawing of a triangle. 10.2 – Shape option (Square/Rectangle) will be implemented for the drawing of a square/rectangle. 10.3 – Shape option (Trapezium) will be implemented for the drawing of a trapezium. 10.4 – Shape option (polygon with up to 8 points) will be implemented for the drawing of a polygon.

11	Class File Format	Class file format **must** be tested to ensure all class files follow the same format. 11.1 – All class files will be recognised as plain text files with the extension "*. names".
12	Annotation File Format	Annotation file format **must** be tested to ensure all annotation files follow the same format. 12.1 – All annotations files will be recognised with the extension "*. annotations". 12.2 – All annotation files will follow the hierarchical date format 5 standard (HDF51)
13	Compatible Image File Format	Compatible image file format **must** be tested to ensure all compatible image files follow the same format. 13.1 – All compatible image files will be recognised with the extension "*.jpg," or "*.png".
14	Data Stored in Annotation Files	Data stored in the annotation files **must** be tested to ensure that the required data is correctly stored in each annotation file. 14.1 – The number of annotated images for each image will be stored in each annotation file. 14.2 – The image file name will be stored in each annotation file. 14.3 – The number of shapes per image for each shape will be stored in each annotation file. 14.4 – The shape type will be stored in each annotation file. 14.5 – Point_1(x, y) will be stored in each annotation file. 14.6 – Point_2(x, y) will be stored in each annotation file. 14.7 – Point_n1(x, y) will be stored in each annotation file.
15	Image Pane	The image pane **must** be tested to ensure the its functionality is correct. 15.1 – The image pane will display the users selected image 15.2 – The image pane will allow the user to select any compatible image.
16	Shape Drawing	The shape drawing **must** be tested to ensure each shape can be drawn on top of the compatible image. 16.1 – The user will be able to draw any shape on top of any compatible image.
17	Annotations File and Filename Selector Options	The annotation file and filename selector options **must** be tested to ensure they include the required options. 17.1 – The user will be able to open and load annotation files. 17.2 – The user will be able to save annotation files.
		Bug Testing
18	Bugs	The application must be debugged to ensure there isn't any fault or error.

Out of Scope Test Number	**Out of Scope Test**	**Out of Scope Description**

1	Virus Scanners	Problems related to virus scanners or filters **are not** supposed to be tested. 1.1 – University computers may not advocate to the running of the application which could lead to the application files being corrupted or deleted.
2	Setup Problems	Problems related to setting up the application **are not** supposed to be tested. 2.1 – The application not being able to be downloaded or installed may not be a fault with the actual application. 2.2 – The application not being compatible with specific devices won't be a fault with the actual application.

Testing Type

The testing type classifies the different methods of testing which are used to test different aspects of an application. The fundamental types of software testing which will be used to test this application are unit, integration, system and acceptance testing. Unit testing is the testing of individual components within the application. Unit testing could be used to test the functionality of the browse button in the class selector pane. Integration testing is the testing of combined individual units. This testing will expose defects between the combined units. This could be used when testing the copying of a shape along with the pasting of a shape. Test stubs will be used to help with integration testing. System testing is the testing of the complete application with the requirements. This testing will require the software tester to complete each test based on the requirements of the application. One of the projects requirements state that the application must allow the user to add/remove classes. To test for this requirement, the software tester would have text for this functionality in the application. Acceptance testing is the testing of the application's acceptability. This test will recognise whether the labelling application is ready and acceptable to be used by the end users. Once all requirements are met, the application should be ready to be used.

Non-functional testing types may also need to be considered to highlight any problems with the application's non-functional requirements. Although this project doesn't require any non-functional requirements, testing the usability of the application could prove important.

Unit tests

In order to present the unit tests used to test the application, test scenario description tables will be used for each unit test.

GUI

Table 14: buttons tested

ID:	TS_BUT_Open_Image	Description:	Simulates a left mouse click on open image button to verify it works and can be clicked.

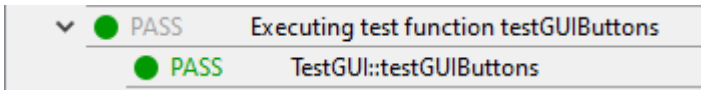
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on open image button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Open Image button */`
	`QPushButton but_openImage;`
	`QTest::mouseClick(&but_openImage, Qt::LeftButton);`
	PASS Executing test function testGUIButtons
	PASS TestGUI::testGUIButtons		

Test date:	18/03/20	Result:	PASSED

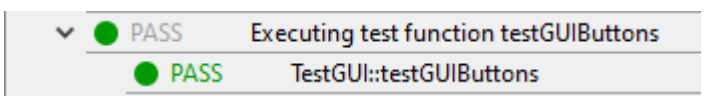
ID:	TS_BUT_Sort Image_Asc	Description:	Simulates a left click on sort image ascending button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on sort image ascending button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		

Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Sort Image Ascending button`
	
	`QPushButton sort_Ascending;`
	
	`QTest::mouseClick(&sort_Ascending, Qt::LeftButton);`
	 		

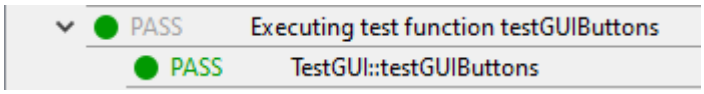
Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Sort Image_Desc	Description:	Simulates a left click on sort image descending button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on sort image descending button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Sort Image Descending button`
	
	`QPushButton sort_Descending;`
	
	`QTest::mouseClick(&sort_Descending, Qt::LeftButton);`
	 		

Test date:	18/03/20	Result:	PASSED

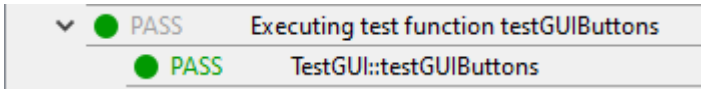
ID:	TS_BUT_Create Class	Description:	Simulates a left click on create class button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on create class button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`//Testing Left Click on Create Class button`
	`QPushButton but_CreateClass;`
	`QTest::mouseClick(&but_CreateClass, Qt::LeftButton);`
	 		

Test date:	18/03/20	Result:	PASSED

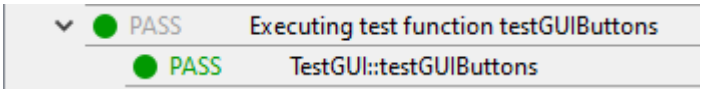
ID:	TS_BUT_Remove Class	Description:	Simulates a left click on remove class button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on remove class button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		

Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`//Testing Left Click on Remove Class button`
	`QPushButton but_RemoveClass;`
	`QTest::mouseClick(&but_RemoveClass, Qt::LeftButton);`
	 		

Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Create Class_File	Description:	Simulates a left click on create class file button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on create class file button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Create Class File button */`
	`QPushButton CreateClassFile;`
	`QTest::mouseClick(&CreateClassFile, Qt::LeftButton);`
	 		

Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Sort Class_Asc	Description:	Simulates a left click on sort class ascending

			button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on sort class ascending button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Sort Class Ascending Button */`
	`QPushButton but_sortClass_Ascending;`
	`QTest::mouseClick(&but_sortClass_Ascending, Qt::LeftButton);`
	PASS Executing test function testGUIButtons
	PASS TestGUI::testGUIButtons		

Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Sort Class_Desc	Description:	Simulates a left click on sort class descending button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on sort class descending button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		

Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s) :	Software Tester (Joshua Ssendagire)		
Individual results:	/** Testing Left Click on Sort Class Descending Button */ QPushButton but_sortClass_Descending; QTest::mouseClick(&but_sortClass_Descending, Qt::LeftButton); PASS Executing test function testGUIButtons PASS TestGUI::testGUIButtons		
Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Help_Rect	Description:	Simulates a left click on help button (Rectangle) to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on help button (Rectangle) using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements :	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s) :	Software Tester (Joshua Ssendagire)		
Individual results:	/** Testing Left Click on Help button (Rectangle) */ QPushButton pushButton_Help; QTest::mouseClick(&pushButton_Help, Qt::LeftButton); PASS Executing test function testGUIButtons PASS TestGUI::testGUIButtons		
Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Help_Sq	Description:	Simulates a left click on help button (Square) to verify It works and can be clicked.

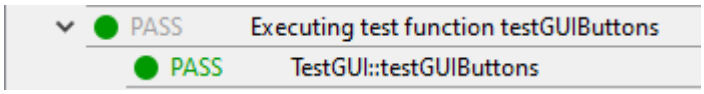
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on help button (Square) using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Help button (Square) */`
	`QPushButton pushButton_HelpSquare;`
	`QTest::mouseClick(&pushButton_HelpSquare, Qt::LeftButt`
	PASS Executing test function testGUIButtons
	PASS TestGUI::testGUIButtons		

Test date:	18/03/20	Result:	PASSED

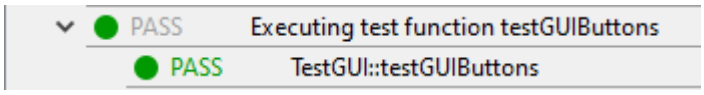
ID:	TS_BUT_Browse_Class	Description:	Simulates a left click on browse class button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on browse class button using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		

Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Browse Class button */`
	
	`QPushButton classBrowse;`
	
	`QTest::mouseClick(&classBrowse, Qt::LeftButton);`
	
	 		

Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Save_Image File	Description:	Simulates a left click on save image file button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on save image file button using QTest.
Quantity/ Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Save Image File button */`
	
	`QPushButton SaveImageFile;`
	
	`QTest::mouseClick(&SaveImageFile, Qt::LeftButton);`
	
	 		

Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Browse_Anno File	Description:	Simulates a left click on browse annotation file button to verify It

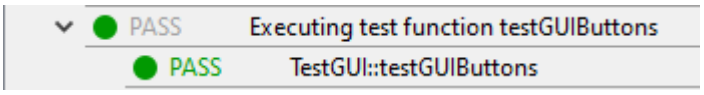
			works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on browse annotation file button using QTest.
Quantity/ Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Browse Annotation File buttor`
	`QPushButton AnnotatedFiles;`
	`QTest::mouseClick(&AnnotatedFiles, Qt::LeftButton);`
	PASS Executing test function testGUIButtons
	PASS TestGUI::testGUIButtons		

Test date:	18/03/20	Result:	PASSED

ID:	TS_BUT_Add_Text	Description:	Simulates a left click on add text button to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on add text button using QTest.

Quantity/ Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Left Click on Add Text button */`
	
	`QPushButton pushButton_addtext;`
	
	`QTest::mouseClick(&pushButton_addtext, Qt::LeftButton)`
	 		

Test date:	18/03/20	Result:	PASSED

Table 15: Check Boxes

ID:	TS_CHKB_SortBy Date	Description:	Simulates a left click on sort by date checkbox to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on sort by date checkbox using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`//Testing Left Click on Sort By Date checkbox`
	
	`QCheckBox checkBox_Date;`
	
	`QTest::mouseClick(&checkBox_Date, Qt::LeftButton);`		

	PASS Executing test function testGUICheckBoxes PASS TestGUI::testGUICheckBoxes		
Test date:	18/03/20	Result:	PASSED

ID:	TS_CHKB_SortBy Filename	Description:	Simulates a left click on sort by filename checkbox to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The button must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on sort by file name checkbox using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`//Testing Left Click on Sort By File Name checkbox`
	
	`QCheckBox checkBox_File;`
	
	`QTest::mouseClick(&checkBox_File, Qt::LeftButton);`
	PASS Executing test function testGUICheckBoxes
	PASS TestGUI::testGUICheckBoxes		

Test date:	18/03/20	Result:	PASSED

Combo Boxes:

ID:	TS_CMB_Image s	Description:	Simulates a left click on images combo box to verify It works and can be clicked.

Test type:	Unit test	Success criteria:	The combo box must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on image combo box using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test and the software developer will have to fix the error.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`//Testing Left Click on Images combo box`
	`QComboBox comboBoxImages;`
	`QTest::mouseClick(&comboBoxImages, Qt::LeftButton);`
	PASS Executing test function testGUIComboBoxes
	PASS TestGUI::testGUIComboBoxes		

Test date:	18/03/20	Result:	PASSED

ID:	TS_CMB_Choose Shape	Description:	Simulates a left click on choose shape combo box to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The combo box must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on the choose shape combo box using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test and the software developer will have to fix the error.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`//Testing Left click on Choose Shape combo box`
	
	`QComboBox comboBox_ChooseShape;`
	
	`QTest::mouseClick(&comboBox_ChooseShape, Qt::LeftButton);`
	PASS Executing test function testGUIComboBoxes
	PASS TestGUI::testGUIComboBoxes		

Test date:	18/03/20	Result:	PASSED

ID:	TS_CMB_Set_ Colour	Description:	Simulates a left click on set colour combo box to verify It works and can be clicked.
Test type:	Unit test	Success criteria:	The combo box must be able to be clicked by the user, in order for this test be successful.
Number of attempts:	1	Comments:	This test simulates a left click on the set colour combo box using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the button is created incorrectly or cannot be left clicked, QTest will fail the test and the software developer will have to fix the error.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

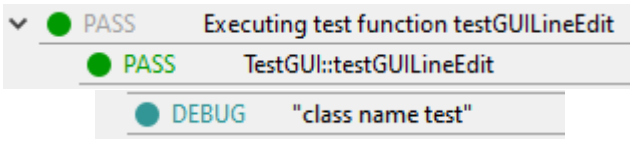
Individual results:	`/** Testing Left click on Set Colour combo box */`
	
	`QComboBox comboBox_SetColour;`
	
	`QTest::mouseClick(&comboBox_SetColour, Qt::LeftButton)`
	PASS Executing test function testGUIComboBoxes
	PASS TestGUI::testGUIComboBoxes		

Test date:	18/04/20	Result:	PASSED

Table 16: Line edit

ID:	TS_LNED_Class Name	Description:	Simulates the writing of text into the line edit field. In this case it's "class name test".
Test type:	Unit test	Success criteria:	The line edit field must be allow text to be entered in order for the test to be successful.

Number of attempts:	1	Comments:	This test simulates the writing of text into a line edit filed using QTest.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If text is unable to be entered into the line edit field or the line edit field is created incorrectly, QTest will fail the test and the software developer will have to fix the error.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing line edit widget by simulating the writing of "class name test" */`
	`QTest::keyClicks(&lineEdit_className, "class name test");`
	`QCOMPARE(lineEdit_className.text(),QString("class name test"));`
	`qDebug() << lineEdit_className.text();`
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_LNED_Class Name	Description:	Tests line edit field to check if its empty.
Test type:	Unit test	Success criteria:	The line edit field must start empty in order for the test to be successful.
Number of attempts:	1	Comments:	This test verifies whether the line edit field is empty using QTest and QVERIFY2.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the line edit field isn't empty, then QTest will fail the test and the software developer will have to correct the error.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing line edit to check if its empty */ QLineEdit lineEdit_className; QVERIFY2(lineEdit_className.text().isEmpty(), "Line Edit is not empty"); qDebug() << lineEdit_className.text(); PASS Executing test function testGUILineEdit DEBUG ""		
Test date:	18/03/20	Result:	PASSED

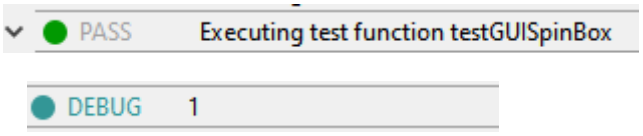
Table 17: Line Edit

ID:	TS_SpBx_Pen Width_Within_ Range	Description:	Testing pen width spin box values within spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value outside of the pen width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing Pen Width spin box values */
	QSpinBox spinBoxPenWidth;
	spinBoxPenWidth.setRange(1,40);
	/** Testing value within Pen Width spin box range */
	spinBoxPenWidth.setValue(20);
	QCOMPARE(spinBoxPenWidth.value(), 20);
	qDebug() << spinBoxPenWidth.value();
	PASS Executing test function testGUISpinBox
	DEBUG 20		

Test date:	18/04/20	Result:	PASSED

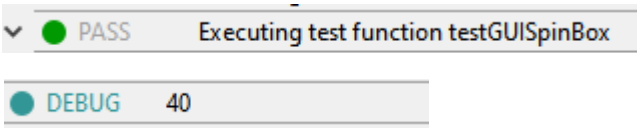
ID:	TS_SpBx_Pen Width_Smaller Range	Description:	Testing pen width spin box values smaller than spin box range
Test type:	Unit test	Success criteria:	The value entered must be smaller than the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is smaller than the range, the test will set the value to the smallest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value smaller than the pen width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Pen Width spin box values */`
	`QSpinBox spinBoxPenWidth;`
	`spinBoxPenWidth.setRange(1,40);`
	`/** Testing value smaller than Pen Width spin box range */`
	`spinBoxPenWidth.setValue(0);`
	`QCOMPARE(spinBoxPenWidth.value(), 1);`
	`qDebug() << spinBoxPenWidth.value();`
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Pen Width_Larger	Description:	Testing pen width spin box values smaller than spin box range
Test type:	Unit test	Success criteria:	The value entered must be larger than the spin box range in order for the test to be successful.

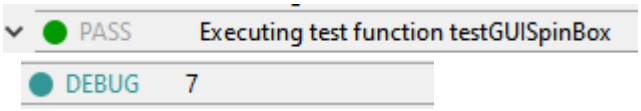
Number of attempts:	1	Comments:	If the value entered is larger than the range, the test will set the value to the largest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value larger than the pen width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Pen Width spin box values */`
	
	`QSpinBox spinBoxPenWidth;`
	
	`spinBoxPenWidth.setRange(1,40);`
	`/** Testing value larger than Pen Width spin box range */`
	
	`spinBoxPenWidth.setValue(41);`
	`QCOMPARE(spinBoxPenWidth.value(), 40);`
	`qDebug() << spinBoxPenWidth.value();`
	
	 		

Test date:	18/04/20	Result:	PASSED

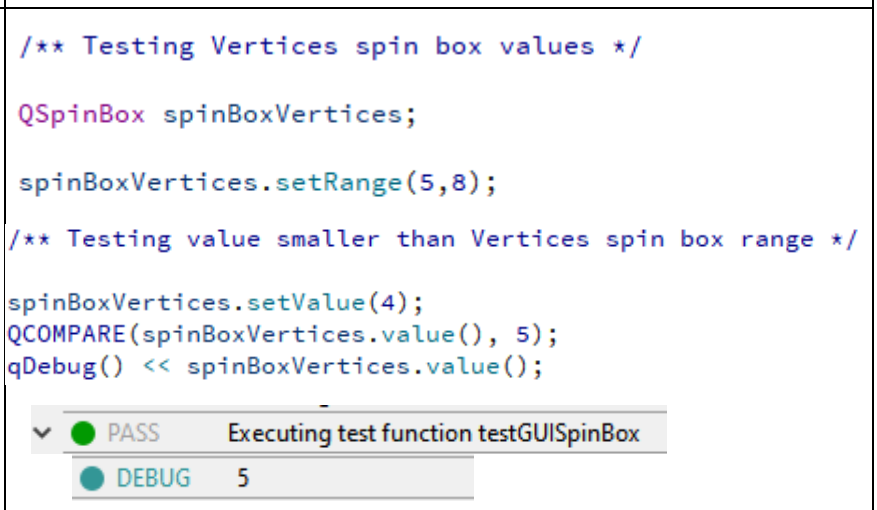
ID:	TS_SpBx_Vertices_Within	Description:	Testing vertices spin box values within spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		

Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value outside of the pen width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Vertices spin box values */`
	
	`QSpinBox spinBoxVertices;`
	
	`spinBoxVertices.setRange(5,8);`
	
	`/** Testing value within Vertices spin box range */`
	
	`spinBoxVertices.setValue(7);`
	`QCOMPARE(spinBoxVertices.value(), 7);`
	`qDebug() << spinBoxVertices.value();`
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertcies_Smaller	Description:	Testing Vertices spin box values smaller than spin box range
Test type:	Unit test	Success criteria:	The value entered must be smaller than the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is smaller than the range, the test will set the value to the smallest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value smaller than the vertices spin box range is entered, then QTest will fail the test and a value within the range must be entered.		

Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertices_Larger	Description:	Testing vertices spin box values smaller than spin box range
Test type:	Unit test	Success criteria:	The value entered must be larger than the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is larger than the range, the test will set the value to the largest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value larger than the vertices spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

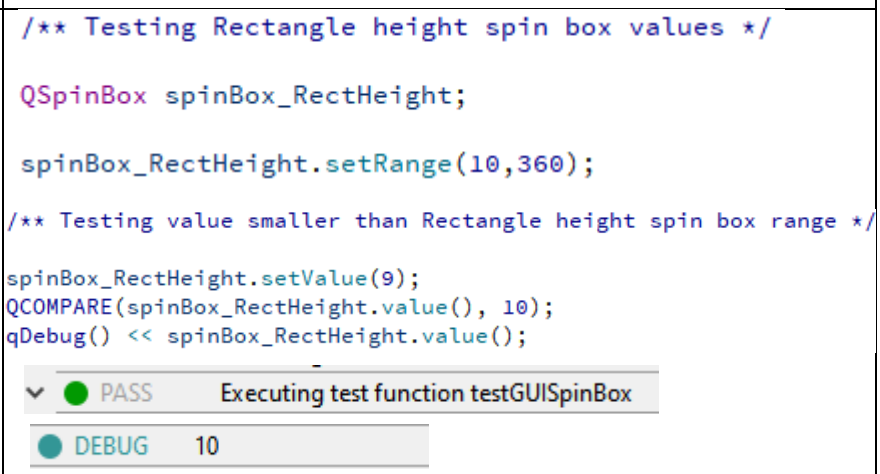
Individual results:	/** Testing Vertices spin box values */
	
	QSpinBox spinBoxVertices;
	
	spinBoxVertices.setRange(5,8);
	
	/** Testing value larger than Vertices spin box range */
	spinBoxVertices.setValue(9);
	QCOMPARE(spinBoxVertices.value(), 8);
	qDebug() << spinBoxVertices.value();
	
	PASS Executing test function testGUISpinBox
	
	DEBUG 8		

Test date:	18/04/20	Result:	PASSED

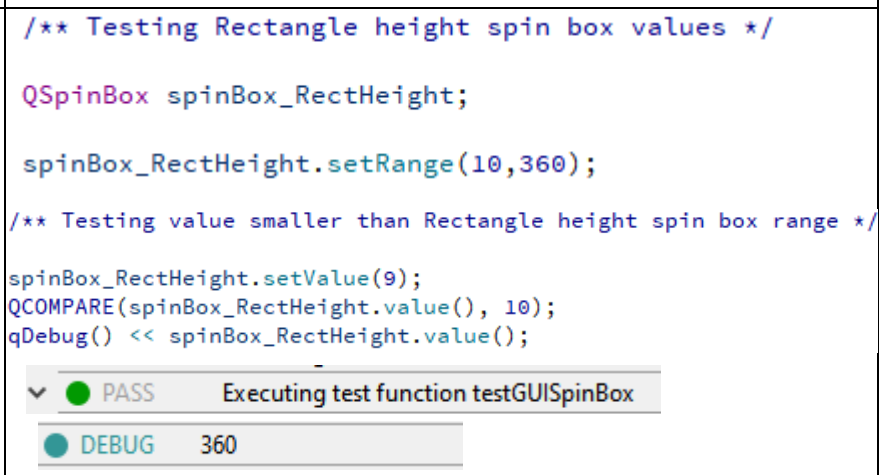
ID:	TS_SpBx_Rect_ Height_Within	Description:	Testing rectangle height spin box values within spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements :	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value outside of the rectangle height spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing Rectangle height spin box values */
	
	QSpinBox spinBox_RectHeight;
	
	spinBox_RectHeight.setRange(10,360);
	
	/** Testing value within Rectangle height spin box range */
	
	spinBox_RectHeight.setValue(90);
	QCOMPARE(spinBox_RectHeight.value(), 90);
	qDebug() << spinBox_RectHeight.value();
	
	PASS Executing test function testGUISpinBox
	
	DEBUG 90		

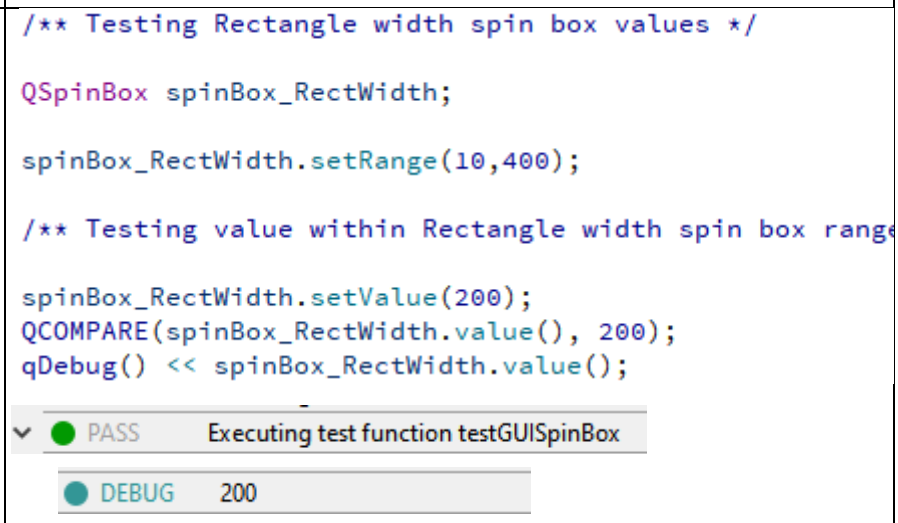
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height_Smaller	Description:	Testing rectangle height spin box values smaller than spin box range
Test type:	Unit test	Success criteria:	The value entered must be smaller than the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is smaller than the range, the test will set the value to the smallest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value smaller than the rectangle height spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

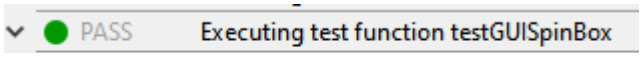
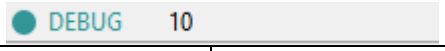
ID:	TS_SpBx_Rect_Height_Larger	Description:	Testing rectangle height spin box values larger than spin box range
Test type:	Unit test	Success criteria:	The value entered must be larger than the spin box range in order for the test to be successful.

Number of attempts:	1	Comments:	If the value entered is larger than the range, the test will set the value to the largest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value smaller than the rectangle height spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Width_Within	Description:	Testing rectangle width spin box values within spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		

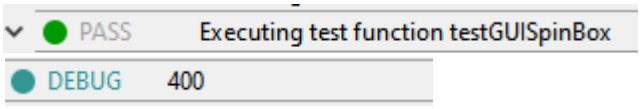
Failure correction procedure:	If a value outside of the rectangle width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Width_Smaller	Description:	Testing rectangle width spin box values smaller than the spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is smaller than the range, the test will set the value to the smallest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value smaller than the rectangle width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing Rectangle width spin box values */
	
	QSpinBox spinBox_RectWidth;
	
	spinBox_RectWidth.setRange(10,400);
	
	/** Testing value smaller than Rectangle width spin box range */
	
	spinBox_RectWidth.setValue(9);
	QCOMPARE(spinBox_RectWidth.value(), 10);
	qDebug() << spinBox_RectWidth.value();
	
	 
	
	 		

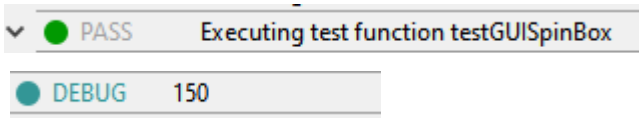
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_ Width_Larger	Description:	Testing rectangle width spin box values larger than the spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is larger than the range, the test will set the value to the largest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value larger than the rectangle width spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

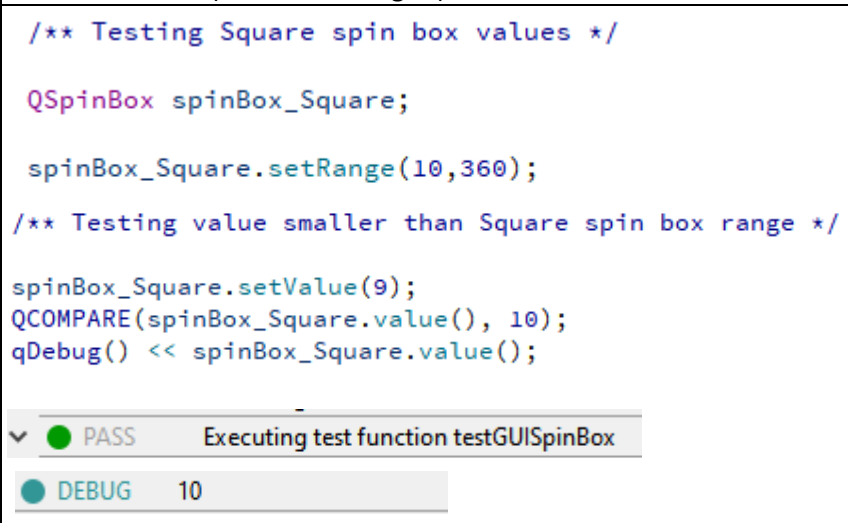
Individual results:	/** Testing Rectangle width spin box values */
	
	QSpinBox spinBox_RectWidth;
	
	spinBox_RectWidth.setRange(10,400);
	
	/** Testing value larger than Rectangle width spin box range */
	
	spinBox_RectWidth.setValue(401);
	QCOMPARE(spinBox_RectWidth.value(), 400);
	qDebug() << spinBox_RectWidth.value();
	
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Sqar_ Within	Description:	Testing square spin box values within spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value outside of the square spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing Square spin box values */
	
	QSpinBox spinBox_Square;
	
	spinBox_Square.setRange(10,360);
	
	/** Testing value witin Square spin box range */
	
	spinBox_Square.setValue(150);
	QCOMPARE(spinBox_Square.value(), 150);
	qDebug() << spinBox_Square.value();
	
	 		

Test date:	18/04/20	Result:	PASSED

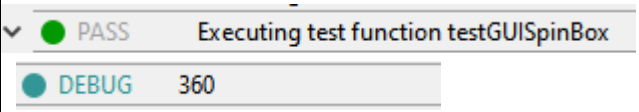
ID:	TS_SpBx_Sqar_ Smaller	Description:	Testing square spin box values smaller than the spin box range.
Test type:	Unit test	Success criteria:	The value entered must be within the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is smaller than the range, the test will set the value to the smallest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value smaller than the square spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

```
/** Testing Square spin box values */

QSpinBox spinBox_Square;

spinBox_Square.setRange(10,360);

/** Testing value smaller than Square spin box range */

spinBox_Square.setValue(9);
QCOMPARE(spinBox_Square.value(), 10);
qDebug() << spinBox_Square.value();
```

```
PASS    Executing test function testGUISpinBox
DEBUG   10
```

ID:	TS_SpBx_Sqar_ Larger	Description:	Testing square spin box values larger than spin box range

Test type:	Unit test	Success criteria:	The value entered must be larger than the spin box range in order for the test to be successful.
Number of attempts:	1	Comments:	If the value entered is larger than the range, the test will set the value to the largest value within the range. This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If a value larger than the square height spin box range is entered, then QTest will fail the test and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Square spin box values */`
	
	`QSpinBox spinBox_Square;`
	
	`spinBox_Square.setRange(10,360);`
	
	`/** Testing value larger than Square spin box range */`
	
	`spinBox_Square.setValue(361);`
	`QCOMPARE(spinBox_Square.value(), 360);`
	`qDebug() << spinBox_Square.value();`
	
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Pen Width_Key_Press_Up	Description:	Testing up arrow keyboard press on pen width spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the pen width spin box from a chosen value within the spin box range. The test will be successful if the

			chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the pen width range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard press on Pen Width spin box */`
	
	`QSpinBox spinBoxPenWidth;`
	
	`spinBoxPenWidth.setRange(1,40);`
	`spinBoxPenWidth.setValue(20);`
	
	`/** Testing up keyboard button press on Pen Width spin box */`
	
	`QTest::keyClick(&spinBoxPenWidth, Qt::Key_Up);`
	`QCOMPARE(spinBoxPenWidth.value(), 21);`
	`qDebug() << spinBoxPenWidth.value();`
	
	PASS Executing test function testSpinBoxKeyPress
	DEBUG 21		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Pen Width_Key_Press_Down	Description:	Testing down arrow keyboard press on pen width spin box from the previous value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the pen width spin box from the previous value within the spin box range. The test will be successful if the previous value goes

			down by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the pen width range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard press on Pen Width spin box */`
	
	`QSpinBox spinBoxPenWidth;`
	
	`spinBoxPenWidth.setRange(1,40);`
	`spinBoxPenWidth.setValue(20);`
	`/** Testing down keyboard button press on Pen Width spin box */`
	
	`QTest::keyClick(&spinBoxPenWidth, Qt::Key_Down);`
	`QCOMPARE(spinBoxPenWidth.value(), 20);`
	`qDebug() << spinBoxPenWidth.value();`
	
	✔ PASS Executing test function testSpinBoxKeyPress
	DEBUG 20		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Pen Width_Key_Press_Up_Range	Description:	Testing up arrow keyboard press on the pen width spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the pen width spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range

			limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the pen width range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing up and down keyboard button press on Pen Width spin box range limit */`
	
	`spinBoxPenWidth.setValue(40);`
	`QTest::keyClick(&spinBoxPenWidth, Qt::Key_Up);`
	`QCOMPARE(spinBoxPenWidth.value(), 40);`
	`qDebug() << spinBoxPenWidth.value();`
	
	✔ PASS Executing test function testSpinBoxKeyPress
	DEBUG 40		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Pen Width_Key_Press_Down_Range	Description:	Testing down arrow keyboard press on the pen width spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the pen width spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below

			the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the pen width range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing up and down keyboard button press on Pen Width spin box range limit */`
	`spinBoxPenWidth.setValue(40);`
	`QTest::keyClick(&spinBoxPenWidth, Qt::Key_Up);`
	`QCOMPARE(spinBoxPenWidth.value(), 40);`
	`qDebug() << spinBoxPenWidth.value();`
	✔ PASS Executing test function testSpinBoxKeyPress
	DEBUG 40		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertices_Key_Press_Up	Description:	Testing up arrow keyboard press on vertices spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the vertices spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one

			after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the vertices range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard button press on Vertices spin box */`
	
	`QSpinBox spinBoxVertices;`
	
	`spinBoxVertices.setRange(5,8);`
	`spinBoxVertices.setValue(7);`
	
	`/** Testing up keyboard button press on Vertices spin box */`
	
	`QTest::keyClick(&spinBoxVertices, Qt::Key_Up);`
	`QCOMPARE(spinBoxVertices.value(), 8);`
	`qDebug() << spinBoxVertices.value();`
	
	✔ PASS Executing test function testSpinBoxKeyPress
	DEBUG 8		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertices_Key_Press_Down	Description:	Testing down arrow keyboard press on vertices spin box from the previous value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the vertices spin box from the previous value within the spin box range. The test will be successful if the previous value goes down by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		

List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the vertices range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard button press on Vertices spin box */`
	`QSpinBox spinBoxVertices;`
	`spinBoxVertices.setRange(5,8);`
	`spinBoxVertices.setValue(7);`
	`/** Testing down keyboard button press on Verticies spin box */`
	`QTest::keyClick(&spinBoxVertices, Qt::Key_Down);`
	`QCOMPARE(spinBoxVertices.value(), 7);`
	`qDebug() << spinBoxVertices.value();`
	PASS Executing test function testSpinBoxKeyPress
	DEBUG 7		

Test date:	18/04/20	Result:	PASSED

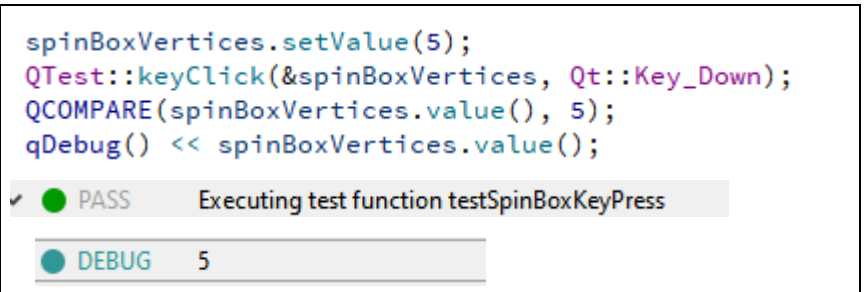
ID:	TS_SpBx_Vertices _Key_Press_Up_Range	Description:	Testing up arrow keyboard press on the vertices spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the vertices spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		

Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the vertices range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

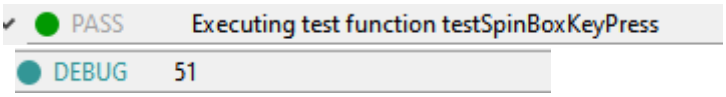
Individual results:	`/** Testing up and down keyboard button press on Vertices spin box range limit */`
	`spinBoxVertices.setValue(8);`
	`QTest::keyClick(&spinBoxVertices, Qt::Key_Up);`
	`QCOMPARE(spinBoxVertices.value(), 8);`
	`qDebug() << spinBoxVertices.value();`
	✔ PASS Executing test function testSpinBoxKeyPress
	DEBUG 8		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertices _Key_Press_Down_Range	Description:	Testing down arrow keyboard press on the vertices spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the vertices spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the vertices range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			

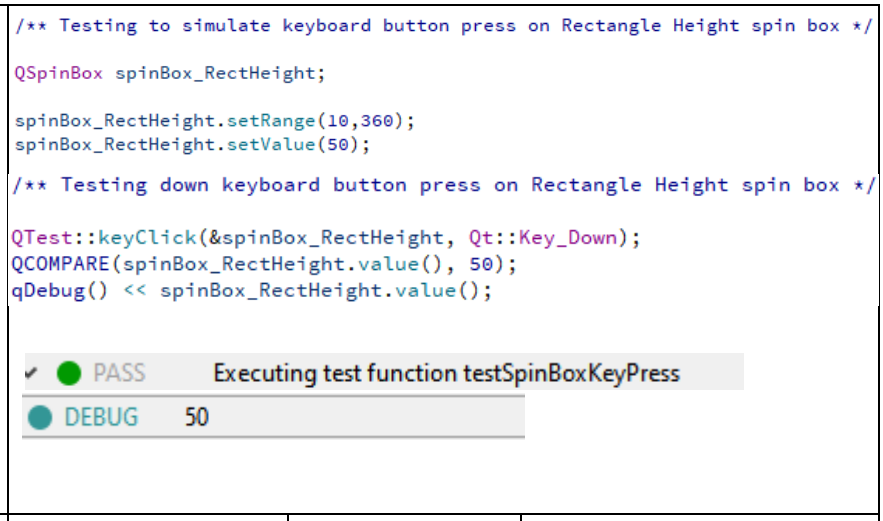
			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height _Key_Press_Up	Description:	Testing up arrow keyboard press on rectangle height spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the rectangle height spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the rectangle height range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

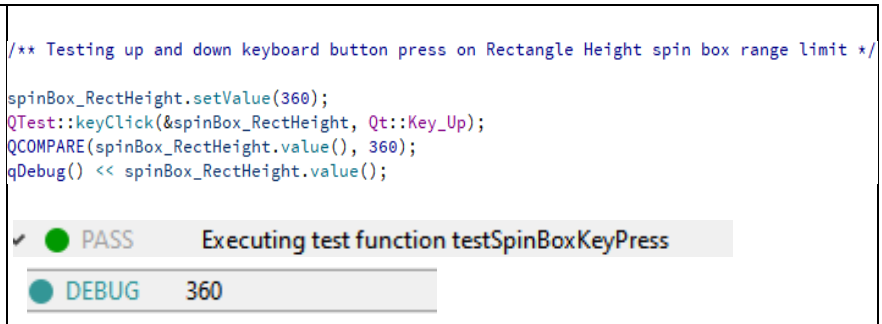
Individual results:	/** Testing to simulate keyboard button press on Rectangle Height spin box */
	
	QSpinBox spinBox_RectHeight;
	
	spinBox_RectHeight.setRange(10,360);
	spinBox_RectHeight.setValue(50);
	
	/** Testing up keyboard button press on Rectangle Height spin box */
	
	QTest::keyClick(&spinBox_RectHeight, Qt::Key_Up);
	QCOMPARE(spinBox_RectHeight.value(), 51);
	qDebug() << spinBox_RectHeight.value();
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height Key_Press_Down	Description:	Testing down arrow keyboard press on rectangle height spin box from the previous value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the rectangle height spin box from the previous value within the spin box range. The test will be successful if the previous value goes down by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the rectangle height range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height_Key_Press_Up_Range	Description:	Testing up arrow keyboard press on the rectangle height spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the rectangle height spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle height range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_ Rect_Height _Key_Press_Down_Range	Description:	Testing down arrow keyboard press on the rectangle height spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the rectangle height spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle height range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`spinBox_RectHeight.setValue(10);`
	`QTest::keyClick(&spinBox_RectHeight, Qt::Key_Down);`
	`QCOMPARE(spinBox_RectHeight.value(), 10);`
	`qDebug() << spinBox_RectHeight.value();`
	PASS Executing test function testSpinBoxKeyPress
	DEBUG 10		

Test date:	18/04/20	Result:	PASSED

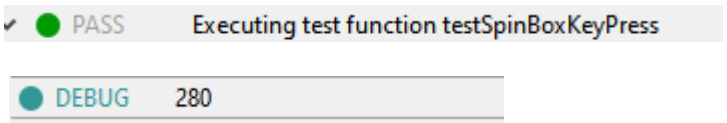
ID:	TS_SpBx_Rect_Width _Key_Press_Up	Description:	Testing up arrow keyboard press on rectangle width spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the rectangle width spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the rectangle width range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard button press on Rectangle Width spin box */`
	`QSpinBox spinBox_RectWidth;`
	`spinBox_RectWidth.setRange(10,400);`
	`spinBox_RectWidth.setValue(280);`
	`/** Testing up keyboard button press on Rectangle Width spin box */`
	`QTest::keyClick(&spinBox_RectWidth, Qt::Key_Up);`
	`QCOMPARE(spinBox_RectWidth.value(), 281);`
	`qDebug() << spinBox_RectWidth.value();`
	PASS Executing test function testSpinBoxKeyPress
	DEBUG 281		

Test date:	18/04/20	Result:	PASSED

<table>
<tr><td>ID:</td><td>TS_SpBx_Rect_Width
Key_Press_Down</td><td>Description:</td><td>Testing down arrow keyboard press on rectangle width spin box from the previous value within the spin box range.</td></tr>
<tr><td>Test type:</td><td>Unit test</td><td>Success criteria:</td><td>The test will simulate the pressing of the down arrow on the keyboard of the rectangle width spin box from the previous value within the spin box range. The test will be successful if the previous value goes down by one after the up arrow is pressed.</td></tr>
<tr><td>Number of attempts:</td><td>1</td><td>Comments:</td><td>This test was created using QTest and QCOMPARE.</td></tr>
<tr><td>Quantity/Quality:</td><td colspan="3">Quality</td></tr>
<tr><td>List of equipment/requirements:</td><td colspan="3">System running any operating system – Windows, Linux, MacOS
Qt creator 5.9 or higher</td></tr>
<tr><td>Setup instructions:</td><td colspan="3">Open application and testing project file and run tst_testgui.cpp file.</td></tr>
<tr><td>Failure correction procedure:</td><td colspan="3">If the previous value within the rectangle width range doesn’t go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.</td></tr>
<tr><td>Engineer(s)/Technician(s):</td><td colspan="3">Software Tester (Joshua Ssendagire)</td></tr>
<tr><td>Individual results:</td><td colspan="3">

```
/** Testing to simulate keyboard button press on Rectangle Width spin box */

QSpinBox spinBox_RectWidth;

spinBox_RectWidth.setRange(10,400);
spinBox_RectWidth.setValue(280);
/** Testing down keyboard button press on Rectangle Width spin box */

QTest::keyClick(&spinBox_RectWidth, Qt::Key_Down);
QCOMPARE(spinBox_RectWidth.value(), 280);
qDebug() << spinBox_RectWidth.value();
```



</td></tr>
<tr><td>Test date:</td><td>18/04/20</td><td>Result:</td><td>PASSED</td></tr>
</table>

ID:	TS_SpBx_Rect_Width_Key_Press_Up_Range	Description:	Testing up arrow keyboard press on the rectangle width spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the rectangle width spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle width range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing up and down keyboard button press on Rectangle Width spin box range limit */`
	`spinBox_RectWidth.setValue(400);`
	`QTest::keyClick(&spinBox_RectWidth, Qt::Key_Up);`
	`QCOMPARE(spinBox_RectWidth.value(), 400);`
	`qDebug() << spinBox_RectWidth.value();`
	✔ PASS Executing test function testSpinBoxKeyPress
	DEBUG 400		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_ Rect_Width_Key_Press_Down_Range	Description:	Testing down arrow keyboard press on the rectangle width spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the

			rectangle width spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle width range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`spinBox_RectWidth.setValue(10);`
	`QTest::keyClick(&spinBox_RectWidth, Qt::Key_Down);`
	`QCOMPARE(spinBox_RectWidth.value(), 10);`
	`qDebug() << spinBox_RectWidth.value();`
	PASS Executing test function testSpinBoxKeyPress
	DEBUG 10		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Sqar _Key_Press_Up	Description:	Testing up arrow keyboard press on square spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the square spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		

List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the square range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard button press on Square spin box */`
	
	`QSpinBox spinBox_Square;`
	
	`spinBox_Square.setRange(10,360);`
	`spinBox_Square.setValue(120);`
	
	`/** Testing up keyboard button press on Square spin box */`
	
	`QTest::keyClick(&spinBox_Square, Qt::Key_Up);`
	`QCOMPARE(spinBox_Square.value(), 121);`
	`qDebug() << spinBox_Square.value();`
	
	✔ PASS Executing test function testSpinBoxKeyPress
	
	DEBUG 121		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_SqarKey_Press_Down	Description:	Testing down arrow keyboard press on square spin box from the previous value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the square spin box from the previous value within the spin box range. The test will be successful if the previous value goes down by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		

Failure correction procedure:	If the previous value within the square range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing to simulate keyboard button press on Square spin box */`
	
	`QSpinBox spinBox_Square;`
	
	`spinBox_Square.setRange(10,360);`
	`spinBox_Square.setValue(120);`
	`/** Testing down keyboard button press on Square spin box */`
	
	`QTest::keyClick(&spinBox_Square, Qt::Key_Down);`
	`QCOMPARE(spinBox_Square.value(), 120);`
	`qDebug() << spinBox_Square.value();`
	
	✔ PASS Executing test function testSpinBoxKeyPress
	
	DEBUG 120		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_SqarKey_Press_Up_Range	Description:	Testing up arrow keyboard press on the square width spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the up arrow on the keyboard of the square width spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the square width range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing up and down keyboard button press on Square spin box range limit */
	
	spinBox_Square.setValue(360);
	QTest::keyClick(&spinBox_Square, Qt::Key_Up);
	QCOMPARE(spinBox_Square.value(), 360);
	qDebug() << spinBox_Square.value();
	
	PASS Executing test function testSpinBoxKeyPress
	
	DEBUG 360		

Test date:	18/04/20	Result:	PASSED

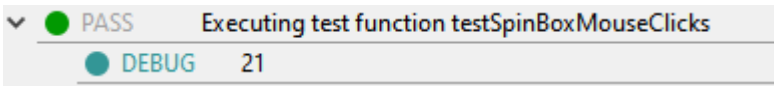
ID:	TS_SpBx_Sqar Key_Press_Down_Range	Description:	Testing down arrow keyboard press on the square spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate the pressing of the down arrow on the keyboard of the square width spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the square width range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	spinBox_Square.setValue(10);
	QTest::keyClick(&spinBox_Square, Qt::Key_Down);
	QCOMPARE(spinBox_Square.value(), 10);
	qDebug() << spinBox_Square.value();
	
	PASS Executing test function testSpinBoxKeyPress
	
	DEBUG 10		

Test date:	18/04/20	Result:	PASSED

<table>
<tr><td>ID:</td><td>TS_SpBx_Pen Width_Clicks_Up</td><td>Description:</td><td>Testing mouse clicks on the up arrow of the pen width spin box from a chosen value within the spin box range.</td></tr>
<tr><td>Test type:</td><td>Unit test</td><td>Success criteria:</td><td>The test will simulate test the mouse clicking of the up arrow on the pen width spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.</td></tr>
<tr><td>Number of attempts:</td><td>1</td><td>Comments:</td><td>This test was created using QTest and QCOMPARE.</td></tr>
<tr><td>Quantity/Quality:</td><td colspan="3">Quality</td></tr>
<tr><td>List of equipment/requirements:</td><td colspan="3">System running any operating system – Windows, Linux, MacOS
Qt creator 5.9 or higher</td></tr>
<tr><td>Setup instructions:</td><td colspan="3">Open application and testing project file and run tst_testgui.cpp file.</td></tr>
<tr><td>Failure correction procedure:</td><td colspan="3">If the chosen value within the pen width range doesn’t go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.</td></tr>
<tr><td>Engineer(s)/Technician(s):</td><td colspan="3">Software Tester (Joshua Ssendagire)</td></tr>
<tr><td>Individual results:</td><td colspan="3">

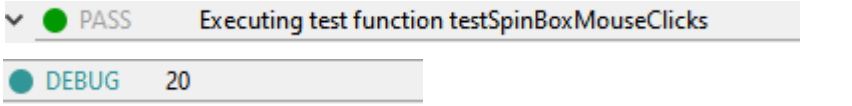
```
/** Testing Mouse clicks on Pen width spin box */

QSpinBox spinBoxPenWidth;

spinBoxPenWidth.setRange(1,40);
spinBoxPenWidth.setValue(20);

QSize size =
spinBoxPenWidth.size();
QPoint upButton=
QPoint(size.width()-2, 2);

/** Testing Mouse clicks on Pen width spin box up and down button */

QTest::mouseClick(&spinBoxPenWidth, Qt::LeftButton, 0, upButton);
QCOMPARE(spinBoxPenWidth.value(), 21);
qDebug() << spinBoxPenWidth.value();
```



</td></tr>
<tr><td>Test date:</td><td>18/04/20</td><td>Result:</td><td>PASSED</td></tr>
</table>

<table>
<tr><td>ID:</td><td>TS_SpBx_Pen Width_Clicks_Down</td><td>Description:</td><td>Testing mouse clicks on the down arrow of the pen width spin box from the previous value within the spin box range.</td></tr>
<tr><td>Test type:</td><td>Unit test</td><td>Success criteria:</td><td>The test will simulate the mouse clicking of down arrow on the pen width spin box from the previous value within the spin box range. The test will be successful if the previous value goes down by one after the up arrow is pressed.</td></tr>
<tr><td>Number of attempts:</td><td>1</td><td>Comments:</td><td>This test was created using QTest and QCOMPARE.</td></tr>
<tr><td>Quantity/Quality:</td><td colspan="3">Quality</td></tr>
<tr><td>List of equipment/requirements:</td><td colspan="3">System running any operating system – Windows, Linux, MacOS
Qt creator 5.9 or higher</td></tr>
<tr><td>Setup instructions:</td><td colspan="3">Open application and testing project file and run tst_testgui.cpp file.</td></tr>
<tr><td>Failure correction procedure:</td><td colspan="3">If the previous value within the pen width range doesn’t go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.</td></tr>
<tr><td>Engineer(s)/Technician(s):</td><td colspan="3">Software Tester (Joshua Ssendagire)</td></tr>
<tr><td>Individual results:</td><td colspan="3">

```
/** Testing Mouse clicks on Pen width spin box */

QSpinBox spinBoxPenWidth;

spinBoxPenWidth.setRange(1,40);
spinBoxPenWidth.setValue(20);
QPoint downButton=
QPoint(size.width()-2, size.height()-2);
QTest::mouseClick(&spinBoxPenWidth, Qt::LeftButton, 0, downButton);
QCOMPARE(spinBoxPenWidth.value(), 20);
```



</td></tr>
<tr><td>Test date:</td><td>18/04/20</td><td>Result:</td><td>PASSED</td></tr>
</table>

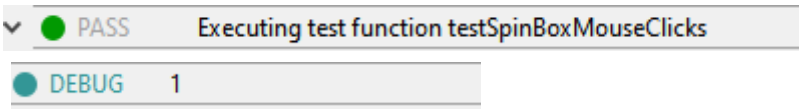
ID:	TS_SpBx_Pen_Width Clicks_Up_Range	Description:	Testing mouse clicks on the up arrow of the pen width spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the up arrow of the pen width spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the pen width range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Mouse clicks on Pen width spin box up and down button at range limit */`
	`spinBoxPenWidth.setValue(40);`
	`QTest::mouseClick(&spinBoxPenWidth, Qt::LeftButton, 0, upButton);`
	`QCOMPARE(spinBoxPenWidth.value(), 40);`
	PASS Executing test function testSpinBoxMouseClicks
	DEBUG 40		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Pen_Width Clicks_Down_Range	Description:	Testing mouse clicks on the down arrow of the pen width spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the down arrow of the pen width spin box from

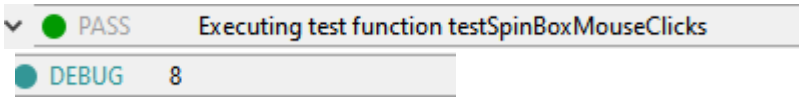
			the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the pen width range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`spinBoxPenWidth.setValue(1);`
	`QTest::mouseClick(&spinBoxPenWidth, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBoxPenWidth.value(), 1);`
	 		

Test date:	18/04/20	Result:	PASSED

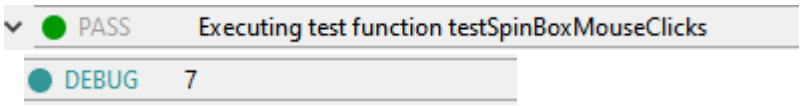
ID:	TS_SpBx_Vertices_Clicks_Up	Description:	Testing mouse clicks on the up arrow of the vertices spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the up arrow on the vertices spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		

Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the vertices range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Mouse clicks on Vertices spin box */`
	
	`QSpinBox spinBoxVertices;`
	
	`spinBoxVertices.setRange(5,8);`
	`spinBoxVertices.setValue(7);`
	
	`spinBoxVertices.size();`
	`QPoint(size.width()-2, 2);`
	`QPoint(size.width()-2, size.height()-2);`
	
	`/** Testing Mouse clicks on Vertices spin box up and down button */`
	
	`QTest::mouseClick(&spinBoxVertices, Qt::LeftButton, 0, upButton);`
	`QCOMPARE(spinBoxVertices.value(), 8);`
	 		

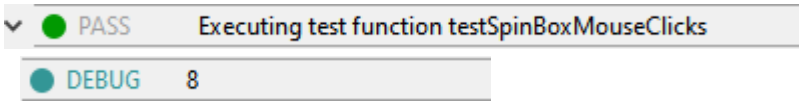
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertices _Clicks_Down	Description:	Testing mouse clicks on the down arrow of the vertices spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the down arrow on the vertices spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the vertices range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`QTest::mouseClick(&spinBoxVertices, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBoxVertices.value(), 7);`
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Vertices_Clicks_Up_Range	Description:	Testing mouse clicks on the up arrow of the vertices spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the up arrow of the vertices spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the vertices range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Mouse clicks on Vertices spin box up and down button at range limit */`
	`spinBoxVertices.setValue(8);`
	`QTest::mouseClick(&spinBoxVertices, Qt::LeftButton, 0, upButton);`
	`QCOMPARE(spinBoxVertices.value(), 8);`
	 		

Test date:	18/03/20	Result:	PASSED

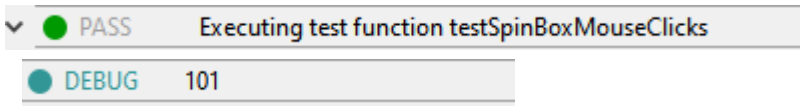
ID:	TS_SpBx_Vertices Clicks_Down_Range	Description:	Testing mouse clicks on the down arrow of the vertices spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the down arrow of the vertices spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the vertices range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`spinBoxVertices.setValue(5);`
	`QTest::mouseClick(&spinBoxVertices, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBoxVertices.value(), 5);`
	PASS Executing test function testSpinBoxMouseClicks
	DEBUG 5		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height _Clicks_Up	Description:	Testing mouse clicks on the up arrow of the rectangle height spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the up arrow on the rectangle height spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.

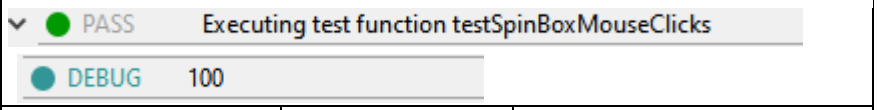
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the rectangle height range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`/** Testing Mouse clicks on Rectangle Height spin box */`
	
	`QSpinBox spinBox_RectHeight;`
	
	`spinBox_RectHeight.setRange(10,360);`
	`spinBox_RectHeight.setValue(100);`
	
	`spinBox_RectHeight.size();`
	`QPoint(size.width()-2, 2);`
	`QPoint(size.width()-2, size.height()-2);`
	
	`/** Testing Mouse clicks on Rectangle Height spin box up and down button */`
	
	`QTest::mouseClick(&spinBox_RectHeight, Qt::LeftButton, 0, upButton);`
	`QCOMPARE(spinBox_RectHeight.value(), 101);`
	
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height_Clicks_Down	Description:	Testing mouse clicks on the down arrow of the rectangle height spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the down arrow on the rectangle height spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.

Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the rectangle height range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`QTest::mouseClick(&spinBox_RectHeight, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBox_RectHeight.value(), 100);`
	 		

Test date:	18/04/20	Result:	PASSED

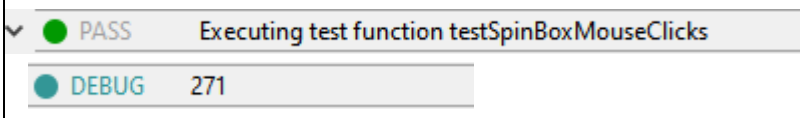
ID:	TS_SpBx_Rect_Height _Clicks_Up_Range	Description:	Testing mouse clicks on the up arrow of the rectangle height spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the up arrow of the rectangle height spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle height range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing Mouse clicks on Rectangle Height spin box up and down button at range limit */ spinBox_RectHeight.setValue(360); QTest::mouseClick(&spinBox_RectHeight, Qt::LeftButton, 0, upButton); QCOMPARE(spinBox_RectHeight.value(), 360); PASS Executing test function testSpinBoxMouseClicks DEBUG 360		
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Height _Clicks_Down_Range	Description:	Testing mouse clicks on the down arrow of the rectangle height spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the down arrow of the rectangle height spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle height range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:	spinBox_RectHeight.setValue(10); QTest::mouseClick(&spinBox_RectHeight, Qt::LeftButton, 0, downButton); QCOMPARE(spinBox_RectHeight.value(), 10); PASS Executing test function testSpinBoxMouseClicks DEBUG 10		
Test date:	18/04/20	Result:	PASSED

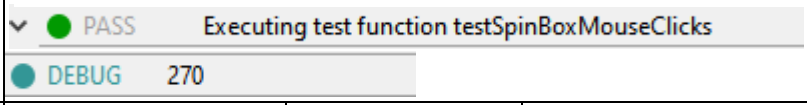
<table>
<tr><td>ID:</td><td>TS_SpBx_Rect_Width
_Clicks_Up</td><td>Description:</td><td>Testing mouse clicks on the up arrow of the rectangle width spin box from a chosen value within the spin box range.</td></tr>
<tr><td>Test type:</td><td>Unit test</td><td>Success criteria:</td><td>The test will simulate the mouse clicking of the up arrow on the rectangle width spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.</td></tr>
<tr><td>Number of attempts:</td><td>1</td><td>Comments:</td><td>This test was created using QTest and QCOMPARE.</td></tr>
<tr><td>Quantity/Quality:</td><td colspan="3">Quality</td></tr>
<tr><td>List of equipment/requirements:</td><td colspan="3">System running any operating system – Windows, Linux, MacOS
Qt creator 5.9 or higher</td></tr>
<tr><td>Setup instructions:</td><td colspan="3">Open application and testing project file and run tst_testgui.cpp file.</td></tr>
<tr><td>Failure correction procedure:</td><td colspan="3">If the chosen value within the rectangle width range doesn’t go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.</td></tr>
<tr><td>Engineer(s)/Technician(s):</td><td colspan="3">Software Tester (Joshua Ssendagire)</td></tr>
<tr><td>Individual results:</td><td colspan="3">

```
/** Testing Mouse clicks on Rectangle Width spin box */

QSpinBox spinBox_RectWidth;

spinBox_RectWidth.setRange(10,400);
spinBox_RectWidth.setValue(270);

spinBox_RectWidth.size();
QPoint(size.width()-2, 2);
QPoint(size.width()-2, size.height()-2);

/** Testing Mouse clicks on Rectangle Width spin box up and down button */

QTest::mouseClick(&spinBox_RectWidth, Qt::LeftButton, 0, upButton);
QCOMPARE(spinBox_RectWidth.value(), 271);
```



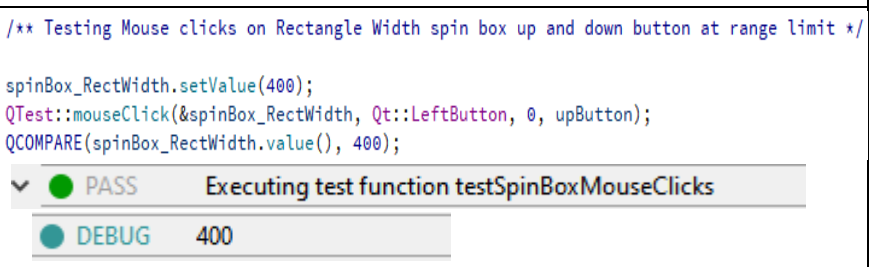
</td></tr>
<tr><td>Test date:</td><td>18/04/20</td><td>Result:</td><td>PASSED</td></tr>
</table>

ID:	TS_SpBx_Rect_Width _Clicks_Down	Description:	Testing mouse clicks on the down arrow of the rectangle width spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the down arrow on the rectangle width spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the rectangle width range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`QTest::mouseClick(&spinBox_RectWidth, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBox_RectWidth.value(), 270);`
	 		

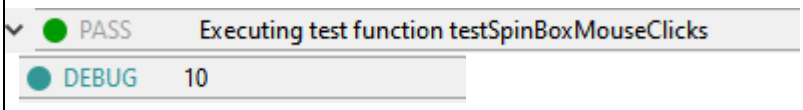
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Rect_Width _Clicks_Up_Range	Description:	Testing mouse clicks on the up arrow of the rectangle width spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the up arrow of the rectangle width spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range

			limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle width range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

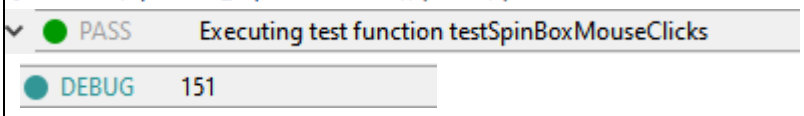
ID:	TS_SpBx_Rect_Width _Clicks_Down_Range	Description:	Testing mouse clicks on the down arrow of the rectangle width spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the down arrow of the rectangle height spin box from the first value within the spin box range. The test will be successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		

Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the rectangle width range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

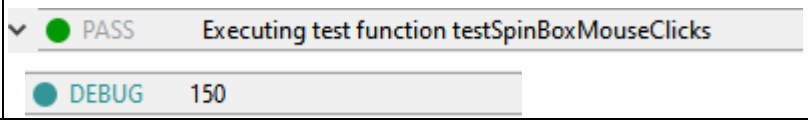
Individual results:	`spinBox_RectWidth.setValue(10);`
	`QTest::mouseClick(&spinBox_RectWidth, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBox_RectWidth.value(), 10);`
	 		

Test date:	18/04/20	Result:	PASSED

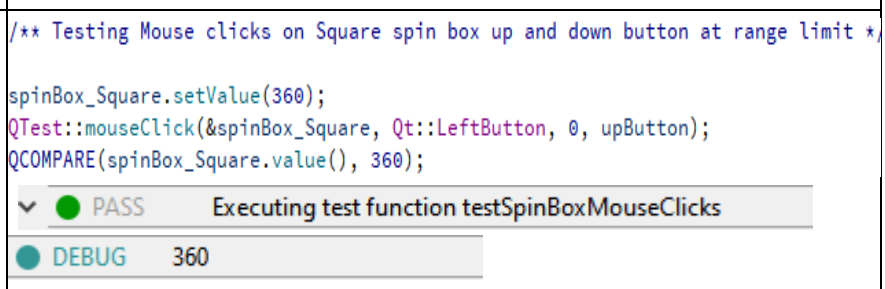
ID:	TS_SpBx_Sqar_ _Clicks_Up	Description:	Testing mouse clicks on the up arrow of the square spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the up arrow on the square spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the chosen value within the square range doesn't go up by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	/** Testing Mouse clicks on Square spin box */
	
	QSpinBox spinBox_Square;
	
	spinBox_Square.setRange(10,360);
	spinBox_Square.setValue(150);
	
	spinBox_Square.size();
	QPoint(size.width()-2, 2);
	QPoint(size.width()-2, size.height()-2);
	
	/** Testing Mouse clicks on Square spin box up and down button */
	
	QTest::mouseClick(&spinBox_Square, Qt::LeftButton, 0, upButton);
	QCOMPARE(spinBox_Square.value(), 151);
	 		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_Sqar_Clicks_Down	Description:	Testing mouse clicks on the down arrow of the square spin box from a chosen value within the spin box range.
Test type:	Unit test	Success criteria:	The test will simulate the mouse clicking of the down arrow on the square spin box from a chosen value within the spin box range. The test will be successful if the chosen value goes up by one after the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the previous value within the square range doesn't go down by one then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:	QTest::mouseClick(&spinBox_Square, Qt::LeftButton, 0, downButton); QCOMPARE(spinBox_Square.value(), 150);  		

Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_SqarClicks_Up_Range	Description:	Testing mouse clicks on the up arrow of the square spin box from the last value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the up arrow of the square spin box from the last value within the spin box range. The test will be successful if the set value doesn't go above the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the square range goes above the last value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		
Individual results:			
Test date:	18/04/20	Result:	PASSED

ID:	TS_SpBx_SqarClicks_Down_Range	Description:	Testing mouse clicks on the down arrow of the square spin box from the first value within the spin box range limit.
Test type:	Unit test	Success criteria:	The test will simulate mouse clicks on the down arrow of the square spin box from the first value within the spin box range. The test will be

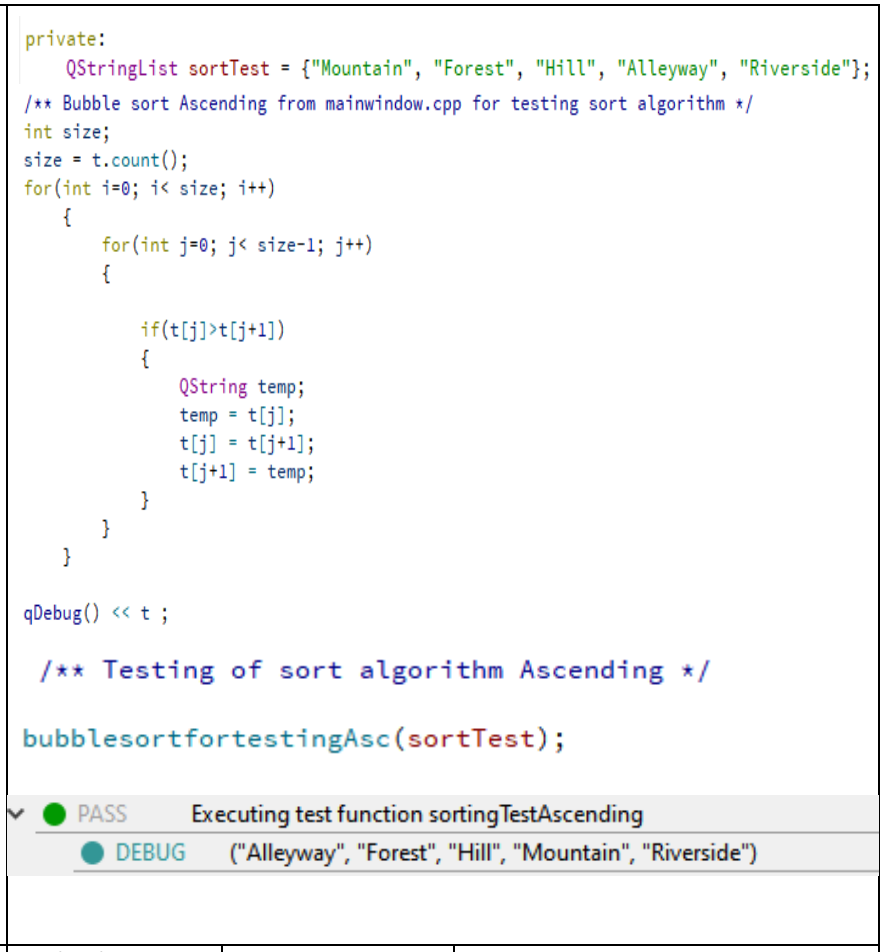
			successful if the set value doesn't go below the range limit the up arrow is pressed.
Number of attempts:	1	Comments:	This test was created using QTest and QCOMPARE.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the value within the square range goes above the first value then QTest will fail the test, showing the expected value and a value within the range must be entered.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:	`spinBox_Square.setValue(10);`
	`QTest::mouseClick(&spinBox_Square, Qt::LeftButton, 0, downButton);`
	`QCOMPARE(spinBox_Square.value(), 10);`
	PASS Executing test function testSpinBoxMouseClicks
	DEBUG 10		

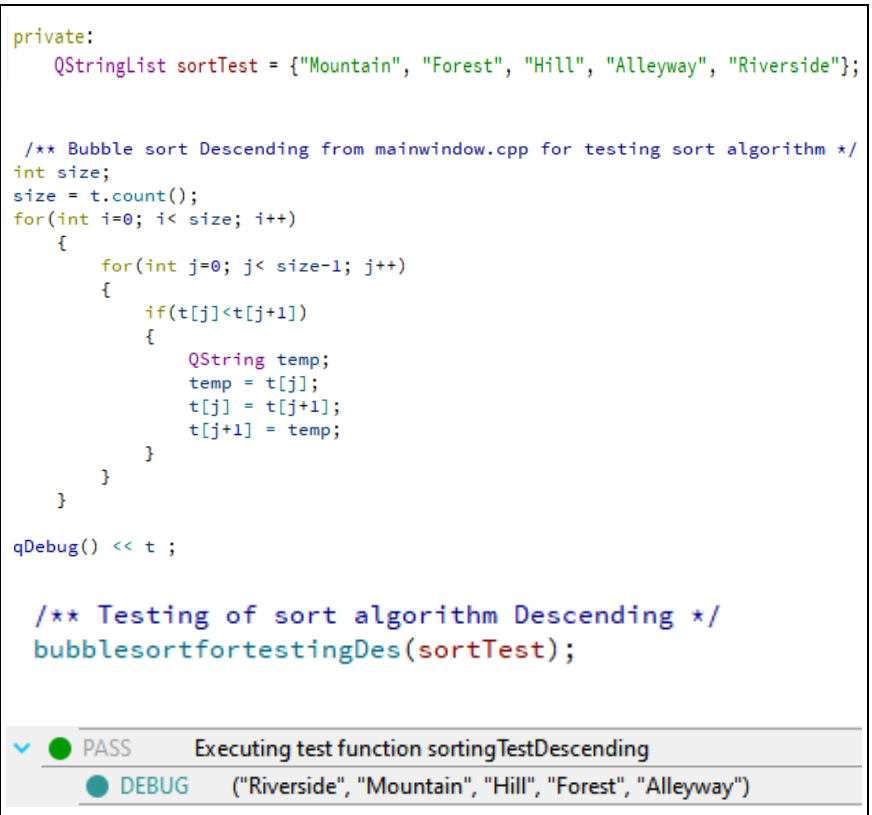
Test date:	18/04/20	Result:	PASSED

Table 18 Sorting:

ID:	TS_ Bubble_ Sort Ascending	Description:	The bubble sort is tested using a list full of test data.
Test type:	Unit test	Success criteria:	The bubble sort must sort the list full of test data in ascending order.
Number of attempts:	2	Comments:	This test verifies the bubble sorts functionality.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the bubble sort test output doesn't sort the list full of test data, the software developer will have to correct the sorting algorithm.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:			
Test date:	18/04/20	Result:	PASSED

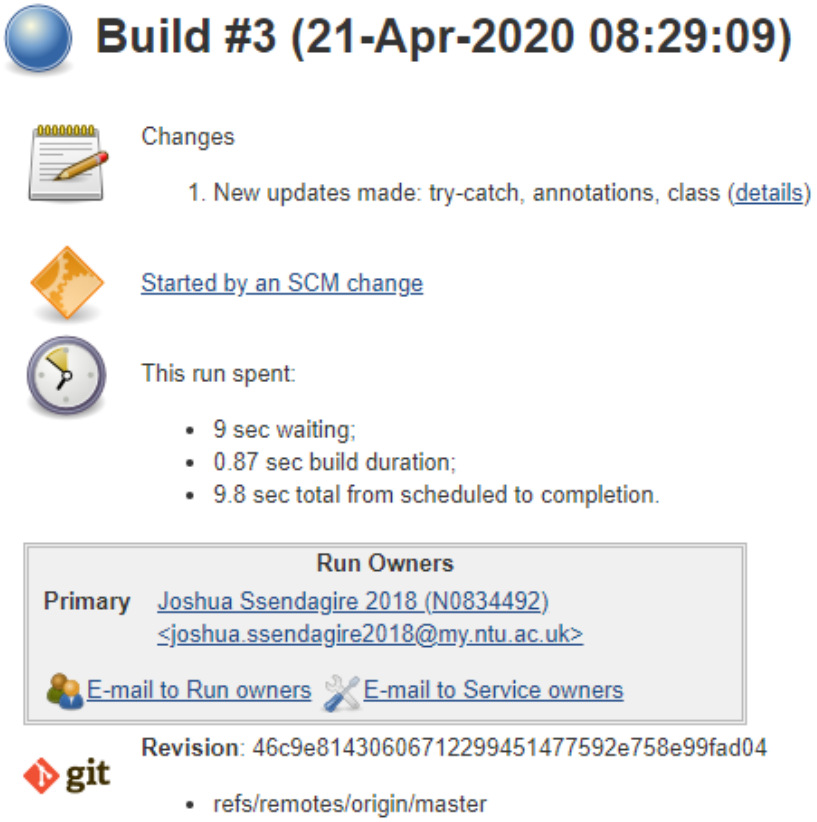
ID:	TS_ Bubble_ Sort Descending	Description:	The bubble sort is tested using a list full of test data.
Test type:	Unit test	Success criteria:	The bubble sort must sort the list full of test data in descending order.
Number of attempts:	2	Comments:	This test verifies the bubble sorts functionality.
Quantity/Quality:	Quality		
List of equipment/requirements:	System running any operating system – Windows, Linux, MacOS Qt creator 5.9 or higher		
Setup instructions:	Open application and testing project file and run tst_testgui.cpp file.		
Failure correction procedure:	If the bubble sort test output doesn't sort the list full of test data, the software developer will have to correct the sorting algorithm.		
Engineer(s)/Technician(s):	Software Tester (Joshua Ssendagire)		

Individual results:			
Test date:	18/04/20	Result:	PASSED

Integration Testing

Continuous integration was utilised and configured with our shared repository using Jenkins, to build and test the code committed to the repository. The continuous integration job is configured to run automatically when the build trigger is triggered. Jenkins has the option of 'Poll SCM' which will cause Jenkins to check the shared repository for any commits since the last build and will run the job. This job doesn't waste as many computational resources as Jenkins other options. A scheduling syntax is needed to tell Jenkins when to run the continuous integration job. The scheduling syntax used in the job was: H 8 * * 1-5. This syntax tells Jenkins to run the job at 8:30AM every day from Monday-Friday. This is beneficial as the job will be run periodically every working day. Changes committed on the weekend will run the job at 8:30PM on the next working day. Running the job everyday isn't necessary and will waste computational resources, so its beneficial to spread the load evenly and have the weekend when jobs are not run.

3.3.2 Poll SCM scheduling syntax



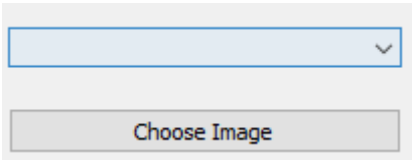
3.32 Successful Poll SCM build

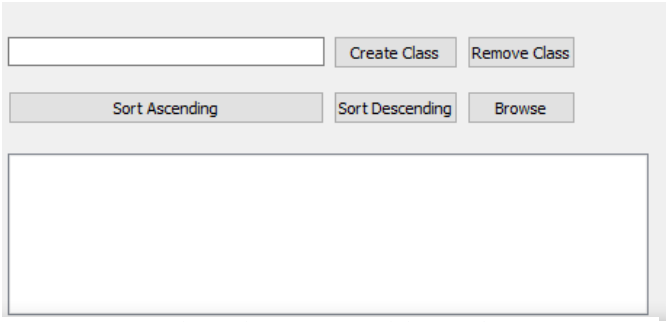
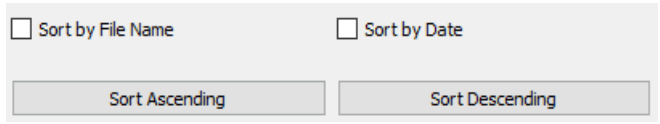
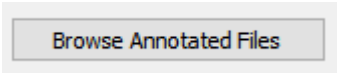
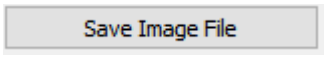
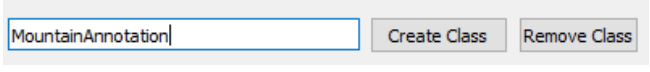
Results using Acceptance Testing

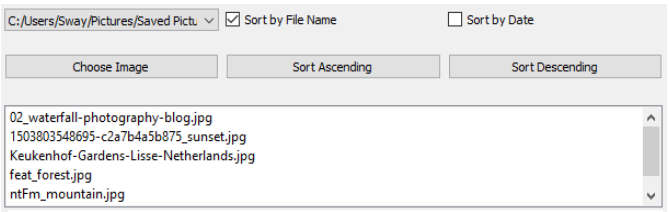
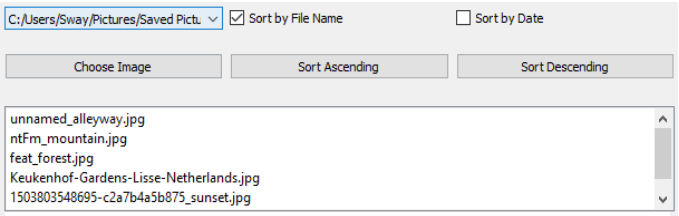
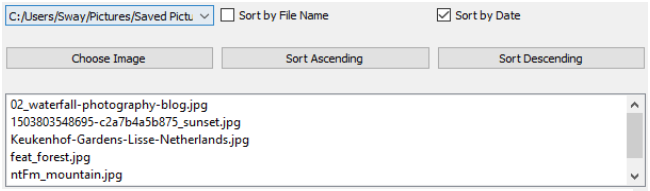
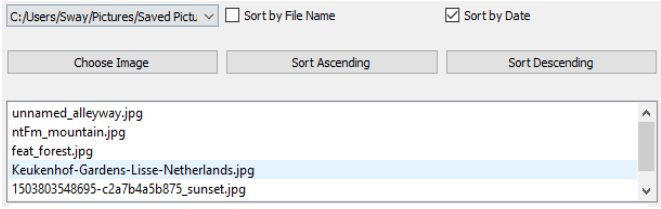
Acceptance testing requires the application to be tested against the requirements of the project to confirm whether the application meets all the requirements and will therefore be accepted by the user.

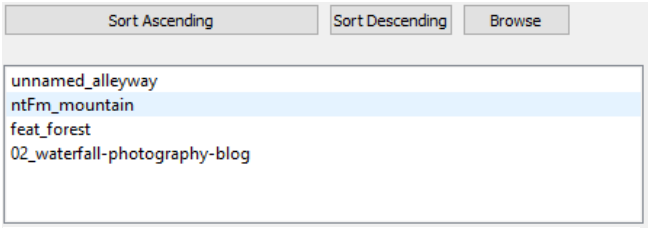
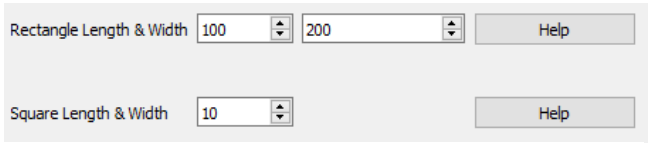
The test video is available online < https://youtu.be/OnlqpTYUO1o>, last accessed <26/04/20>

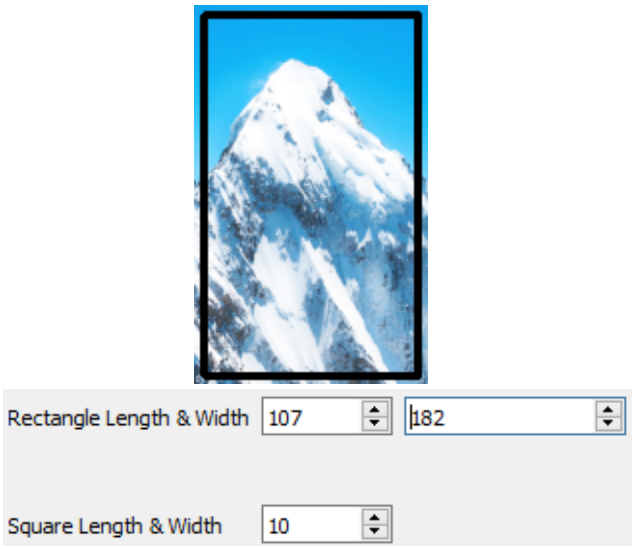
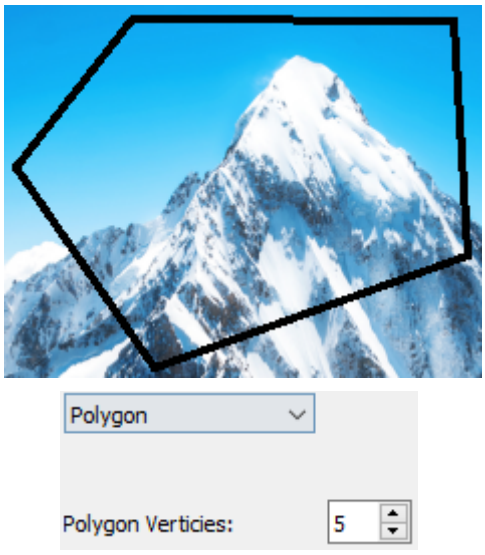
Table 19: Results

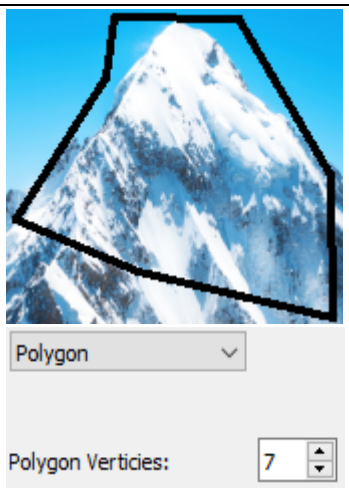
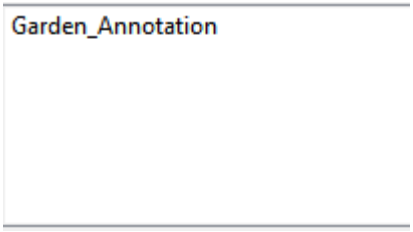
Requirement Name & Number	Requirement(s)	Requirement Evidence
1 Buttons	1.1 - The target folder which contains all compatible images will have buttons for browsing through the folder. 1.2 - The class selector pane will have a browse button for navigating	 *1.1 Browse button (Choose image) for browsing through the target folder.*

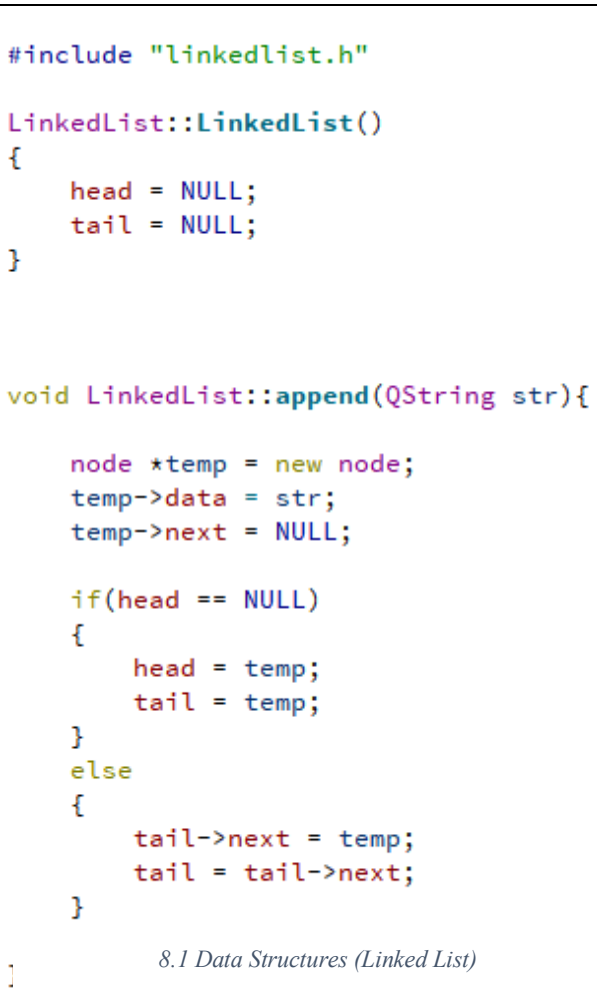
	through folders to select class files. 1.3 - An add and remove button will be implemented for adding and removing classes. 1.4 - An ascend and descend button will be implemented for sorting compatible images files by filename and file date. 1.5 - Open button will be implemented for opening annotation files. 1.6 - Save button will be implemented for saving annotation files. 1.7 - Confirm button will be implemented to confirm user requests. For example, button to confirm the overwriting of a file if it already exists **(Future work).** 1.8 - Cancel button will be implemented to cancel user requests **(Future work).** 1.9 - Help button could be implemented to help the user navigate the application.	 *1.2 Class selector pane with browse button*  *1.3 Add and remove button*  *1.4 Ascend and descend button for sorting*  *1.5 Browse annotation files*  *1.6 Save image file*  *1.9 Help buttons*
2 Text Fields	2.1 – Text fields will be implemented for naming annotation files.	 *2.1 Text fields implemeted*
3 Appending File Changes	3.1 – The addition/removal of	

	classes will be appended to the classes file **(Future work).** 3.2 – The saving of annotation files will be appended to annotation folder **(Future work).**	
4 Sorting/ Searching Algorithms	4.1 – Sorting/searching algorithms will be implemented to sort ascending/descending compatible images by filename and file date. 4.2 - Sorting/searching algorithms will be implemented to sort ascending/descending all classes listed in classes pane.	 *4.1 Sort Images Ascending by File Name*  *4.1 Sort Images Descending by File Name*  *4.1 Sort Images Ascending by Date*  *4.1 Sort Images Descending by Date*  *4.2 Sort Classes Ascending*

		 *4.2 Sort Classes Descending*
5 AutoSave using threads	5.1 – Auto save will be implemented to automatically save annotation files every minute using threads **(Future work).**	
6 Shape Operations	6.1 – The ability to Increase shape sizes will be implemented to increase the shape size 6.2 – The ability to move the vertices of polygons will be implemented to allow the user to modify the shape of polygons **(Future work).** 6.3 – The ability to delete shapes will be implemented to remove shapes which aren't needed (**Future work).** 6.4 – The ability to copy shapes will be implemented to duplicate shapes (**Future work).** 6.5 – The ability to paste shapes will be implemented to paste duplicated images **(Future work).**	 *6.1 Increase size shapes*   *6.1 Increase shape sizes*

	6.6 – The ability to visualise the name of the class on top of the shape will be implemented to recognise the class name of each shape.	 *6.1 Increase shape sizes*  *6.2 Ability to move Polygon Verticies*

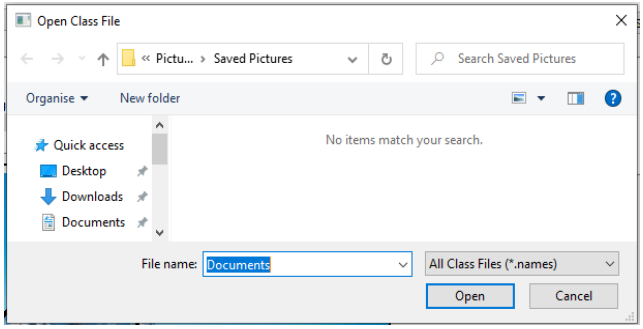
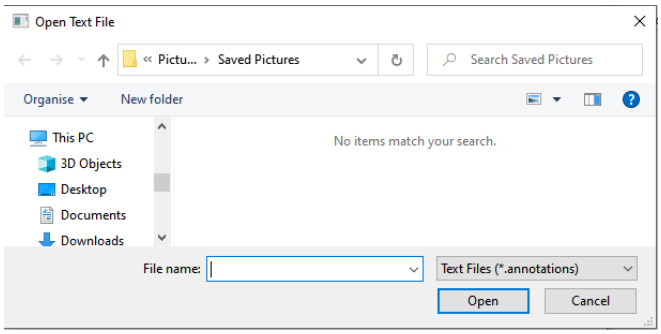
		 *6.2 Ability to move Polygon Verticies (by choosing amount of verticies and clicking each vertex positon on top of image)*  *6.6 Visualise class name*
7 Modifying Annotation Files	7.1 - The user will be able to change the name of an existing file **(Future Work).**	

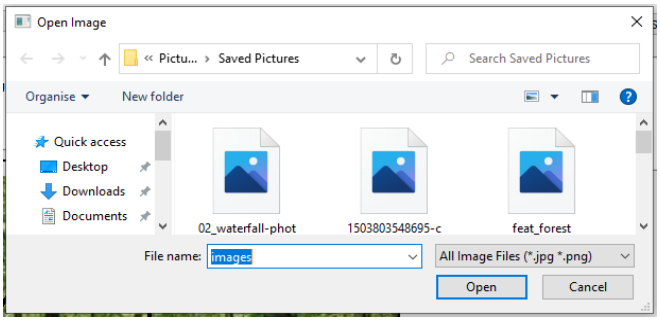
8 Data Structures	8.1 – Data structures will be used to store data in memory through linked lists.	*8.1 Data Structures (Linked List)*  *8.1 Data Structures (Linked List)*
9 Error and Exception Handling	9.1 – If a file already exists, a warning will be displayed to the user to ask the user to confirm the	

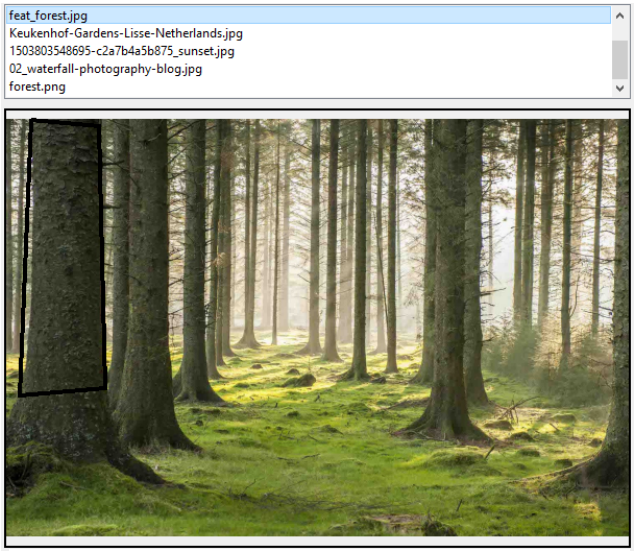
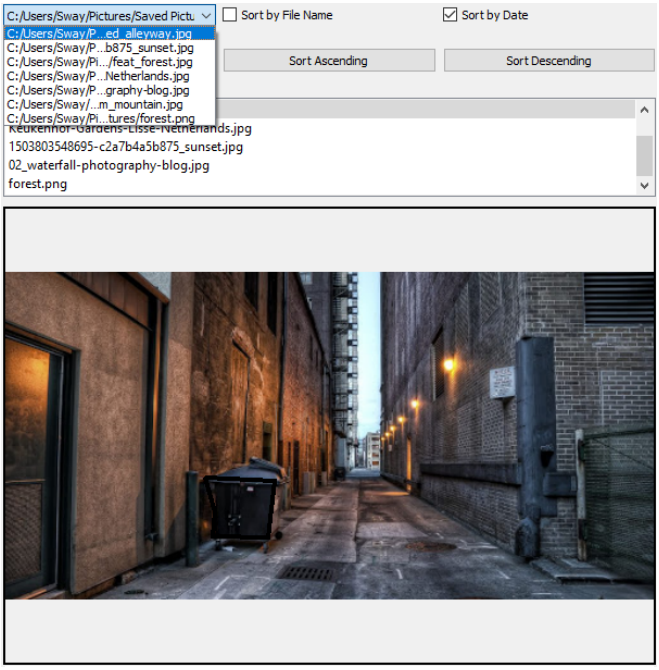
```
#include "linkedlist.h"

LinkedList::LinkedList()
{
    head = NULL;
    tail = NULL;
}



void LinkedList::append(QString str){

    node *temp = new node;
    temp->data = str;
    temp->next = NULL;

    if(head == NULL)
    {
        head = temp;
        tail = temp;
    }
    else
    {
        tail->next = temp;
        tail = tail->next;
    }
}
```

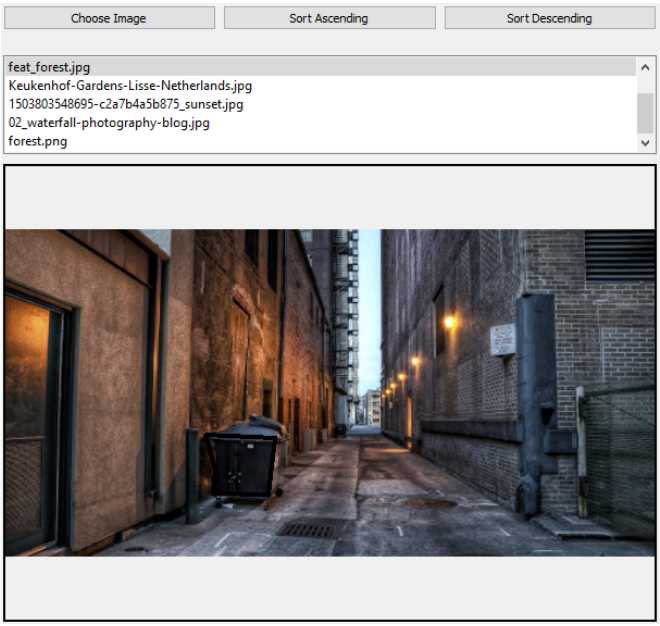
```
int LinkedList::size(){
    node* temp = head;
    int i = 0;
    while (temp != 0){
        temp = temp -> next;
        i++;
    }
    return i;

}
```

	overwrite of the file **(Future work).**	
10 Shape Options	10.1 – Shape option (Triangle) will be implemented for the drawing of a triangle. 10.2 – Shape option (Square/Rectangle) will be implemented for the drawing of a square/rectangle.	 *10.1 Shape Option (Triangle)*  *10.2 Shape Option (Sqaure)*  *10.2 Shape Option (Rectangle)*  *10.2 Shape Option (Trapezium)*

		 *10.2 Shape Option (Trapezium)*
11 Class File Format	11.1 – All class files will be recognised as plain text files with the extension "*. names".	 *11.1 Class Files *.names extension*
12 Annotation File Format	12.1 – All annotations files will be recognised with the extension "*. annotations". 12.2 – All annotation files will follow the hierarchical date format 5 standard (HDF51) **(Future Work).**	 *12.1 Annotation Files *. annotation*

13 Compatible Image File Format	13.1 – All compatible image files will be recognised with the extension "*.jpg," or "*.png".	feat_forest.jpg Keukenhof-Gardens-Lisse-Netherlands.jpg 1503803548695-c2a7b4a5b875_sunset.jpg 02_waterfall-photography-blog.jpg forest.png *13.1 Image files *.jpg, *.png extension*  *13.1 Image files *.jpg, *.png extension*
14 Data Stored in Annotation Files	14.1 – The number of annotated images for each image will be stored in each annotation file **(Future Work).** 14.2 – The image file name will be stored in each annotation file **(Future Work).** 14.3 – The number of shapes per image for each shape will be stored in each annotation file **(Future Work).** 14.4 – The shape type will be stored in each annotation file **(Future Work).** 14.5 – Point_1(x, y) will be stored in each annotation file. **(Future Work).** 14.6 – Point_2(x, y) will be stored in each annotation file **(Future Work).** 14.7 – Point_n1(x, y) will be stored in each	

	annotation file **(Future Work).**	
15 Image Pane	15.1 – The image pane will display the users selected image 15.2 – The image pane will allow the user to select any compatible image.	 *15.1 Image pane displaying selected image*  *15.2 Image pane allows user to select chosen images*

		 *15.2 Image pane allows user to select images using the choose image button*
16 Shape Drawing	16.1 – The user will be able to draw any shape on top of any compatible image.	 *15.2 Image pane allows user to select images using the choose image button*
17 Annotations File and Filename Selector Options	17.1 – The user will be able to open and load annotation files **(Future Work).** 17.2 – The user will be able to save annotation files **(Future Work).**	

Risks & Issues

Considering a risk analysis has been created for the project, a risk analysis will also need to be completed for the testing phase of the project. This risk analysis will highlight the risks when testing and how to mitigate these risks. The table includes a risk, probability scale of a risk occurring, impact scale of risk, the impact on the testing and prevention method. Probability Scale: 1 – Very Unlikely, 2 – Moderately Unlikely, 3 - Likely, 4 – Moderately highly likely, 5 – Very likely. Impact Scale: 1 – Low impact, 2 – Moderately low impact, 3 - Likely, 4 – Moderately highly impact, 5 – High impact.

Table 20: Risk & Issues

Risk	Probability (1-Very Unlikely, 5-Very Likely	Impact (1-Low Impact, 5-High Impact)	Impact on testing	Prevention method
Software tester's lack of experience testing.	3	5	Having a lack of experience in testing could lead to inaccurate testing, which would result in unrecognised defects and a broken application.	This lack of experience can be avoided by conducting thorough background research into software testing. Resources such as the internet could lower the impact of this risk.
Project schedule falls behind meaning less time for testing,	3	5	The project schedule falling behind would lead to the testing schedule falling behind too. This would also leave less time for testing which would impact the integrity of the application and could impact the	Following the project schedule closely and creating hard deadlines will minimise the risk of the project schedule falling behind and less time for testing.

			quality of the testing.	
Loss of software tester	2	5	If for some reason the software tester is to leave or fails to continue to communicate with the team, the testing phase of the project will fail. This will become a huge problem, if the rest of the team aren't familiar with test plan.	If the software tester leaves the project, one of the group members should step up and take responsibility. The team should document that the team has lost their software tester during the project. If the software tester leaves after creating the test plan, this document will help them understand the role of the software tester.
Poor test plan	3	4	A poor test plan can cause bugs in the application. This would have a bad impact on the final project and lead to inaccuracies in testing.	A poor test plan can cause problems. A generous amount of time should be given to create a good test plan. By creating a test plan in the early development stages there will be fewer chances of poor testing as more focus will be given to the testing stage.

Significant changes in user requirements/introduction of new requirements	4	5	The requirements of the project should be fully investigated and agreed before the specification. If there is a change or an addition to the requirements this could lead to the testing process becoming longer or more difficult depending on the change of requirement.	The team will need to discuss any new requirements which are introduced. This will allow the software tester to prepare for the testing of this requirement.
Availability of software (resources)	4	4	Software that is needed for testing may not be available on every operating system. This could lead to inaccuracies in testing, if these resources aren't made available.	The software tester will have to find an alternative resource for testing.
Conflicting test priorities	2	5	While testing, there might be conflicting test priorities where the software tester might prioritise the wrong type of testing at the wrong time.	The software tester will have to create a detailed test plan and follow it closely.
Product complexities	4	5	If the product become too complex, this could lead to the testing also becoming complex. As a	The software tester needs to prepare accordingly for the product complexities with help from

			result, the software tester might fail to test the application properly.	the internet and the team.
Poor communication	3	4	Lack of communication between the software tester and the rest of team will result in a lack of cohesion through the project. This will result in team members not being able to understand all parts of the project including the testing phase.	This can be solved by using a communication tools such as Slack, where members can continuously communicate to share work and update all members with the progress of the test phase.

Test Logistics

The Test Logistics highlight the members of the team who will be responsible for testing as well as when testing will take place. In the team, Joshua Ssendagire is the Software Tester and therefore responsible for the testing the application. The Software Tester must understand the project and its requirements, be able to cooperate, be meticulous and have a strong desire for accuracy. The team was confident with their choice of Software Tester as they possess all these qualities. The project plan explains that testing should occur after development and implementation stages are complete.

Test Objectives

Within the testing phase of the project, there are objectives and goals which need to be defined to ensure that the application is debugged and without any fault before the submission.

Test Objectives	Test Objective Description
Find and Prevent defects/faults in application	One of the main objectives of testing is to find and prevent defects/faults in the application. Finding defects/faults will improve the quality of the application as well as give the software tester a better understanding of how to prevent these faults from affecting

	the end goal of the project. This will reduce the risk of software inaccuracies before the submission of the application.
Verify and validate project requirements	System and Acceptance testing allows the testing of the application against the requirements of the project. In general, testing will verify and validate whether all project requirements have been met.
Evaluate the application's performance	Testing will help to evaluate aspects of the application such as performance and usability. The application will be tested for its ease of use as one of the requirements is to have a simple GUI. Although the performance of the application isn't defined in the requirements of the project, it is important for the software tester to test the application against non-functional aspects.
Ensure quality of application	Testing allows for the ensuring of quality of the application. All bugs, defects and faults will be tested for and handled accordingly which will in turn ensure that the application will be of a high quality.

Test Criteria

The test criteria is a very important aspect within the testing phase. The test criteria provides the software tester with an understanding of how to judge the success rate of the test phase. From testing, the software tester will be able to conclude whether the application hasn't failed any test cases and can move on to the next stage of development. This is the Exit criteria. Or the application has failed too many test cases and the testing must be suspended until these test cases are passed. This is the Suspension criteria.

The Exit criteria for this project is: 90% of all critical test cases must pass. This high pass rate percentage will ensure high quality of the application. Not only will this push the team to work to a high standard but will also provide more efficient and accurate way of working moving forward in the project if this percentage is met. The team is allowing for 10% of all critical test cases to not pass to advocate for problems when programming and testing.

The Suspension criteria for this project is: if team members report that 45% of test cases failed, then testing should be suspended until the software developer and software architect fixes the failed cases. This percentage rate below 50% will allow for the development team to program efficiently to avoid the suspension of testing. 45% isn't too high or too low, which allow for some error if needed.

Resource Planning

A Resource plan lists all resources from human to system which are needed to complete the project. Resource planning is important within the test plan as it will help the software tester complete the schedule and estimation for the project.

Table 21: Resource planning

Human Resource Number	Human Resource Member	Human Resource Task
1	Project Manager	1.1 – Manage the team and whole project. 1.2 – Preparing, submitting and presenting project plan. 1.3 – Help implement labelling application using C++. 1.4 – Help create UML diagrams. 1.5 – Deliver structured report. 1.6 – Submit report.
2	Software Architect	2.1 – Preparing, submitting and presenting the project design (UML diagrams). 2.1.1 – Use case Diagram(s) 2.1.2 – Class Diagram(s) 2.1.3 – Sequence Diagram(s) 2.1.4 – State Diagram(s) 2.1.5 – Component Diagram(s) 2.2 – Design GUI. 2.3 – Help implement labelling application using C++. 2.4 – Deliver structured report.
3	Software Developer	3.1 – Preparing, submitting and presenting reference manual. 3.2 – Create contribution guide. 3.3 – Implement labelling application using C++. 3.4 – Help create UML diagrams. 3.5 – Deliver structured report.
4	Software Tester	4.1 – Preparing, submitting and presenting the test plan. 4.1.1 – Identifying the best test techniques/tools/approach. 4.2 – Use test plan to test application. 4.2.1 – Test application, record results and fix defects. 4.2.2 – Implement test cases. 4.3 – Help create UML diagrams. 4.4 – Help Implement labelling application using C++ 4.5 - Deliver structured report

System Resource Number	System Resources	System Resource Description
1	Computer	A computer is needed to access the labelling application.
2	Application	Install and run the labelling application.
3	Test tool	A test tool like Jenkins and QTest for unit, integration and acceptance testing.

Test Environment

The testing environment is the environment of software and hardware, where the software tester executes test cases. The test environment for the labelling application requires the software developer and software tester to get involved. These two members will need to work closely and discuss the labelling application to create an accurate test environment. To test the labelling application successfully:

Hardware – PC/Laptop is required.

Software – Linux, MacOS or Windows 7 and above is required.

Jenkins must be installed/accessed for integration testing.

Qt must be installed for unit/acceptance testing.

Network – No internet is required. Testing can be executed offline.

Schedule & Estimation

Test estimation manages and determines how long a task takes to complete. Effort within the test is also estimated, which is an essential task in test management. The table below annotates the testing phase tasks, subtasks and the members responsible for the completion of these tasks.

Table 22: Schedule & Estimation

Task	Subtask	Member
Analyse product and product requirements.	Discuss product with group member for greater understanding of product specification.	Project Manager, Software Architect, Software Developer, Software Tester.
Create Test Plan	Create test strategy. Create test objectives. Create test criteria.	Software Tester.
Create test cases	Discuss test cases with group members. Revise and change test cases based on discussion.	Project Manager, Software Architect, Software Developer, Software Tester Software Tester
Execute test cases	Build test environment. Execute test cases. Review test case results.	Software Tester Software Tester Software Tester

	Discuss Exit and Suspension criteria.	Project Manager, Software Architect, Software Developer, Software Tester
Record and report application defects	Create defect report Report/fix defects	Project Manager, Software Architect, Software Developer, Software Tester. Software Tester

Test Deliverables

During testing, there are deliverables. Each deliverable throughout the testing phase will need to be met. Below are the deliverables for the testing phase:

Deliverable 1 – Test Plan

Deliverable 2 – Test Cases

Deliverable 3 – Test Strategy

Deliverable 4 - Unit Tests

Deliverable 5 – Integration Tests

Deliverable 6 – Acceptance Tests

Deliverable 7 – Project Sign Off

Conclusion and Future Work

In conclusion, the majority of the tests executed in application had passed. In terms of unit testing, all 11 tests were successful. Tests for linked lists haven't been created, however this is something that could be done as future work of application. In terms of integration testing, the majority of our test builds were a success, which means there wasn't any error when building after committing changes to the application. For acceptance testing, the application was able to meet the majority of the requirements. In addition, the requirements that were not met will be able to be met as future work of the application.

The team has created an application where the user is able to open images into the application draw shapes on top of the image and label the contents of the image as a class. The team has implemented sorting and linked lists which were mandatory requirements and have also included some extra features such as the ability to change colours of the shape and the width of the lines.

Performance

Currently the application is able to do perform most of the functionality that were set out. The teams main aim was to deliver appropriate data structures for the storage of data in the memory, sorting algorithms the ability to draw shapes and save annotations; this can be done by the application the team has produced.

However, the application can perform the copy and pasting of shapes and the saving of images with shapes drawn on them. Along with this the team was not able to implement the ability to autosave.

Future work in regards to performance: the future aim of the team would be implementing threading to improve performance of the program so the autosaving functionality can be performed. To increase performance of the program the team could use different frameworks o draw shapes as the code can be seen quite extensive specifically in regards to

Computational efficiency

The computational efficiency in the program can be improved significantly as the code in to draw shapes is very long and inefficient however other parts of the program are efficient. Also, the sorting algorithm we have chosen is the bubble sort which has a time complexity of O(n); a more efficient sorting algorithm would the quick sort algorithm which has O(n^2).

Reliability and Portability

The application we have created is reliable at the moment with no major errors that have occurred in the testing. The application is not portable as it requires qt creator to work and has not been deployed.

Scalability

The project is highly scalable as there some functionality missing which an worked on and improved. Once the future improvements are made the project can be scaled to an extent to which it can be used for CNN. However, this is only after improvements have been made to the application.

Overall, for the future there are a significant number of changes and improvements that can be made specifically in file handling, rendering shapes and making the code more efficient by using concepts like inheritance for shapes and more.

References

Agilemodeling.com. 2020. *UML 2 Deployment Diagrams: An Agile Introduction*. [online] Available at: <http://www.agilemodeling.com/artifacts/deploymentDiagram.htm> [Accessed 24 March 2020].

Concise Software. 2020. *ATAM - The Architecture Tradeoff Analysis Method. What It Is?*. [online] Available at: <https://concisesoftware.com/architecture-tradeoff-analysis-method-atam/> [Accessed 15 April 2020].

Erder, M., 2020. *Architecture Tradeoff Analysis Method - An Overview | Sciencedirect Topics*. [online] Sciencedirect.com. Available at: <https://www.sciencedirect.com/topics/computer-science/architecture-tradeoff-analysis-method> [Accessed 15 April 2020].

Qin, Z., Xing, J. and Zheng, X., n.d. *Software Architecture*.

Shen, X., 2002. *An Architecture Tradeoff Analysis Of Postgresql*.

Visual-paradigm.com. 2020. *UML Class Diagram Tutorial*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/> [Accessed 10 April 2020].

Visual-paradigm.com. 2020. *What Is Activity Diagram?*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/> [Accessed 12 April 2020].

Visual-paradigm.com. 2020. *What Is Component Diagram?*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/> [Accessed 15 April 2020].

Visual-paradigm.com. 2020. *What Is Deployment Diagram?*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/> [Accessed 24 March 2020].

Visual-paradigm.com. 2020. *What Is Sequence Diagram?*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/> [Accessed 15 April 2020].

Visual-paradigm.com. 2020. *What Is State Machine Diagram?*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/> [Accessed 16 April 2020].

Visual-paradigm.com. 2020. *What Is Use Case Diagram?*. [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> [Accessed 17 April 2020].

Guru99.com. 2020. *How To Create Test Plan*. [online] Available at: <https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html/> [Accessed 15 March].

Overall Reflection

Overlooking on the work the group has done as the project manager I believe the group got off to a good start with the project plan, project design and the beginning of the development. During the development there were a few problems which meant certain task had to be reassigned to other members of the group. The testing stage was thoroughly prepared for but due to a delay in the development stages the testing stage had to be pushed back slightly. There are some functionality missing which have been documented throughout the report and can be areas in the future that can be worked on.

Overall the team has put in a good effort with a good project design, most of the functionality is achieved with a few extra features and a good test plan with appropriate testing.

Appendix

Appendix A

Reference manual can be found in the documentation folder.