Group 25

Software Design and Implementation Coursework

Project manager - Bethan Rees;
Software Architect - Michael Gibbs;
Software Developer - Cait Burke;
Software Tester - Harry Evans
4-26-2020

Lab tutor – Kayode Owa

Abstract

Artificial Intelligence is used to aim to replicate human capabilities in machines, it is being applied in many areas including Computer Vision. Convolutional Neural Networks (CNN) are being used for classification. The CNNs need to use a labelling tool. We propose to implement a labelling application using C++ to annotate datasets. The application will have the following functionalities, searching for images, drawing and labelling shapes, classifying the shapes into groups and saving of the annotated image. In truth we accomplished a working project however we are not fully able to edit shapes once they have been placed. The application is able to search, load and save images, as well as classify, draw and label shapes.

Contents

Revision history

Version	Issue date	Stage	Changes	Authors
0.1	02/04/2020	Draft	Initial draft and outline	Bethan Rees (PM)
0.2	06/04/2020	Draft	Initial inserts of data	Bethan Rees (PM)
1	13/04/2020	Version 1	All completed deliverables inserted (deliverables not finalised)	Bethan Rees (PM)
1.1	20/04/2020	Version 1		Michael Gibbs (SA)
1.2	22/04/2020	Version 1	Addition of an abstract and formatting	Bethan Rees (PM)
1.3	22/04/2020	Version 1	Addition of Explanations	Harry Evans (ST)
1.4	24/04/2020	Version 1	Addition of Time Plan	Bethan Rees (PM)
1.5	24/04/2020	Version 1	Addition of user manual	Cait Burke (SD)
2	25/04/2020	Version 2	Addition of tests, inclusion of conclusion and overall reflection	Bethan Rees (PM)
2.1	25/04/2020	Version 2	Addition of conclusion and explanation to class diagram	Michael Gibbs (SA)
2.2	25/04/2020	Version 2	Finalising of conclusion and overall reflections.	Bethan Rees (PM)
2.3	26/04/2020	Version 2	Addition of GUI mock-up	Michael Gibbs (SA)
2.4	26/04/2020	Version 2	Finalising conclusion	Harry Evans (ST)
3	26/04/2020	Final Version	Finalising of document	Bethan Rees (PM)

List of tables

List of images

Introduction

For our assignment we have developed a labelling application using C++ for annotating Datasets for being used by CNNs. The application is a labelling tool that allows Datasets to be inserted. The inserted datasets can be sorted and searched through until your desired dataset is selected. The application allows your dataset to be classified and annotated. The classifications can be loaded and sorted through. The classifications can store shapes. The shapes are drawn on the image and labelled as to annotate the dataset. We have used C++ and QT to develop this application.

Background research

List of Libraries and Tools

During the design phase of a project we need to consider what software tools and libraries are going to be used in order to create the outcome quickly and efficiently. Planning which software you are going to use means you can allocate time to potentially learning how to use the software as well as actually planning/designing the piece you are working on. Applying this additional planning can save exponential amounts of time, depending on the size of the project. This is mainly because if you decide to use some software then you and your team will need to then learn how to use it. This process can take lots of time especially when others come to critique work as they don't know how to add improvements because they're unfamiliar with the software.

We need to consider which C++ libraries to install and import into our project because if we were to try and code everything from nothing our project will take a very long time to complete; especially when another library can do lots of the work for us. Picking a C++ library can be a very difficult process as it deeply affects the project. In our case the team have little to no experience using any external C++ libraries. Therefore, we need to make sure a reliable, easy-to-use library is chosen that will allow us to build a stable graphical user interface (GUI) and minimise the time to do it.

C++ GUI Libraries

The C++ GUI library we are going to be using is Qt. It's a free, open source collection of GUI libraries which is highly rated in software implementation. While researching GUI libraries, many forums and recommendations were for Qt. This is because, as of Qt 5, it allows you to build a "windowing system, which lets you create user interfaces, and includes advanced features such as displaying charts” (David Bolton, 2016).

UML Design Software

We were looking for a UML design software that everyone could download, learn and use quickly, while also looking for something that can turn a diagram straight into C++ code. We found Visual Paradigm which we are going to use as the main design tool because we can build all the UML diagrams we require as well as transfer the class diagram to C++ code. This will save lots of time in the future because of the number of classes, methods and attributes we are dealing with. Visual Paradigm is also a free software to use at our level which is also a positive when developing without a set budget.

Image Processing Library

We are going to be using Magick++, which is “the object-oriented C++ API to the ImageMagick processing library” (Magick++, no date). Magick++ provides many functions and classes that are made for processing many different image formats. This is beneficial to us because we need to be able to load as many different “compatible” files as possible. This library gives us access to achieve this without manually coding these functions, which if we did, could take months, if not years.

CI/CD Tools

We are going to be looking at learning and implementing a CI/CD tool called Jenkins. This tool will help us continually deliver a version of our project periodically. Jenkins is a piece of software that we can target to retrieve certain data while our software is running. For example, we can get it to collect any crashes and exceptions and from this information we can then see where the main bugs are and get onto fixing them before the next version release.

Generating Documentation from Annotated C++ Sources Tools

For out project we are going to be using Doxygen. Doxygen is a tool we are going to use to shave lots of time off getting to the end goal. We can use our UML diagrams to start coding and use Doxygen to generate some documentations from the code (providing it’s commented thoroughly), all the while adding to the contribution guide. This will be very beneficial to have because we can then read the documentation before adding or making further improvements.

Design

Requirements List

The full requirements list can be found in appendix 1.

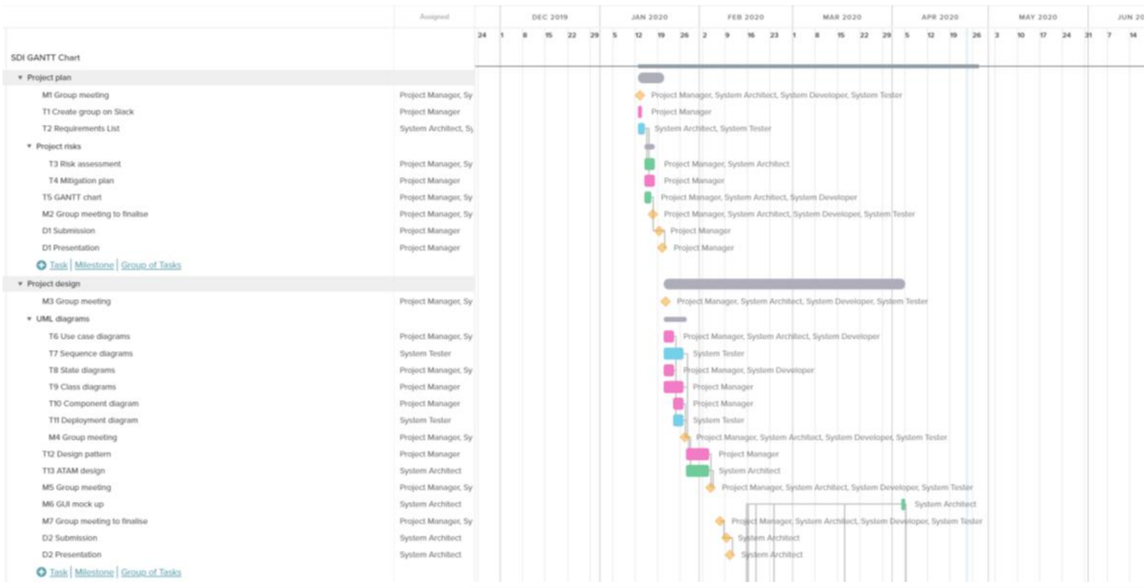
Requirement Number	Label	Requirement Description			Requirement Implementation	Task(s)
1	T14	Simple GUI where the user **MUST** be able to navigate to the chosen file			Making use of file handling techniques such as read, write and save file.	Familiarise ourselves with file handling techniques from C++.
2	T26	System **SHOULD** list the compatible image files (e.g. *.jpg *.png) available in that folder.			Configuring the system to display the available file formats.	Obtain available file formats.
3	T25	The files **SHOULD** be sorted into:	Ascending Order	By the file name	Allow user to select whether files are to be sorted into ascending or descending order. Allows the user to select the option.	Research some efficient sorting algorithms, such as merge and insertion.
4	T25			By the file date		
5	T25		Descending Order	By the file name		
6	T25			By the file date		
7	T14	Users **MUST** be able to navigate and select class folders.			Insert a browse button to each class to improve navigation experience. Class files identified as plain text files corresponds to a class.	Insert browse button. Specify which text file corresponds to which class.

Risk assessment

The full risk assessment can be found in appendix 2.

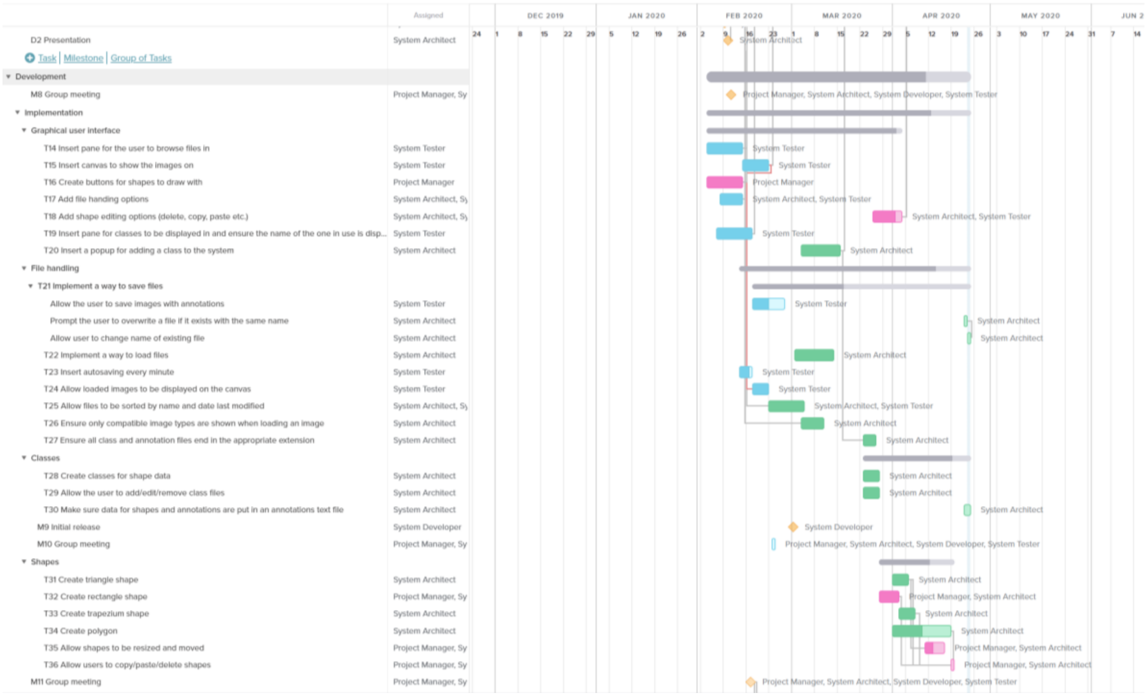
Risk	Probability (1- Very Unlikely, 5- Very Likely).	Impact (1- Low, 5- High).	Impact on project.	Prevention Method/ Ways to mitigate.
Bugs in code	5	3	Finding bugs in your code can be very time consuming to deal with and some may be very threatening to how the software may run.	The best way to potentially prevent having bugs in our code is to have a longer planning period, such that when we come to implement the code, there is a clear layout of what needs to be done. Otherwise, we may miss a feature and could potentially break another section of code trying to introduce new code.
Loss of source code	2	5	This could have a major impact due to the project needing to be reinvented from scratch or the last saved code.	We will make sure that we have frequent back-ups of the source code as well as saving it on a cloud-based server. We could potentially upload the current progress onto GitHub to get feedback and share any changes with the group right away all the while keeping the save safe.
Technical issues	3	3	Technical issues could potentially lead to loss of code if code is not periodically saved. It could also increase project time and costs as we may need to wait for the issues to be resolved.	To prevent this, we will regularly update our software and ensure that we keep back-ups before making large changes. Technical issues can be hard to predict and therefore this is the main way we can go about trying to reduce the risk.

Time Plan

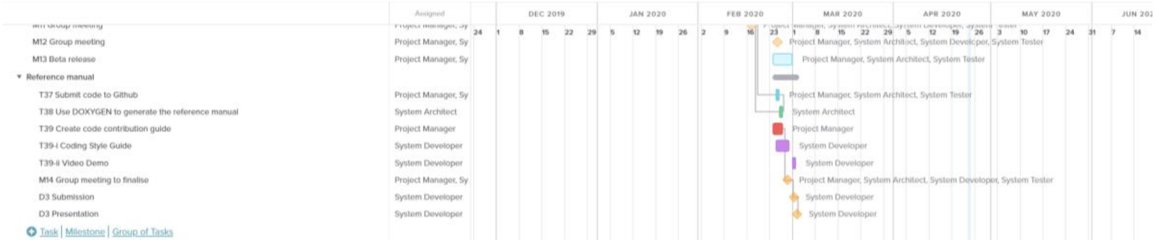


SDI GANTT Chart

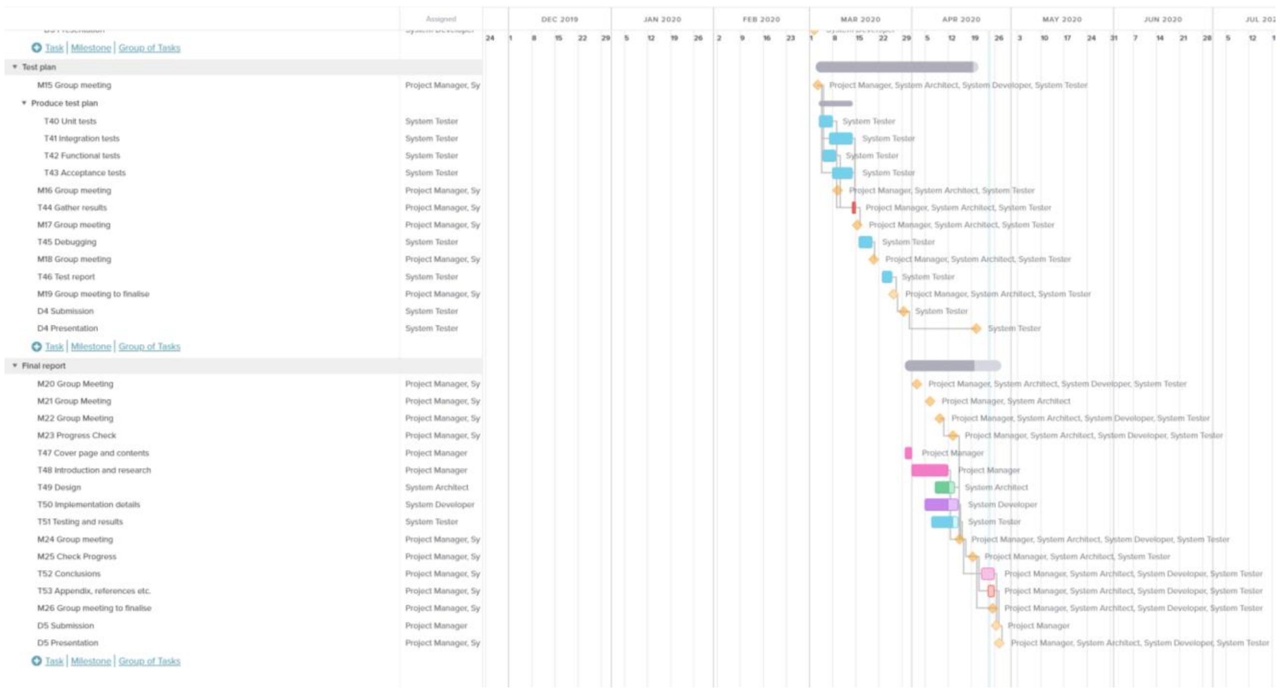
	Assigned
▾ Project plan	
M1 Group meeting	Project Manager, Sy
T1 Create group on Slack	Project Manager
T2 Requirements List	System Architect, Sy
▾ Project risks	
T3 Risk assessment	Project Manager, Sy
T4 Mitigation plan	Project Manager
T5 GANTT chart	Project Manager, Sy
M2 Group meeting to finalise	Project Manager, Sy
D1 Submission	Project Manager
D1 Presentation	Project Manager
Task | Milestone | Group of Tasks	
▾ Project design	
M3 Group meeting	Project Manager, Sy
▾ UML diagrams	
T6 Use case diagrams	Project Manager, Sy
T7 Sequence diagrams	System Tester
T8 State diagrams	Project Manager, Sy
T9 Class diagrams	Project Manager
T10 Component diagram	Project Manager
T11 Deployment diagram	System Tester
M4 Group meeting	Project Manager, Sy
T12 Design pattern	Project Manager
T13 ATAM design	System Architect
M5 Group meeting	Project Manager, Sy
M6 GUI mock up	System Architect
M7 Group meeting to finalise	Project Manager, Sy
D2 Submission	System Architect
D2 Presentation	System Architect
Task | Milestone | Group of Tasks	



	Assigned
D2 Presentation	System Architect
Task | Milestone | Group of Tasks	
▾ Development	
M8 Group meeting	Project Manager, Sy
▾ Implementation	
▾ Graphical user interface	
T14 Insert pane for the user to browse files in	System Tester
T15 Insert canvas to show the images on	System Tester
T16 Create buttons for shapes to draw with	Project Manager
T17 Add file handing options	System Architect, Sy
T18 Add shape editing options (delete, copy, paste etc.)	System Architect, Sy
T19 Insert pane for classes to be displayed in and ensure the name of the one in use is disp...	System Tester
T20 Insert a popup for adding a class to the system	System Architect
▾ File handling	
▾ T21 Implement a way to save files	
Allow the user to save images with annotations	System Tester
Prompt the user to overwrite a file if it exists with the same name	System Architect
Allow user to change name of existing file	System Architect
T22 Implement a way to load files	System Architect
T23 Insert autosaving every minute	System Tester
T24 Allow loaded images to be displayed on the canvas	System Tester
T25 Allow files to be sorted by name and date last modified	System Architect, Sy
T26 Ensure only compatible image types are shown when loading an image	System Architect
T27 Ensure all class and annotation files end in the appropriate extension	System Architect
▾ Classes	
T28 Create classes for shape data	System Architect
T29 Allow the user to add/edit/remove class files	System Architect
T30 Make sure data for shapes and annotations are put in an annotations text file	System Architect
M9 Initial release	System Developer
M10 Group meeting	Project Manager, Sy
▾ Shapes	
T31 Create triangle shape	System Architect
T32 Create rectangle shape	Project Manager, Sy
T33 Create trapezium shape	System Architect
T34 Create polygon	System Architect
T35 Allow shapes to be resized and moved	Project Manager, Sy
T36 Allow users to copy/paste/delete shapes	Project Manager, Sy
M11 Group meeting	Project Manager, Sy



	Assigned
M12 Group meeting	Project Manager, Sy
M13 Beta release	Project Manager, Sy
▾ Reference manual	
T37 Submit code to Github	Project Manager, Sy
T38 Use DOXYGEN to generate the reference manual	System Architect
T39 Create code contribution guide	Project Manager
T39-i Coding Style Guide	System Developer
T39-ii Video Demo	System Developer
M14 Group meeting to finalise	Project Manager, Sy
D3 Submission	System Developer
D3 Presentation	System Developer
Task | Milestone | Group of Tasks	



	Assigned	
▾ Test plan		
M15 Group meeting	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
▾ Produce test plan		
T40 Unit tests	System Tester	System Tester
T41 Integration tests	System Tester	System Tester
T42 Functional tests	System Tester	System Tester
T43 Acceptance tests	System Tester	System Tester
M16 Group meeting	Project Manager, Sy	Project Manager, System Architect, System Tester
T44 Gather results	Project Manager, Sy	Project Manager, System Architect, System Tester
M17 Group meeting	Project Manager, Sy	Project Manager, System Architect, System Tester
T45 Debugging	System Tester	System Tester
M18 Group meeting	Project Manager, Sy	Project Manager, System Architect, System Tester
T46 Test report	System Tester	System Tester
M19 Group meeting to finalise	Project Manager, Sy	Project Manager, System Architect, System Tester
D4 Submission	System Tester	System Tester
D4 Presentation	System Tester	System Tester
▾ Final report		
M20 Group Meeting	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
M21 Group Meeting	Project Manager, Sy	Project Manager, System Architect
M22 Group Meeting	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
M23 Progress Check	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
T47 Cover page and contents	Project Manager	Project Manager
T48 Introduction and research	Project Manager	Project Manager
T49 Design	System Architect	System Architect
T50 Implementation details	System Developer	System Developer
T51 Testing and results	System Tester	System Tester
M24 Group meeting	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
M25 Check Progress	Project Manager, Sy	Project Manager, System Architect, System Tester
T52 Conclusions	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
T53 Appendix, references etc.	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
M26 Group meeting to finalise	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester
D5 Submission	Project Manager	Project Manager
D5 Presentation	Project Manager, Sy	Project Manager, System Architect, System Developer, System Tester

Contribution Guide

Code of Conduct

1. **Introduction**
   Our code of conduct serves as a guide to the actions of team members. This includes our values for ethics, diversity and inclusion as well as unifying our team. The code of conduct also sets the basic requirements and expectations within our workplace and is always expected to be upheld by all team members.

2. **Our team**
   Every team member of our team must up-hold and represent our code of conduct as well as following and being responsible for relevant laws. Understanding laws that apply to your role and the team is important, as failure to uphold the laws may result in disciplinary action. All team members can ask for help when something is unclear before taking action. It is expected that all team members are informed about their expectations and know where to go for answers if these are unclear.

3. **Fair treatment**
   We value fair treatment and inclusion of all. Honesty and respect should be present within all interactions between team members or third parties. We recognise your rights as a team member as nothing in this code of conduct is intended to remove or limit your rights under the legal law. In the nature of our business, we collect and store minimal personal information such as entered pictures and annotations. With this we ensure that no third parties will have access to the information. We are also committed to providing a safe and inviting work environment for all team members. We expect all team members to promptly report unsafe conditions to the project manager.

4. **Conflicts of interest**
   We always aim to act in the best interest of our team, meaning one's personal interests must never influence actions in relation to the team. A situation where a conflict of interest may arise could cause tension and mistrust within the team and therefore is to be avoided. A

potential conflict may not be easy to recognise straight away but they usually involve personal gain and may create and incentive to benefit yourself, friends or family. Should a conflict of interest arise we will help assess the situation and deem the appropriate course of action.

5. **Reporting concerns**
   If you believe that there had been a violation of the code or the law, concerns may be reported anonymously. It is preferable to have your name and contact information so that if necessary you may contacted directly and we will do all to protect your identity. Concerns are not to be ignored as a failure to report could result in harm to the team and its reputation. Retaliation is any action that may deter an individual from reporting a concern or participating in an investigation. We prohibit retaliation of any kind against someone who participate in internal investigations or the reporting of a code concern. We do not tolerate false accusations as they deter resources and can infer mistrust and this would be a violation of our code.

6. **Conclusion**
   In summary we always value all team members and expect all members to be guided by this code and to uphold it with pride.

(Johnson&Johnson, 2019; Alphabet Investor Relations, no date; The Coca-Cola Company, 2018)

Reporting bugs and Enhancing code

Reporting bugs is vital to improve and ensure smooth running of code. In reporting bugs the information must include:

Name of Error	
Severity	
Description	
Proof	
Expected results	
Actual results	
Steps to reproduce	
Additional information	
Contact information	

If a change is suggested then the person who suggested the change within the team is responsible for implementing the change or acquiring the means to do so, making sure to reference the coding style guide. (Tatham, 1999)

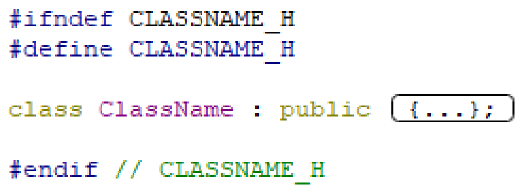
Continuous Integration/ Continuous Develop (CI/CD)
For our continuous integration and continuous development, we are using GitHub and Jenkins. This ensures that we are continually delivering an up to date version of our project. Our GitHub commits are done on regular bases at 3+ times a week and are continually referencing the coding style guide.

Coding style guide

C++ version
C++11 should be used within Qt 5.14.1

Header file conventions
Each .cpp file should have an associated .h file with the exception of the .cpp file that contains the main() function.

Header files should be self-contained and end in .h. They should include all other headers the file needs and have header guards as formatted:



```
#ifndef CLASSNAME_H
#define CLASSNAME_H

class ClassName : public {...};

#endif // CLASSNAME_H
```

The header file should contain the declaration for all the functions of the class and its public and private variables.

Scoping conventions
Code should be placed in the relevant namespace, which has a unique name.
Local variables should be initialised within its function's declaration.

Class conventions
Avoid calling any functions within the class constructor.
Structs are only to be used for passive objects which carry data, anything else used is a class.
Use a struct over a tuple.
Inheritance must be public when used.
Class data members must be private, unless they are constants.
When defining a class, the public section should be first, followed by protected, and finally private.

Function conventions
Use return values instead of output parameters where possibly in order to improve readability and performance. Input parameters should always be before output parameters.
Write short functions for them to be easy to read and modify.

Naming conventions
All names must be clear to anyone else reading the code.
File names should be lowercase and not include any special characters except underscores or dashes.
Variable names should be in lowercase with underscores between the words.
Function names should start with a capital letter for each word and should not have underscores between the words.

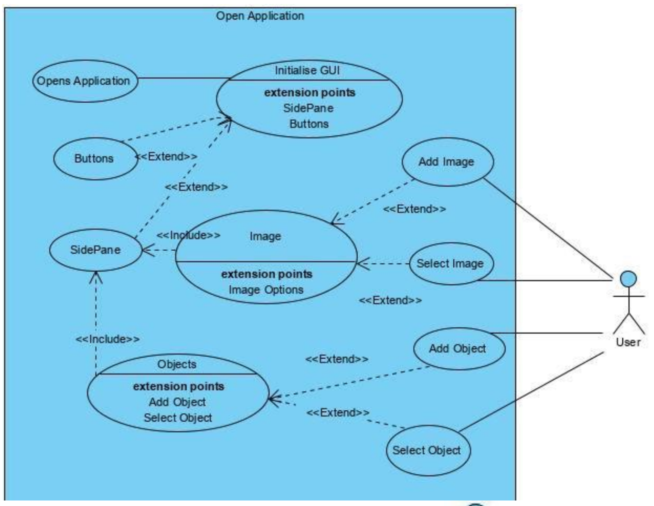
Comment style
Comments should be included regularly to make the code easier to read.
Use the // syntax.
Ensure each class and function has a comment briefly describing what it does.
Variables should have a comment briefly describing what they are.
Comments are not needed where the code states the obvious. They should describe why the code is doing what it does.

Formatting
Each line of code should be no longer than 80 characters, unless it is a comment. If a comment goes onto a new line it should be started on this line.
Non-ASCII characters should be avoided.
When indenting, use 4 spaces for each indentation and avoid using tab.
Functions - the return type and parameters should be included in the function declaration. Parameters should be on the same line if possible, if needed a function call is to be used.

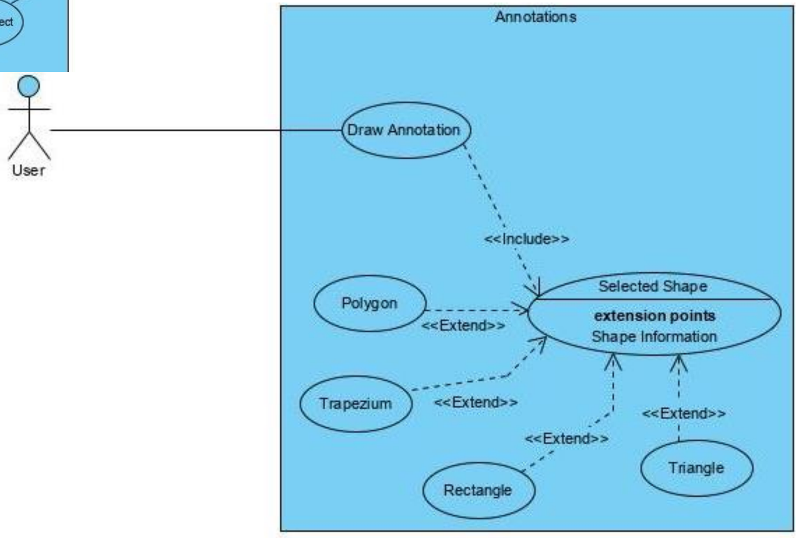
(Google C++ Style Guide, no date)

Implementation

Use Case

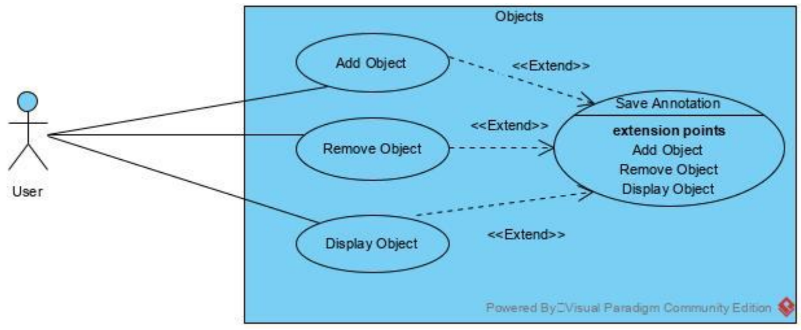


The Application is loaded and initialises the GUI. The Application displays the SidePane and Buttons. The SidePane displays image options and Object Options. The User can add or select and image for the GUI to display. The User can add and select Objects to be displayed on the image.



The User can annotate an image by selecting and adding shapes. The shape options are Polygon, Trapezium, Rectangle or Triangle. Shapes have the option to be copied and pasted, edited, and deleted.

The User can add, remove and display objects on the image. The objects contain saved annotations.

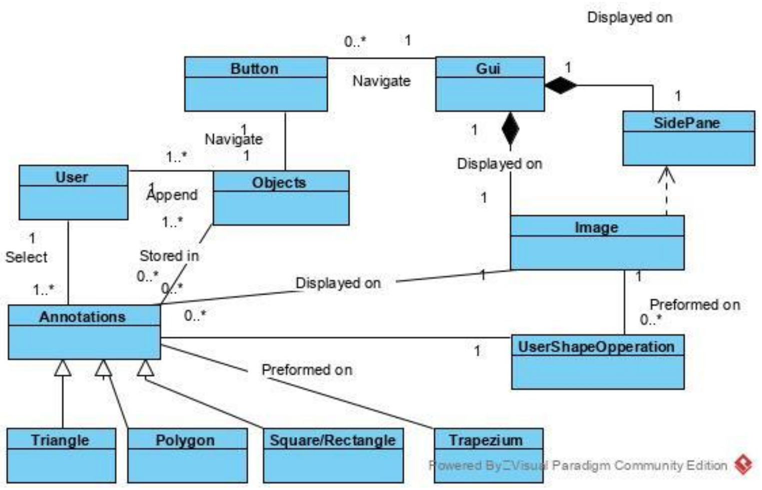


There is an option to sort the files in either pane by the name or date they were last modified, in both ascending or descending order. The files are automatically saved every minute.

All these use cases reflect the main requirements we need to follow, especially to do with the shape drawing. I believe they show what we need to achieve by the end of the project. Requirement 38 states the user must be able to delete a shape which the final use case demonstrates will be able to happen. This is just one example of the many that these use-cases illustrate.

Class Diagram

This version of the class diagram does not include any attributes or operators. This should make the relationships easier to follow.



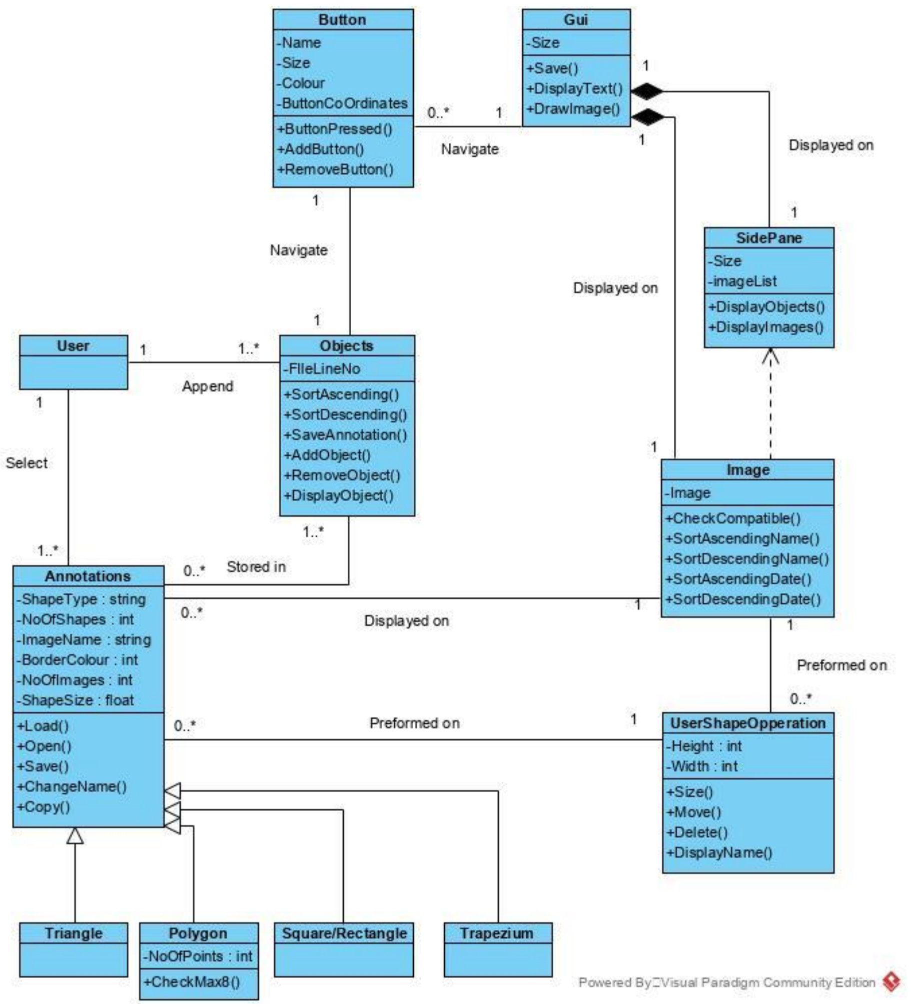
The version below is more in depth, including public and private methods as well as attributes, along with their type where necessary.

In making this class diagram we decided to use the use-case as a guideline. From there we could gain the classes and some of their relationships with other classes. This is very important as we can get an accurate version of what we are trying to create.

Here we have decided to use several classes as well as 4 extended classes – for the annotations. These are going to take all the same properties of the annotations class as well as some additional methods that will differ from the others. There is a class to represent the user and the user interaction however this will not be in the final solution itself, just simply an actor in the system.

In our class diagram we have tried to keep as many methods private as possible as this will improve the efficiency of the system. This is very important when considering building a program as performance is a large part of functionality.

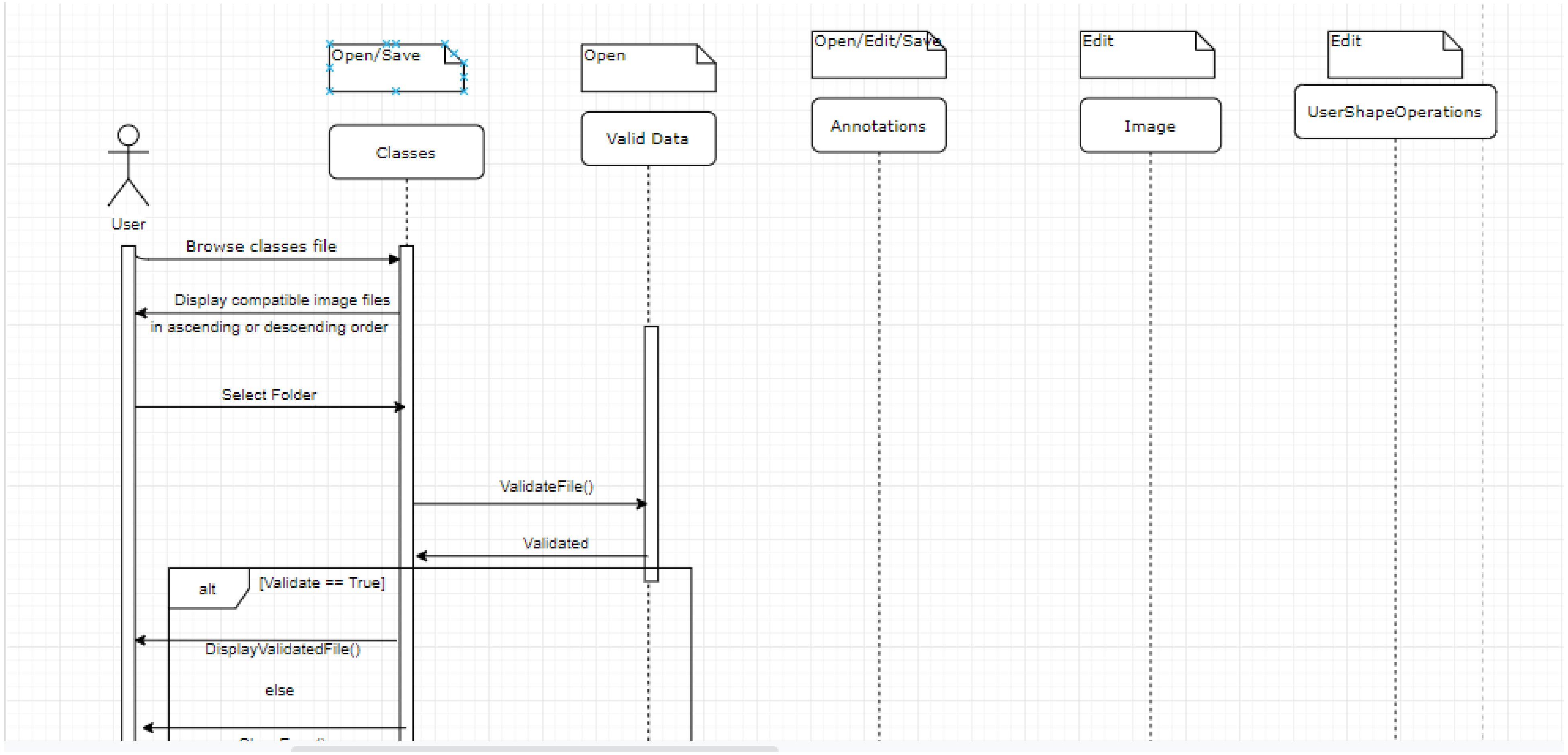
However, let it be noted that we are using Qt, which does contain many different libraries of which Visual Studio will not have. I say this because there may need to be some changes in how the classes are linked depending on how a Qt application may be built (especially in the UI). Given more time and additional contributing members, we could gain more of an understanding at this stage and be ready to program in Qt before we tackle the main problem.

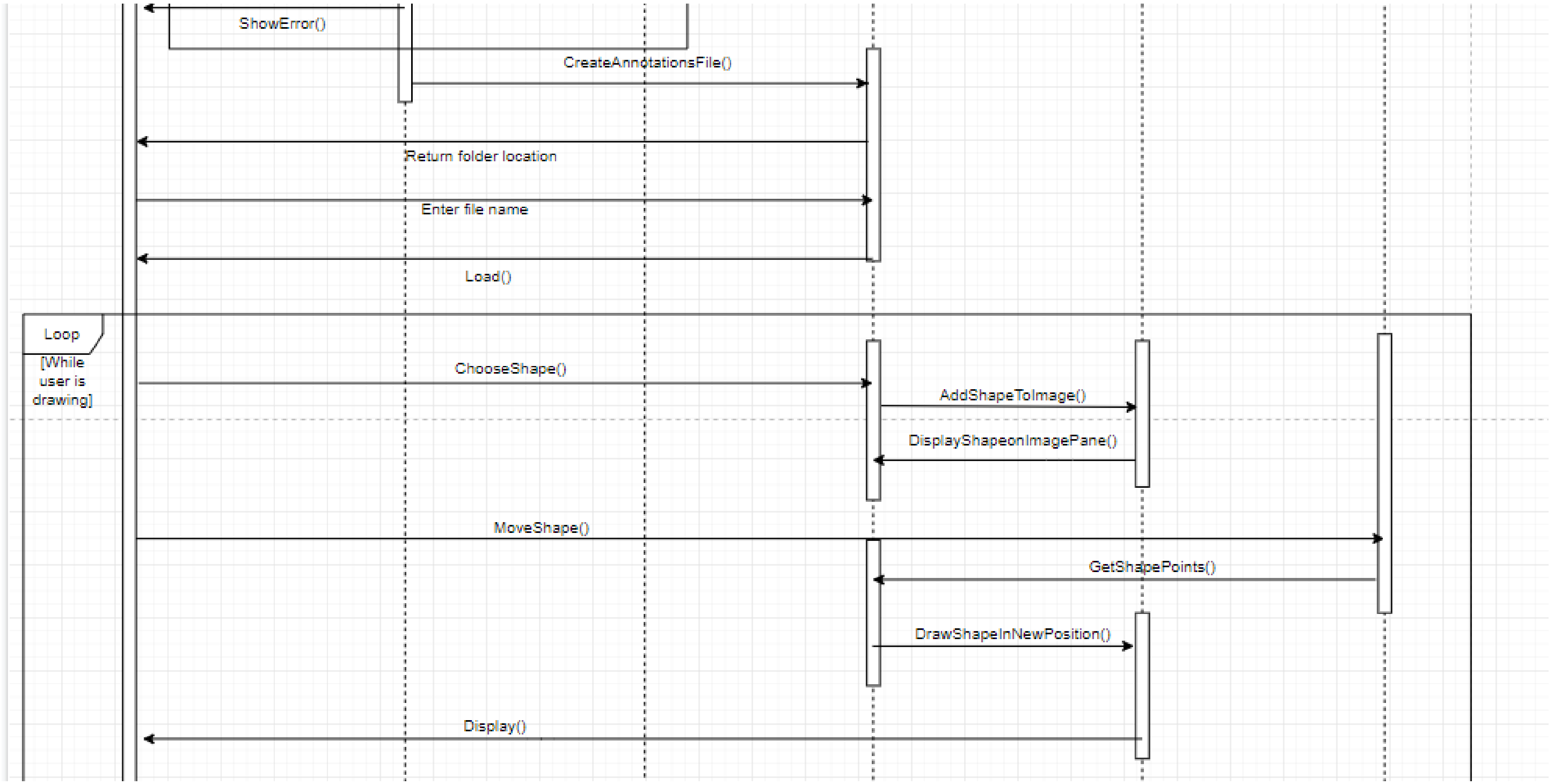


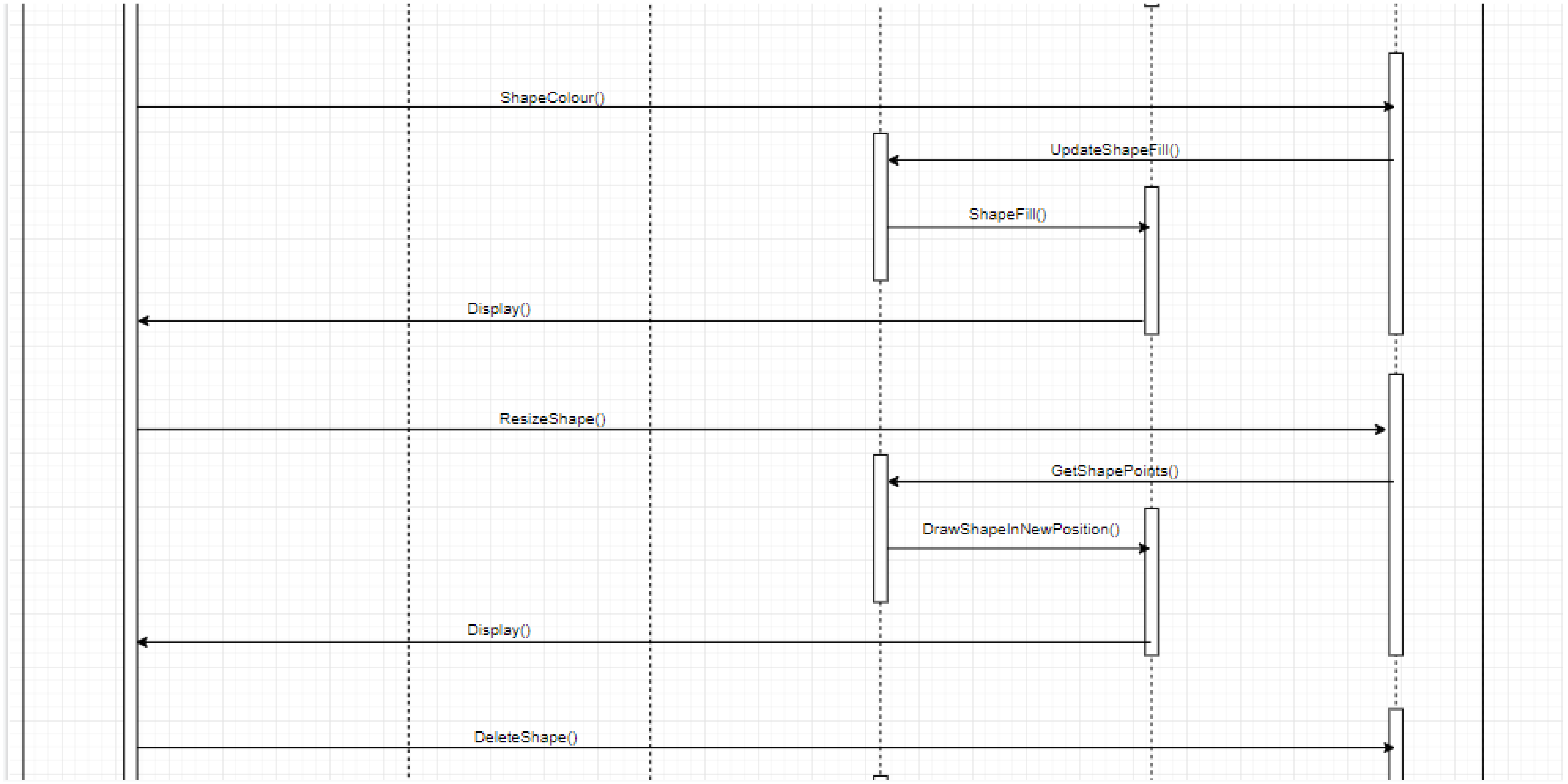
The relationships are:

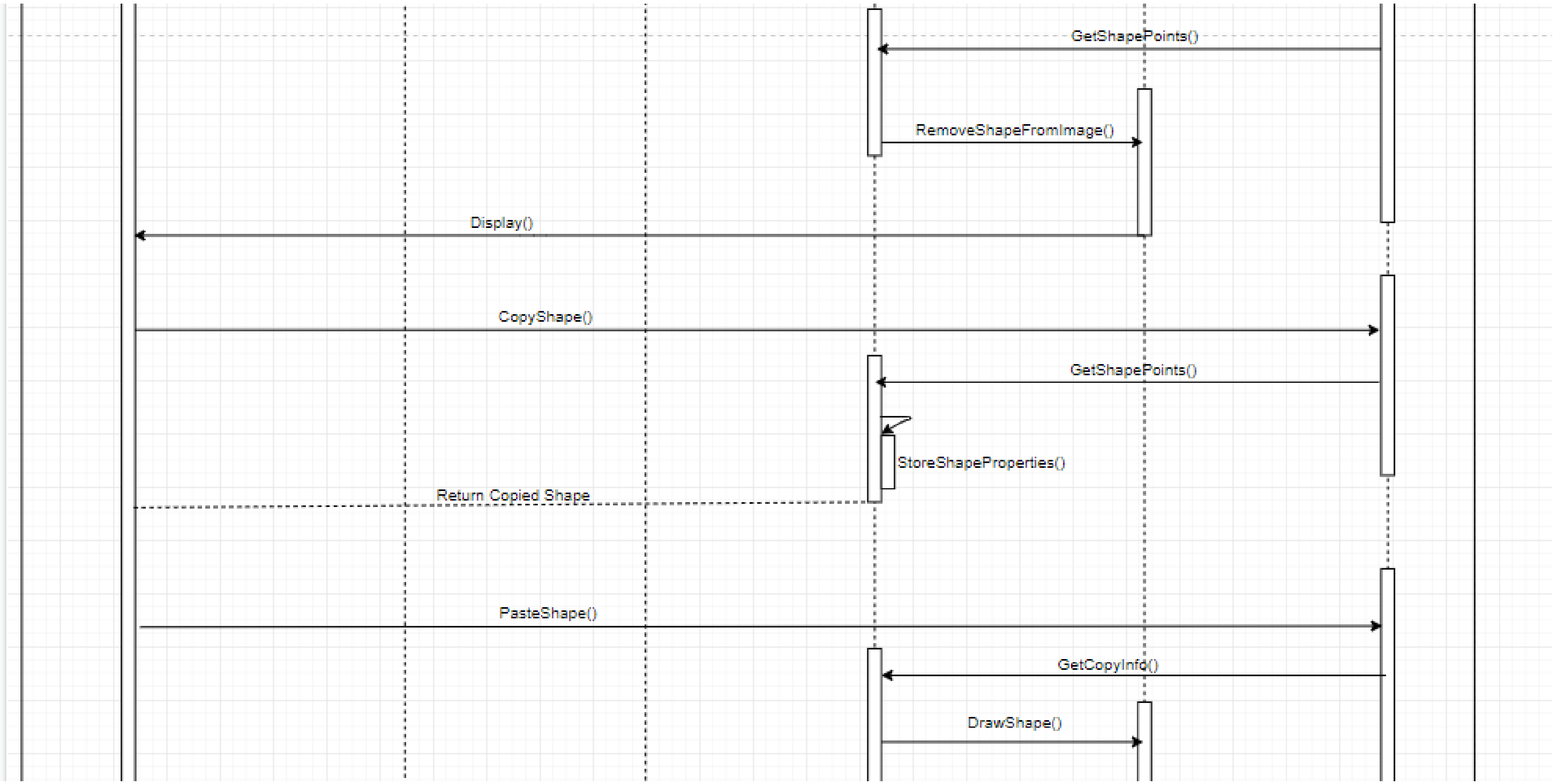
- The button class contains all the information to add or remove a button as well as registering if the button was pressed. There is one button per Object, however the GUI can contain many buttons.
- The GUI displays all the necessary information to the user such as the text and image. The GUI can only display one image and one side pane.
- The side pane is used to select the image from the image class to be displayed on the GUI.
- The Images are stored as a list so that they can be sorted ascending or descending by name or date. The image can display zero or more annotations. The image can also be affected by zero or more shape operations.
- The user can use shape operations on each annotation on one image.
- The annotations contain all the information for the shapes allowing them to be loaded, saved and copied as well as changing the name of the stored annotations file. The user can select many annotations to edit and the annotations can be affected by zero or more shapes. Many annotations can be stored in a single object.
- The object class store the file line number so that the objects can be sorted ascending and descending. The objects can also be saved, added, removed and displayed. One object can store zero or more annotations and the user can add or remove many objects.
- Triangle extends the annotations class.
- Polygon extends the annotations class.
- Square/Rectangle extends the annotations class.
- Trapezium extends the annotations class.

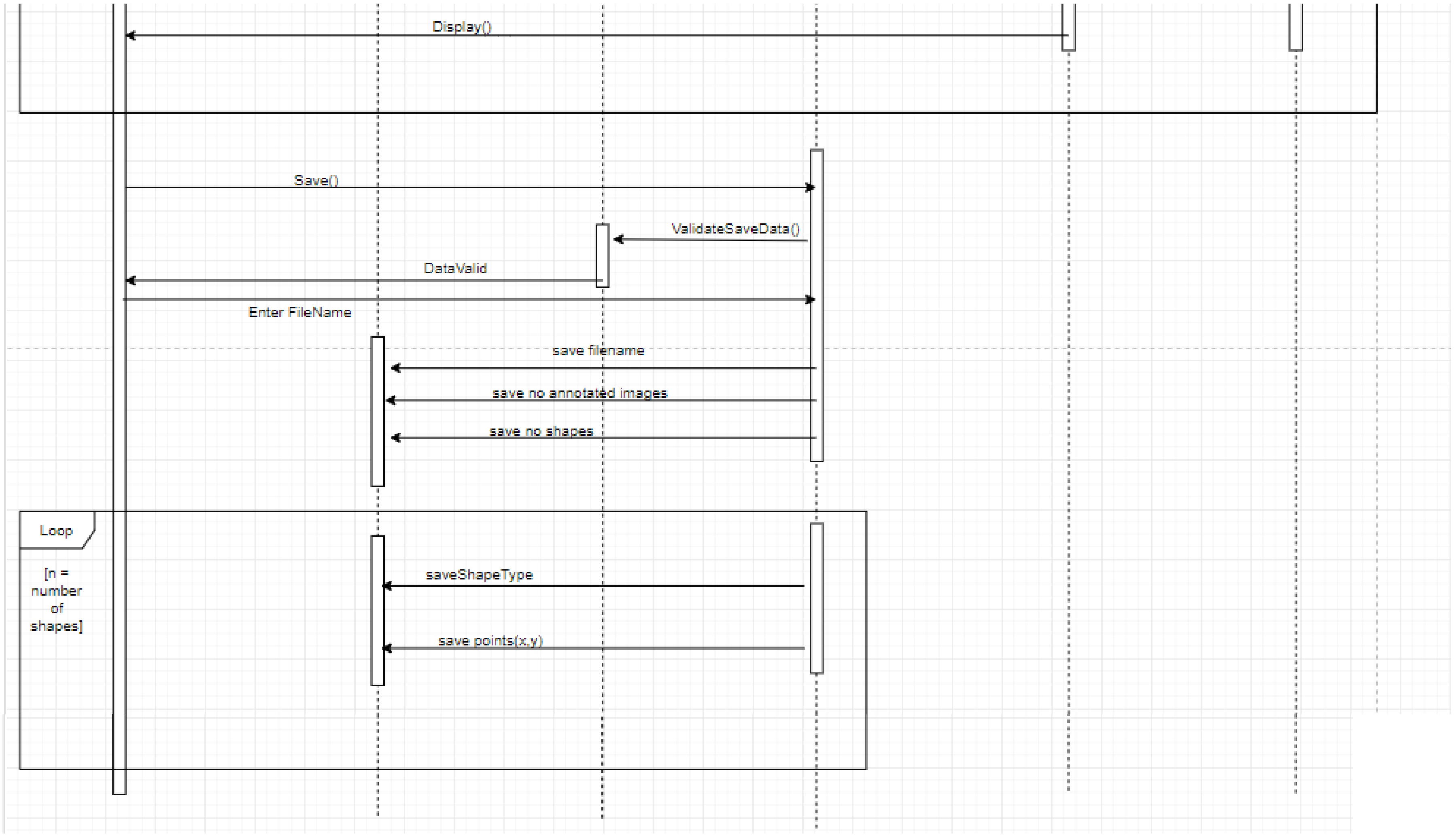
Sequence Diagram




ShowError()
CreateAnnotationsFile()
Return folder location
Enter file name
Load()
Loop
[While user is drawing]
ChooseShape()
AddShapeToImage()
DisplayShapeonImagePane()
MoveShape()
GetShapePoints()
DrawShapeInNewPosition()
Display()


ShapeColour()
UpdateShapeFill()
ShapeFill()
Display()
ResizeShape()
GetShapePoints()
DrawShapeInNewPosition()
Display()
DeleteShape()

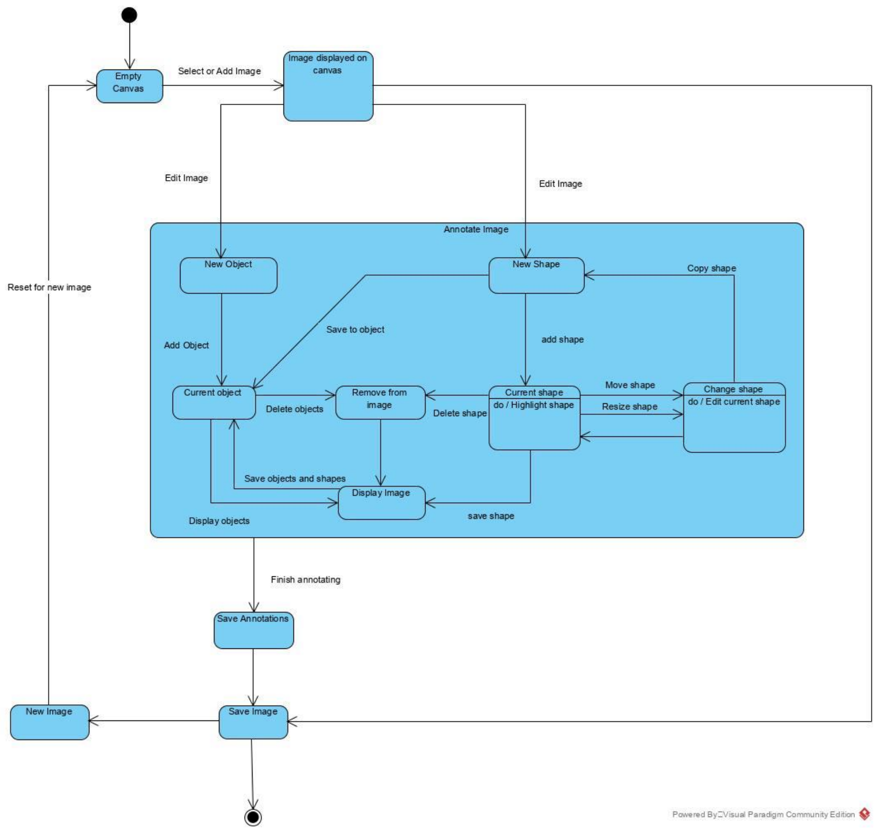

GetShapePoints()
RemoveShapeFromImage()
Display()
CopyShape()
GetShapePoints()
StoreShapeProperties()
Return Copied Shape
PasteShape()
GetCopyInfo()
DrawShape()


Display()
Save()
ValidateSaveData()
DataValid
Enter FileName
save filename
save no annotated images
save no shapes
Loop
[n = number of shapes]
saveShapeType
save points(x,y)

Explanation of Sequence Diagram
Opening a file:
The user browses the objects files in ascending or descending order. All compatible files are displayed to the user who then selects a folder.
The folder is then validated to make sure there are no errors when opening it. If there are no errors, the folder is validated and can be shown to the user.
An annotations file is then created, and the folder location is returned. The user can name this file. If the filename already exists then, an error is shown to the user who then enters another file name. The annotations file and image pane are loaded to the screen, ready for the user to annotate.
Annotating the image pane:
This section of the diagram is in a loop because the user can repeat this process an unlimited amount of times until they want to save and exit. Every sequence in the loop is an action which can be performed by the user.
The user starts off by selecting a shape, this shape is then added to the image and displayed on the image pane for the user to see.
The user can perform actions on the shape such as move, drag a vertex, change colour, resize, delete, copy and paste.
Saving an annotation:
When the user has finished annotating an image pane, they select the save option. They will have the option to change the filename or it will stay the same.
As the program is designed to save every 60 seconds anyway, this is the last time the annotation will be saved, and the user will be returned to the start of the program.
The for loop in the save sequence runs for how many shapes there are on the annotation as each shape has its own data to be saved such as shape type and the shape coordinates.
(Visual Paradigm, 2020)

State Diagram

Below is a state diagram to show how the programs state will change from an initial state to an end

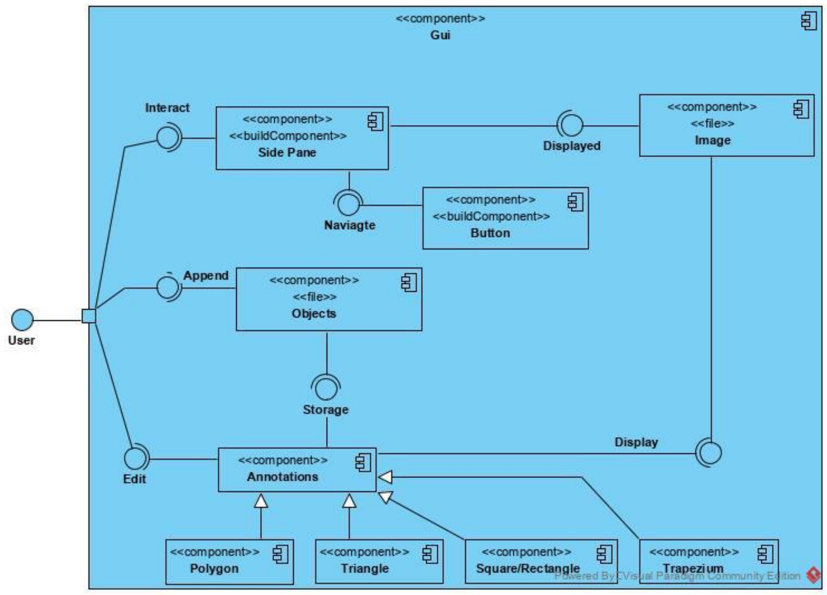


state. A state diagram is very important to see how the system changes state to deal with problems and/or data. This can give us an understanding of how the data will move and where we can expect problems as well as mitigate deadlocks.

When the program is opened it initially has an empty canvas. The user then selects or adds an image to display on it. They then can edit the image by adding a new object or a new shape. They are also able to save the image so that they may pick a new image. The user can choose to add as many shapes as they want as well as edit saved shapes. They can edit saved shapes by moving, resizing or deleting shapes. Once the shape is saved it can be displayed on the image. Once the user has finished annotating, they can save the image. If they choose to annotate a new image, the state is the empty canvas once again.

Component Diagram

Below is the component diagram of the system. The links and behaviours are also elaborated a little more below.



<<component>> Gui
Interact
<<component>> <<buildComponent>> Side Pane
Displayed
<<component>> <<file>> Image
Naviagte
<<component>> <<buildComponent>> Button
Append
<<component>> <<file>> Objects
User
Storage
Display
Edit
<<component>> Annotations
<<component>> Polygon
<<component>> Triangle
<<component>> Square/Rectangle
<<component>> Trapezium
Powered By Visual Paradigm Community Edition

This diagram contains a range of components that will be effective when it comes to the implementation of the project. We have 9 components here within the system on a software level. This will help us to understand how the software will interact with each other. On a basic level this is an explanation of the diagram.

The user can interact with the side pane or append the objects file. The user is also able to edit annotations. The side pane uses buttons to navigate the GUI so that it can display a supplied image. The annotations are stored in the objects file and are displayed on the image. The triangle, polygon, trapezium and square/rectangle extend the annotations as shapes that can be drawn.

Explanation of the Searching and Sorting Algorithm

Search Algorithm

For our search algorithm, we implemented a binary searching algorithm. We implemented this algorithm because it is a must that the search list is in ascending order so therefore, it also involves our sorting algorithm. In our program, the main task that the search does is to help the user find their desired image. It works by firstly looking at the middle element in the array, then if our desired value is greater than the middle value, we can instantly disregard the first half of the list. This process is then repeated with only the second half of the array and so on until the middle element is equal to our desired value, we then have the index in which the desired image is located.

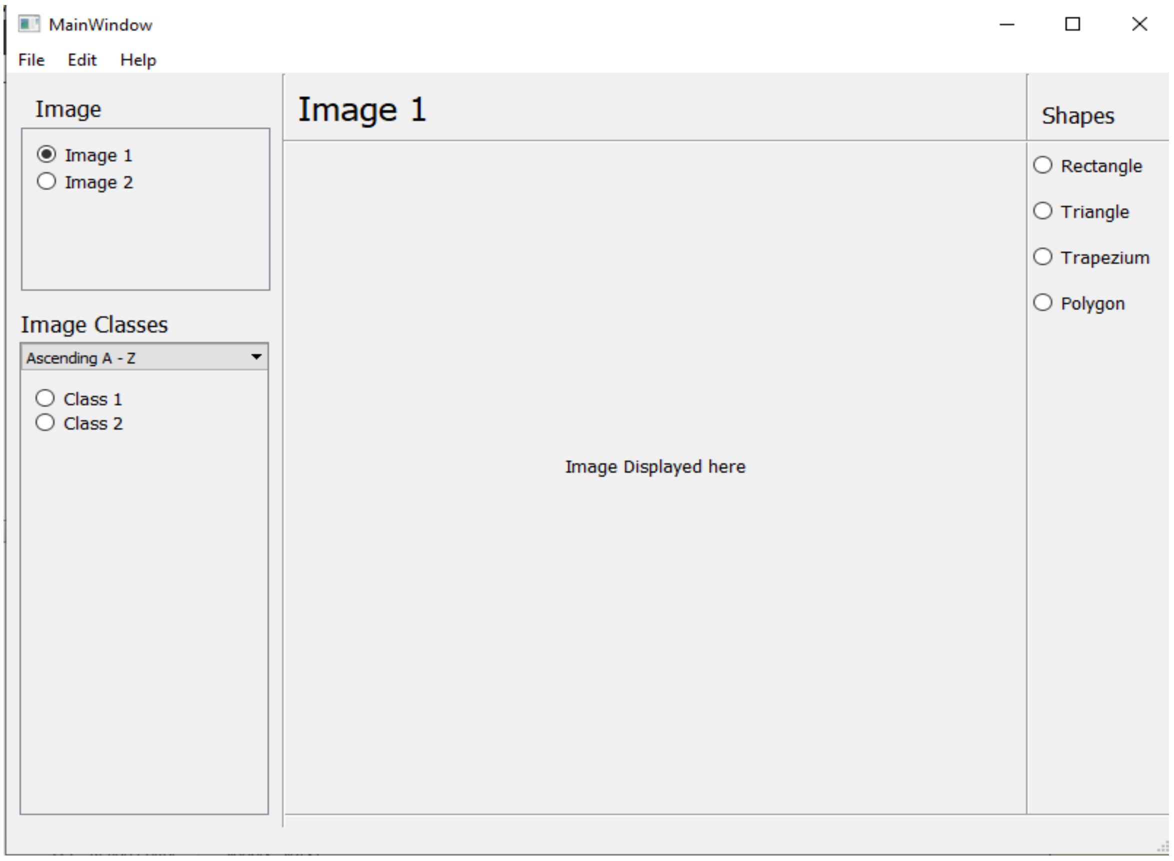
With an average time complexity of O(log n), we implemented this search because it is on average much faster than a linear search therefore reducing the time the user will have to wait if there is a large amount of images to search for. For example, an array of 1 million elements, on average it would take a linear search 500,000 comparisons whereas a binary search would only take 20. The main disadvantage of this algorithm is that it is harder to implement than a linear search and that it requires the array to already be sorted although this isn't a problem considering we have also implemented a sort algorithm regardless.

Sort Algorithm

In the case of our Algorithm, we implemented a bubble sort. With an average time, complexity of O(n^2), this algorithm works by firstly comparing the first two elements, the larger of the two elements is placed second. Whichever of the two is greater is then compared with the third element, they are compared and swapped if need be and then the third and fourth element are compared, this process repeats until the end of the array when after each pass, the corresponding element will be in its correct position. For e.g. At the end of the first pass, the greatest element will be at the end of the array and after the second pass, the second greatest element will be at the second last position of the array and so on. For our program, we used a sorting algorithm to sort the images by name or date and the classes by name. The user has the option to choose if they want the order to be ascending or descending. The main disadvantage of us using a bubble sort is that it's not the best algorithm to deal with extremely long arrays. However, considering the user will mainly be focused on one image at a time, they will rarely be in the position to have so many images open that it's too noticeable when sorting the images or classes.

GUI Mock-up

Using Qt, I have created a GUI mock-up to show what the UI may look like in a basic form.



You can see where some of the classes have come into play, most of which are physically visible, such as the Image class and the side pane, located on the left. However, there are some classes that remain behind the scenes such as the annotation class for now as there is no image loaded to annotate on. I think that having this tool will give us lots of headway in future development of this project because we know what we are trying to achieve, through a bare-boned copy. This knowledge can be used to avoid mistakes as well as a reference on what to aim to achieve when looking at layout.

Testing

Test Plan

Introduction

The Test plan is created to give the developers an indication and approach to resources and schedule of all testing activities of the project.

The plan identifies what areas to be tested, the features to be tested, the types of testing and who is responsible for testing.

Test Logistics

-Who will test it?

Mainly the software tester, but other members will also help

-When will the tests occur?

The testing will occur once the following have been completed:

- The project plans
- The software architecture
- The software has been created

Test Objective

The test objectives are to verify the functionality of the program and to highlight and bugs or defects which need attention.

Bounds of Use

When the application is left open for a long period of time, we don't expect much to change in terms of the performance of the program. One bound could be when the user adds many shapes, the time for the program to execute adding new shapes will deteriorate as more shapes are being dealt with. This will also occur when the user wants to save their annotation, as the more shapes and images to save, the longer the program will take to execute and save.

Another bound is the search algorithm, if we implemented a linear search, the average time complexity would be O(n) whereas if we implemented a binary search, the average time complexity would be O(log n) which is a lot quicker than a linear search. Therefore, the time for a search in our program will depend on which search algorithm we implement.

Likewise, for sorting algorithms, the algorithm we implement will affect how quickly the program will execute that function. The most likely sort would be a bubble sort which has an average time complexity of O(n^2) whereas a quicker algorithm such as a merge sort has an average time complexity of O(nlog(n)).

We also must consider how the program will cope when left open for a long time. The main issue will be threading which will trigger an autosave every 60 seconds, the longer the application is left open, the more data being saved which may not be of any use if nothing has changed in the last x amount of minutes. One way to overcome this maybe to allow the user to have the option to toggle autosave on and off, that way autosave will most likely only be on when the user is active. Another way is to compare data from each save and if the data isn't changing then a pop-up message could appear asking the user if they are still active.

Test Criteria

-Suspension Criteria

I reported that at least 30% of test cases have failed, current testing will be suspended and reported to the development team. This will allow time for the development team to address testing issues and hopefully mitigate further testing failures.

-Exit Criteria

Highlight the criteria that a successful completion of a test phase.

- Run rate: It is compulsory to be 100% unless a clear reason is given. This is calculated by the number of tests executed divided by the total number of test cases.
- Pass rate is 80%. It is also compulsory to achieve this pass rate. This is calculated by the number of tests passed divided by the number of tests executed.

Test Type

Throughout our development and testing stages, we have identified a good amount of testing types in order to highlight as many defects before our product is released. These include:

- Unit Testing- This type of testing is when an individual component of the program is tested so we know they are fit for use. This allows us to isolate each part of the program to ensure they are correct. It also allows us to find defects early in the development cycle, this includes bugs in the developer's code as well as parts of the specification or requirements which are missing. This type of unit testing is therefore a type of *WhiteBox* test technique. White box testing is when the internal structure and implementation is known to the developer/tester.
- Integration Testing- This typically happens after unit testing. It is when you take parts of the code which have been unit tested and integrate them to make sure parts of the program combine and execute as expected. This will allow us to expose defects in the interaction of our program.
- Functional Testing- This type of testing tests each function by giving the appropriate input parameter, this can be normal, boundary or erroneous data. Our system should catch erroneous inputs without breaking the program and retrieve another input.
- Acceptance Testing- This is the last stage of our testing and is performed by the end user. It is like *BlackBox* testing in that that the end user isn't going to see the internal structure of the program. The user will only test the inputs and outputs of the program.

White box and Black box testing

White box testing is a technique of testing in which the tester has knowledge of the internal structure of the program. It will be carried out by developers and software testers. It will test every function of the program as well as the logic to make sure it executes the way it was intended. This will allow us to isolate sections of code to test they work correctly before being integrated into the system.

Black box testing is a kind of trial and error testing in which the internal structure of the program is hidden from the tester/end user. The tester will therefore have control over the inputs and outputs of the program and by testing different types of inputs, I.e. erroneous and boundary, we can highlight how well the program deals with invalid data instead of crashing.

(Mahak_jain, no date**)**

Resource Planning

-System Resource

No.	**Resources**	**Descriptions**
1	Test tool	Use of DevOp's such as Travis or Jenkins to test software.
2	Computer	Need up to date version of visual studio in order to run program. 1.8 GHz or faster processor. 2 GB of RAM; 4 GB of RAM recommended https://docs.microsoft.com/en-us/visualstudio/productinfo/vs2017-system-requirements-vs
3	File handling	File explorer to store Image and Classifier folders
4	Network	System can run on NTU and home networks
5	Server	Install required software

-Human Resource

No.	**Member**	**Tasks**
1	Project Manager	• Manage the whole project • Requirements List • Gantt Chart and Scheduling • Risk Assessment/Mitigation

		• Use of monitoring tools
2	Software Architect	• Use case diagram • Sequence Diagram • State Diagram • Class Diagram • Component Diagram • GUI • Operations and attributes identified • Cohesion and coupling • Association • ATAM design • Design patterns and/or MVC • Deployment diagram • UML notation
3	Software Developer	• Contribution Guide • Style Guide • Reference Manual • Functionality • Data structures • User interface • Use of VCS tools • Exception handling • Access modifiers
4	Software Tester	• Test Plan • Results and screenshots • Report defects to developers • Documentation • Test report • Use of debugging tools • Use of DevOp tools

Features to be tested

Test No.	Requirement Description	Expected Outcome

1	Simple GUI where the user **MUST** be able to navigate to the chosen file			User can navigate the interface to select a file
2	System **SHOULD** list the compatible image files (e.g. *.jpg *.png) available in that folder.			Compatible image files will be listed with the required format.
3	The files **SHOULD** be sorted into:	Ascending Order	By the file name	The files will successfully be sorted into ascending order by the file name.
4			By the file date	The files will be sorted into ascending order by the file date
5		Descending Order	By the file name	The files will successfully be sorted into descending order by the file name.
6			By the file date	The files will be sorted into descending order by the file date
7	Users **MUST** be able to navigate and select class folders.			A browse button which will allow the user to navigate to their chosen file.
8	Files in this folder **MUST** have a *.names extension.			All files will have a *.names extension
9	All classes **SHOULD** be listed in a classes pane.			All classes will be listed in either ascending or descending order.
10	The file line number **MUST** be preserved.			The file line number will be preserved in the appropriate folder.
11	Users **MUST** be able to add classes.			The user will be prompted too add a class through a button.
12	There **SHOULD** be a way for the user to name the class right away.			The user will be made to name the class on creation.
13	Users **MUST** be able to remove classes.			The user will be prompted to remove a class using a button
14	The class file **MUST** be able to be appended to.			The user will be allowed to edit the classes file appending what shape they want to use.
15	Users **SHOULD** be able to use the following shapes:	Triangle		A class for each shape will be used storing the appropriate attributes and methods for each one.
16		Rectangle		
17		Trapezium		
18		Polygon		
19	The polygon shape **MUST** have up to 8 points, including the start.			The program will only allow between 3 and 8 points.

	Data: The number of points	Normal: 5	The shape will be drawn with 5 points
		Boundary: 8	The shape will be drawn with 8 points
		Erroneous: 10	The program will return an error prompting the user to enter a valid number of points.
20	The user **MUST** (only) be able to use the shapes to draw on top of the image.		The system will check the current position of the shape and if the shape is outside of the image boundaries, an error will occur and the shape will be placed back in it's original position.
21	The shapes **MUST** have no fill (only the outline)		The system won't allow the user to change the shape fill colour.
22	Shapes **MUST** be displayed on the image.		The system will only allow shapes to be drawn within specific boundaries of the image pane.
23	There **SHOULD** be an annotations file (with extension *.annotations).		Files will be loaded successfully with the correct extension.
24	User **SHOULD** be able to save the image with annotations.		File handling options will be available to the user giving them the option to save their annotation. This save will happen successfully.
25	The system **SHOULD** prompt the user to overwrite the file if another exists with the same name.		The system will take an input from the user for a filename. This filename will be compared to other filenames to check for duplicates. If there is a duplicate, an error will occur and the user will have to input another filename.
26	User **SHOULD** be able to change the name of an existing file.		The system will allow the user to change a filename. The same algorithm will be used to check for a duplicate filename.

27	Annotations **MUST** follow the hierarchical data format 5 (HDF5).		All annotations will follow HDF5.
28	The following data **MUST** be stored in each annotation file:	Number of annotated images	This data can be stored as a list or linked list???
29		Image file name	Will be appended to the list and algorithm applied to check for duplicates.
30		Number of shapes per image	Will be successfully stored as an integer.
31		Shape type	Will be successfully stored as an attribute.
32		Point_1(X,Y)	Will be successfully stored as 2 integers.
33		Point_2(X,Y)	
34		Point_n(X,Y)	
35	The selected image **SHOULD** be displayed on the image pane.		The coordinates/boundaries of the image pane will be stored, so the image will be placed within this boundary.
36	The user **SHOULD** be able to increase the size of a shape using the mouse.		The system will take in user input from the mouse which will be used to resize the shape. An algorithm will be used to make sure the shape is not resized outside of the image boundaries. If this occur, the shape will revert back to its original size.
37	The user **SHOULD** be able to move the vertex of a polygon with the mouse.		Input from the mouse will be taken when a vertex has been clicked on. This will then allow the user to drag the selected vertex and place it into a new position. The shape will be then filled in its new position. If the vertex has been dragged outside of the image boundaries, then an error will occur, and the shape will be reverted to its original position.
38	User **MUST** be able to delete a shape.		The system will allow the user to select a shape and

		delete it from the image pane.
39	User **SHOULD** be able to copy shapes.	The system will allow the user to copy a shape. The size and dimensions of the shape will be stored.
40	User **SHOULD** be able to paste shapes.	The copied information of the shape will be used to create a new shape when the user chooses the paste option. If no shape has been copied, then an error will occur notifying the user that there is no copy information.
41	The name of the class **SHOULD** be visible at the top of the shape.	The name of the class will be successfully displayed at the top of the shape.
42	The application **MUST** be automatically saved.	The system will overwrite the last save file automatically.
43	The autosave **MUST** happen every minute.	A clock system will be used and every 60 seconds, the previous save file will be overwritten will the new save file.
44	The autosaving process **MUST** be done using threads.	The system will use threads to save each part of the program.
45	Storing data **MUST** use a data structure from the first term.	The system will make efficient use of a data structure to store data.
46	All sorting algorithms **MUST** be from the first term.	The system will use sorting algorithms from the first term.
47	All searching algorithms **MUST** be from the first term.	The system will use searching algorithms efficiently from the first term.

Features not to be tested

These are features not to be tested because they are not included in the requirements lists

- Operational
    - Compatible with Windows and iOS
- Performance
    - Shapes on the image pane will load within 2 seconds

- Security
    - Admin access for higher level functionalities, I.e. file handling
- Cultural
    - Personal information is protected in compliance with the Data Protection Act.

Use Case Scenarios

Throughout the testing process, we will ensure that our tests follow the system in place in our use cases, this is to ensure thorough testing from the start of the application to the end. The use cases identify all of what the user can do so this will be a kind of acceptance testing which is performed by the end user. We have various use cases such as the navigation of the GUI, adding and removing objects and the drawing of shapes onto the canvas. All these use cases will be strongly followed to ensure the specification guidelines are met. By achieving these functionalities, we will know that the user is able to accomplish everything they are supposed to be able to do.

Risks and Issues

Please refer to the risk assessment in appendix 2.

Stress Failure Conditions

One stress condition could be the handling of multiple users at a time as there would be delays over a server. However, our program has been designed to run on one computer so only one user can use it at a time.

Another condition is the handling of vast amounts of data in the form of shapes, images and classifiers. Because we must save the annotations in a format, there should be a very structured approach to this issue. We could tackle this issue by having a dataset which controls the number of images and classifiers the user wants to add to their annotation.

As for drawing shapes onto the canvas, the performance of the program will be more demanding as it has more shapes to deal with. We will have to carry out tests on how stable to program is at dealing with many shapes. This will also depend on the capabilities of the device the user is running the program on.

Automation Tools

The most likely testing automation tools we'll implement are QTest and Jenkins. QTest is the most appropriate considering it is an in-built library on the QT application, this will allow me to carry out unit tests.

We will use Jenkins for the continuous integration of our system so that the Jenkins will build periodically or every time there's a Github commit so that it is easier for developers to introduce changes and highlight the error if a Jenkins build is unsuccessful.

Schedule and Estimation

Please refer to the Gantt chart in the project plan.

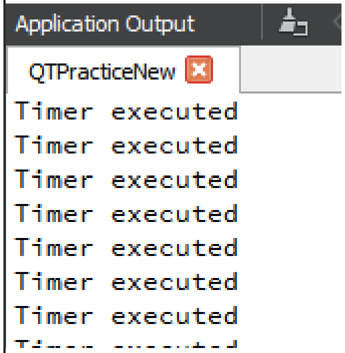
Test Deliverables

- Before the testing phase

    - Test Plan
- During Testing
    - Results and screenshots
    - Use of DevOp tools
    - Workarounds
- After Testing
    - Test Report

Unit Test Cases

This diagram shows an example of our unit test cases the full collection of unit testing cases can be found under the folder “DOCUMENTATION” within the test report.

ID	testCaseMyTimer	**Description:**	Creates a timer
Test Type: Unit	Quantity/Quality	**Success criteria:**	Pass
Number of attempts:	1	**Comments**	This timer is created for the saving function.
List of equipment/requirements	Qt with QTesting		
Setup instruction	Run the program (Timer starts as soon as program starts)		
Failure correction procedure	Report back to developer		
Engineer(s)/Technician(s)	Harry Evans-Software Tester		
Individual results:	Totals: 3 passed, 0 failed, 0 skipped, 0 blacklisted, 1ms ********* Finished testing of MyTimer *********  Application Output QTPracticeNew Timer executed Timer executed Timer executed Timer executed Timer executed Timer executed Timer executed		
Test Date:	29/2/20	**Result**	Creates a timer

Test Cycle

This shows a summary of how many tests have been passed, failed and total executed.

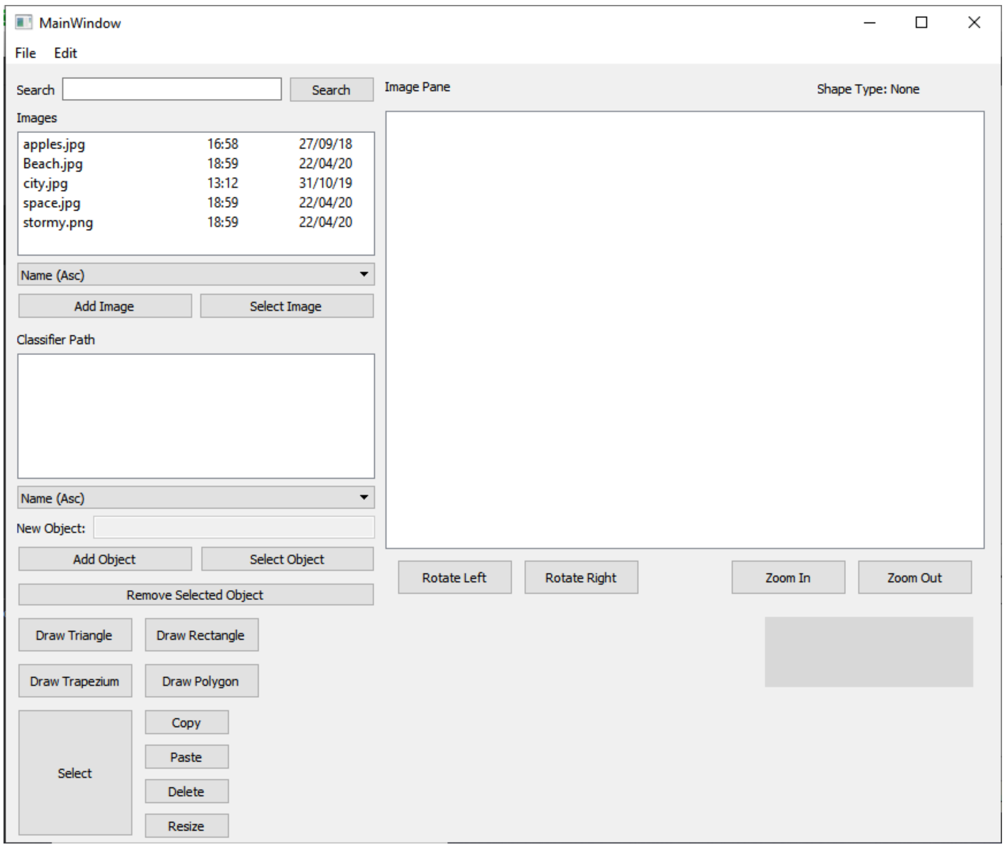
EXECUTED	PASSED	44
	FAILED	11
	(Total) TESTS EXECUTED (PASSED + FAILED)	55
PENDING		0
IN PROGRESS		0
BLOCKED		0
(Sub-total) TEST PLANNED (PEDNING + IN PROGRESS + BLOCKED + TEST EXECUTED		55

Test Defects

The test defects table can be found under the folder "DOCUMENTATION" within the test report. The table shows the bugs/problems we faced, when we tackled them as well as finally overcoming them, while noting the difficulty of the problem.

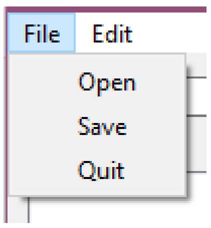
User Manual and instruction of the software

Our software is to be used to display images and annotate certain objects in the image by displaying an outline around them. This guide talks through each feature of the program, what it does and how to access it. Below is an image of the interface.



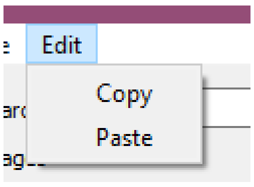
apples.jpg	16:58	27/09/18
Beach.jpg	18:59	22/04/20
city.jpg	13:12	31/10/19
space.jpg	18:59	22/04/20
stormy.png	18:59	22/04/20

File menu



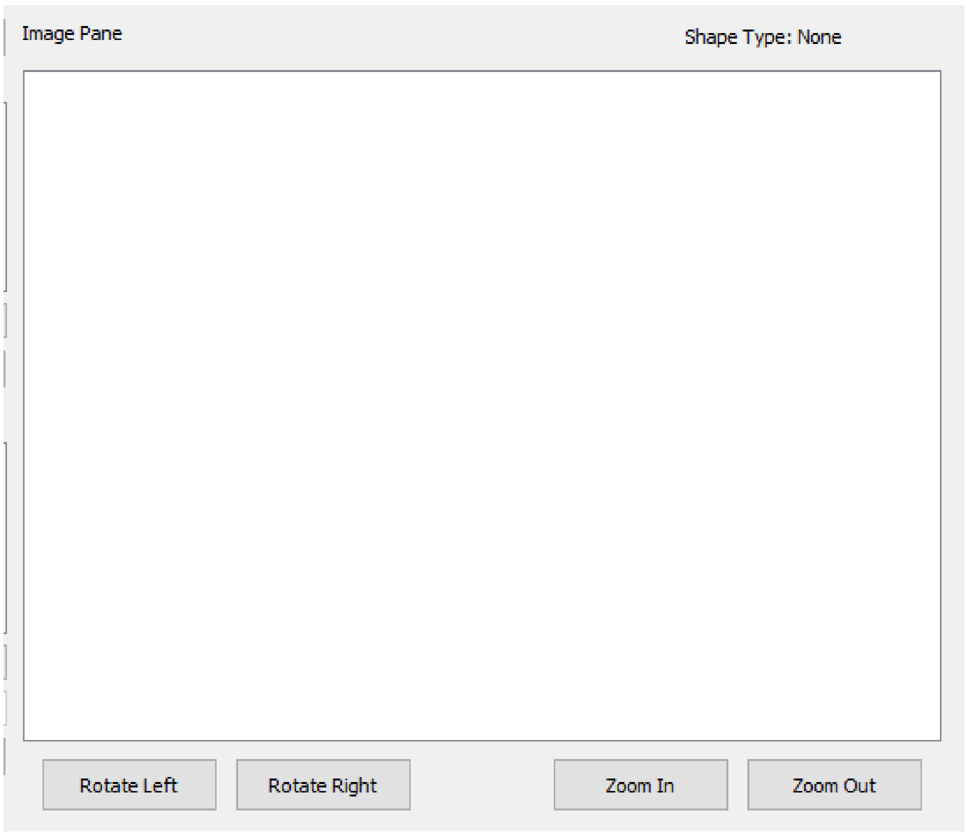
- Open – Navigate the files on your computer to find the image you would like to annotate.
- Save – Save the image with the annotations you have added.
- Quit – quit the program

Edit menu



- Copy – copy the shape currently selected to the clipboard.
- Paste – insert the image which is currently in your computer's clipboard.

Canvas



The image which has been selected is displayed on the canvas, so you can add shapes and annotations to it by clicking on the desired area.

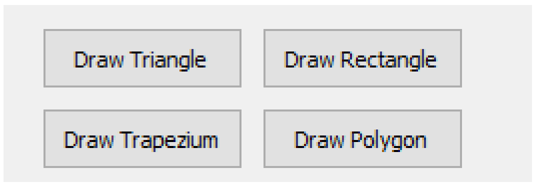
In order to rotate and zoom on the image, the corresponding buttons are below the canvas.

Shape Creation

Shapes are used to highlight parts of an image (an annotation). They are placed by clicking and dragging the mouse over the area you would like the shape to be displayed, then releasing the mouse. Make sure there is a shape type selected as well as an object selected otherwise nothing will happen. We have produced four different kinds of shapes:

- Triangle
- Rectangle

- Trapezium
- Polygon – any shape which can be up to eight points. Clicking back onto the first point you plotted will finish the drawing of the shape. When you plot your eighth point the shape is created by automatically joining the eighth point to the first.



This will produce a shape with a black outline, no fill and also put the class type into the centre of the shape.

Shape Operations

A shape is selected on the canvas by clicking on it after clicking on the select button. The type of shape which is selected is displayed in the top right corner.

When a shape is selected you can copy it, paste it, delete it or resize it.

Moving a shape can be done by selecting the desired shape and dragging it to the new location.

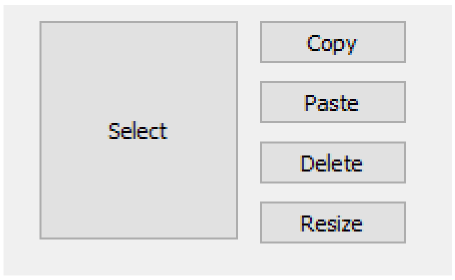


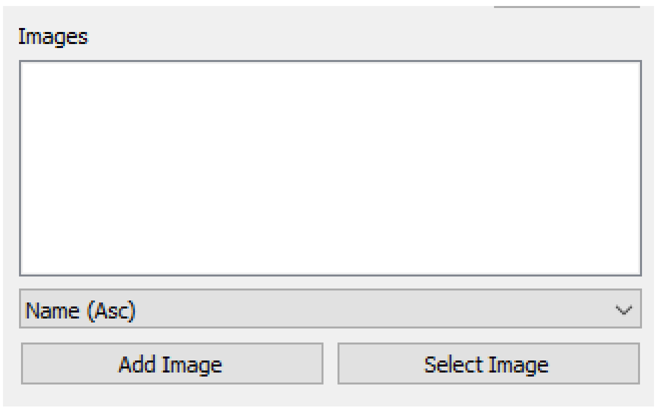
Image selection

Clicking the ‘Add Image’ button allows you to browse the files on your PC in order to find the desired image. This image is then added to the list.

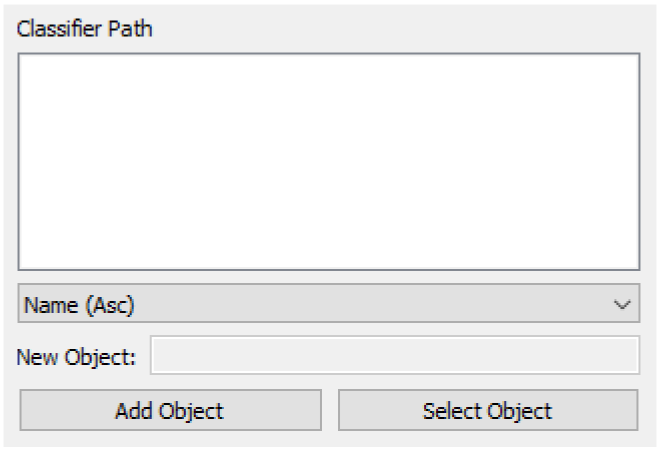
Click on the image in the list that you want to put on the canvas. Then press select image to confirm this and place it on the canvas. The text of the selected image will go red.

Images can also be organised into an ascending or descending order, by their name or the date that they were created.

You can search for a particular image by using the search bar at the top, which will highlight the corresponding text in green. When using this function please make sure that the list is in ascending order by name.



Classifier selection

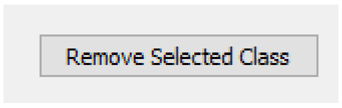


Classifiers can be added to show where a certain object on the image is. This is referred to by the shape outlining it and it has a corresponding label. Once a classifier is added to the list, you can select it to bound it to a shape by clicking ‘Select Object’.

A new object can be added by pressing ‘Add Object’ and then typing the desired object into the ‘New Object’ bar.

They can be sorted like the images, into ascending or descending order, by their name or the date that they were created.

There is also a button to remove a class that is currently selected.



Conclusions and future work

In conclusion we created a working project which implement most needed functionalities.

Our project has good performance and an easy to navigate GUI. The GUI allows the user to search for images, select an image, select a class (or object) and draw a variety of shape to the canvas. The image and classes list can be sort into ascending and descending order. The GUI is also on a timer which implements a thread based automatic saving function to increase computational efficiency. The project is quite portable if the user has QT and has the main files downloaded considering that Qt is cross platform.

In terms of software testing, in future work we will start completing the test defects table as soon as development has begun. This will allow us to keep up to date keep track of bugs in our code so we can inform the software developers immediately, allowing developers to turn their attention to the bugs more efficiently and maintain the set test suspension criteria and keep development running smoothly.

Overall Reflection

Overall, we completed as much of the project as we could in the allotted time frame, which is most of the requirements. Given more time and more effort from all team members we believe that all functionalities could have been implemented. The functionalities that we didn't have the time to complete were mainly related to the shapes in that, we are unable to delete an added shape, we are unable to resize an added shape and we did not get to adding the customisable polygon shape. All of which were attempted but unfinished in the given time and were removed from the final code, however all accessible in a previous version of the GitHub. We were also unable to copy and paste a shape but if the previously mentioned were to be completed we believe that copying and pasting a shape would not be difficult to implement.

In reflection, we know how much more resources and time we would need to complete the project. More time on this project could have been very useful as certain bugs and chunks of code took longer than anticipated to code. With all this being said, we know that what we have produced fits most of the working criteria, which is what we set out to do. If we did have more time from the start, we may have taken a different methodological approach to the problem. Perhaps use a more agile method or a JAD approach to the problem. We think that this could have been beneficial as it promotes coding throughout the project on several different final versions.

References list

Research for GUI Library
David Bolton, "5 Cross-Platform GUIs for C++", Dice, (18/11/16), available at:
https://insights.dice.com/2016/11/18/5-cross-platform-guis-for-c/
[date accessed: 04/02/20]

Qt Website
Qt, (2020), "Qt", available at: https://www.qt.io/
[date accessed 04/02/20]

UML Design Tool
Visual Paradigm, (no date), available at: https://www.visual-paradigm.com/
[date accessed 04/02/20]

Image Processing Library
Magick++, (no date), available at:
http://www.imagemagick.org/Magick++/?ImageMagick=euqj79qcd73925ive9hf8sme42
[date accessed 06/02/20]

CI/CD Tool
Gregory Boissinot, Michal Turek, Marco Steffan, (08/19), CPPCHECK. Available at:
https://plugins.jenkins.io/cppcheck
[date accessed 07/02/20]

Generating Documentation from Annotated Code Tool
Dimitri van Heesch, (2018), Doxygen, available at: http://www.doxygen.nl/
[date accessed 07/02/20]

Contribution guide
Johnson&Johnson, (2019) available at: https://www.jnj.com/_document?id=00000159-69fe-dba3-afdb-79ffcdd60000
[date accessed 28/02/2020]

Alphabet Investor Relations, (no date) available at: https://abc.xyz/investor/other/code-of-conduct/
[date accessed 28/02/2020]

The Coca-Cola Company, (2018) available at: https://www.coca-colacompany.com/policies-and-practices/code-of-business-conduct
[date accessed 28/02/2020]

Tatham, S, (1999) available at: https://www.chiark.greenend.org.uk/~sgtatham/bugs.html
[date accessed 29/02/2020]

Coding style guide
Google C++ Style Guide, (no date), available at: https://google.github.io/styleguide/cppguide.html
[date accessed 27/02/20]

Chapter 1: Writing a Unit Test
The Qt Company Ltd, (2020), available at: https://doc.qt.io/qt-5/qttestlib-tutorial1-example.html
[date accessed 7/03/20]

Differences between Black Box Testing vs White Box Testing
Mahak_jain, (no date), available at: https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/
[date accessed 20/04/20]

How to Create a Test Plan
Guru99, (2020), available at: https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html
[date accessed 25/02/20]

Test Summary Reports Tutorial
Guru99, (2020), available at: https://www.guru99.com/how-test-reports-predict-the-success-of-your-testing-project.html
[date accessed 10/03/20]

Jenkins GitHub Integration: Install Git Plugin
Guru99, (2020), available at: https://www.guru99.com/jenkins-github-integration.html
[date accessed 19/03/20]

What is Sequence Diagram?
Visual Paradigm, (2020), available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/
[date accessed 05/02/20]

Appendix

Appendix 1: Requirements list

Requirement Number	Label	Requirement Description			Requirement Implementation	Task(s)
1	T14	Simple GUI where the user **MUST** be able to navigate to the chosen file			Making use of file handling techniques such as read, write and save file.	Familiarise ourselves with file handling techniques from C++.
2	T26	System **SHOULD** list the compatible image files (e.g. *.jpg *.png) available in that folder.			Configuring the system to display the available file formats.	Obtain available file formats.
3	T25	The files **SHOULD** be sorted into:	Ascending Order	By the file name	Allow user to select whether files are to be sorted into ascending or descending order. Allows the user to select the option.	Research some efficient sorting algorithms, such as merge and insertion.
4	T25			By the file date		
5	T25		Descending Order	By the file name		
6	T25			By the file date		
7	T14	Users **MUST** be able to navigate and select class folders.			Insert a browse button to each class to improve navigation experience. Class files identified as plain text files corresponds to a class.	Insert browse button. Specify which text file corresponds to which class.
8	T27	Files in this folder **MUST** have a *.names extension.			Plain text files have an extension which corresponds to a class.	Use the extension "*.names".

9	T19	All classes **SHOULD** be listed in a classes pane.		Classes listed in ascending or descending order.	Research algorithms used in sorting.
10	M9	The file line number **MUST** be preserved.		Preserve the file line number in an appropriate folder.	We can assign the file line number to the class as an attribute.
11	T20, T29	Users **MUST** be able to add classes.		Prompt the user with an option to add a class through a button.	Research a way in which another menu can be displayed.
12	T20	There **SHOULD** be a way for the user to name the class right away.		Give the user the option to name the class as it's created.	Check constraints of class names, such as special characters.
13	T29	Users **MUST** be able to remove classes.		Prompt the user with an option to remove a class through a button.	Further research a way in which another menu can be displayed.
14	T29	The class file **MUST** be able to be appended to.		Allow the user access to edit the class file.	Look into ways to append to class file.
15	T31	Users **SHOULD** be able to use the following shapes:	Triangle	Include a class for each shape perhaps with the amount of sides being a constant attribute.	Familiarise ourselves with shape implementations.
16	T32		Rectangle		
17	T33		Trapezium		
18	T35		Polygon		
19	T35	The polygon shape **MUST** have up to 8 points, including the start.		Include upper and lower boundary for the user selections of the number of points.	Display an error if value entered outside of the boundary.

20	M9	The user **MUST** (only) be able to use the shapes to draw on top of the image.	Check whether the shape lies on top of the image. User can draw the shape on top of an image.	Insert if statement to return a Boolean if shape is or isn't on top of the image.
21	T18	The shapes **MUST** have no fill (only the outline)	This option will be hidden from the user.	Disable this option from the user.
22	M9	Shapes **MUST** be displayed on the image.	Keep shape drawing within boundaries of the image border.	Include a maximum and minimum values which must not be exceeded.
23	T27	There **SHOULD** be an annotations file (with extension *.annotations).	Load files successfully with this extension.	Familiarise ourselves with extensions and file handling techniques.
24	T21	User **SHOULD** be able to save the image with annotations.	Display file handling options to the user.	Familiarise ourselves with file handling techniques.
25	T21	The system **SHOULD** prompt the user to overwrite the file if another exists with the same name.	Check for duplicates files. Display pop-up to prompt the user.	Display error message to the user allowing them to enter another input.
26	T21	User **SHOULD** be able to change the name of an existing file.	Allow the user access to edit existing file names.	Implement a button that when pressed clears the name of the file and allows you to rename as if new.

27	M9	Annotations **MUST** follow the hierarchical data format 5 (HDF5).		Apply HDF5 to all annotations.	Construct research on what HDF5 is and what it requires.
28	T30	The following data **MUST** be stored in each annotation file:	Number of annotated images	Allows data to be easily accessed and/or changed	Store as an integer.
29	T30		Image file name		Store as a string.
30	T30		Number of shapes per image		Store as an integer.
31	T30		Shape type		Store as an attribute.
32	T30		Point_1(X,Y)		Store as 2 integers.
33	T30		Point_2(X,Y)		
34	T30		Point_n(X,Y)		
35	T15	The selected image **SHOULD** be displayed on the image pane.		The coordinates/boundaries of the image pane will be stored, allowing the image to be place within the boundary.	Allowing user selection. Store image pane properties. Displaying an image in the correct position.
36	T35	The user **SHOULD** be able to increase the size of a shape using the mouse.		User input from the mouse will be used to enlarge or decrease the size of an image. The image must not be rescaled outside of the boundary of the image pane.	A click and drag mechanism to move and rescale the image. Store maximum boundary variable of image pane.

37	T35	The user **SHOULD** be able to move the vertex of a polygon with the mouse.	User input from the mouse will be used to click a vertex and drag to another position. The vertex can't be dragged outside of the image pane.	Keep track of the coordinates of the vertex being dragged so new vertex can be placed. Store boundary variable of image pane which dragged vertex can't exceed.
38	T36	User **MUST** be able to delete a shape.	The user will have the option to remove a shape from the image pane.	Keep track of all vertex coordinates of the shape. Remove the shape using the vertex coordinates.
39	T36	User **SHOULD** be able to copy shapes.	The user will have the option to copy a shape from the image pane.	Size and dimensions of shape will be stored and replicated.
40	T36	User **SHOULD** be able to paste shapes.	The user will have the option to paste a shape from the image pane.	The copied information will be used to place a new shape of the same properties on the pane. Display the new shape at the current mouse position.
41	T19	The name of the class **SHOULD** be visible at the top of the shape.	Display name so it is visible to the user.	We should have the name displayed in an object as an attribute.
42	T23	The application **MUST** be automatically saved.	Overwrite the current save file to avoid having loads of unwanted files.	Familiarise ourselves with file handling techniques to save the program information to a file.
43	T23	The autosave **MUST** happen every minute.	The use of a clock system could be implemented.	Use a clock to keep track of time and every 60 seconds, save the program to a file.
44	T23	The autosaving process **MUST** be done using threads.	Use a thread to save each part of the program quickly and efficiently.	Research how to implement threads in C++.

45	M9	Storing data **MUST** use a data structure.	Data structure will make the program more efficient, easier to store and locate the data that we need.	Research which data structures would be the most efficient for the specification we have been given.
46	M9	**MUST** use a sorting algorithm.	Use a function to implement an efficient sorting algorithm for large sets of data, such as a quick sort.	Research which sorting algorithms are used in popular applications and why.
47	M9	**MUST** use searching algorithm.	Use a function to implement an efficient searching algorithm for large sets of data to find the specified piece of information in an efficient amount of time.	Research which searching algorithms are used in popular applications and why.

Appendix 2 : Risk Assessment

Risk	Probability (1- Very Unlikely, 5- Very Likely).	Impact (1- Low, 5- High).	Impact on project.	Prevention Method/ Ways to mitigate.
Bugs in code	5	3	Finding bugs in your code can be very time consuming to deal with and some may be very threatening to how the software may run.	The best way to potentially prevent having bugs in our code is to have a longer planning period, such that when we come to implement the code, there is a clear layout of what needs to be done. Otherwise, we may miss a feature and could potentially break another section of code trying to introduce new code.
Loss of source code	2	5	This could have a major impact due to the project needing to be reinvented from scratch or the last saved code.	We will make sure that we have frequent back-ups of the source code as well as saving it on a cloud-based server. We could potentially upload the current progress onto GitHub to get feedback and share any changes with the group right away all the while keeping the save safe.
Technical issues	3	3	Technical issues could potentially lead to loss of code if code is not periodically saved. It could also increase project time and costs as we may need to wait for the issues to be resolved.	To prevent this, we will regularly update our software and ensure that we keep back-ups before making large changes. Technical issues can be hard to predict and therefore this is the main way we can go about trying to reduce the risk.
Running out of time due to poor time management	1	5	The impact of poor time management is large as it may lead to not submitting our project on time.	Efficient allocation of group member responsibilities and deadlines for each member to complete their objective will decrease the chances of running out of time. Frequent checks on members progress can also help to decide whether a member needs some help with their

				part/task. This could largely help with the time efficiency of the project.
Requirements not being well established and poorly documented	2	5	This could have a huge impact because if the requirements are poor then this means that when it comes to design and implementation some may be missed. This may not sound like much but if we give the client an incomplete system, they certainly won't want any future projects with us.	To prevent this, we would need to take time and care when constructing a requirements list. It is one of the most important documents which we will go through thoroughly and make sure we have covered every point. As well as this we would make sure everything is laid out in a way that is easy to read and easy to construct a test plan.
Issues with integrating the software across platforms	1	5	Although this isn't a direct requirement, it is often common for software to be available on multiple platforms. The impact of a new requirement as large as that is huge as it will require us to build at least a new build of the system or change language to something like Java. The lack of portability across platforms would reduce the availability of the application greatly.	Preventing this is very difficult because it is something that, if the client really wants, should be included in the initial specification because of the amount of planning this does accrue. All we can do to potentially prevent this is to make the code as efficient and easy to read as possible as if there is a change of requirements later then we at least know the algorithms we need to use. We could use existing software to make the code easier to integrate efficiently and more usable across platforms.

Usability of the application	3	4	Having an unclear GUI and a complicated navigation system will confuse the user and they may not be aware of some features of the program. They would also be less likely to use the program again making it less popular.	In software testing, we could carry out questionnaires on how easy to use our system is. We can gather feedback on ways to make the GUI clearer and more understandable.
Small project team size	5	3	The project may take longer to complete as our team is of a small size and system requirements could be overlooked.	We will plan efficiently to reduce the risk of overlooking factors as well as to spread the work evenly.
Integration of a new requirement	4	4	Introducing a new requirement after the planning section would lead to a high impact as we would need to readjust areas of our code and integrate the necessary code to make the project run smoothly	We will make sure we have documented and planned all our requirements in detail to ensure we have covered all possibilities. Any new requirements requested during the project will be added to our requirements and management will allocate an amount of time to implement this request.
Secure handling of user information	1	5	Although there is no specific requirement saying to use personal data, we believe that the use of personal data should always be a high priority to keep safe. It is vital that if we	We will ensure all data is stored privately and is classified from other users. Encryption could also be used to add security to data which could be breached. We will abide by the Data Protection Act

			do need to use personal data, it is kept safe otherwise this would cause catastrophic damage to us as a group as well as the company we distribute to.	1998. Data will also be deleted upon user request or when they finish the program.
User chooses inappropriate image.	3	5	If the user uploads an inappropriate image, and this is shared on other platforms such as social media. This will give our program a very poor and misinformed reputation.	We could either instruct the user to only upload an appropriate image, so they are made aware that it is forbidden. Or we could have a range of images already on the program and not have an image upload option.
Poor member contribution	1	4	Depending on the amount of contribution, this member gives and how important of a task it is being completed, it could have a major impact on meeting deadlines and giving time to other parts of the project when one group member is filling in for someone else’s work.	If poor member contribution is having a detrimental effect on completing a task, then this can be reported to the module leader, so they are informed. We will also note who has had the most impact on each task, so everyone gets an equal workload.

Sickness/illness	2	3	Depending on the severity of the illness, it shouldn't have too large an effect on the outcome of the project.	Precise planning a time management will allow us to mitigate this issue, giving us an allowance for illness.
Disagreement among group members	1	3	A disagreement could cause arguments and disruption between group members wasting time we must complete that task.	To minimise disagreements, we would make sure every task is delegated fairly. If there is a disagreement, a compromise will be attempted but ultimately it will be up to whoever is leading the current section of the project. E.g. Software Developer
Poor work balance between group members	2	4	It will be costly on time as one member will do more work in the same space of time as another member isn't doing any work. The work should be spread equally all the time, not just sections of the project as each section will take longer to complete.	Good project management and allocation will minimise this risk as allowing members to have an equal balance on each section of the project will give the best chance of completing the objective.
Group member motivation	2	3	Poor motivation will cause disruption and the objective may not be completed to its potential.	We can have discussions on what tasks group members are more confident on and will enjoy doing more allocating them equally to do this task.

Reference manual
See the “html” folder.