Software Design & Implementation Project

Group 42

Lab Tutor David Adama

Barry O’Connor (Project Manager) – N0813926

Soham Jaiswal (Software Architect) – N0820198

Dovydas Skackovas (Software Developer) – N0800445

Ramunas Povilaitis (Software Tester) – N0805114

Plagiarism Declaration

This report and the software it documents is the result of my own work, other contributing group members are acknowledged. Any contributions to the work by third parties, other than tutors, are stated clearly below this declaration. Should this statement prove to be untrue I recognise the right and duty of the Board of Examiners to take appropriate action in line with the university's regulations on assessment.

Name: Barry O'Connor ID No: N0813926

Name: Soham Jaiswal ID No: N0820198

Name: Dovydas Skackovas ID No: N0800445

Name: Ramunas Povilaitis ID No: N0805114

Abstract

The SDIAnnotations project was set as coursework for the Software Design and Implementation module at Nottingham Trent University. The aim of the coursework was to design and develop annotations software for the purposes of training an Artificial Intelligence to recognise objects in photographs by highlighting the objects and storing the information. For example, this process can be used to help driverless cars recognise a pedestrian, traffic lights and stop signs. The resulting annotations files can then be read, alongside the images to perform the process of Machine Learning.

The team proposes to create an annotations application which will allow a user to import images, classes (lists of objects) and use their mouse to be able to draw shapes on top of images. These shapes will be separate from the image and will be adjustable when drawn to allow for better highlighting of areas of interest within the images.

The research undertaken to achieve this has involved several tools used for drawing and manipulating shapes as well as several software development platforms and practices. This has resulted in a cross-platform tool which can be compiled and used on several operating systems.

The resulting software provides a quick and user-friendly way to annotate multiple images, allowing for large numbers of photographs to be processed in a session.

Revision History

Version	Issue Date	Stage	Changes	Author
1	22/04/2020	Created	Structure laid out	Barry
2	22/04/2020	Draft	Project Manager information added	Barry
3	23/04/2020	Draft	Abstract completed	Barry
4	23/4/2020	Draft	Background Research Completed	Soham
5	24/4/2020	Draft	Test Scenarios completed	Soham
6	24/4/2020	Draft	Added ideas to Conclusions	Soham
7	24/4/2020	Draft	Implementation (D2) completed	Soham
8	25/04/2020	Draft	Gantt Chart breakdown added	Barry
9	25/04/2020	Draft	Software standards	Barry
10	26/04/2020	Draft	Updated testing (numbering was wrong). Added in some test videos for missing tests	Barry
11	26/04/2020	Draft	Introduction	Barry
12	26/04/2020	Draft	Results section	Barry
13	26/04/2020	Final	Final check, tweaks and reorder of contents tables	Barry

List of Contents

Contents

List of Tables

List of Images

Introduction

Artificial Intelligence (AI) has become a subject which attracts huge amounts of attention. The prevalence of the technology has led to applications ranging from Medical usage in Endoscopy (LEENHARDT, R., et al., 2020) and early detection of skin melanoma (D. Gavrilov, et al., 2018). In terms of driverless cars which are controlled by AI, both Google (Waymo, 2019) and Microsoft (Microsoft, 2019) are researching and investing in the technology alongside many other household names. This research offers large sums of money, with a UK startup company managing to raise $20 million, becoming one of the UK's best funded ventures in this field (Telegraph, 2019). This attention and scope gives the subject a great deal of heft and so many companies and projects are focusing on that field.

In order to train an AI, Machine Learning is employed. This process has many methods and algorithms to achieve this, among them is Supervised Learning and Unsupervised Learning (Nichols, J.A., Herbert Chan, H.,W. and Baker, M.A.B., 2019). Unsupervised learning is where the AI's programming allows it to learn by itself, a good example of this might be a Roomba which learns by recognising when it bumps into furniture, eventually learning the layout of the room.

Supervised Learning in comparison, involves something similar to teaching a child how to speak. If you show the child an apple and repeat the word "Apple" while holding it, the child eventually associated the sound with the object. This is similar to the way that the AI behind driverless cars are taught. The operators highlight many images containing, for example a stop sign, identifying the area of the image which represents a stop sign. Eventually the AI will come to associate that shape with a stop sign and take the appropriate action.

This process is what the SDIAnnotations project was created for. By creating software which allows a user to highlight areas of an image and to associate those areas, or annotations, with real life objects like tree, dog, person, car the supplication creates a way to teach an AI. This project aims to create a robust, cross-platform, user friendly way of doing this.

Background Research

As part of the process of building the application, research had to be completed into the various tools and C++ libraries needed to complete the various tasks associated with the creation of an annotation application. The following table (Table 1) summarises these tools and libraries and the factors leading to their consideration for use in the project.

Table 1 - a table describing C++ libraries and tools used

Library/Tools	Reasons for choosing this library/tool
QT for GUI and Drawing Shapes (Company, T.Q, 2019)	● Multi-platform ● Huge library aimed towards GUI development ● The Qt translation system, using linguist is easy to use and makes supporting multiple languages easy ● The GUI layout system where the widgets resize themselves according to a layout. ● QString does everything in Unicode and handles the conversions from different codecs very easily. ● Good customer support. ● Access to source code.
Doxygen for code Commenting (Doxygen, 2019)	● Easy to use and integrate ● Can be written within code, for easy maintenance ● Great support for C++
Draw.io for UML Diagrams (draw.io, 2019)	● Easy to use ● Free ● 3. Multipurpose use
Jenkins (continuous integration) (Jenkins, 2019)	● Free ● Open Source ● Fully Automated & Ease of use
OpenCV (researched but opted to use Qt native drawing tools instead) (OpenCV, 2019)	● Free ● Open Source ● Suited for Computer Vision Projects ● Easy Integration with Visual Studios ● Multiple Language is supported

	● Multiple OS types Supported ● Requires additional files and libraries
HDF5 (researched but opted to use Qt native JSON tools instead) (HDF5group, 2019)	According to the product information (hdf5.org ● Free ● Open Source ● Requires additional files and libraries ● Hierarchical data format ● Portable and extensible ● Cross platform ● Suitable for “Dig Data”

QT is a cross-platform application development framework for desktop, embedded and mobile. It is a framework written in C++. A pre-processor, the MOC (Meta-Object Compiler), is used to extend the C++ language with features like signals and slots.

Doxygen is a documentation generator, a tool for writing software reference documentation. ... Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code.

Draw.io is a free diagramming application that allows users to create and share diagrams within a web browser. ... Users are able to create and edit a variety of diagrams including flowcharts, org charts, process diagrams, ER diagrams, UML, network diagrams etc.

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is a library used for Image Processing. It is mainly used to do all the operations related to Images.

Design

Requirements

The following table (Table 2) lists the requirements identified for the SDIAnnotations project, based upon the coursework specification provided. Each requirement is based upon a need identified by the client (the module staff in this case) within the specification and any additional requirements identified by the team.

The table contains a number, a brief description of the requirement, a column discussing the implications of the requirement in terms of the project and finally a column for the related tasks which have been linked to individual tasks in the Gantt chart later in this section (figure XXX).

The requirements in this document have been identified using the MoSCoW method. This method prioritises requirements as follows:

M - The resultant software MUST meet this need. These are non-negotiable requirements which must be met.

S - The resultant software SHOULD meet this need. These requirements should be implemented where possible but the project will not depend on completing these

C - The resultant software COULD meet this need. These are requirements which could be included so long as they don't affect the project negatively in terms of time or workload.

W - The resultant software WOULD meet this need. These requirements may be held back for future development and tend to encompass features which are not core to the system and can act as a goal to work towards over several development cycles.

The table also includes requirements placed upon each member by the coursework, which includes individual portions of work and submissions.

Table 2 - Requirements gathered from the coursework specification documents

No	Requirement Description	Requirement Implication	Tasks
Functional Requirements			
1	MUST allow the user to import image files.	The GUI must provide a browse button which allows for the selection of images from an operating system dialog box.	**T10:** display an Open Dialog Box with the filter set to compatible image types. (Open File)
2	MUST populate an Image array with the selected images.	When a user has selected the images an array must be populated with all compatible images selected.	**T10:** populate the Image Array with the selected images. (Read from file)

3	COULD display a message if image(s) aren't compatible.	If no compatible images are found a message could be displayed to the user.	**T10:** Implement functionality to display a warning message box.
4	MUST populate the Image list box object within the GUI with the images in the array.	When completed, the contents of the array MUST populate a list object within the GUI of all images.	**T10:** populate the Image list box with the contents of the Image Array. (Read from file)
5	Must use a sort algorithm from term 1.	The project must implement and use a sort algorithm from term 1.	**T14:** Implement a sort algorithm to allow various elements within the project to be sorted.
6	Must use a search algorithm from term 1.	The project must implement and use a search algorithm from term 1.	**T25:** Implement a search algorithm to allow various elements within the project to be searched.
7	MUST allow the Image list box to be sortable by file name and file date (asc, desc).	The Image list box must be sortable by file name and file date (asc, desc).	**T15:** implement sort functionality in the Image list box clicking headings will sort the list box contents.
8	MUST display the selected image in the GUI.	When an item in the Image list box is clicked the currently selected image must be displayed in the GUI.	**T10:** implement functionality to display the selected image in the Image Pane
9	MUST allow the user to open class files.	The GUI must provide a browse button which allows for the selection of class files from an operating system dialog box.	**T16:** display an Open Dialog Box with the filter set to "*.names".

10	MUST populate a Class array with the selected classes.	When a user has selected the class file an array must be populated with all classes in the file.	**T16:** populate the Class Array with the selected images.(File Open)
11	MUST populate the Class list box object within the GUI with the classes in the array.	When completed, the contents of the class array must populate the Class list box object within the GUI of all classes.	**T16:** populate the Class list box with the contents of the Class Array.(Load from file, Close file)
12	MUST preserve the order (line number)of the classes in the class file	The order of classes within the class file must be preserved.	**T16:** implement checks to ensure operations on the class file do not change line numbers of classes.
13	MUST allow the user to add classes to the class file.	The user must be able to add (append) classes to the class file from the GUI.	**T18:** Implement functionality to add a class, appending it to the end of the Class file. (File Append)
14	MUST allow the user to delete classes from the class file.	The user must be able to delete classes from the class file from the GUI. **This must not change the line numbers of the classes in the file.**	**T19:** Implement functionality to delete classes from the Class file. (delete from file)
15	MUST allow the user to sort the list of Classes	The user must be able to sort classes in the list.	**T17:** implement sort functionality in the Class list box clicking headings will sort the list box contents.
16	MUST allow the user to select a triangle, rectangle, trapezium, polygon (up to 8 points)	The following shapes must be available to select in the GUI - triangle, rectangle, trapezium, polygon (up to 8 points).	**T8:** Create clickable GUI elements for each of the 4 shapes.
17	Must allow the user to be able to draw the shapes over (but not on) the image.	The user must be able to draw the chosen shape over but not on the image.	**T21:** Create a transparent drawing layer which sits above the image.

18	MUST only use outlines for the shapes, no fill.	The shape must be an outline only (no fill).	**T21:** Ensure that all shapes are created as outlines without a fill.
19	SHOULD allow the user to scale (resize) the selected shape with the mouse.	The user should be able to scale the selected shape using the mouse. **This will update the associated annotation coordinates.**	**T24:** Implement functionality to scale the selected shape and update the associated annotation.
20	SHOULD allow the user to move the vertices of the selected polygon using the mouse.	The user should be able to move the vertices of the selected polygon using the mouse. **This will update the associated annotation coordinates.**	**T24:** Implement functionality to adjust the vertices of the selected polygon and update the associated annotation.
21	SHOULD allow the user to delete the selected shape using the mouse.	The user should be able to delete the selected shape using the mouse. **This will delete the associated annotation.**	**T29:** Implement functionality to delete the selected shape and the associated annotation.
22	SHOULD allow the user to copy and paste the selected shape using the mouse.	The user should be able to copy and paste the selected shape using the mouse. **This will copy the associated annotation into a new annotation.**	**T29:** Implement functionality to copy the selected shape and the associated annotation.
23	SHOULD display the class name above all shapes.	All shapes should display the associated class name above it in text.	**T21:** Implement functionality to display the class name in text above each shape.
24	MUST use the JSON for annotations files and use an ".annotations" extension. **(changed to JSON format as group decision)**	Annotation files must be JSON format with an ".annotations" extension.	**T13:** Research how to write to and read from JSON format files.

25	MUST allow the user to search for and open an annotations file.	The user must be able to open an existing annotations file by using a browse button which allows for the selection of an annotation from an operating system dialog box.	**T26:** display an Open Dialog Box with the filter set to *.annotations. Implement functionality to read from JSON format files. (Load from File, Close File)
26	MUST display associated annotations over the image. When an item in the Image list box is clicked.	When the user switches between images, any annotations associated with the image must be displayed.	**T12:** Implement functionality to search for annotations associated with the current image when the user changes image.
27	MUST store annotations in memory using a linked list.	The ability to remove annotations from anywhere in the list (not just the front and end) suggests using a linked list to store the annotations.	**T22:** Implement linked list functionality to store and manipulate the annotations in the application.
28	MUST allow the user to save the current set of annotations to a file.	The user must be able to save the current set of annotations to a file. This will involve a save button in the GUI.	**T22:** Implement functionality to write to JSON format files. (Save to file, Close file)
29	MUST alert the user if the save will overwrite an existing file.	If saving would overwrite an existing file, the user must confirm or cancel the overwrite action.	**T26:** Implement an OK_Cancel message box to alert the user.
30	MUST be able to change the name of an existing file.	The user must be able to change the name of an existing file.	**T26:** Implement functionality to change the name of an annotations file.
31	MUST include the following information in the annotations file: 1.Number of annotated images; For each Image	The annotation file must use hierarchical data to store the required data:	**T22:** Implement the given structure for data when handling annotation files.

	2. Image file name; 3. Number of shapes per image; For each shape 4. Shape type; 5. Point_1 (x, y); 6. Point_2 (x, y); 7. Point_n (x, y);		
32	SHOULD automatically save the application every minute	The application should automatically save the annotation file every minute, this will require a recurring internal timer.	**T28:** Implement a timer to trigger the auto save functionality using existing code from **T22**. (Save to file)
33	MUST perform the auto save using threads.	Any auto save must be done using threads.	**T28:** Implement functionality to use threads to perform **T22**.
34	COULD change the colour of annotations when drawn to help with visibility if contrast isn't high enough to be suitably visible	change the colour of a shape using the popup menu	**T64** implement functionality to change the colour of a shape using the popup menu
Non Functional			
35	MUST use C++ language	The application must use the C++ language	Ensure all code, frameworks and API's are C++.
36	MUST use a simple GUI	The application must use a simple GUI	**T8**: design and implement a simple GUI for the project.
37	MUST include at least one class which uses private member functions for common functionality that	The application must include at least one class which uses private member functions for common functionality that should not be exposed in the public interface.	**T2:** implement at least one class which uses private member functions for common functionality that should not be exposed in the public interface.

	should not be exposed in the public interface.		
38	MUST include at least one place where error and exception handling have been used.	The application must include at least one place where error and exception handling have been used.	**T3:** implement at least one instance of error and exception handling.
39	MUST include appropriate testing cases and testing libraries.	The application must include appropriate testing cases and testing libraries.	**T23, T27 and T30:** include appropriate testing cases and testing libraries in the project.
40	SHOULD utilise concurrent programming when appropriate.	Concurrent programming should be used when appropriate.	**T4:** seek opportunities to handle tasks concurrently
41	MUST research and decide upon an API for handling images.	The team must research and decide upon an API for handling images.	**T6, T9:** research and decide upon an API for handling images.
42	MUST research and decide upon a method of creating the GUI.	The team must research and decide upon a method of creating the GUI.	**T6:** research and decide upon a method of creating the GUI.
43	MUST produce a Gantt chart.	The Project Manager must produce a Gantt chart.	**T36:** produce a Gantt chart.
44	MUST produce a Risk Analysis table.	The Project Manager must produce a Risk Analysis table.	**T34:** produce a Risk Analysis table.
45	MUST produce a Requirements Table.	The Project Manager must produce a Requirements Table.	**T35:** produce a Requirements Table.
46	MUST submit his work to Dropbox.	The Project Manager must submit his work to Dropbox.	**T37, M2, D1:** Project Manager must submit his work to Dropbox

47	MUST present his work to the Lab Tutor.	The Project Manager must present his work to the Lab Tutor.	**T38, M4:** Project Manager must present individual work to the Lab Tutor.
48	MUST submit the final documentation.	The Project Manager must submit the final documentation.	**T32, M20 and D5:** Project Manager must submit the final documentation.
49	MUST submit a video demonstration	The Project Manager must submit a video demonstration of the project.	**T62, M20, D5:** Project Manager must submit a video demonstration of the software
50	Each team member must submit a contributions Form	Each team member must complete and submit a Contributions Form to the correct folder on Dropbox.	**T63, M20, D5:** Each member must submit a completed Contributions Form to Dropbox
51	MUST produce UML diagrams and description.	The System Architect must produce UML diagrams and descriptions.	**T42, T43, T44, T45 and T46:** produce UML diagrams and description.
52	MUST produce GUI Mock-up.	The System Architect must produce GUI Mock-up.	**T47:** produce GUI Mock-up.
53	MUST select the API for handling images.	The System Architect must select the API for handling images.	**T39:** select the API for handling images.
54	MUST select the method of Creating a GUI.	The System Architect must select the method of Creating a GUI.	**T40:** select the method of Creating a GUI.
55	MUST produce a list of these and any other tools needed.	The System Architect must produce a list of these and any other tools needed.	**T41:** produce a list of these and any other tools needed.

56	MUST submit his work to Dropbox.	The System Architect must submit his work to Dropbox.	**T49, M8 and D2:** System Architect must submit his work to Dropbox.
57	MUST present his work to the Lab Tutor.	The System Architect must present his work to the Lab Tutor.	**T50, M9:** System Architect must present individual work to the Lab Tutor.
58	MUST use a GitHub repository for storing the project code.	The team must use a GitHub repository for storing the project code.	**T51:** use a GitHub repository for storing the project code.
59	MUST produce a contribution guide for the Git repository.	The Software Developer must produce a contribution guide for the Git repository.	**T52:** produce a contribution guide for the Git repository.
60	MUST produce the reference manual for the software.	The Software Developer must produce the reference manual for the software.	**T53:** produce the reference manual for the software.
61	MUST produce a document detailing any features not implemented.	The Software Developer must produce a document detailing any features not implemented.	**T54:** produce a document detailing any features not implemented.
62	MUST submit his work to Dropbox.	The Software Developer must submit his work to Dropbox.	**T55, M12 and D3:** Software Developer must submit his work to Dropbox.
63	MUST present his work to the Lab Tutor.	The Software Developer must present his work to the Lab Tutor.	**T56, M13:** Software Developer must present individual work to the Lab Tutor.
64	MUST produce the test plan.	The Software Tester must produce the test plan.	**T59:** produce the test plan.

65	MUST decide how to test continual updates to the repository.	The Software Tester must decide how to test continual updates to the repository.	**T57:** decide how to test continual updates to the repository.
66	MUST choose any testing tools or frameworks which are needed to test the software.	The Software Tester must choose any testing tools or frameworks which are needed to test the software.	**T58:** choose any testing tools or frameworks which are needed to test the software.
67	MUST submit his work to Dropbox.	The Software Tester must submit his work to Dropbox.	**T60, M17 and D4:** Software Tester must submit his work to Dropbox.
68	MUST present his work to the Lab Tutor.	The Software Tester must present his work to the Lab Tutor.	**T61, M18:** Software Tester must present individual work to the Lab Tutor.

Risk Assessment

Risk assessment is an essential part of any project. By taking time to anticipate potential issues, rating them by severity and producing contingencies, the impact of many common issues can be reduced and their impact upon the progress and ultimate success of the project can be mitigated.

The following table (Table 3) describes the complete list of risks identified for the SDI Annotations project. Each risk is listed with ratings for Probability (how likely the risk will occur) and Impact (to what degree the risk will hamper the progress of the project. Each of these is rated from 1 to 5, where 1 represents a low impact or probability and 5 represents a high impact or probability. The table further discusses the details of the impact on the project and suggests strategies to help mitigate the effect upon the project in the final two columns.

The Risk Analysis for this project follows the SWOT method of risk analysis which is as follows:

Strengths - aspects of the project or team which give the project an advantage over other teams. For example this might include utilising a team member's previous experience in a specific field if it is related to the current.

Weaknesses - aspects of the team or project which may act as a disadvantage. For example this might include lack of access to certain types of hardware or software.

Opportunities - areas where the project or team can make use of trends, new software or existing strengths to create options and advantages. For example a new piece of documentation software may make parts of the project faster or richer in features.

Threats - factors which may affect the team or project negatively, for example bad timekeeping skills or bad time management.

Table 3 - a list of risks associated with the development of the application

Risk	**Probability** 1(low) - 5 (high)	**Impact** 1(low) - 5 (high)	**Impact on Project**	**Mitigation**
Illness / Absence	5	3	Illness or absence is unavoidable in any project and can have a high probability of having an impact on project timescales if not addressed.	Team members should notify the Project Manager when they are sick or will be absent. This allows the Project Manager to control the impact, redistribute workload and keep on top of deadlines in the case of prolonged absence.
Loss of Work	4	4	The impact of losing work through media corruption or	The group will use GitHub to store the development code

			user error is a strong possibility and has a high impact on the progress of the project.	which all members will work on. Backups will be made daily so that we have a solid system in place to recover from any loss.
Versionin g Conflicts	5	2	With multiple team members working on code simultaneously, versioning conflicts may occur.	Using GitHub will allow users to post code to the repository which will automatically stop conflicting code from committing. Members must then resolve this conflict themselves or communicate the issue to the team.
Human Error (Bad code committe d)	3	3	It is possible that a team member may commit some code which breaks the software. This will take time to fix unless rollbacks are implemented.	GitHub allows for rollbacks on repositories so if this occurs the group can rollback to an earlier version without the bad code.
GitHub Corruptio n	1	4	There is a possibility that the GitHub repository itself could become corrupted or unusable in some way.	Regular, incremental offline backups of the repository should be taken and stored in a safe place on a regular basis.
Inability to complete some features of the work	3	4	Due to the varying technical experience of the group coding and projects in general, this has a medium probability of happening.	This can be mitigated within the group by providing and allowing time to examine existing solutions and discussing and having a clear priority on core features. All features which are core to the project should be completed first, with any additional features only attempted when the core features are working.
Bugs in code	5	4	Bugs in code can be difficult to find and can push timescales for the project out as members try and find and fix the bug.	Regular testing, especially employing unit testing or test driven development can help minimise the possibility of serious bugs emerging. Fluidity in

				the project timing should also be used to allow for this.
Technical issues	3	3	Technical issues with computer equipment, flash drives, internet provider etc can all have an impact on project timescales.	Group members can always access the PC's at university if their own laptop or PC breaks or they lose internet.
Varying Experien ce and Degree Subjects	5	3	Experience among the team in the different languages used varies greatly All members of the team don't have the same experience in web technologies, databases and C#.	For some this will be partially remedied during Term 2 due to relevant modules switching to advanced topics. For everyone else, training must be included in the project plan so that everyone can learn something useful from the project and use those skills to contribute in a meaningful way to the development of the project.
Lack of knowledg e of relevant products	2	4	Understanding how the software the team is working on will function is a useful ability to have. Team members who have not used similar software or don't have experience of GUI's might struggle to understand what a File Open Dialog is or how a dropdown list works.	The team understands the software well enough due to in-class demos and the components used in the software will be very familiar to anyone who has used a graphical Operating System such as Windows. This gives this particular risk a very low chance of happening. Discussion of the features, aims and outcomes of the project will help clarify any misunderstanding of these products.
Losing a member of the group	1	5	The probability that we will permanently lose a member of the group is low but should be considered. This could be through medical issues, bereavement or even dropping out of University completely. This would severely impact	Should this happen it will be necessary for that person to communicate this to the team as soon as possible. The team will need to organise a team meeting as soon as possible. Handling this situation well will require the Project Manager to reallocate

			the workload and the project overall.	responsibilities in the short term within the group so that the project can continue and the group should seek help and advice from relevant tutors and module leader in finding a long term solution. If the team member simply stops communicating the Project Manager should discuss this with the team and arrange for members to be swapped. The non-communicative member will be designated as Tester and their duties will be completed by the other members as much as possible.
Lack of engagement from team members	4	4	The timescales for the project are such that if a team member stops communicating, contributing or frequently misses meetings that they will struggle to keep up and the project may suffer.	Having meetings at reasonable intervals will also allow the project manager to gauge engagement of members. A member who doesn't contribute or misses several meetings can be flagged for further contact with the aim of resolving any issues that might be causing that particular member to disengage.
Missed deadlines or deadlines overrunning	2	3	The timescales for the project are such that if a team member misses deadlines or deadlines overrun, the project will face issues. Each part of the project relies on previous parts being completed so this will have a cascading effect if unchecked.	This can be mitigated by having reasonable milestones in the project so that the project manager can be aware of any issues that arise in a reasonable time. If any work takes longer than expected or doesn't get completed, the project manager will be able to spot this early. This will allow the project manager to reassign tasks or team members to deal with any issues as needed.

Disagreement between members	2	2	If a disagreement occurs between members this can halt progress until a decision is made and may cause issues if one member feels they were overruled.	Where possible the group should resolve issues internally. Where a decision can't be made in good time, our Lab Tutor has been defined as having the final vote.
Keeping motivation high	5	3	With some students dealing with multiple simultaneous group projects for the duration of this term, it may be difficult to keep team members focused on the project. This might lead to missed deadlines or lack of interest.	It is important to have milestones to keep people focused and to back that up with good project management - praise, giving team members meaningful contribution and giving enough time to complete tasks.
Conflicting deadlines / Workload	5	4	With 4 group projects running at the same time this term, there are conflicts between deadlines and workload with each project having different demands on time.	The only real solution to this is meticulous timekeeping with regard to the project plan. Group members need to work steadily towards goals and divide their time between all projects. Communicating deadlines with the project manager will allow them to build flexibility into the schedule around those deadlines.
Time Management	2	4	Poor time management among members of the group could lead to issues delivering the project on time.	The project manager must stick to the project plan and make sure the project continues. Group members must be aware of upcoming deadlines and work towards those continually across the term.
Poor Scheduling	2	3	Poor understanding of the complexity of tasks involved could result in underestimating time needed for various tasks.	Sufficient time must be given to each task so that there is flexibility built into the schedule to account for unforeseen circumstances.

Inadequately defined Requirements	1	4	Risks to the project caused by requirements that are inadequately defined	Lack of granularity could cause confusion among team members on how to implement requirements and could result in sections of the code having to be rewritten to accommodate missing or badly defined requirements. The project manager must ensure that all requirements are stated clearly, communicated well and are met in subsequent testing strategies and coding milestones.
COVID-19 (Additional Update)	1	5	Due to the COVID-19 pandemic, major disruption and changes have and will happen to the group.	In order to combat this we must take steps to stay in touch via Slack and be prepared to hold meetings on MS Teams. The usual structure and timing of meetings can change to accommodate different time zones. Communication will not only help the group stay focused but will also help to combat feelings of isolation for members of the group.

List of Tasks / Time Plans

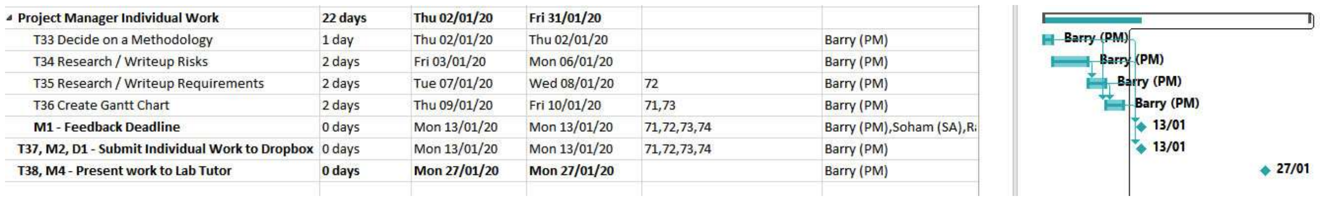
A Gantt chart was created to help manage team members, assign jobs and to track deadlines. This has been split into smaller pieces for ease of discussion. The project involved deadlines for each member of the team and these were kept in a separate are of the chart to aid clarity and because these were ongoing individual pieces of work which would happen alongside the main project.

Individual Submissions

Project Manager

The project manager task involved the creation of the Risk Assessment table (Table 2), Requirements table (Table 3) and the Gantt chart being discussed in this section.

Figure 1 shows the breakdown of the tasks, milestones and deliverables for this portion of the assignment



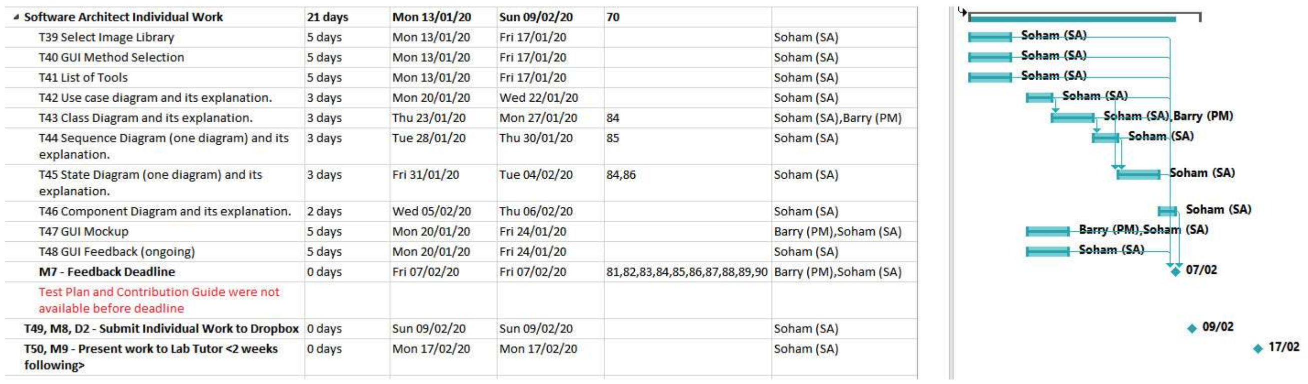
Project Manager Individual Work	**22 days**	**Thu 02/01/20**	**Fri 31/01/20**		
T33 Decide on a Methodology	1 day	Thu 02/01/20	Thu 02/01/20		Barry (PM)
T34 Research / Writeup Risks	2 days	Fri 03/01/20	Mon 06/01/20		Barry (PM)
T35 Research / Writeup Requirements	2 days	Tue 07/01/20	Wed 08/01/20	72	Barry (PM)
T36 Create Gantt Chart	2 days	Thu 09/01/20	Fri 10/01/20	71,73	Barry (PM)
M1 - Feedback Deadline	0 days	Mon 13/01/20	Mon 13/01/20	71,72,73,74	Barry (PM),Soham (SA),R
T37, M2, D1 - Submit Individual Work to Dropbox	0 days	Mon 13/01/20	Mon 13/01/20	71,72,73,74	Barry (PM)
T38, M4 - Present work to Lab Tutor	**0 days**	**Mon 27/01/20**	**Mon 27/01/20**		Barry (PM)

Figure 1 - Gant chart section displaying the Project Manager's submission

Software Architect

The Software Architect deliverable involved the creation of several diagrams including Use Case, Class, Sequence, State and Component diagram. This submission also included the selection and explanation of why the core libraries and tools were chosen.

Figure 2 shows the breakdown of the tasks, milestones and deliverables for this portion of the assignment. Please note that both the required Contributions Guide and Test Plan were not produced in time to be included in the decisions made by the Systems Analyst. This will be discussed as part of the main Gantt later in this chapter.



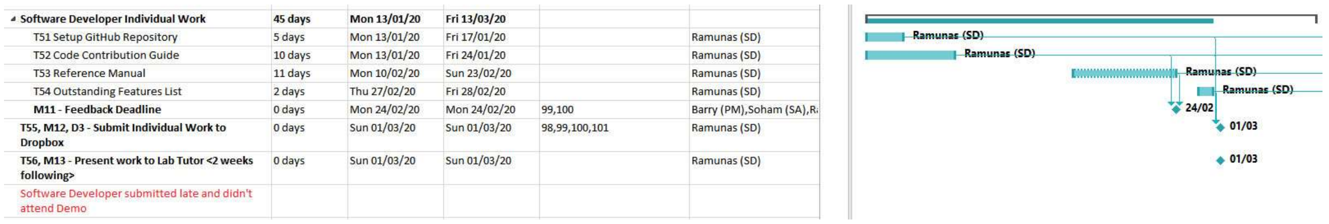
Software Architect Individual Work	**21 days**	**Mon 13/01/20**	**Sun 09/02/20**	**70**	
T39 Select Image Library	5 days	Mon 13/01/20	Fri 17/01/20		Soham (SA)
T40 GUI Method Selection	5 days	Mon 13/01/20	Fri 17/01/20		Soham (SA)
T41 List of Tools	5 days	Mon 13/01/20	Fri 17/01/20		Soham (SA)
T42 Use case diagram and its explanation.	3 days	Mon 20/01/20	Wed 22/01/20		Soham (SA)
T43 Class Diagram and its explanation.	3 days	Thu 23/01/20	Mon 27/01/20	84	Soham (SA),Barry (PM)
T44 Sequence Diagram (one diagram) and its explanation.	3 days	Tue 28/01/20	Thu 30/01/20	85	Soham (SA)
T45 State Diagram (one diagram) and its explanation.	3 days	Fri 31/01/20	Tue 04/02/20	84,86	Soham (SA)
T46 Component Diagram and its explanation.	2 days	Wed 05/02/20	Thu 06/02/20		Soham (SA)
T47 GUI Mockup	5 days	Mon 20/01/20	Fri 24/01/20		Barry (PM),Soham (SA)
T48 GUI Feedback (ongoing)	5 days	Mon 20/01/20	Fri 24/01/20		Soham (SA)
M7 - Feedback Deadline	0 days	Fri 07/02/20	Fri 07/02/20	81,82,83,84,85,86,87,88,89,90	Barry (PM),Soham (SA)
Test Plan and Contribution Guide were not available before deadline					
T49, M8, D2 - Submit Individual Work to Dropbox	0 days	Sun 09/02/20	Sun 09/02/20		Soham (SA)
T50, M9 - Present work to Lab Tutor <2 weeks following>	0 days	Mon 17/02/20	Mon 17/02/20		Soham (SA)

Figure 2 - Section of the Gantt chart displaying the Software Architect submission

Software Developer

The Software Developer submission included setting up the GitHub repository, development of a Contributions Guide so that other members of the group would know how to code for the project and documentation of the software alongside a list of outstanding features. More time was given to this and the Software Tester submission so that they had time to deliver the required draft versions of the Contribution Guide and Test Plan to the Software Architect.

This deadline was missed and the Software Developer submitted late. The demonstration was also missed completely, forcing the group to adjust members and discuss options with the Module Leader.

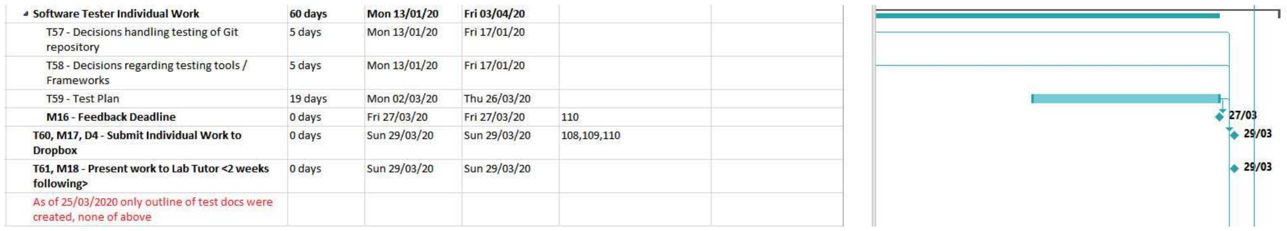


Software Developer Individual Work	**45 days**	**Mon 13/01/20**	**Fri 13/03/20**		
T51 Setup GitHub Repository	5 days	Mon 13/01/20	Fri 17/01/20		Ramunas (SD)
T52 Code Contribution Guide	10 days	Mon 13/01/20	Fri 24/01/20		Ramunas (SD)
T53 Reference Manual	11 days	Mon 10/02/20	Sun 23/02/20		Ramunas (SD)
T54 Outstanding Features List	2 days	Thu 27/02/20	Fri 28/02/20		Ramunas (SD)
M11 - Feedback Deadline	0 days	Mon 24/02/20	Mon 24/02/20	99,100	Barry (PM),Soham (SA),R
T55, M12, D3 - Submit Individual Work to Dropbox	0 days	Sun 01/03/20	Sun 01/03/20	98,99,100,101	Ramunas (SD)
T56, M13 - Present work to Lab Tutor <2 weeks following>	0 days	Sun 01/03/20	Sun 01/03/20		Ramunas (SD)
Software Developer submitted late and didn't attend Demo					

Figure 3 - Gantt chart section showing the Software Developer submission

Software Tester

The Software Tester was tasked with making decisions regarding the handling of testing on GitHub, decisions regarding any testing tools required and development of the Test Plan for the project.

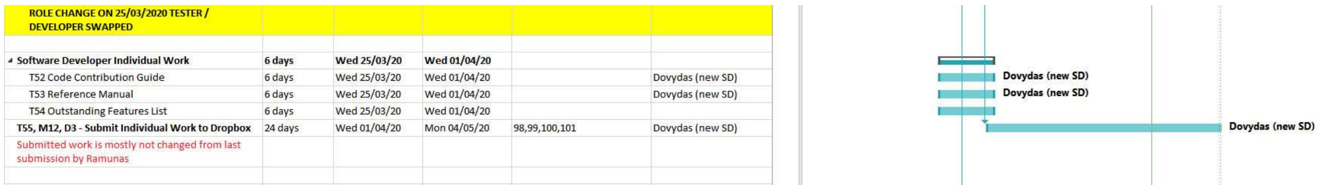


Software Tester Individual Work	**60 days**	**Mon 13/01/20**	**Fri 03/04/20**		
T57 - Decisions handling testing of Git repository	5 days	Mon 13/01/20	Fri 17/01/20		
T58 - Decisions regarding testing tools / Frameworks	5 days	Mon 13/01/20	Fri 17/01/20		
T59 - Test Plan	19 days	Mon 02/03/20	Thu 26/03/20		
M16 - Feedback Deadline	0 days	Fri 27/03/20	Fri 27/03/20	110	
T60, M17, D4 - Submit Individual Work to Dropbox	0 days	Sun 29/03/20	Sun 29/03/20	108,109,110	
T61, M18 - Present work to Lab Tutor <2 weeks following>	0 days	Sun 29/03/20	Sun 29/03/20		
As of 25/03/2020 only outline of test docs were created, none of above					

Figure 4 - Gantt chart section detailing the Software Developer Submission

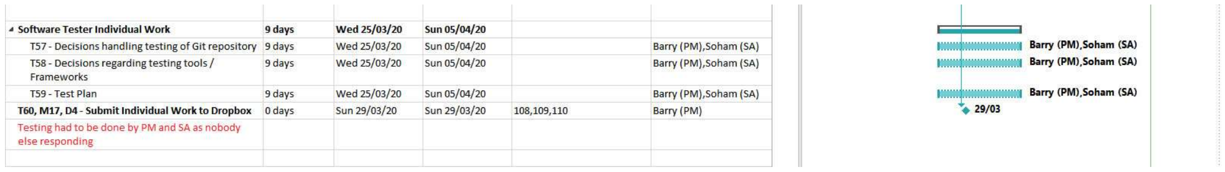
Software Developer and Software Tester Role Switch

Unfortunately the team lost one member during the project and roles had to be adjusted accordingly on the 25/03/2020 due to this, additional sections in the Gantt chart had to be added to accommodate the switch. The following figures (Figure 5 and Figure 6) show the change in roles and the distribution of the outstanding tasks among members.



ROLE CHANGE ON 25/03/2020 TESTER / DEVELOPER SWAPPED					
Software Developer Individual Work	**6 days**	**Wed 25/03/20**	**Wed 01/04/20**		
T52 Code Contribution Guide	6 days	Wed 25/03/20	Wed 01/04/20		Dovydas (new SD)
T53 Reference Manual	6 days	Wed 25/03/20	Wed 01/04/20		Dovydas (new SD)
T54 Outstanding Features List	6 days	Wed 25/03/20	Wed 01/04/20		
T55, M12, D3 - Submit Individual Work to Dropbox	24 days	Wed 01/04/20	Mon 04/05/20	98,99,100,101	Dovydas (new SD)
Submitted work is mostly not changed from last submission by Ramunas					

Figure 5 - Gantt chart section detailing the amended Software Developer submission

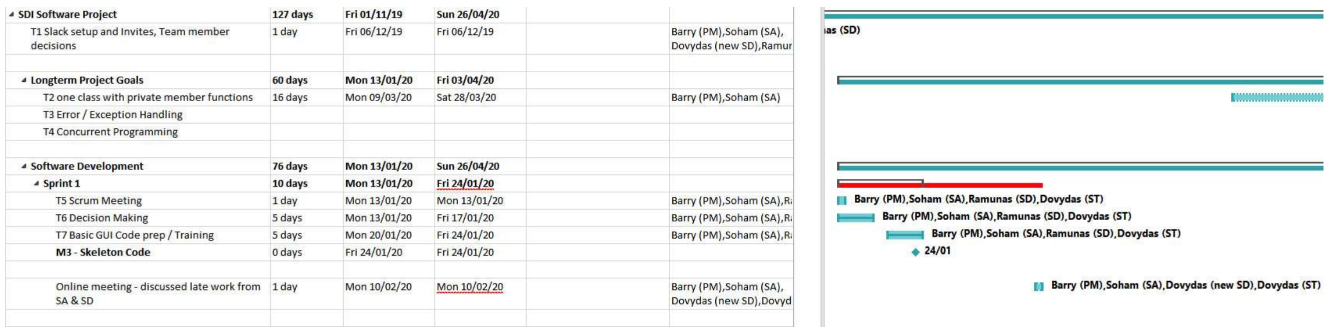


Software Tester Individual Work	**9 days**	**Wed 25/03/20**	**Sun 05/04/20**		
T57 - Decisions handling testing of Git repository	9 days	Wed 25/03/20	Sun 05/04/20		Barry (PM),Soham (SA)
T58 - Decisions regarding testing tools / Frameworks	9 days	Wed 25/03/20	Sun 05/04/20		Barry (PM),Soham (SA)
T59 - Test Plan	9 days	Wed 25/03/20	Sun 05/04/20		Barry (PM),Soham (SA)
T60, M17, D4 - Submit Individual Work to Dropbox	0 days	Sun 29/03/20	Sun 29/03/20	108,109,110	Barry (PM)
Testing had to be done by PM and SA as nobody else responding					

Figure 6 - Gantt chart section detailing the distribution of tasks for the Tester Submission

Project Flow

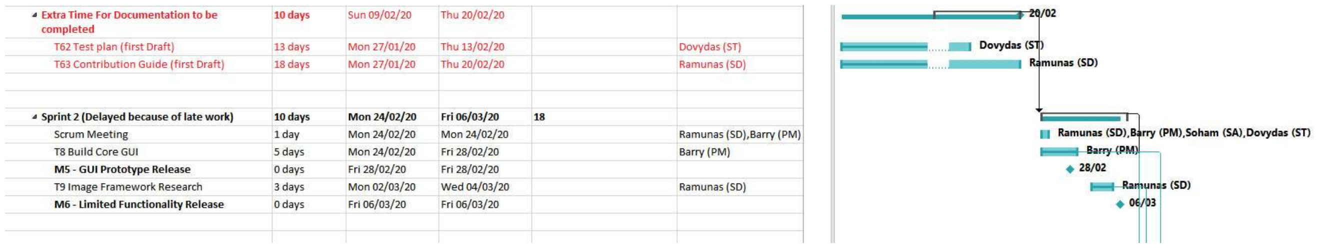
Figure 7 shows the initial period within the Team. Long-term project goals were set up as well as Slack and Trello. The initial tasks were focused heavily on decision making, leaving plenty of time for team members to work on their individual submissions. The Project Manager moved items around within the project plan to give both the Software Developer and Software tester 2 clear weeks to create the draft versions of their documentation needed by the Software Architect.


as (SD)
Barry (PM),Soham (SA),Ramunas (SD),Dovydas (ST)
Barry (PM),Soham (SA),Ramunas (SD),Dovydas (ST)
Barry (PM),Soham (SA),Ramunas (SD),Dovydas (ST)
24/01
Barry (PM),Soham (SA),Dovydas (new SD),Dovydas (ST)

SDI Software Project	**127 days**	**Fri 01/11/19**	**Sun 26/04/20**		
T1 Slack setup and Invites, Team member decisions	1 day	Fri 06/12/19	Fri 06/12/19		Barry (PM),Soham (SA), Dovydas (new SD),Ramur
Longterm Project Goals	**60 days**	**Mon 13/01/20**	**Fri 03/04/20**		
T2 one class with private member functions	16 days	Mon 09/03/20	Sat 28/03/20		Barry (PM),Soham (SA)
T3 Error / Exception Handling					
T4 Concurrent Programming					
Software Development	**76 days**	**Mon 13/01/20**	**Sun 26/04/20**		
Sprint 1	**10 days**	**Mon 13/01/20**	**Fri 24/01/20**		
T5 Scrum Meeting	1 day	Mon 13/01/20	Mon 13/01/20		Barry (PM),Soham (SA),R
T6 Decision Making	5 days	Mon 13/01/20	Fri 17/01/20		Barry (PM),Soham (SA),R
T7 Basic GUI Code prep / Training	5 days	Mon 20/01/20	Fri 24/01/20		Barry (PM),Soham (SA),R
M3 - Skeleton Code	0 days	Fri 24/01/20	Fri 24/01/20		
Online meeting - discussed late work from SA & SD	1 day	Mon 10/02/20	Mon 10/02/20		Barry (PM),Soham (SA), Dovydas (new SD),Dovyd

Figure 7 - Initial section of the Gantt chart showing the early days of the project

Despite this, both draft documents did not appear before the deadline. A meeting was called the following day to discuss this and the documentation was promised to be delivered.

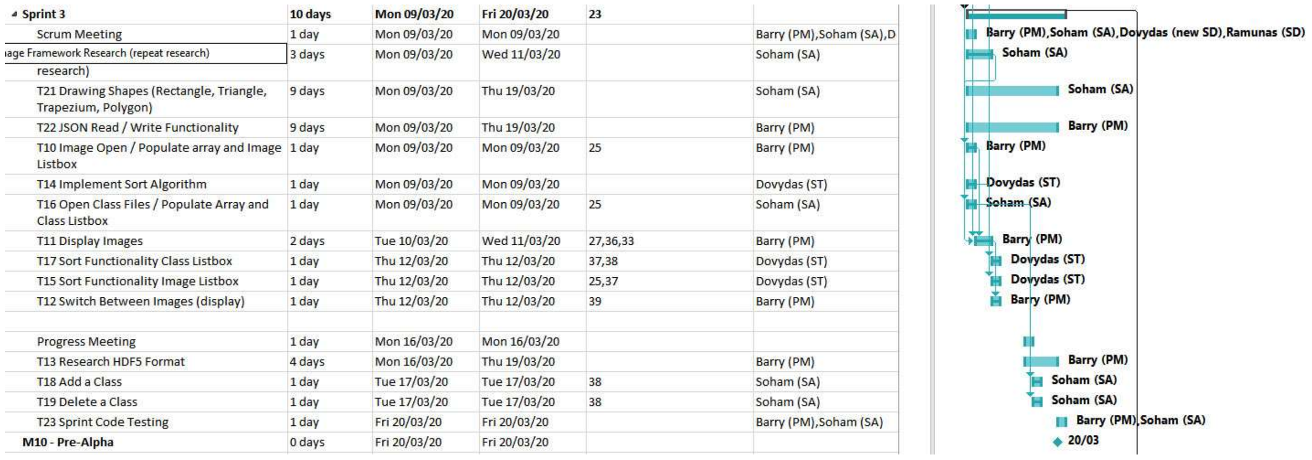


Extra Time For Documentation to be completed	**10 days**	Sun 09/02/20	Thu 20/02/20		
T62 Test plan (first Draft)	13 days	Mon 27/01/20	Thu 13/02/20		Dovydas (ST)
T63 Contribution Guide (first Draft)	18 days	Mon 27/01/20	Thu 20/02/20		Ramunas (SD)
Sprint 2 (Delayed because of late work)	**10 days**	**Mon 24/02/20**	**Fri 06/03/20**	**18**	
Scrum Meeting	1 day	Mon 24/02/20	Mon 24/02/20		Ramunas (SD),Barry (PM)
T8 Build Core GUI	5 days	Mon 24/02/20	Fri 28/02/20		Barry (PM)
M5 - GUI Prototype Release	0 days	Fri 28/02/20	Fri 28/02/20		
T9 Image Framework Research	3 days	Mon 02/03/20	Wed 04/03/20		Ramunas (SD)
M6 - Limited Functionality Release	0 days	Fri 06/03/20	Fri 06/03/20		

Figure 8 - Sprint 2 within the Gantt chart

Figure 8 shows the delays caused by this, ranging from a few days with regard to the Testing documentation to a delay of almost 2 weeks in terms of the Contribution Guide. On the advice of the Module Leader, the Project manager stopped the group from progressing onto coding until this was in place.

During Sprint 2, which focused mainly on researching the various frameworks and libraries, the Software Developer gradually stopped participating, culminating in his total absence by the end of this sprint. This caused a huge amount of delay as other team members had to cover tasks and research everything from scratch. Until the research was completed, we simply couldn't begin to code and so the project suffered and timescales had to be adjusted and tasks reorganised.

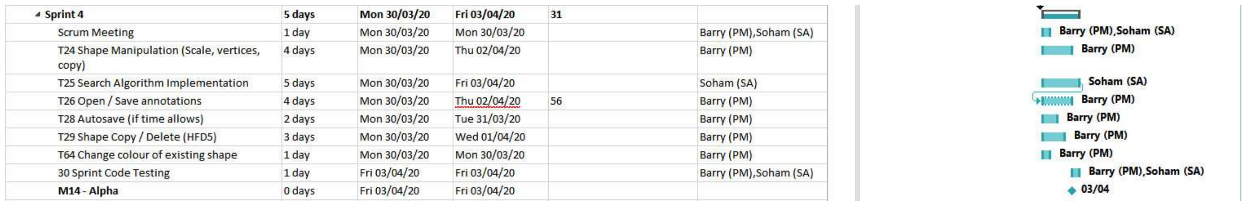


Sprint 3	**10 days**	**Mon 09/03/20**	**Fri 20/03/20**	**23**	
Scrum Meeting	1 day	Mon 09/03/20	Mon 09/03/20		Barry (PM),Soham (SA),D
ıage Framework Research (repeat research) research)	3 days	Mon 09/03/20	Wed 11/03/20		Soham (SA)
T21 Drawing Shapes (Rectangle, Triangle, Trapezium, Polygon)	9 days	Mon 09/03/20	Thu 19/03/20		Soham (SA)
T22 JSON Read / Write Functionality	9 days	Mon 09/03/20	Thu 19/03/20		Barry (PM)
T10 Image Open / Populate array and Image Listbox	1 day	Mon 09/03/20	Mon 09/03/20	25	Barry (PM)
T14 Implement Sort Algorithm	1 day	Mon 09/03/20	Mon 09/03/20		Dovydas (ST)
T16 Open Class Files / Populate Array and Class Listbox	1 day	Mon 09/03/20	Mon 09/03/20	25	Soham (SA)
T11 Display Images	2 days	Tue 10/03/20	Wed 11/03/20	27,36,33	Barry (PM)
T17 Sort Functionality Class Listbox	1 day	Thu 12/03/20	Thu 12/03/20	37,38	Dovydas (ST)
T15 Sort Functionality Image Listbox	1 day	Thu 12/03/20	Thu 12/03/20	25,37	Dovydas (ST)
T12 Switch Between Images (display)	1 day	Thu 12/03/20	Thu 12/03/20	39	Barry (PM)
Progress Meeting	1 day	Mon 16/03/20	Mon 16/03/20		
T13 Research HDF5 Format	4 days	Mon 16/03/20	Thu 19/03/20		Barry (PM)
T18 Add a Class	1 day	Tue 17/03/20	Tue 17/03/20	38	Soham (SA)
T19 Delete a Class	1 day	Tue 17/03/20	Tue 17/03/20	38	Soham (SA)
T23 Sprint Code Testing	1 day	Fri 20/03/20	Fri 20/03/20		Barry (PM),Soham (SA)
M10 - Pre-Alpha	0 days	Fri 20/03/20	Fri 20/03/20		

Figure 9 - Gantt chart section showing the third Sprint

Figure 9 shows the team attempting to make up for approximately a month's worth of lost time by having the Software Architect work on one section of the research and code while the Software tester and Project Manager worked on the initial code.

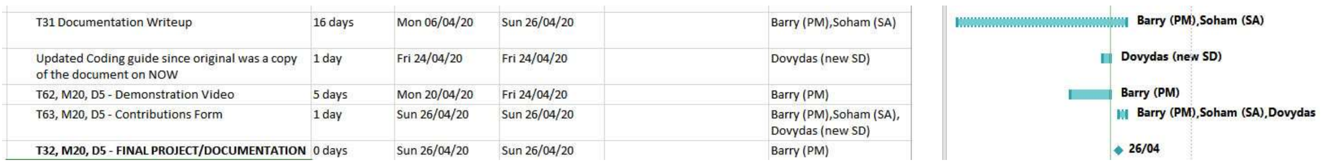
The Project Manager by this point had wholly taken over the role of Software Developer and gradually, the team began to progress, slowly catching up in terms of time.



Sprint 4	**5 days**	**Mon 30/03/20**	**Fri 03/04/20**	**31**	
Scrum Meeting	1 day	Mon 30/03/20	Mon 30/03/20		Barry (PM),Soham (SA)
T24 Shape Manipulation (Scale, vertices, copy)	4 days	Mon 30/03/20	Thu 02/04/20		Barry (PM)
T25 Search Algorithm Implementation	5 days	Mon 30/03/20	Fri 03/04/20		Soham (SA)
T26 Open / Save annotations	4 days	Mon 30/03/20	Thu 02/04/20	56	Barry (PM)
T28 Autosave (if time allows)	2 days	Mon 30/03/20	Tue 31/03/20		Barry (PM)
T29 Shape Copy / Delete (HFD5)	3 days	Mon 30/03/20	Wed 01/04/20		Barry (PM)
T64 Change colour of existing shape	1 day	Mon 30/03/20	Mon 30/03/20		Barry (PM)
30 Sprint Code Testing	1 day	Fri 03/04/20	Fri 03/04/20		Barry (PM),Soham (SA)
M14 - Alpha	0 days	Fri 03/04/20	Fri 03/04/20		

Figure 10 - The section of the Gantt chart documenting the final Sprint

Figure 10 shows the final coding sprint in which the team worked through the requirements list and tried to finish everything that they felt was possible in the allotted time. By this point, the Software Tester had become the Software Developer, and like their predecessor, the amount of activity from them dropped to nothing. This final Sprint, as demonstrated, was carried out by the Project Manager and Software Architect. This sprint was originally intended to be used as a testing phase, however it had to be used to finish off the software.



T31 Documentation Writeup	16 days	Mon 06/04/20	Sun 26/04/20		Barry (PM),Soham (SA)
Updated Coding guide since original was a copy of the document on NOW	1 day	Fri 24/04/20	Fri 24/04/20		Dovydas (new SD)
T62, M20, D5 - Demonstration Video	5 days	Mon 20/04/20	Fri 24/04/20		Barry (PM)
T63, M20, D5 - Contributions Form	1 day	Sun 26/04/20	Sun 26/04/20		Barry (PM),Soham (SA), Dovydas (new SD)
T32, M20, D5 - FINAL PROJECT/DOCUMENTATION	0 days	Sun 26/04/20	Sun 26/04/20		Barry (PM)

Figure 11 - The section of the Gantt chart covering documentation

Finally, figure 11 shows the final part of the project, again the new Software Developer had little input although it should be noted that he did replace the current Contributions Guide with a version which wasn't a copy of the version on NOW.

Assumptions

The following assumptions were made during the development of this software:

Development

- Each member of the team will use the software on a device with a keyboard and mouse (or other method of text entry and selection).
- Each member will have an active internet connection for research and software acquisition purposes.
- Open Source software will be used where possible to keep costs to a minimum.
- The team will follow Agile Methodologies throughout development.
- The team will develop the software in the C++ language and all plugins, API's and tools used will be compatible with this language.
- The project scope will not change drastically after sign off (the specification will not change).
- The software will exist as a standalone application without the need for an internet connection.
- During development, the team will adhere to guidelines within the BCS Code of Conduct.

End User

- Due to the precise nature of the shapes, accessibility may be an issue. Without a precise method of selecting and adjusting vertices available, this process may be beyond the physical limitations of some users.
- Each user will use the software on a device with a suitable text input and position tracking device such as a keyboard and mouse.
- The end user should not require an internet connection to use this software.
- The end user should not need to install any other products to use this application

Adopted Coding Standards

Initially the team began coding in standard C++ but when OpenCV didn't fit with our needs, the switch to Qt native graphics classes resulted in a necessary switch to Qt based versions of the language (QString, QVector etc). This was necessary since the classes in question used them heavily so switching to this style was inevitable.

In terms of actual code, the team adopted a strategy of keeping the code clean and understandable.

Variable names, classes and function names were all allocated a style and an instruction to keep the name so clear that it should not require comments to allow the user to understand the meaning. So, instead of using "i" for a loop counter, we would use "loopcount". Classes would be named for what they were, always as a capitalised word – Controller, Model, View etc. Functions would be named for what they did, always with camel case – updateAnnotation() etc.

Commenting would take 2 styles. The main, documentable style would apply to header files, since they contain the entire class including properties and methods. This would follow the style set out by Doxygen and allow for comprehensive documentation to be automated. Within the .cpp files themselves, comments would be focused more on explaining why the code was structured as it was or explaining what was happening in complex portions of the code.

Indentation would be important as well as using full versions of the "if" statement with braces ({}). This would help with readability and with error checking, since it is too easy to misread where the statement ends without these.

White space would be used to help with readability, breaking the code into smaller steps of a few lines.

The full version of the software Programming Style Guide can be found in the DOCUMENTATION folder.

Implementation

Use Case Diagram

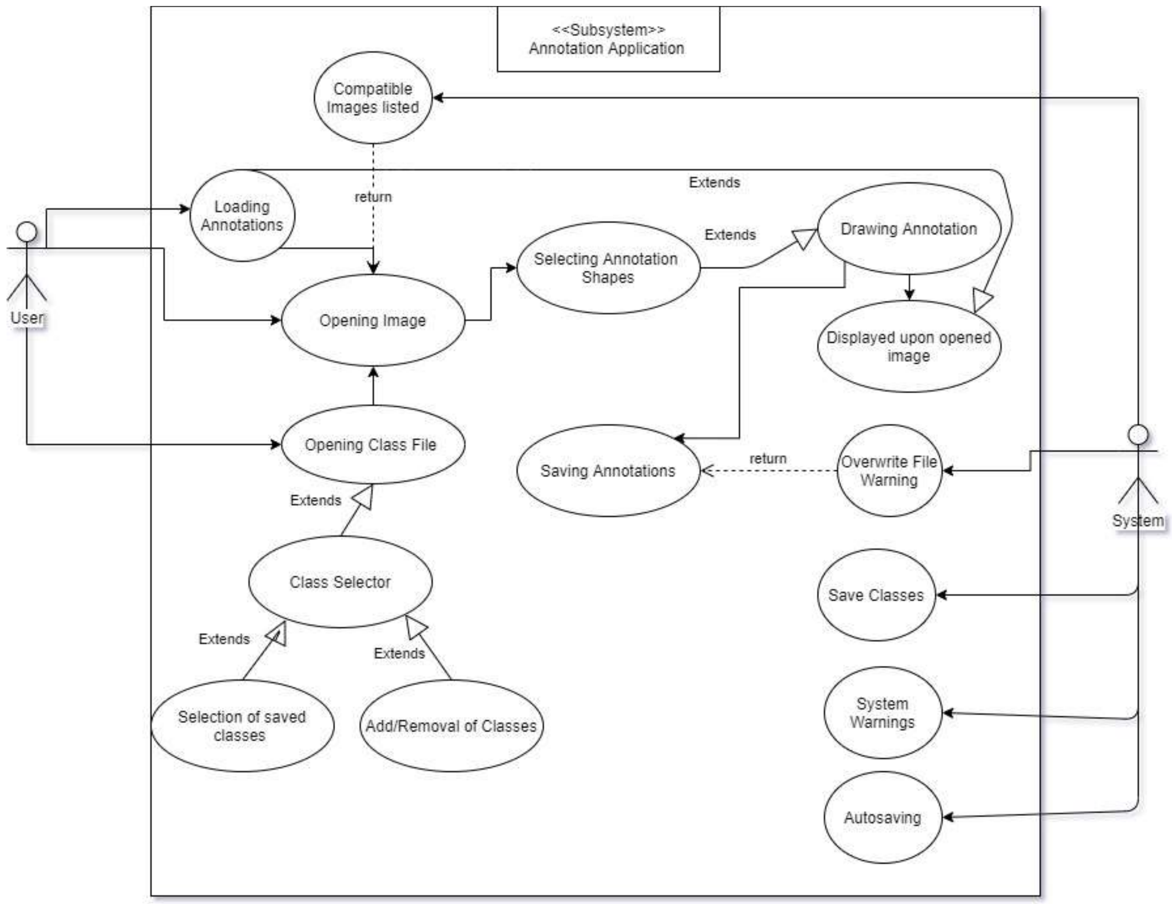


Figure 12 - Depicting Use Case Scenario

Explanation:

Users can load Images, Class files or Annotations after starting the application. Users can then add or remove classes or load previously saved classes after opening a class file. After opening an image, the user can load annotations, which will be displayed on screen, if the related image is opened.

Users can select shapes, which include Triangle, rectangle, circle, polygon (8 points) and draw shape on the image, which will be displayed live. The User can save annotations, if overwriting, a system warning will be displayed. The system can also auto save Annotations at a fixed time interval.

Class Diagram

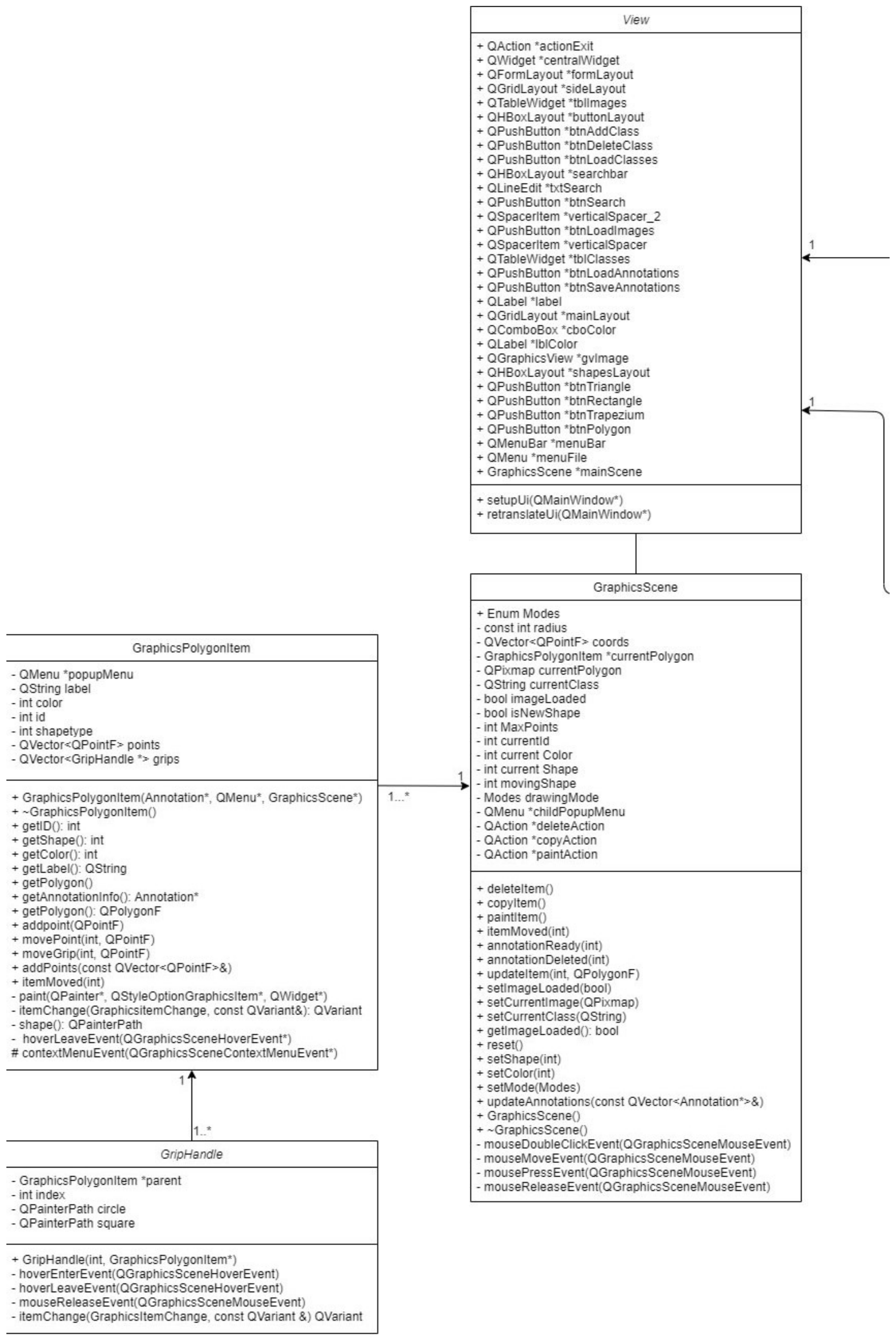

View
+ QAction *actionExit
+ QWidget *centralWidget
+ QFormLayout *formLayout
+ QGridLayout *sideLayout
+ QTableWidget *tblImages
+ QHBoxLayout *buttonLayout
+ QPushButton *btnAddClass
+ QPushButton *btnDeleteClass
+ QPushButton *btnLoadClasses
+ QHBoxLayout *searchbar
+ QLineEdit *txtSearch
+ QPushButton *btnSearch
+ QSpacerItem *verticalSpacer_2
+ QPushButton *btnLoadImages
+ QSpacerItem *verticalSpacer
+ QTableWidget *tblClasses
+ QPushButton *btnLoadAnnotations
+ QPushButton *btnSaveAnnotations
+ QLabel *label
+ QGridLayout *mainLayout
+ QComboBox *cboColor
+ QLabel *lblColor
+ QGraphicsView *gvImage
+ QHBoxLayout *shapesLayout
+ QPushButton *btnTriangle
+ QPushButton *btnRectangle
+ QPushButton *btnTrapezium
+ QPushButton *btnPolygon
+ QMenuBar *menuBar
+ QMenu *menuFile
+ GraphicsScene *mainScene
+ setupUi(QMainWindow*)
+ retranslateUi(QMainWindow*)
1
1
GraphicsScene
+ Enum Modes
- const int radius
- QVector<QPointF> coords
- GraphicsPolygonItem *currentPolygon
- QPixmap currentPolygon
- QString currentClass
- bool imageLoaded
- bool isNewShape
- int MaxPoints
- int currentId
- int current Color
- int current Shape
- int movingShape
- Modes drawingMode
- QMenu *childPopupMenu
- QAction *deleteAction
- QAction *copyAction
- QAction *paintAction
+ deleteItem()
+ copyItem()
+ paintItem()
+ itemMoved(int)
+ annotationReady(int)
+ annotationDeleted(int)
+ updateItem(int, QPolygonF)
+ setImageLoaded(bool)
+ setCurrentImage(QPixmap)
+ setCurrentClass(QString)
+ getImageLoaded(): bool
+ reset()
+ setShape(int)
+ setColor(int)
+ setMode(Modes)
+ updateAnnotations(const QVector<Annotation*>&)
+ GraphicsScene()
+ ~GraphicsScene()
- mouseDoubleClickEvent(QGraphicsSceneMouseEvent)
- mouseMoveEvent(QGraphicsSceneMouseEvent)
- mousePressEvent(QGraphicsSceneMouseEvent)
- mouseReleaseEvent(QGraphicsSceneMouseEvent)
GraphicsPolygonItem
- QMenu *popupMenu
- QString label
- int color
- int id
- int shapetype
- QVector<QPointF> points
- QVector<GripHandle *> grips
+ GraphicsPolygonItem(Annotation*, QMenu*, GraphicsScene*)
+ ~GraphicsPolygonItem()
+ getID(): int
+ getShape(): int
+ getColor(): int
+ getLabel(): QString
+ getPolygon()
+ getAnnotationInfo(): Annotation*
+ getPolygon(): QPolygonF
+ addpoint(QPointF)
+ movePoint(int, QPointF)
+ moveGrip(int, QPointF)
+ addPoints(const QVector<QPointF>&)
+ itemMoved(int)
- paint(QPainter*, QStyleOptionGraphicsItem*, QWidget*)
- itemChange(GraphicsitemChange, const QVariant&): QVariant
- shape(): QPainterPath
- hoverLeaveEvent(QGraphicsSceneHoverEvent*)
# contextMenuEvent(QGraphicsSceneContextMenuEvent*)
1...*
1
1
1..*
GripHandle
- GraphicsPolygonItem *parent
- int index
- QPainterPath circle
- QPainterPath square
+ GripHandle(int, GraphicsPolygonItem*)
- hoverEnterEvent(QGraphicsSceneHoverEvent)
- hoverLeaveEvent(QGraphicsSceneHoverEvent)
- mouseReleaseEvent(QGraphicsSceneMouseEvent)
- itemChange(GraphicsItemChange, const QVariant &) QVariant

Figure 13 - Left hand side of the Class Diagram

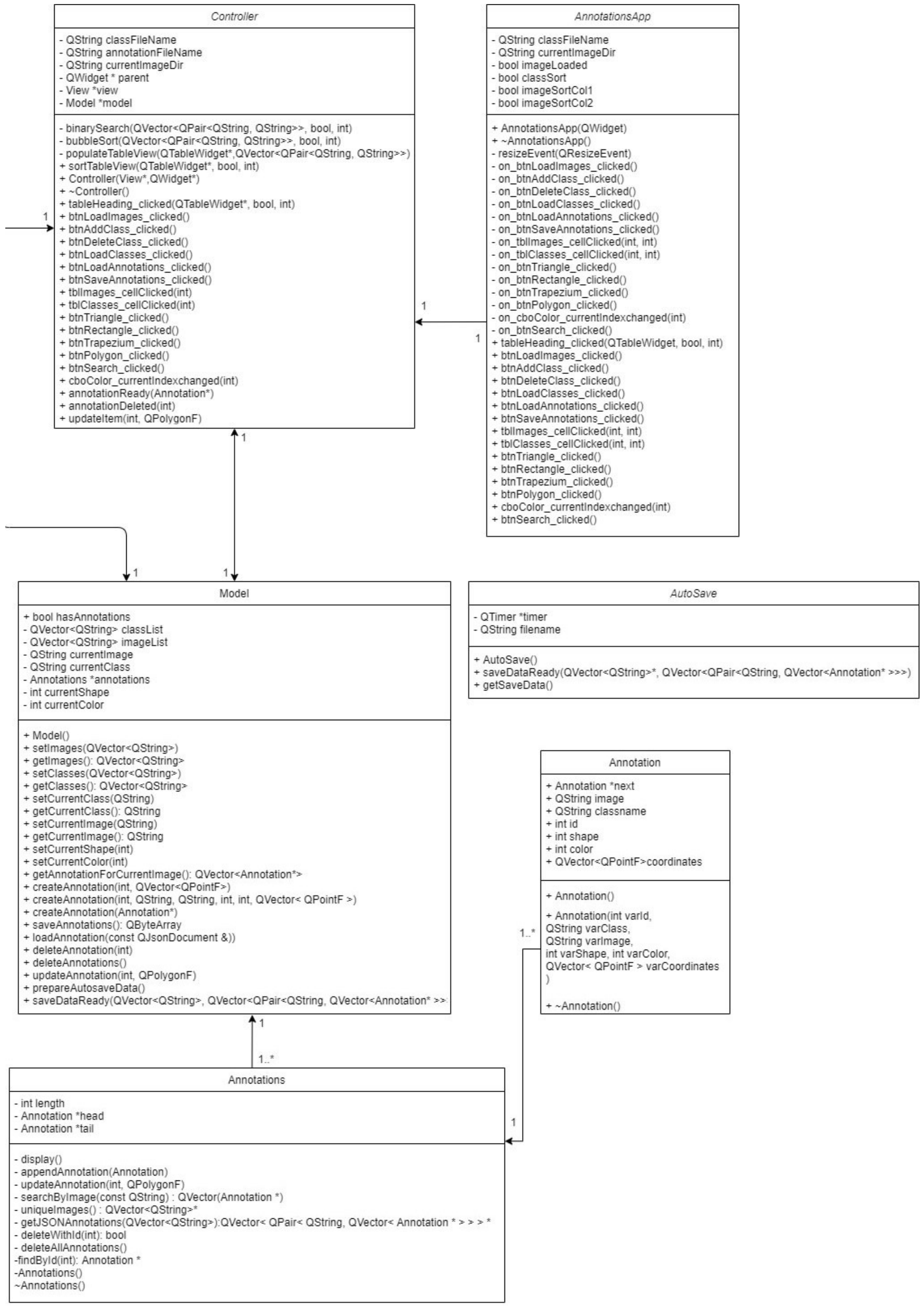

Controller
- QString classFileName
- QString annotationFileName
- QString currentImageDir
- QWidget * parent
- View *view
- Model *model
- binarySearch(QVector<QPair<QString, QString>>, bool, int)
- bubbleSort(QVector<QPair<QString, QString>>, bool, int)
- populateTableView(QTableWidget*,QVector<QPair<QString, QString>>)
+ sortTableView(QTableWidget*, bool, int)
+ Controller(View*,QWidget*)
+ ~Controller()
+ tableHeading_clicked(QTableWidget*, bool, int)
+ btnLoadImages_clicked()
+ btnAddClass_clicked()
+ btnDeleteClass_clicked()
+ btnLoadClasses_clicked()
+ btnLoadAnnotations_clicked()
+ btnSaveAnnotations_clicked()
+ tblImages_cellClicked(int)
+ tblClasses_cellClicked(int)
+ btnTriangle_clicked()
+ btnRectangle_clicked()
+ btnTrapezium_clicked()
+ btnPolygon_clicked()
+ btnSearch_clicked()
+ cboColor_currentIndexchanged(int)
+ annotationReady(Annotation*)
+ annotationDeleted(int)
+ updateItem(int, QPolygonF)
AnnotationsApp
- QString classFileName
- QString currentImageDir
- bool imageLoaded
- bool classSort
- bool imageSortCol1
- bool imageSortCol2
+ AnnotationsApp(QWidget)
+ ~AnnotationsApp()
- resizeEvent(QResizeEvent)
- on_btnLoadImages_clicked()
- on_btnAddClass_clicked()
- on_btnDeleteClass_clicked()
- on_btnLoadClasses_clicked()
- on_btnLoadAnnotations_clicked()
- on_btnSaveAnnotations_clicked()
- on_tblImages_cellClicked(int, int)
- on_tblClasses_cellClicked(int, int)
- on_btnTriangle_clicked()
- on_btnRectangle_clicked()
- on_btnTrapezium_clicked()
- on_btnPolygon_clicked()
- on_cboColor_currentIndexchanged(int)
- on_btnSearch_clicked()
+ tableHeading_clicked(QTableWidget, bool, int)
+ btnLoadImages_clicked()
+ btnAddClass_clicked()
+ btnDeleteClass_clicked()
+ btnLoadClasses_clicked()
+ btnLoadAnnotations_clicked()
+ btnSaveAnnotations_clicked()
+ tblImages_cellClicked(int, int)
+ tblClasses_cellClicked(int, int)
+ btnTriangle_clicked()
+ btnRectangle_clicked()
+ btnTrapezium_clicked()
+ btnPolygon_clicked()
+ cboColor_currentIndexchanged(int)
+ btnSearch_clicked()
1
1
1
1
1
1
1
Model
+ bool hasAnnotations
- QVector<QString> classList
- QVector<QString> imageList
- QString currentImage
- QString currentClass
- Annotations *annotations
- int currentShape
- int currentColor
+ Model()
+ setImages(QVector<QString>)
+ getImages(): QVector<QString>
+ setClasses(QVector<QString>)
+ getClasses(): QVector<QString>
+ setCurrentClass(QString)
+ getCurrentClass(): QString
+ setCurrentImage(QString)
+ getCurrentImage(): QString
+ setCurrentShape(int)
+ setCurrentColor(int)
+ getAnnotationForCurrentImage(): QVector<Annotation*>
+ createAnnotation(int, QVector<QPointF>)
+ createAnnotation(int, QString, QString, int, int, QVector< QPointF >)
+ createAnnotation(Annotation*)
+ saveAnnotations(): QByteArray
+ loadAnnotation(const QJsonDocument &))
+ deleteAnnotation(int)
+ deleteAnnotations()
+ updateAnnotation(int, QPolygonF)
+ prepareAutosaveData()
+ saveDataReady(QVector<QString>, QVector<QPair<QString, QVector<Annotation* >>
AutoSave
- QTimer *timer
- QString filename
+ AutoSave()
+ saveDataReady(QVector<QString>*, QVector<QPair<QString, QVector<Annotation* >>>)
+ getSaveData()
Annotation
+ Annotation *next
+ QString image
+ QString classname
+ int id
+ int shape
+ int color
+ QVector<QPointF>coordinates
+ Annotation()
+ Annotation(int varId,
QString varClass,
QString varImage,
int varShape, int varColor,
QVector< QPointF > varCoordinates
)
+ ~Annotation()
1..*
1
1..*
1
Annotations
- int length
- Annotation *head
- Annotation *tail
- display()
- appendAnnotation(Annotation)
- updateAnnotation(int, QPolygonF)
- searchByImage(const QString) : QVector(Annotation *)
- uniqueImages() : QVector<QString>*
- getJSONAnnotations(QVector<QString>):QVector< QPair< QString, QVector< Annotation * > > > *
- deleteWithId(int): bool
- deleteAllAnnotations()
-findById(int): Annotation *
-Annotations()
~Annotations()

Figure 14 - Right hand side of the Class Diagram

Explanation:

Figures 12 and 13 demonstrate the Class Diagram used in the development of the project. The Model-View-Controller (MVC) Architecture is being followed. The Controller is responsible for handling & processing inputs from the user, View is responsible for updating GUI elements on the screen, and Model stores all the data about the Image & Annotations.

Controller-View-Model are unique classes; hence they have one on one connection between them.

- 1 GraphicsPolygonItem can have many GripHandles & each GripHandle can have 1 GraphicsPolygonItem, so 1-1..* relation
- 1 GraphicsPolygonItem belongs to 1 GraphicsScene & 1 GraphicsScene can have multiple GraphicsPolygonItem, so 1-1..* relation
- 1 GraphicsScene can have 1 View & 1 View can have 1 GraphicsScene, so 1-1 relation.
- 1 Model can have many Annotations & 1 Annotations can have 1 Model, so 1-1..* relation.
- 1 Annotations can have many Annotation & each Annotation can belong to 1 Annotations, so 1-1 relation.
- 1 Controller can have 1 AnnitationsApp & 1 AnnotationsApp can have 1 Controller, so 1-1 relation.
- Autosave communicates with other objects but does not have a relationship since it operates within a separate thread.

Sequence Diagram

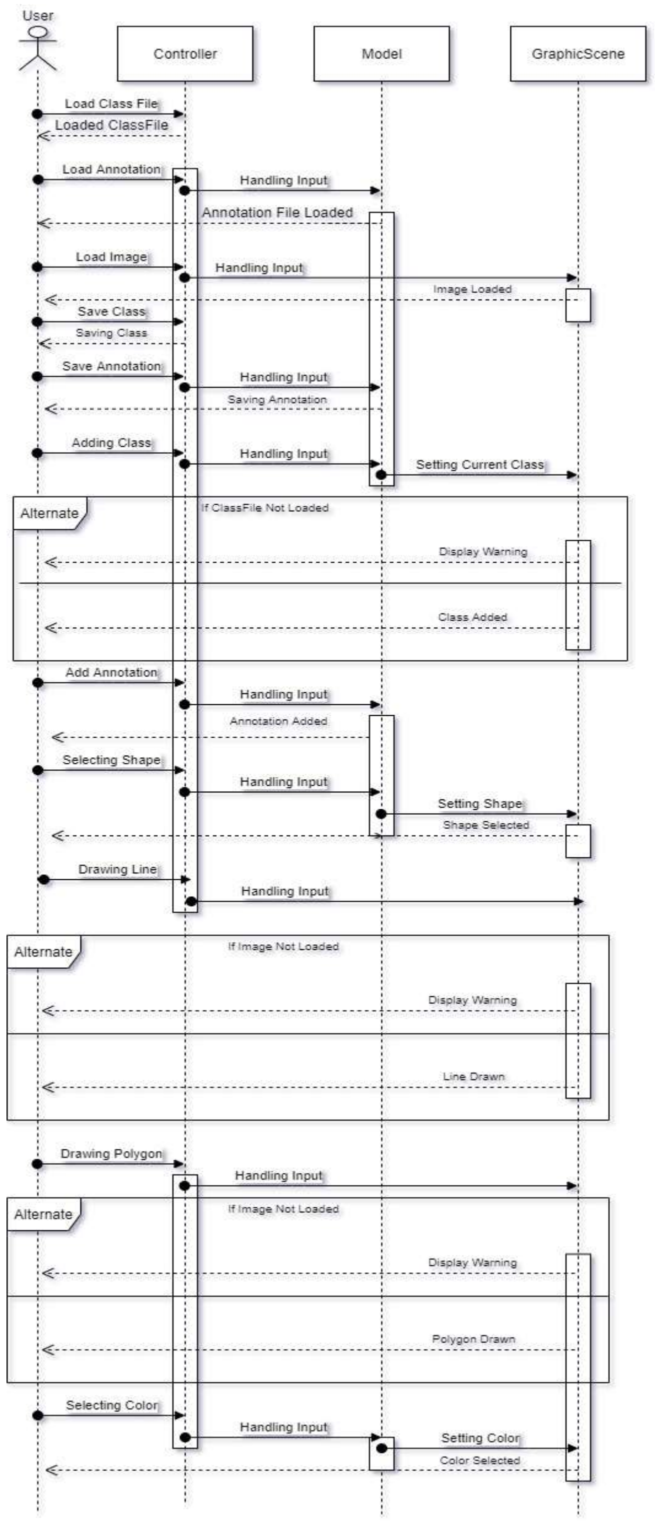


Figure 15 - Sequence diagram developed during the design phase of the project

Explanation

- User inputs Load Class File, Controller returns with Loaded Class File.
- User inputs Load Annotation File, Controller Handles the input, Model returns with Loaded Annotation File.
- User inputs Load Image, Controller Handle the input, GraphicsScene returns with Loaded Image.
- User inputs Save Class, Controller returns with saved class.
- User inputs Save Annotation, Controller Handles the Input, Model returns with saved Annotation.
- User inputs Add Class, Controller Handle the Input, if Class File is not loaded, GraphicsScene returns with a System Warning, otherwise returns with Class File Added
- User inputs Add Annotations, Controller handles the input, Model returns with Annotation Added.
- User inputs Select Shape, Controller handles the Input, GraphicsScene returns with Shape Selected.
- User inputs Draw Line, Controller handles the input, if an image has been loaded, GrahpicsScene returns with line drawn, otherwise returns with a system warning.
- User inputs Draw Polygon, Controller handles the input, if an image has been loaded, GrahpicsScene returns with polygon drawn, otherwise returns with a system warning.
- User inputs select Colour, Controller handles the input, Model, sets the Colour, GraphicsScene, returns with Colour Selected.

State-Machine Diagram

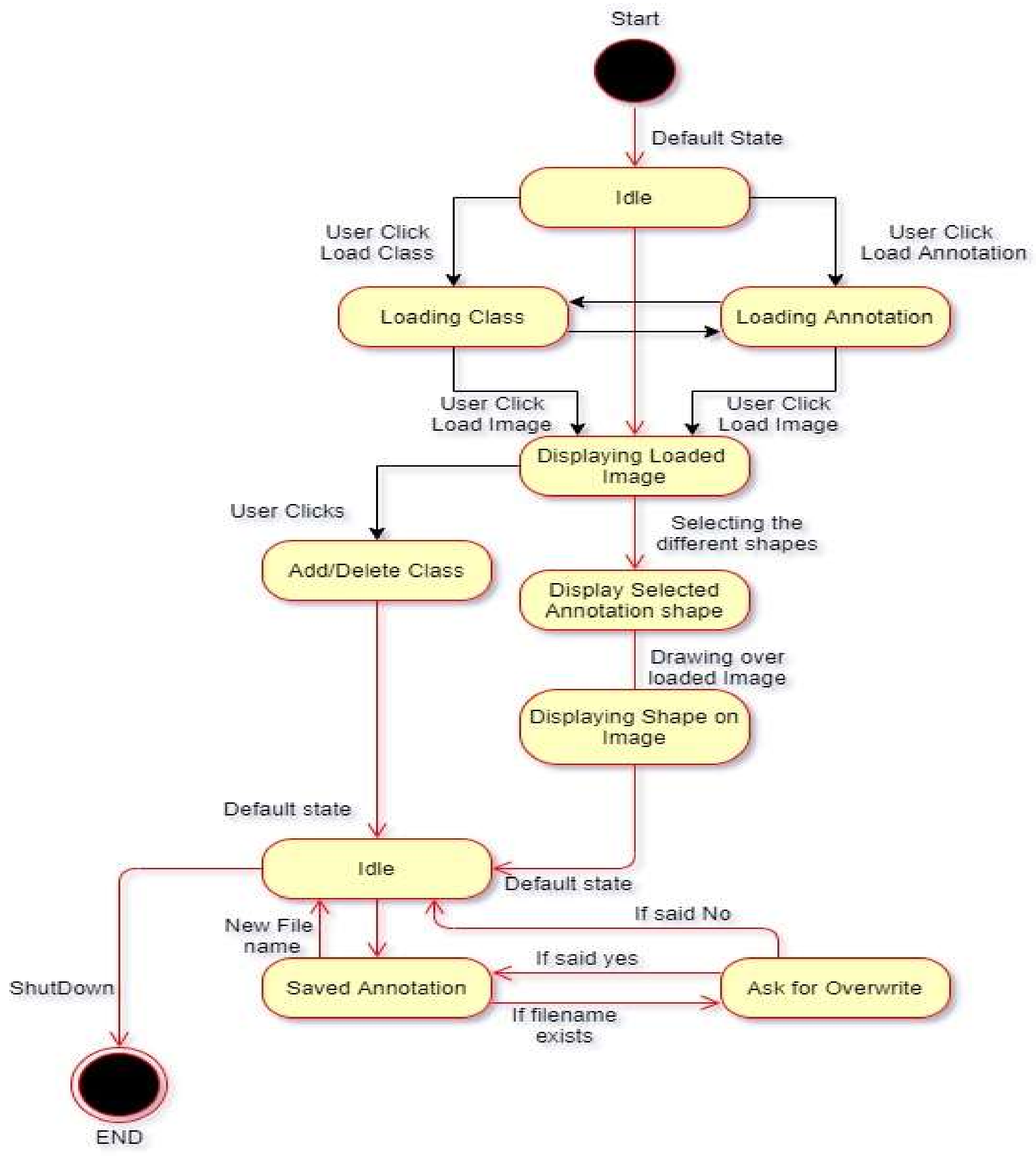


Figure 16 - Depicting State Machine Diagram

Explanation:

Idle is the default state of the application, Users can either load Class/Annotation/Image after starting the application. Annotation shapes can only be drawn after the user has opened an image; the system will then display the image with annotation shapes drawn, on the canvas defined. System will return to its default state, the user can save the annotation, if it's a new file name system will save it, and return to its idle state, otherwise ask for overwrite, depending on the user input, will either save or directly return to its idle state. The system will return to its end after the user clicks shutdown.

Component Diagram

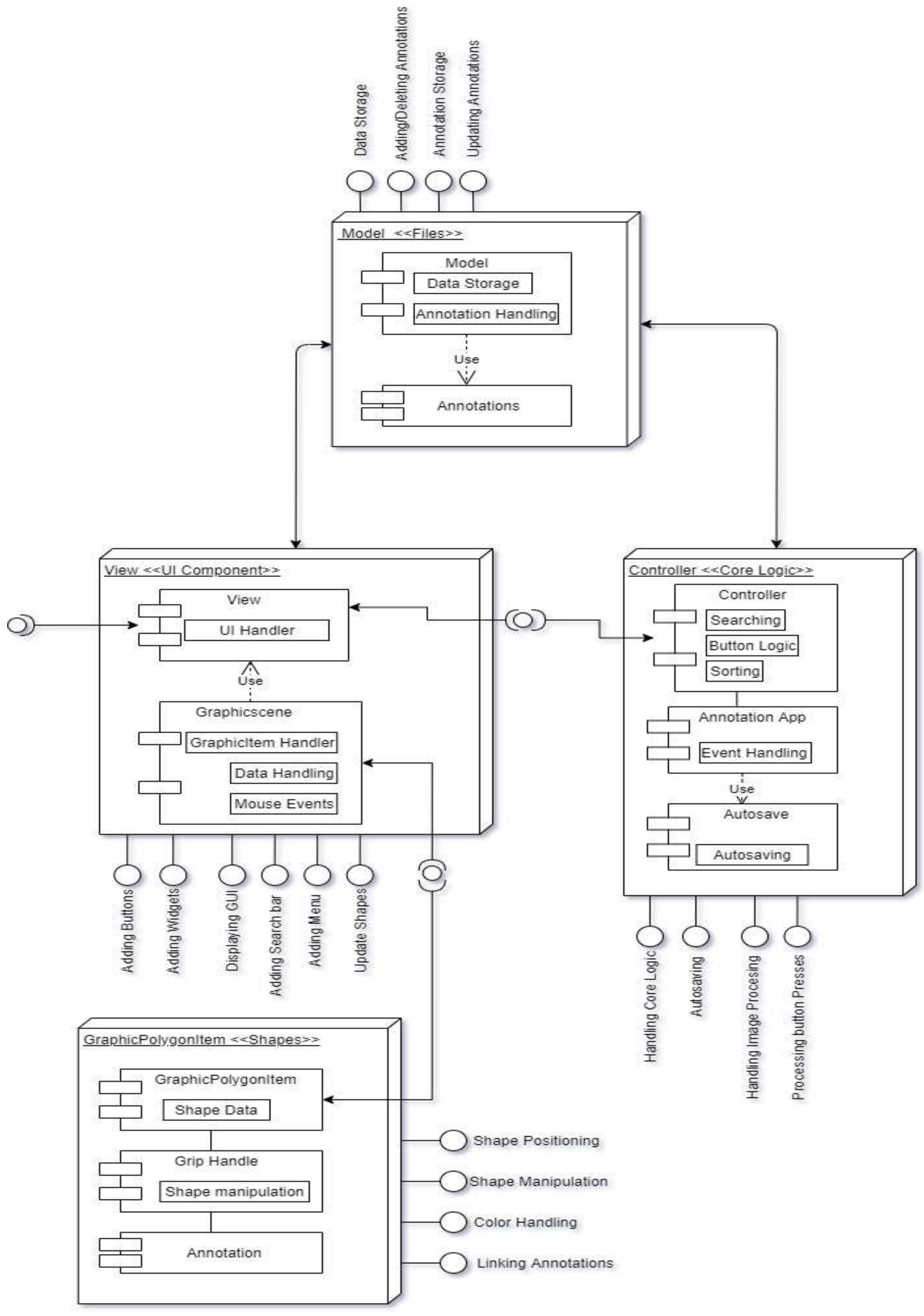


Figure 17 - Depicting Component Diagram

Explanation:

The System has 3 Major Components which exist within the AnnotationsApp application level class:

View: Consists of View which handles the UI, along with adding various buttons and widgets, and updating elements live on screen. This also contains and GraphicsScene which handles the main area for drawing (Graphics Scene / GraphicsPolygonItem / GripHandles) which allow the user to draw and adjust shapes.

Controller: Consists of controller, which acts as a facilitator between the View and Model classes and reacts to user interactions.

Model: Consists of model and Annotation, handles data storage, core application logic and annotation editing.

GUI Mockup

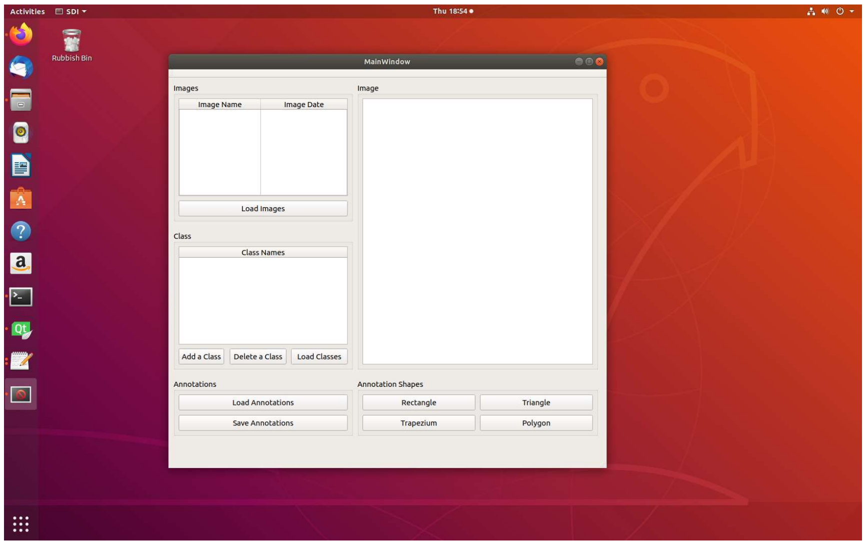


Figure 18- Depicting the original GUI Mockup

Explanation

In terms of design, the GUI uses field sets to group relevant controls so all controls relating to images are within one defined area of the screen. The same is true for Annotations, Classes and Shapes. The layout follows the logical order from top left that a user will take.

So initially they will need to load images to work from, then they will need to load classes, finally they will either load existing annotations or create new ones. It may be useful to grey out controls so a specific order can be made clear.

The Image and Class list boxes will use clickable headers to allow for sorting. This follows the default behaviour of similar controls on windows so this will be an action user will be familiar with. The mock-up is not to exact scale, where possible the image portion of the layout will take up as much space as possible since this is the main work area of the annotation tool.

Test Scenarios

The information in this chapter refers to the contents of the Testing Report and Test Plan, which can be found in the TESTING folder. All tables and images referenced in this section of the document refer to the contents of that document. A full demonstration of the software can also be found (Available online, https://youtu.be/9qrxwFJ4zW0, last accessed 26/04/2020) which shows all the functionality of the software.

Test-1:
Description: This test demonstrates the ability for a user to click on the Load Images button which opens an Open File dialog, populated with the correct image types as a filter.
Result: Success
Refer to: Table 1, Figure 1B

Test-2:
Description: This test demonstrates the application's ability to populate the Images List of the GUI with the images selected by the user.
Result: Success
Refer to: Table 2, Figure 2

Test-3:
Description: This test demonstrates the ability to sort the Images list according to image name, the images show a before and after scencario.
Result: Success
Refer to: Table 3, Figure 3

Test-4:
Description: Refactoring code to include a strict image filter rendered the need to warn users of incompatible images so this test was made redundant.
Result: N/A
Refer to: N/A

Test-5:
Description: The sort function (clicking on the header) should sort images files according to name
Result: Success
Refer to: Table 5, Figure 5a, 5b

Test-6:
Description: Demonstrate the ability to use the search facility to search the image list for a given item
Result: Success
Refer to: Available online, https://www.youtube.com/watch?v=ccOZ5M1tSTk , last accessed 26/04/2020

Test-7:
Description: Demonstrate the ability for the image list box to sort by image name ascending and descending order
Result: Success
Refer to: Table 7, Figure 7a, 7b also Available online, https://youtu.be/6O_J4nTYSWU , last accessed 26/04/2020

Test-8:
Description: Demonstrate the ability of the user to load a selected image into the canvas and display it
Result: Success
Refer to: Table 8, Figure 8

Test-9:
Description: Demonstrate the ability of the user to open Class Files using a system Open File dialog
Result: Success
Refer to: Table 9, Figure 9

Test-10:
Description: This test demonstrates the application's ability to populate the Class List of the GUI with the contents of the selected class file selected by the user.
Result: Success
Refer to: Table 10, Figure 10

Test-11:
Description: This test demonstrates the application's ability to populate the Class List of the GUI (and therefore the structure behind it) with the contents of the selected class file selected by the user.
Result: Success
Refer to: Table 11, Figure 11

Test-12:
Description: Demonstrate the preservation of the classes after deleting an item from the list
Result: Success
Refer to: Table 12, Figure 12a, 12b

Test-13:
Description: Demonstrate the ability of a user to add a new class to the class list
Result: Success
Refer to: Table 13, Figure 13a, 13b

Test-14:
Description: Demonstrate the ability of a user to delete an existing class from the class list
Result: Success
Refer to: Table 14, Figure 14

Test-15:
Description: Demonstrate the ability to sort the Class list by asc and desc order
Result: Success
Refer to: Table 15, image 15a, 15b, Available online, https://youtu.be/L8F68cMUsTs, last accessed 26/04/2020

Test-16:
Description: Demonstrate the ability for the user to draw various shapes on an image
Result: Success
Refer to: Table 16, Figure 16

Test-17:
Description: Demonstrate the ability to draw on and move without affecting the image
Result: Success
Refer to: Table 17, Figure 17a, 17b

Test-18:
Description: Demonstrate the ability to draw shapes which consist of an outline only
Result: Success
Refer to: Table 18, Figure 18

Test-19:
Description: Demonstrate the ability to resize a shape, showing that the functionality exists.
Result: Success
Refer to: Table 19, Available online, https://youtu.be/XyPjJ55ldsA, last accessed 26/04/2020

Test-20:
Description: Demonstrate the ability to adjust the vertices of an existing shape
Result: Success
Refer to: Table 20, Available online, https://youtu.be/XyPjJ55ldsA, last accessed 26/04/2020

Test-21:
Description: Demonstrate deleting the shape using the mouse and it being removed from memory
Result: Success
Refer to: Table 21, Figure 21a, 21b

Test-22:
Description: Demonstrate copying a shape via the context menu and show it being duplicated.
Result: Success
Refer to: Table 22, Figure 22a, 22b

Test-23:
Description: Demonstrate that the class name appears above each shape
Result: Success
Refer to: Table 23, Figure 23

Test-24:
Description: Display the file contents showing that it's formatted in the correct JSON format
Result: Success
Refer to: Table 24, Figure 29

Test-25:
Description: Demonstrate the use of an Open Dialog Box with the filter set to *.annotations.
Result: Success
Refer to: Table 25, Figure 25

Test-26:
Description: Demonstrate that the shapes exist above the images and that the shapes are associated with specific images.
Result: Success
Refer to: Table 26, Available online, https://youtu.be/qlzrTnaMncg, last accessed 26/04/2020

Test-27:
Description: Demonstrate the use of a Linked List in the application
Result: Success
Refer to: Table 27, Figure 27 and also demonstrated in the Unit Tests available in the SOURCE folder

Test-28:
Description: Demonstrate ability to save annotations
Result: Success
Refer to: Table 28, Figure 28 and also myfile.annotations in the TESTS folder

Test-29:
Description: Demonstrate that the system requires confirmation for overwrite of the annotations file
Result: Success
Refer to: Table 29, Figure 29

Test-30:
Description: Demonstrate that a user can rename the annotations file.
Result: Success
Refer to: Table 30, Figure 30

Test-31:
Description: JSON Format should match the requested format.
Result: Success
Refer to: Table 31, Figure 31, File is available in the myfile.annotations in the TESTS folder

Test-32:
Description: Demonstrate the auto save feature creating a new auto save file with the correct content.
Result: Success
Refer to: Table 32, Success - Available online, https://youtu.be/IwK7Jfci8Vk, last accessed 26/04/2020

Test-33:
Description: Any auto save must be done using threads.
Result: Success
Refer to: Table 33. This is only really demonstrable via code inspection. In the code, the Autosave class handles this. A sample autosave file is available in the TESTS folder.

Test-34:
Description: Any auto save must be done using threads.
Result: Success
Refer to: Table 34. Available online, https://youtu.be/QXvY3xZM2j0, last accessed 26/04/2020

Test-35:
Description: Demonstrate that the application can handle a large volume of images and shapes onscreen at once.
Result: Success
Refer to: Table 34. Available online, https://youtu.be/QXvY3xZM2j0, last accessed 26/04/2020

Unit, Integration, Functional Acceptance tests

Unit Tests

In terms of Unit Tests, the team developed a set of Boost framework tests which are available in the Source directory of this submission.

The tests are contained in a single file but are split into three separate test suites, one for the sort functionality, one for the search functionality and one to test the functionality of the linked list

Sort Functionality tests

In the test suite "SortTestSuite" there are several tests, each designed to test different aspects of the Bubblesort implementation used within the project. Due to the fact that it must sort both images and dates, it takes in a vector consisting of pairs of strings, the first of the pair is the image name, the second a string created from the file last modified date.

The initial set of data is an unsorted list of 5 images with various names and dates.

The tests compare this initial list to a manually sorted "expected results" list so that the test can determine if the results match what is expected of the function.

Initially the tests are basic tests to find out if the function handles a list with no content, moving on to checking that the first of the pairs can be sorted in ascending and descending value. Then the dates are checked to make sure they can be sorted in both directions.

All tests in this section pass.

Search Functionality Tests

In the test suite "SearchTestSuite" there are several tests, each designed to test different aspects of the Binary Search implementation. There is a limitation with binary search in that the search term is used for comparison against the midpoint of the vector, this works very well for numeric data or when the search term is a complete string but it does force the search to limit itself to the first part of the string if searching for a partial string. Therefore it is only possible to search correctly when searching for an entire string or a string that begins with the given search term.

The searches use the same initial data as the previous tests and check to ensure that the function does not return one of the values when passed a string that does not match.

Secondly the function is tested to ensure that when passed a substring of an existing image name, the correct image is returned. This is expanded upon by searching for a term which is featured in two of the image names (beach and beautiful). Searching for "bea" could return either but since a binary search depends on first being sorted, only the image beginning with "beach" should be and is returned.

Finally a search is completed, searching for the entire string as an edge case and a final check is made for an existing string with additional characters just to make sure that the presence of an existing string does not throw off the code.

All tests in this section pass.

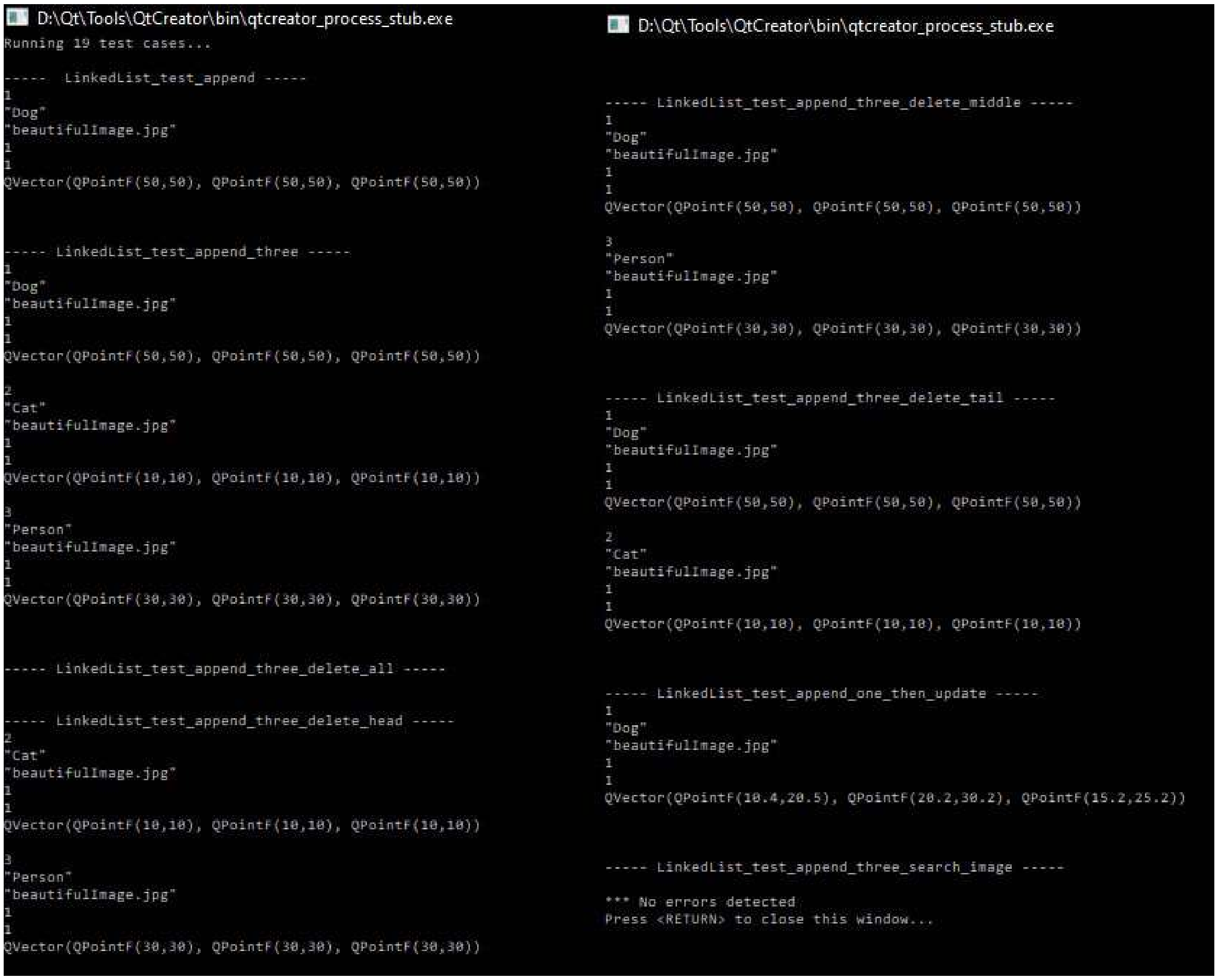
Linked List Tests

In the final test suite “LinkedListTestSuite” the tests are focused on the implementation of the Linked List within the program.

The tests in this section cover the following:

- Appending a single node to an empty list
- Appending several annotations to an empty list
- Deleting the head of an existing list
- Deleting the tail of an existing list
- Deleting one of the middle nodes in the linked list
- Deleting all nodes within an existing list
- Appending a node and updating the position values (emulating a user moving a shape)
- Searching within the linked list for a given image name and comparing the number of nodes returned to ensure that all matches are found.

Figure 19 displays the output from these tests, showing that each test within the 3 sections passes.



```
D:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Running 19 test cases...

-----  LinkedList_test_append -----
1
"Dog"
"beautifulImage.jpg"
1
1
QVector(QPointF(50,50), QPointF(50,50), QPointF(50,50))


----- LinkedList_test_append_three -----
1
"Dog"
"beautifulImage.jpg"
1
1
QVector(QPointF(50,50), QPointF(50,50), QPointF(50,50))

2
"Cat"
"beautifulImage.jpg"
1
1
QVector(QPointF(10,10), QPointF(10,10), QPointF(10,10))

3
"Person"
"beautifulImage.jpg"
1
1
QVector(QPointF(30,30), QPointF(30,30), QPointF(30,30))


----- LinkedList_test_append_three_delete_all -----

----- LinkedList_test_append_three_delete_head -----
2
"Cat"
"beautifulImage.jpg"
1
1
QVector(QPointF(10,10), QPointF(10,10), QPointF(10,10))

3
"Person"
"beautifulImage.jpg"
1
1
QVector(QPointF(30,30), QPointF(30,30), QPointF(30,30))
```

```
D:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe


----- LinkedList_test_append_three_delete_middle -----
1
"Dog"
"beautifulImage.jpg"
1
1
QVector(QPointF(50,50), QPointF(50,50), QPointF(50,50))

3
"Person"
"beautifulImage.jpg"
1
1
QVector(QPointF(30,30), QPointF(30,30), QPointF(30,30))


----- LinkedList_test_append_three_delete_tail -----
1
"Dog"
"beautifulImage.jpg"
1
1
QVector(QPointF(50,50), QPointF(50,50), QPointF(50,50))

2
"Cat"
"beautifulImage.jpg"
1
1
QVector(QPointF(10,10), QPointF(10,10), QPointF(10,10))


----- LinkedList_test_append_one_then_update -----
1
"Dog"
"beautifulImage.jpg"
1
1
QVector(QPointF(10.4,20.5), QPointF(20.2,30.2), QPointF(15.2,25.2))


----- LinkedList_test_append_three_search_image -----

*** No errors detected
Press <RETURN> to close this window...
```

Figure 19 - a display of the results from the unit testing suites

Integration Tests

This project combines the sort functionality and the search functionality, since a Binary Search must be sorted before it can function. This functionality is tested for integration during the unit tests discussed in the previous section and the results of the tests can be seen in Figure 19.

There is no direct integration between these components and the linked list within this program.

Acceptance Tests

The testing structure used in developing the test plan covers all the requirements for the application, which are tested and documented in the Testing Report and Test Plan, which can be found in the TESTING folder. These, alongside the demonstration video(Available online, https://youtu.be/9qrxwFJ4zW0, last accessed 26/04/2020), demonstrate that the application achieves the goals it was designed to meet

Stress Testing

The system was put through a stress test, while running several resource heavy applications like Adobe Premiere Pro, OBS (streaming software and video recording), Fireworks, Word etc. 1267 images were loaded and approximately 960 annotations were created by repeatedly saving and appending saved annotations to current annotations. The results can be seen in the testing documentation and the system displays no real slowdown during this process.

Results

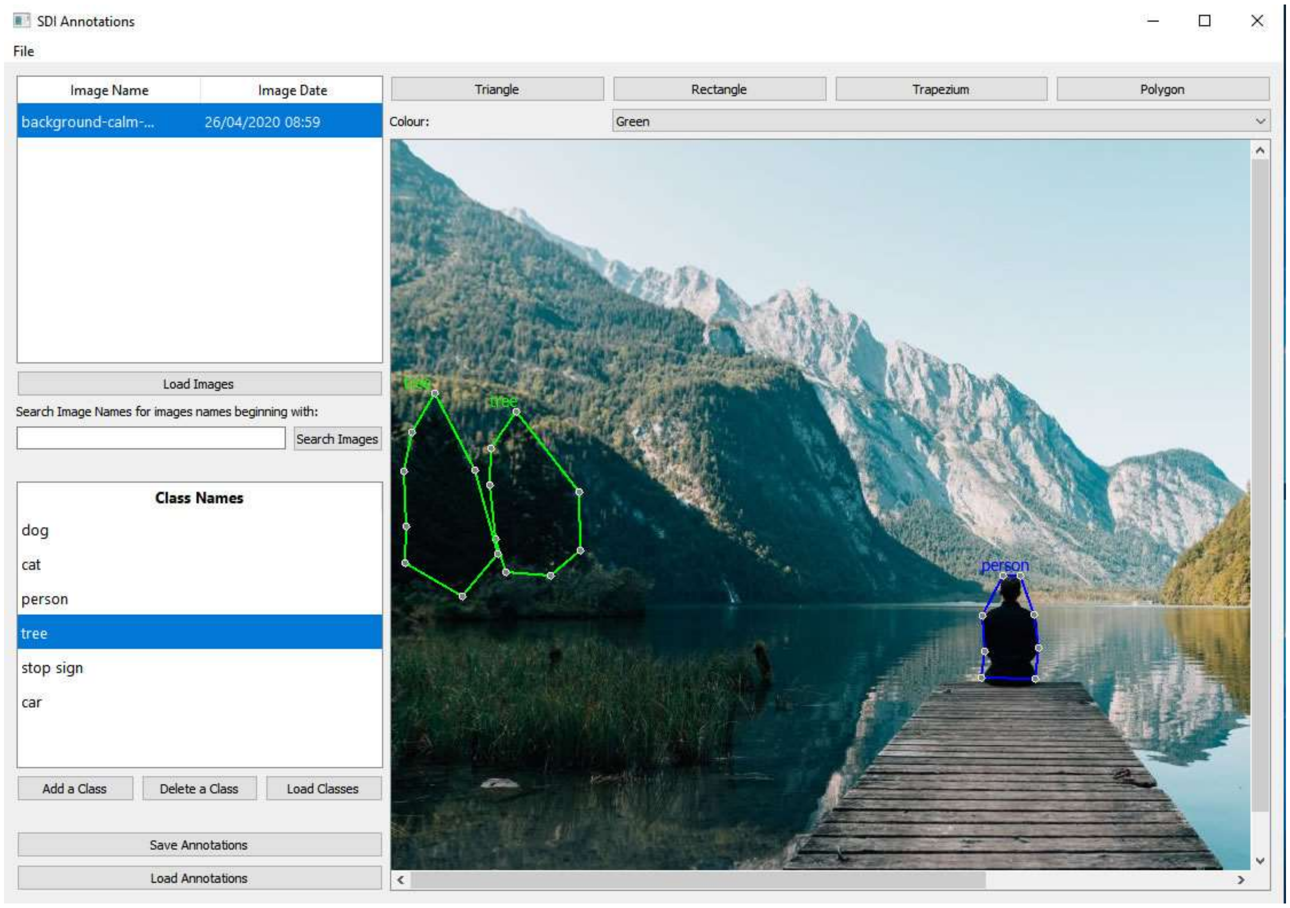


Figure 20 - Image of the final software

Figure 20 shows the final version of the software in use. The image view area resizes when the application does and allows for scrolling so even with a small (800x600) screen, the annotation process works well.

Users can chose between different images, different classes and colours when drawing annotations. As an extra feature the user can even change the colour of an existing annotation to help with dark colours on light areas.

Figure 21 shows the application maximised with some variation in the choice of annotation, showing several different classes, colours and shapes used to good effect to identify parts of the picture which might be of interest. The process of drawing the annotations is smooth and intuitive and feels like the basis for a great piece of software.

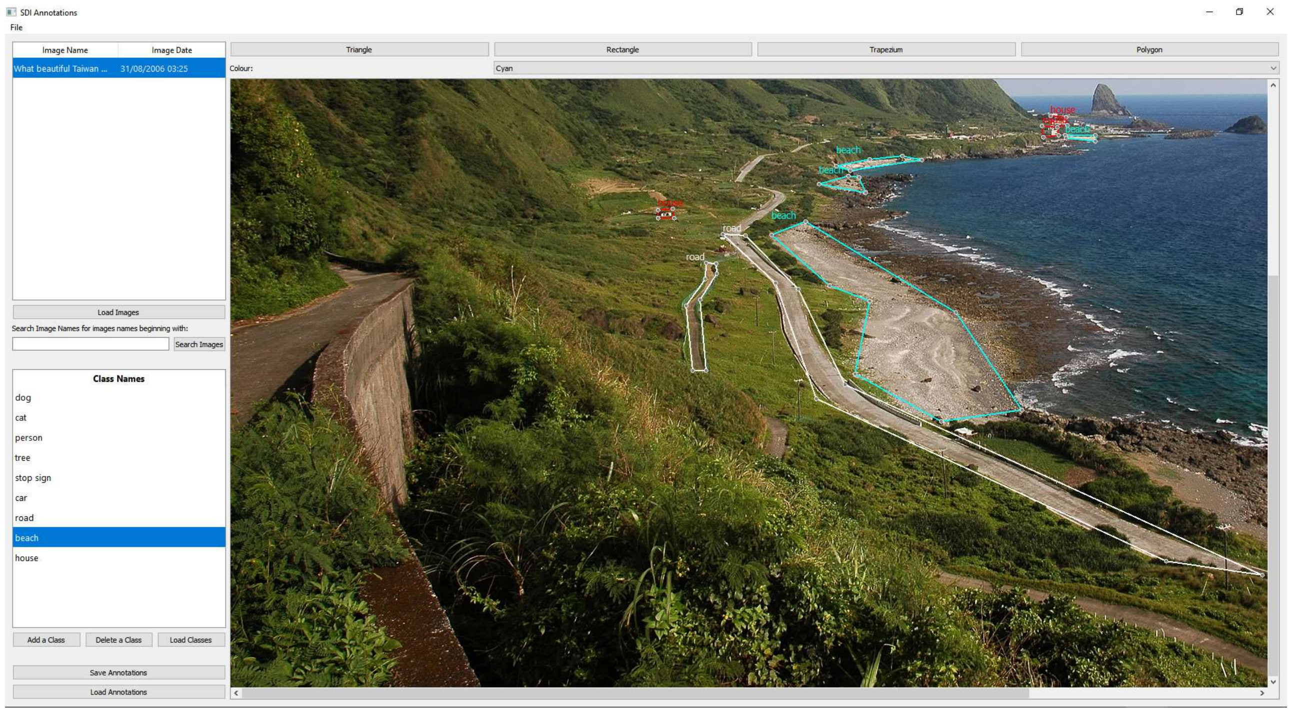


Figure 21 – Full size image and detailed annotations

Testing

The testing for this project was severely hampered by the tester not implementing a test plan early on. With a solid test plan in place the team could have created suitable unit tests to run on Jenkins and removed a lot of the eventual manual testing required.

Despite this, the team has produced testing to cover all of the requirements, unit tests and stress tests on the system which, while not optimal, prove a good level of reliability and usability in the system.

Portability

Ideally, the application would be available as a standalone executable with all related dll's and files incorporated into a single file, unfortunately Qt requires a full licence version of the software to be able to do this. While this makes it possible, the costs were prohibitive in the case of this project.

The source code is developed in Qt which allows for compilation using several compilers, giving the software cross-platform reach. This must be handled by the end user since their choice of compiler and machine environment is beyond the ability of the team to predict. However the compilation of this application would be identical to many open source software currently available and therefore would not be beyond the capability of users familiar with the process. To make the process simple, a thorough guide could be produced to walk users through the steps needed to complete this process.

This software has been tested on Windows 10 and Ubuntu and functions well on both.

Efficiency

The main bottleneck in the system occurs when saving Annotations. This is because the Qt JSON system operates in several for loops, creating a fairly intensive point. However, under stress testing with over 1000 images and many hundreds of shapes onscreen at once, there was no obvious slowdown on the system.

By definition (University of Wisconsin, 2018), the for loops will run in $O(n^2)$.

The outer loop will run n times where n is the number of unique images. This gives a value of $O(n)$ for the outer loop.

The inner loop will run m times where m is the number of annotations. This gives a complexity of $O(m)$ for the inner loop.

Since we don't know which is bigger, this gives us an overall value of $O(n) * O(m)$ which is equivalent to $O(n+m)$ or $O(\max(n,m))$.

However we have a third loop which handles the coordinate values inside each annotation which can consist of 1 to 8 vertices. If we factor this in we get a best case scenario of $O(1)$ and a worst case scenario of $O(8)$.

When combined with the other loops this results in a range of between $O(n+m+1)$ and $O(n+m+8)$.

Security

Security does not feature highly in this application. None of the data stored is personalised or consists of sensitive data, removing the need to keep such data private.

Reliability

Several tests were executed involving leaving the software running for long periods of time (often overnight or for 24 hours straight) while working on other things or using the computer for other tasks and no noticeable negative side effects were witnessed.

Maintainability

The code for this project has been developed with readability in mind and the classes used (factoring in the close relationship between the graphics items within Qt itself) are, for the main part, self-contained and limited to one job (high Cohesion). For example the Autosave class simply handles auto save functionality. The GraphicsPolygonItem, Griphandle and GraphicsScene (while interconnected due to their nature) communicate using signals which allows for a low level of coupling.

Scalability

In terms of scalability, several tests were performed, opening over 1200 images simultaneously and copying as many annotations as could be reasonably managed (approximately 960) and no system slowdown was noticed. This leads the team to believe that the application is already fairly scalable, although there may potentially be room within the code to increase efficiency and increase the level of scalability.

Complexity

Sort efficiency of the implemented Bubble Sort functionality

This is a simple sort functionality which takes a list as its source and moves through the list comparing adjacent elements. If the left element is greater than the right, they are swapped over and the next two elements are checked. This is repeated until the whole list is sorted.

The best scenario for Bubble Sort is when the array is already sorted. This would then give a complexity of O(n) where n is the number of elements.

The worst scenario would be where the vector is sorted but in the opposite order. So if we were to sore a vector which was already in descending order into ascending order. This would give a result of O being n + (n+1) + (n+2) and so on. This simplifies to O = (n*(n+1)/2) which is equivalent to O(n^2)

Search efficiency of the implemented Binary Search functionality

This search algorithm works on a sorted list and takes the middle element of the list as a reference point. If this middle point is the search term then we have found what we are looking for and can stop. This would give a complexity of O(1).

If the search term is greater than the middle element, the first half of the list can be discarded, otherwise the second half of the list can be discarded. Then the process repeats again. The midpoint is found and the comparison begins again. This is repeated until the search term is found.

In mathematical terms this equates to asking how many times we need to half the number of elements in the list, which is represented by n, until we get 1. This results in n being halved each step of the process, resulting in O(log n).

Conclusions & Future Work

Despite the issues faced by the team, the end product is of good quality and meets almost all of the core requirements with the exception of HDF5 format files. The decision to change from this to JSON was taken after spending time researching the format but never being able to use it in any meaningful way. A point was reached where the Project Manager decided to stop working on this format and move to one which we knew we could complete.

Aside from the HDF5 format the team were able to complete the other features and added a few extra features to add value to the application. These include being able to change the colours of the annotation shapes so that users can swap to a brighter colour on a dark background or a darker colour on a light background. This could be reprised for a future version of the software.

In terms of testing, there could be improvements. The strategy the team employed was non-existent mainly due to the Software Tester repeatedly being asked to create one and not doing so, the end result being that the Project Manager had to come up with a set of tests and implement those with the Software Architect. This ended up being a last minute affair and could, with a little effort earlier in the project have resulted in a much more robust set of testing.

It would be interesting to see if it would be possible to expand upon the current solution to see if the accessibility issues could be handled in a better way. With the precision needed to draw shapes, keyboard access is not adequate to allow someone who is unable to use a mouse to really use the application. Perhaps there are alternatives such as joystick devices, drawing surfaces, or equipment featuring head-mounted cameras which could be utilised to make the application more accessible.

While developing the software the team became aware of the issue with contrasting colours. Some of the pen colours (blue for example) are too dark to be used on dark areas of images. The application offers users the opportunity to change or update the colours used for shapes but perhaps it would be useful to some users who have colour blindness or sight issues a high contrast mode. This could scan the pixels surrounding the vectors as they are placed or adjusted and dynamically change the pen colour so that a specific contrast ratio is kept at all times. This could be paired with a zoom functionality to allow users greater control over the view of the activity.

The application could also be expanded to utilise hotkeys to bind certain functions, making it easier to use. This functionality was attempted using QShortcut within Qt but while the action happened, the slots connecting the shortcut to the eventual event handler never fired. The team did not have time to research this further and had to abandon the functionality.

The team also felt that it would be good to have a colour picker dialog, instead of having fixed colour choices, although the team did eventually implement all the supported colours within Qt.

The GUI colour scheme could be improved upon, Qt has QSS (Qt Style Sheets) which allow developers to theme certain parts of the GUI. The application already makes use of these to a lesser extent in retaining the highlighted option in the Image and Class lists when the controls lose focus. This could be expanded upon to create themes.

Reference List

Company, T.Q., [2019?]. *Why Qt - Get started today* [online]. Available at: https://www.qt.io/why-qt [Accessed Apr 25, 2020].

D. Gavrilov, A. Melerzanov, N. Schelkunov and A. Gorodilov, "Artificial Intelligence Image Recognition Inhealthcare," *2018 International Conference on Artificial Intelligence Applications and Innovations (IC-AIAI)*, Nicosia, Cyprus, 2018, pp. 24-26.

Doxygen, 2019. *Generate documentation from source code.* [online]. Available at: http://www.doxygen.nl/index.html [Accessed 26/04/ 2020].

draw.io, [2019?] *Diagrams For Everyone, Everywhere.* [online]. Available at: https://drawio-app.com/ [Accessed Jan 26, 2020].

HDF5group, 2019. The HDF5 Library & File Format. [online]. Available at: https://www.hdfgroup.org/solutions/hdf5/ [Accessed Feb 23, 2020].

Jenkins, [2019?]. *Jenkins User Documentation*. [online]. Available at: https://www.jenkins.io/doc/index.html [Accessed March 13, 2020].

Leenhardt, R., et al., 2020. *CAD-CAP: a 25,000-image database serving the development of artificial intelligence for capsule endoscopy*. Endoscopy International Open, 8 (3), E415-E420.

Microsoft, [2019?]. *Autonomous Vehicle Solutions.* [online]. Available at: https://www.microsoft.com/en-gb/industry/automotive/autonomous-vehicle-deployment [Accessed Apr 26, 2020].

Nichols, J.A., Herbert Cchan, H.,W. and Baker, M.A.B., 2019. *Machine learning: applications of artificial intelligence to imaging and diagnosis*. Biophysical Reviews, 11 (1), 111.

OpenCV, [2019?]. *About OpenCV*. [online]. Available at: https://opencv.org/about/ [Accessed 26/04/ 2020].

Telegraph, 2019. Cambridge start-up Wayve to become one of Britain's best-funded driverless car ventures in $20m deal. Telegraph.co.uk.

University of Wisconsin, [2018?]. *Complexity and Big-O Notation*. [online]. Available at: http://pages.cs.wisc.edu/~vernon/cs367/notes/3.COMPLEXITY.html [Accessed Apr 26, 2020].

Waymo, [2019?]. *Driverless Car Technology*. [online]. Available at: https://waymo.com/ [Accessed Apr 26, 2020].

Overall Reflection

The PM became aware of issues within the group regarding the use of Slack. 50% of the group wanted to use Discord and comments on this continued well into March. This "dislike" of the choice of communication tools caused severe issues in managing the project - 50% of the team would simply not be reachable on Slack for weeks on end. When they did come online it was only for a few minutes so it was extremely difficult to get any decisions from the group as a whole. The PM had to spend a lot of wasted time sending emails and reminders that were never replied to.

The post count for each member on Slack varies wildly, showing very clearly the vastly different engagement and activity from each member.

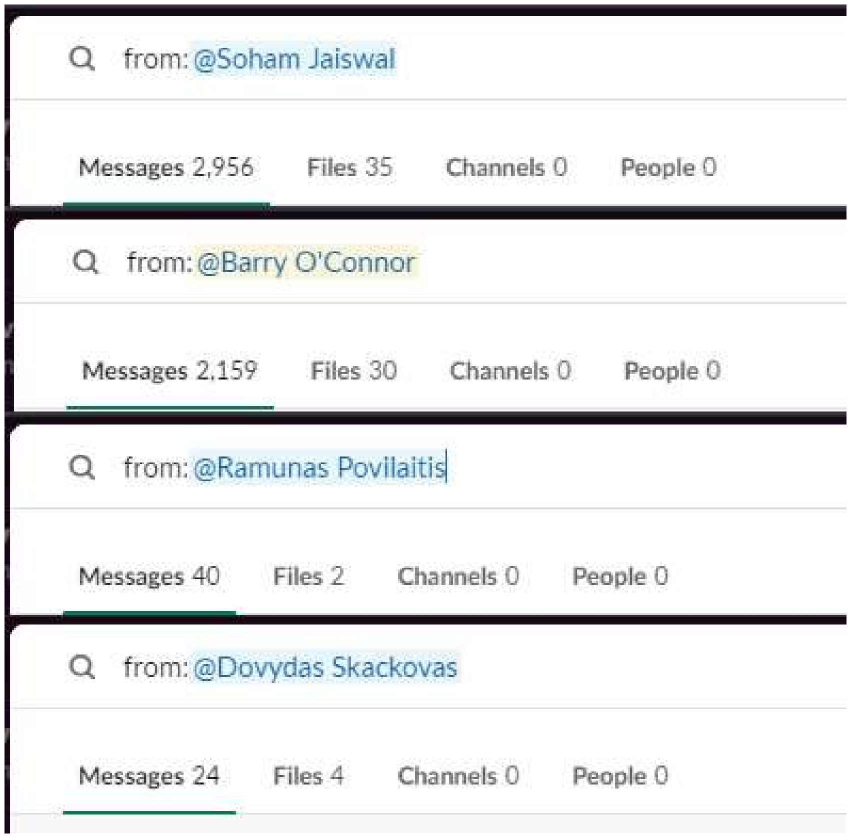

from: @Soham Jaiswal
Messages 2,956 Files 35 Channels 0 People 0
from: @Barry O'Connor
Messages 2,159 Files 30 Channels 0 People 0
from: @Ramunas Povilaitis
Messages 40 Files 2 Channels 0 People 0
from: @Dovydas Skackovas
Messages 24 Files 4 Channels 0 People 0

Figure 22 - screenshot of the various Slack message counts

Ramunas from the very start had a habit of disappearing and being uncontactable for periods of 4-6 weeks. None of the team were able to contact him during these times and it very quickly caused major issues for the group. Ramunas was the team Software Developer and tasked with researching OpenCV and coming up with an implementation of that so that the team could concentrate on other areas. It is unknown whether he did the research because around the time of his submission, he simply stopped communicating completely. All attempts to contact him failed (on Slack, via email, Facebook, Discord).

This halted progress among the group entirely and the project manager had to step in and take over the role of Software Developer at this point to allow the team the ability to keep moving forward. This involved all of the tasks associated with software developer - including management of the entire codebase, GitHub, decisions regarding code direction, coding style and so on. As of writing, nobody within the group has heard anything more from Ramunas.

It should be mentioned that Soham also took over some of the research and work originally meant for Ramunas, allowing the group to progress despite the setbacks encountered. Despite having his own deadlines and commitments, he accepted any extra work he was asked to do and remained active and supportive in general. The project Gantt chart clearly shows how much extra work the PM and SA took on board.

It should also be mentioned that Dovydas wasn't exactly an active or productive member. His contributions were few and tended to be of as little effort as he could manage.

When Ramunas became Tester, the PM inherited the role and had to take over the documentation for this. The document at this point (a few days before the D4 deliverable) was little more than a few headers in a document. He had been repeatedly asked to go through the requirements list and come up with some basic tests that we could do as a group to act as acceptance tests. In the end, the SA and PM had to complete the entirety of the testing task.

In short, Dovydas has contributed around 2 or 3 hours' worth of work to the project. Even without the issues forcing additional work upon people, this pales in comparison to others in the group.

With hindsight, perhaps a different approach was needed with this group. A firmer, less lenient PM may have forced a better quality of work and better engagement out of Dovydas and Ramunas.

In the end it feels like an opportunity missed. The team had 2 very able developers in Barry and Ramunas and a wide ranging skill set. This should have been a fun, enjoyable project to work on with it being in an area very few of us had worked on. With a little effort from all 4 members, the resultant software could have been excellent.