System Design and Implementation
Labelling Application

Development Report

Jordan Rudge(N0833118)
Harry Dale(N0803132)
Liam Mullins(N0801237)
Tanmay Poddar(T0093879)

Tutor: David Adama

1 - Abstract

This report covers the planning, designing and development of our Software Design and implementation application in which we endeavoured to create an object classification/annotation system that aims at mimicking a Convolutional Neural Network (CNN).
We used an agile methodology during development due to the small group size and the variety of roles we each had. It allowed us to have an autonomy of the tasks contained within those roles, while also allowing contribution of other members.
After many hurdles and issues including a non compliant group member and the Covid-19 pandemic, we believe we have achieved our goal to create the software with room for additional work in the future to make the application a fully fledged system.

Our software tester Tanmay did not contribute to any aspect of this project for the entirety of this project and is the reason for lack of testing within our project.

2 - Revision History

2.1 Application

Version	Issue Date	Stage	Changes	Author/s
1.0	7/2/2020	Early stages of development	Files are created for the application	Harry dale Jordan Rudge Liam Mullins
1.1	21/2/2020	Early stages of development	Main Gui is created	Harry dale Jordan Rudge Liam Mullins
1.2	10/3/2020	Alpha	Images can be imported, sorted and displayed	Harry dale Jordan Rudge
1.3	28/3/2020	Alpha	Classes can be made/ imported and sorted	Harry dale
1.4	10/4/2020	Beta	Shapes can be drawn to the canvas to then be moved	Harry dale Jordan Rudge
1.5	24/4/2020	Beta	Classes chosen are attached to the boxes when drawn	Harry dale Jordan Rudge
1.6	29/4/2020	Late stages of development	Boxes can be saved using JSON files	Liam Mullins
1.7	2/5/2020	Final update	Help section included	Jordan Rudge

2.2 Report

Version	Issue Date	Stage	Changes	Author/s
1.0	11/3/2020	Early stages	Contents page linked. plan started	Jordan Rudge Harry Dale Liam Mullins
1.1	16/3/2020	Early stages	Plan added	Jordan Rudge
1.2	21/3/2020	Planning stage	Implementation details started	Jordan Rudge Harry Dale Liam Mullins
1.3	26/3/2020	Planning stage	Background research Requirements Risk analysis Task list time plan	Jordan Rudge Harry Dale Liam Mullins
1.4	4/4/2020	Planning stage	Monitoring tools Coding standards Code contribution guide	Harry dale
1.5	9/4/2020	Designing Stage	Adopted coding standards Use case diagrams State diagrams	Jordan Rudge Harry Dale Liam Mullins
1.6	14/4/2020	Designing Stage	Sequence diagrams Class diagrams Component diagrams	Jordan Rudge Harry Dale Liam Mullins
1.7	19/4/2020	Implementation stage	Results added	Liam Mullins

1.8	24/4/2020	Implementation stage	Conclusions and individual contributions added	Jordan Rudge Harry Dale Liam Mullins
1.9	29/4/2020	Final stages	Individual reflections References added	Jordan Rudge Harry Dale Liam Mullins
2.0	3/5/2020	Final stage	Appendix appended	Jordan Rudge Harry Dale Liam Mullins

3 - Contents

4 - List of Tables

5 - List of Images

6 - Introduction

Convolutional Neural Networks (CNN) are a continuing and flourishing computer science topic, which work hand in hand with other machine learning strategies. Many different programs can be made from using this approach and the world is constantly evolving to make simple jobs faster and more efficient. In today's society it is evident that computers are becoming much smarter as time goes. This means that applications that can help the general public can be made to make their lives much easier, with an application such as this it would mean an CNN would be able to recognise and find exactly what a user is looking for by cross referencing images they have been given with a wide array of other learnt images.

We were tasked with creating an application that would be used in training a CNN to allow it to automatically recognise real world objects and be able to label them. Our application would allow the user to load many images into a canvas and load a series of class names. Once loaded the user would be able to select a class and draw squares onto the image with the associated class names as annotations identifying aspects of the image. The user would then be able to save and load the squares size and location and attached annotation for later use in a real CNN program.

This report covers our journey in creating this application, from planning, design, development to the results and conclusion.

7 - Plan

7.1 - Background Research

Visual Studio
We are using Visual Studio as our IDE as we have all learnt to code C++ on it. In addition it has various features including live assistance and quick debugging. The community edition is also free so we are able to work independently at home/away from University.

Visio
We are using Visio to produce a mockup design of the GUI, due to its versatility and variety of shapes, making it very easy to realise designs.

Github
We are using Github as a repository for our code, which will allow us to easily coordinate and contribute individually or as a team towards each new subsequent commits to the application. It also features a convenient system to highlight issues with projects that need addressing, along with their importance and a timeframe they need to be sorted by.

draw.io
During planning and design we have been using GDrive to easily and conveniently access a shared space to upload and continually update our work. Therefore it was handy for us to use draw.io, as it is built into GDrive. It also has many of the features provided by other priced applications, while draw.io is free.

Qt
We are using Qt to create our GUI for our application, as it is one of the most highly rated GUI libraries, with full range of functionality for our needs including buttons, canvases, and file panes. Our software architect also has prior experience working with Qt, so along with its features, we thought it would be most appropriate. We also plan to use QGraphicsScene and QGraphicsItem classes for use in drawing and moving shapes.

Doxygen
We are using Doxygen as it is a quick and professional method of producing documentation for our code. It does this by compiling comments from code into a format of our choice.

JSON
We are using JSON as it is a great alternative to databases when using large and complex datasets. We can use it to manage our data in “groups” and “datasets”.

7.2 - Requirements - Table 1

Requirement Number	Description	Implications	Tasks
File Handling			
1	The application must be able to read common image file extensions (.JPEG . PNG)	Program will detect selected image files and be able to open and display said files.	Create GUI to display images and involve file handling to read
2	User can open and load annotation files	Program will load and display user selected file	Integrate file navigation and handling and display files within GUI.
3	Save annotation files.	Program will save changes from the user	Integrate file handling and save button in GUI.
4	When saving a warning appears if a file exists. Option to overwrite.	Program will give a dialogue box for warning	Produce an error handler for duplicate file name event to help inform the user
5	Users can change the name of the annotation file.	Display dialogue box to allow user input new name for system	Produce a system for renaming a file
6	Store image file name in annotation file.	Program will locate, open and add image file name to annotation file	Integrate file navigation, editing and handling.
7	Store number of shapes per image.	Program will locate, open and add number of shapes per image to annotation file	
8	Store shape type.	Program will locate, open and add the shape type to the annotation file.	
9	Store coordinates of shape.	Program will locate, open and add coordinates of shape to the annotation.	

Search/Sort/Data Structures			
1	Compatible images in the folder must be able to be sorted by ascending and descending file name.	Program will sort files and display then in alphabetical order using “the chosen sort”	Select and implement an appropriate sorting algorithm.
2	Compatible images in the folder must be able to be sorted by ascending and descending file date.	Program will sort files by date and time from ascending and descending.	
3	Classes selector with browse button to navigate and select class files .	Program will detect classes file and be able to navigate to desired files and open	Integrate file navigation and handling
4	All classes should have the possibility of ordering in ascending or descending inside a class pane.	Program will sort classes using chosen sort alphabetically	Integrate one of chosen sort
5	Classes file line number must be kept the same.	Line numbers won’t be changed to allow programs to use it for sorting.	Line number preserved for use with chosen sorting algorithm.
6	Users will be able to add and remove classes and append in the class file.	Program will have option to add or remove classes via GUI	Integrate file navigation and handling
7	Use of binary trees for data handling	Program will utilise binary trees for all data handling needs	Create binary trees to ensure data is controlled within the project
8	The annotations must follow the hierarchical data format HDF5 or JSON.	The program must use HDF5/JSON file formatting to annotate images	Implement an annotation system using HDF5 JSON.
9	Storing data must be completed using the data structures from 1st term.	The program will use a data structure to be able to store and retrieve files and images.	Select and implement an appropriate data structure.
10	Must use sort and	The program will	Select and implement

	search algorithm implemented in 1st terms labs	implement a search and sorting algorithm learnt in 1st term studies.	an appropriate searching and sorting algorithm.
Graphical Libraries/frameworks			
1	Must have a simple GUI	Use GUI to allow the user to view images and draw shapes on images. Also allowing for file navigation, creation and deletion	Research and implement a suitable library to allow creation of a GUI
2	Select appropriate of graphical library	Program will use premade libraries for its creation to allow user ease of use and clear feedback	Select and implement an appropriate graphical library
3	Users must be able to draw one of shapes detailed in the specification: Rectangle.	Program will be able to draw detailed shapes plus the option to edit the location of individual vertices of the polygon.	GUI to display image and then drawing shapes option. Shapes drawn superimposed on image
4	User will not be able to annotate image other than with the use of the preinstalled shapes	Only predetermined shapes available for user in the program	Implement a drop down list in the GUI of predetermined shapes for the user to choose from.
5	Shapes use outlines and do not fill with colour	Program will draw shapes unfilled but with a border.	Shapes will be displayed as unfilled in GUI.
6	Shapes should be displayed on the image	Program will superimpose the shapes onto the image.	Integrate superimposition of shapes within GUI.
7	Shape size must be customizable	Program will have a shape resizing button when drawing shapes.	Integrate shape editing buttons within GUI for resizing.
8	The images the user has selected must be present in the image pane	The program must have the images they have selected inside the application in an image pane.	Create an explorer for ordering files.
9	Use of arrays for	The program will use	Utilise array structures

	images	arrays for image storage and reprisal	for store images
10	User must be able to move whole shapes with mouse	Program responds to user command for shape move and moves it to new user specified position	Integrate a function that encompasses shape editing.
12	User will be able to remove shapes	program deletes desired shape from the image	
13	User can visualise name of class on top of the shape	Program will display the name of the class on top of the shape.	
14	User can copy paste shapes to mouse cursor	program responds to the copy action by taking the selected shape and pasting it at selected position	Implement an individual copy and paste system within the pane
System architecture			
1	Clear System Modelling diagram (UML) with descriptions	Diagrams for the system need to be made to accurately assess what can be done within the app	Create UML designs such as sequence, use case etc.
2	Select graphical library (OpenCV/OpenGL)	A graphical library will be needed to allow shape drawing to be carried out inside the application.	Research and weigh up the best library for drawing shapes.
3	Select GUI design library	We will need to design a GUI therefore a library/framework will be needed.	Research and weigh up the best GUI design for our application.
Software Dev			
1	Produce clear contribution guide	Group members need to keep on top of work and make sure everyone is making contributions to the project	Set up Git hub and Trello to set up task association for each person
2	Code and	Development of code	Setup Git repository

	documentation must be stored in a git repository .	over time shown on Github.	and upload program along with documentation.
Software Test			
1	The use of continuous integration tools	The program must be connected to github to be able to integrate updates from all members	Set up github to allow multiple people to work on the program
2	Produce a clear and understandable Test Plan	The report must have an adequate test plan including pictures and extensive reviewing of the software	Produce a test plan to test for all found requirements
3	Ensure that the program will run on all computers	The program must be compatible and run on the university computers.	Run the finished program on at least 3 different computers.

7.3 - Risk Analysis - Table 2

Risk	Probability (1 Very Unlikely - 5 Very Likely)	Impact (1 Very Low - 5 - Very High)	Risk Level	Impact on Project	Mitigation Plan
Bugs in code	5	4	**Severe**	Could prevent the project achieving all its requirements .	Implement thorough testing of all aspects of the code. Could also use agile methodology to fix bugs with every increment.
Computer crashes/software loss	2	2	**Low**	Could lose code worked on since last save.	Perform regular saves and upload regularly to github.
Team member(s) not	3	5	**High**	Will increase other members'	Ensure team members are included in any and all team

participating in project				workload. As a result they will have less time to complete their respective work, possibly will do a worse job overall.	correspondence and have access to the work. Otherwise assign their tasks equally among the team and give adequate time to complete.
Members Dropping out of university	1	5	**Low**	Losing an entire member means a large workload is then moved onto the remaining team which will affect quality and satisfaction with the project.	Discuss with the professor ,tutor etc. What steps to take and consider aspects such as deadline extensions or a new working member. To ensure work is at the highest level without burnout
Poor time manageme nt	2	5	**Medium**	May not be able to finish the project to a desired standard.	Make a detailed project plan with goals and timelines to stick to.
Delays caused by a task delaying dependent tasks	3	4	**High**	Could have less time completing dependent tasks and therefore could be done to a worse standard.	Encourage team members to ask for help if necessary.
Planning is not good enough to support project developmen	2	5	**Medium**	May miss targets and deadlines due to poor organisation and planning.	Ensure detail and effort is put into the plan and perhaps also do research into planning techniques.

t.					
GUI too complex	3	5	**High**	Users may have trouble navigating the application. May not be able to use it to its fullest capabilities if at all.	Follow clear UX design principles and ensure GUI is usable in testing. Perhaps get someone outside of the group to try it out.
Prioritisation of other projects	3	5	**High**	Team members also belong to other group projects and will have to prioritise them at some point, which could cause delays when reaching deadlines.	Regularly meet and discuss progress. Start work as soon as possible to avoid risking delays when other projects need doing.
Poor team communication	2	4	**Medium**	Could lead to members not fully understanding the tasks/goals. Work might have to be reallocated and deadlines missed.	Establish multiple ways of reaching every team member. Provide regular updates in regards to progress and meetings.
Member illness	2	4	**Medium**	Members could miss a chunk of development of the project. Work will need to be reallocated	Start work early to make sure that delays can be managed without harming the rest of the project.

				and deadlines could be missed.	
Poor version control	3	3	**Medium**	Adjacent programming on the same file could result in crashes and lost code due to conflict	Use Git to reduce risk. Will be easier for members to follow changes to code. Can easily restore old versions of the code.
Member conflicts	2	4	**Medium**	May be design disagreements between members. Could slow down the design process and put the team behind schedule.	If members disagree on design decisions then will be put to vote. If there are equal votes then the project manager will have a deciding vote.
Member Motivation to project	2	4	**Medium**	Members may produce lower quality submissions due to their lack of motivation to the project and could entirely fail to reach deadlines.	Maintain regular meetings and update feelings and opinions on project to adapt work load and contributions to ensure high member satisfaction and motivation
Unfair Work Balance	2	2	**Low**	Work must be evenly assigned and If a member must cover for another member due to illness, non contribution etc. Their	Regular team meetings and agreed adaptable workloads will allow the problematic team members workloads to be split evenly and fairly among the team with agreed submissions from each member, not just placed on them without

				additional workload may be seen as unfair and produce work of a lower quality and be incomplete. Which could affect motivation and drive for the project	their consent.

7.3.1 - Risk Level Key - Table 3

Probability

5	5	10	15	20	25
4	4	8	12	16	20
3	3	6	9	12	15
2	2	4	6	8	10
1	1	2	3	4	5
	1	**2**	**3**	**4**	**5**

Impact

Low	
Medium	
High	
Severe	
Critical	

7.4 - Task List Time Plans

We used a Gantt chart to assign tasks and monitor the timeline of our completion of the tasks. It was also very useful for the project manager in keeping track of which tasks were assigned to whom and how far along said tasks are along to completion.

You can find our Gantt Chart at appendix N

7.5 - Assumptions

- All members of the team have access to hardware to develop the application.
- All members have access to the internet.
- All members are computer literate.

7.6 - Use of Monitoring Tools

During the project planning stage we utilised the business software Slack to easily communicate, share ideas and show development of the work. Thanks to this software we can have a meeting without being in the same country which has been an issue over the christmas break period. We have also linked our Github repository to slack so we can see our work during our online meetings easily and clearly. This also allows for well managed version control of the source code of our project, and can also allow us to recover any old code if new code is lost. Utilising Google Drive we were able to collaborate on a single document all at one time, as well as this Google Drive allows for strong management of edits and changes as many users can roll back or specifically pin-point additions or removals without direct communication between group members.

7.7 Methodologies

For our chosen methodology we used an agile extreme programming approach. This approach was chosen because it gave our team lots of flexibility while building the program, we were also able to improve the software quality and responsiveness to changing requirements within the team and time limit provided by using this approach.

7.7 - Adopted Coding Standards

Producing a clear coding styles guide for a group is important to ensure consistency is utilised throughout, it makes our code look professional and easily readable. It assists any future developers with common styles and techniques we used such as the use of camel case and our tabbing styles to allow anyone to understand the format and structure of our code.

Full coding standards found at Appendix 2.

7.8 - Code Contribution guide

Whilst working with a group that potentially is ever growing and evolving it is important to ensure that all group members are clear about how to treat contribution of our project and how to treat fellow group members as to ensure a stress free work environment.

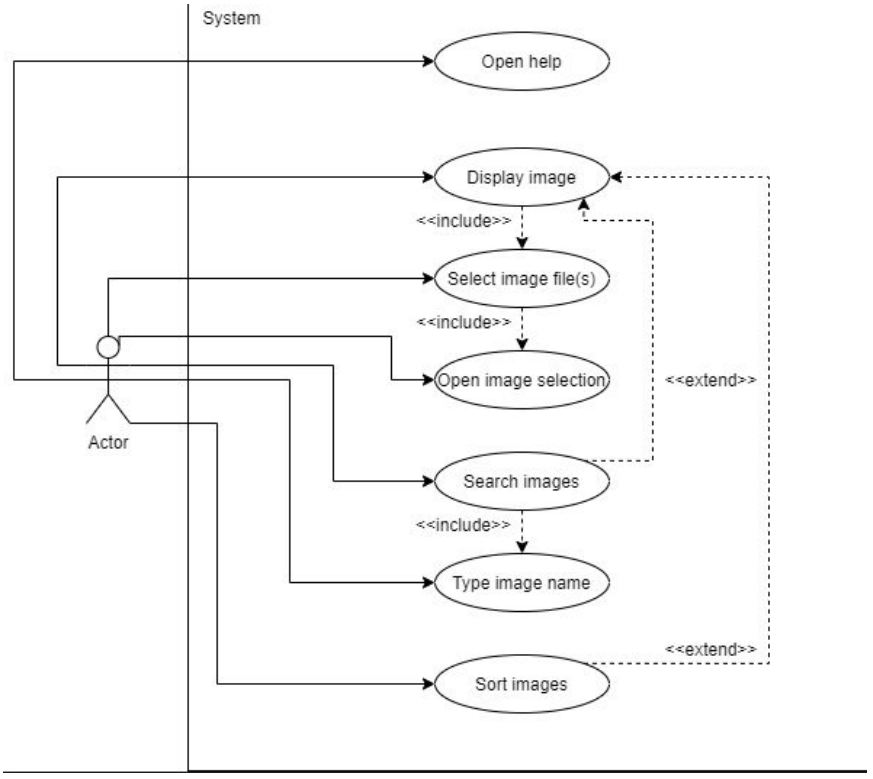
More specifically The code contribution guide included aspects relating to the use of github, bug reporting Continuous Integration/Continuous Development Strategy (CI/CD) and a code of conduct. These clear instructions and rules should be easy for any current and future developer to understand to allow for the more efficient development system.

Full Code Contribution Guide found at Appendix 3.

9 - Implementation details

9.2 - Use-Case Diagram

9.2.1 Images

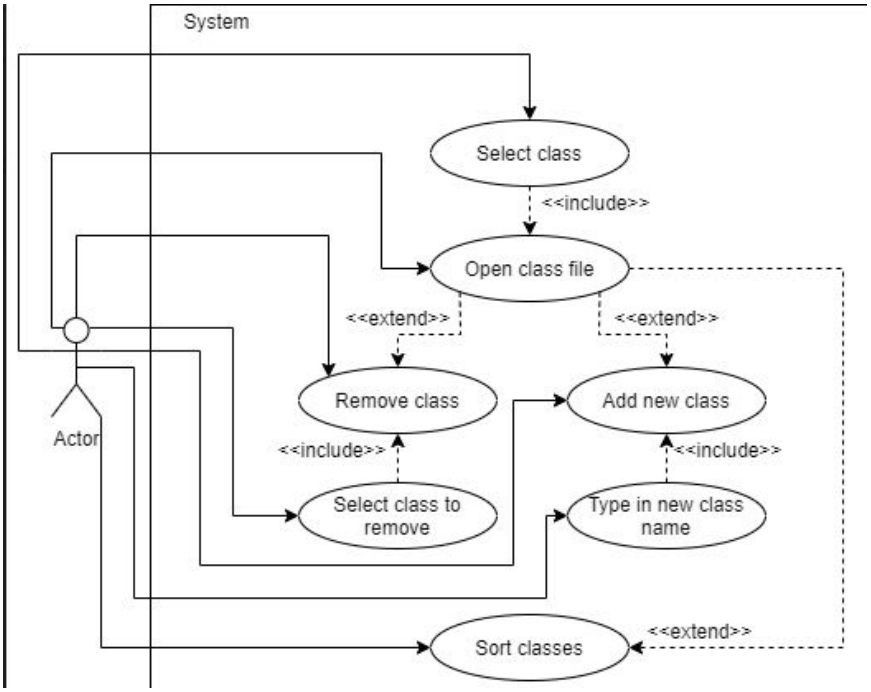


(Figure 1 Image Diagram)

This Use case diagram shows the relationship between the actor and the

images used. The actor is able to choose images from the file dialog and display them in the application. From here the actor can search for specific images and also sort in alphabetical order or date imported. The use cases complete requirements 1 and 2 for file handling and 2 in the data structures requirements with the ability to display common images and display compatible images in a folder that can be sorted in different ways.

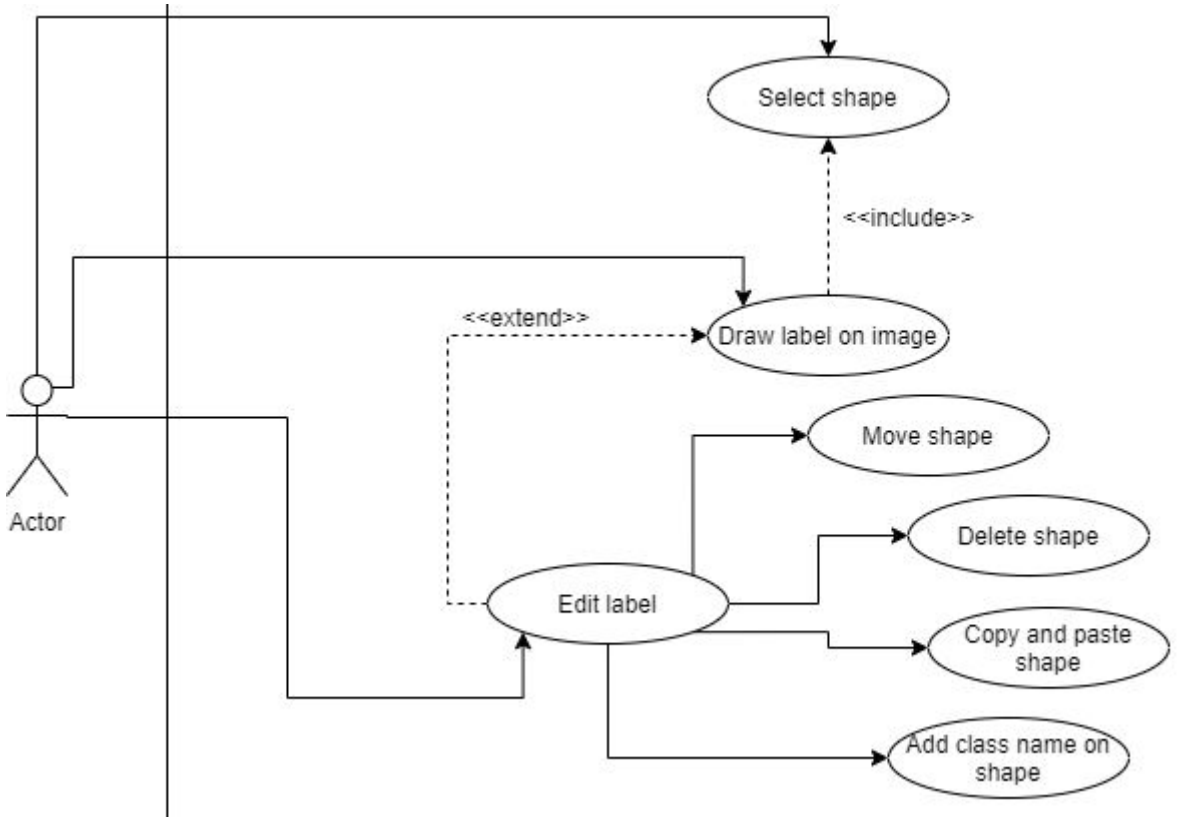
9.2.2 Classes



(Figure 2 Class diagram)

In this diagram the actor is able to select a class from the file dialog to open it inside the application or add their own class by typing in a name and classes. From here the actor is able to select and remove class files and sort them by alphabetical and date imported.
This use case completes requirements 3, 4, 5, 6, 7 of the data structures requirements which allows a user to select, remove, sort and display classes.

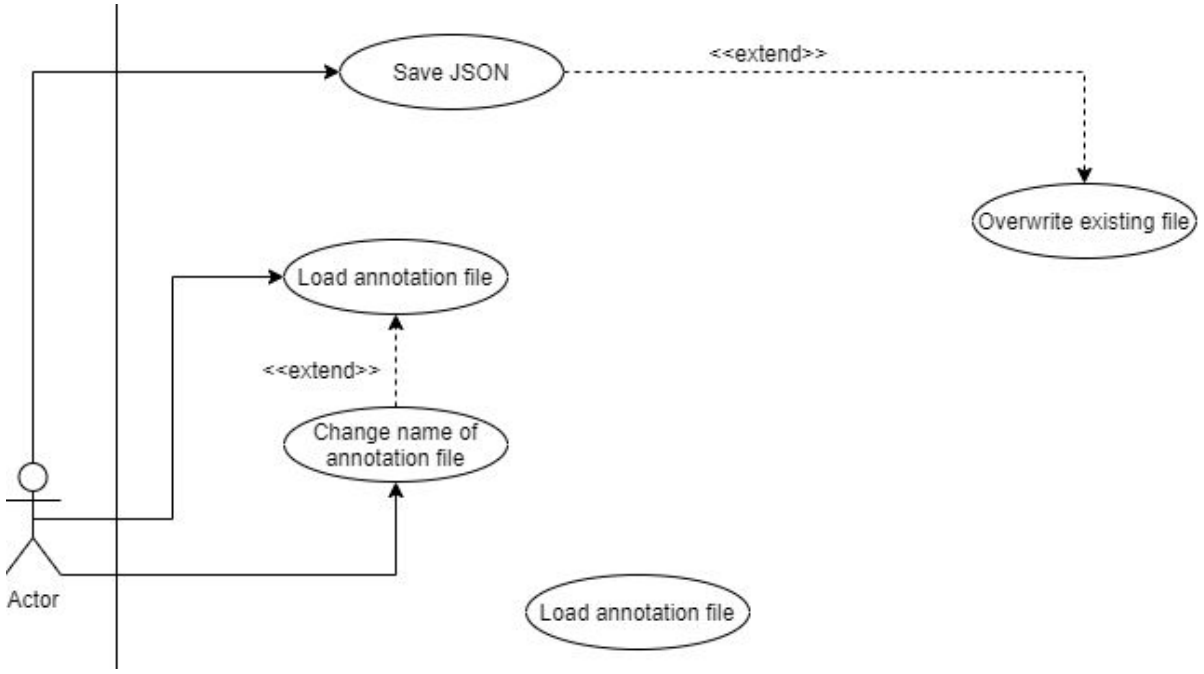
9.2.3 Drawing Labels



(Figure 3
Drawing Diagram)

In this diagram the actor is able to select a square and draw it onto the image. Once drawn the actor is able to edit the shape by redrawing it and moving it on the screen. This use case completes requirement 3,4,5,6,8,10,12,13,14 in the graphics framework requirements which allows the user to choose a shape, edit the shape and name the shape in the program and copy and paste it.

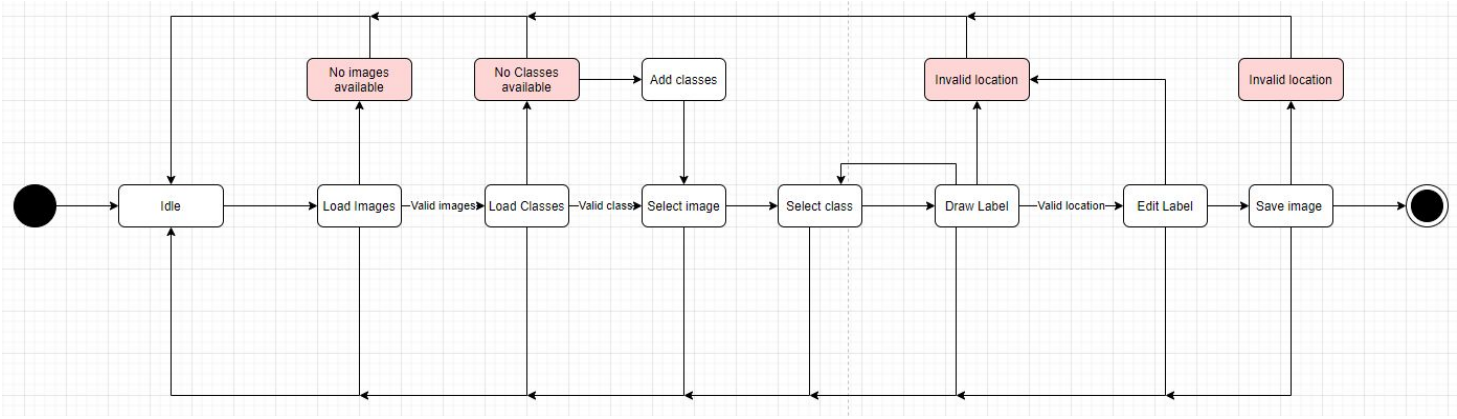
9.2.4 File Handling



(Figure 4
File Handling Diagram)

The diagram shows the relationship between the actor and the saving of files. The actor is able to save the file in JSON Format. The JSON file will have the file name, and for each image it will have the number of shapes per image, with each shape will have its own shape type and their coordinates. This use case completes requirement 8 due to being able to load and save files in a JSON format.

9.3 State Diagram

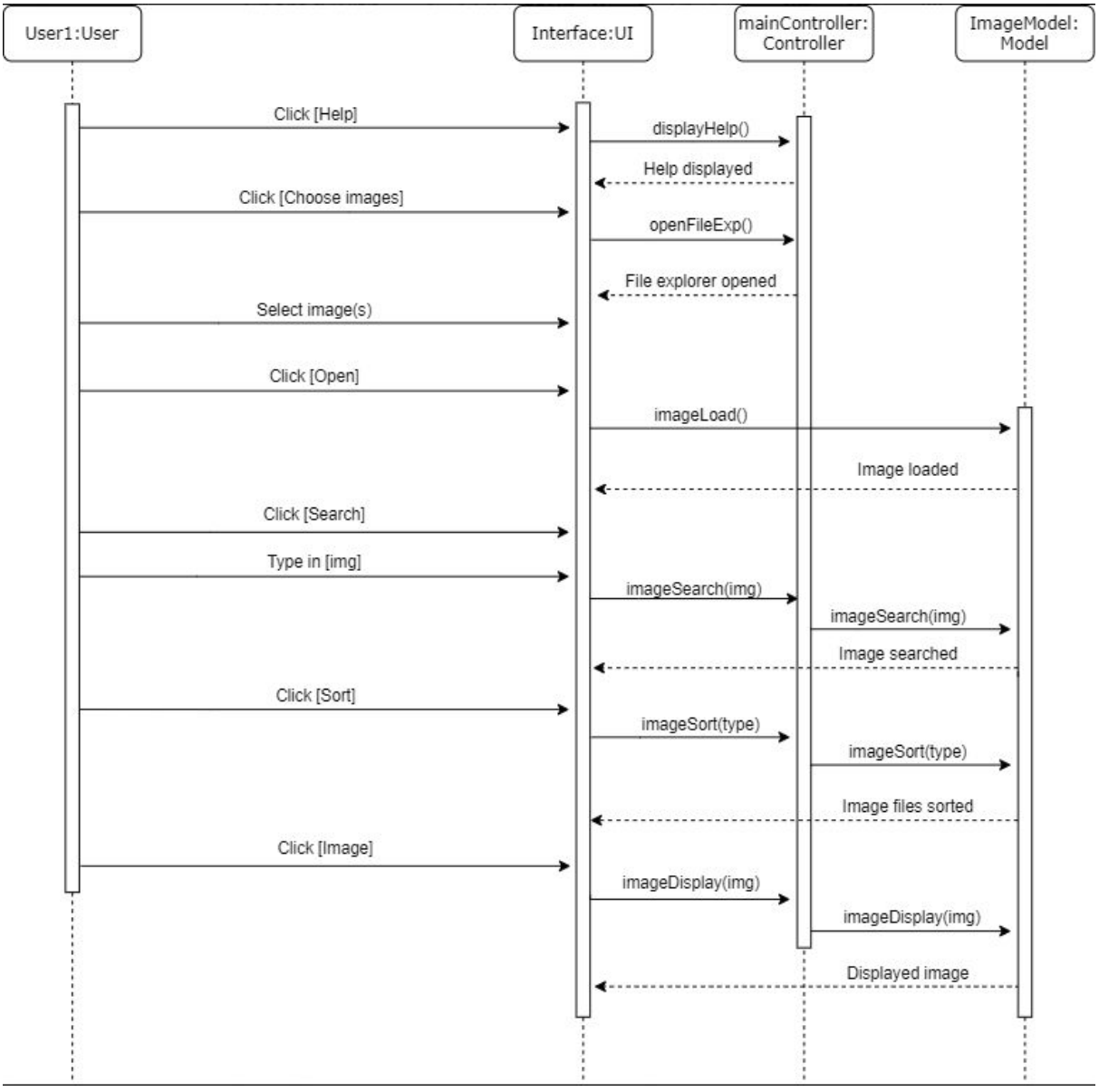


(Figure 2
State Diagram)

This state diagram shows the steps the application takes from start to end. First the user has to load the images and classes, if none are available then the application will go back to idle, the user can also add a class if none are available. Once the images and classes are loaded the user must select an image and a class to draw a label on. From there a user can add as many labels as they want by selecting the class they want. Each label can be edited in different ways, when the user is finished they can save the image and the files will be saved to the user's selected drive. This diagram fills requirements to be able to load images and classes to be drawn on an image using various shapes and saved.

9.4 Sequence Diagram

9.4.1 Images

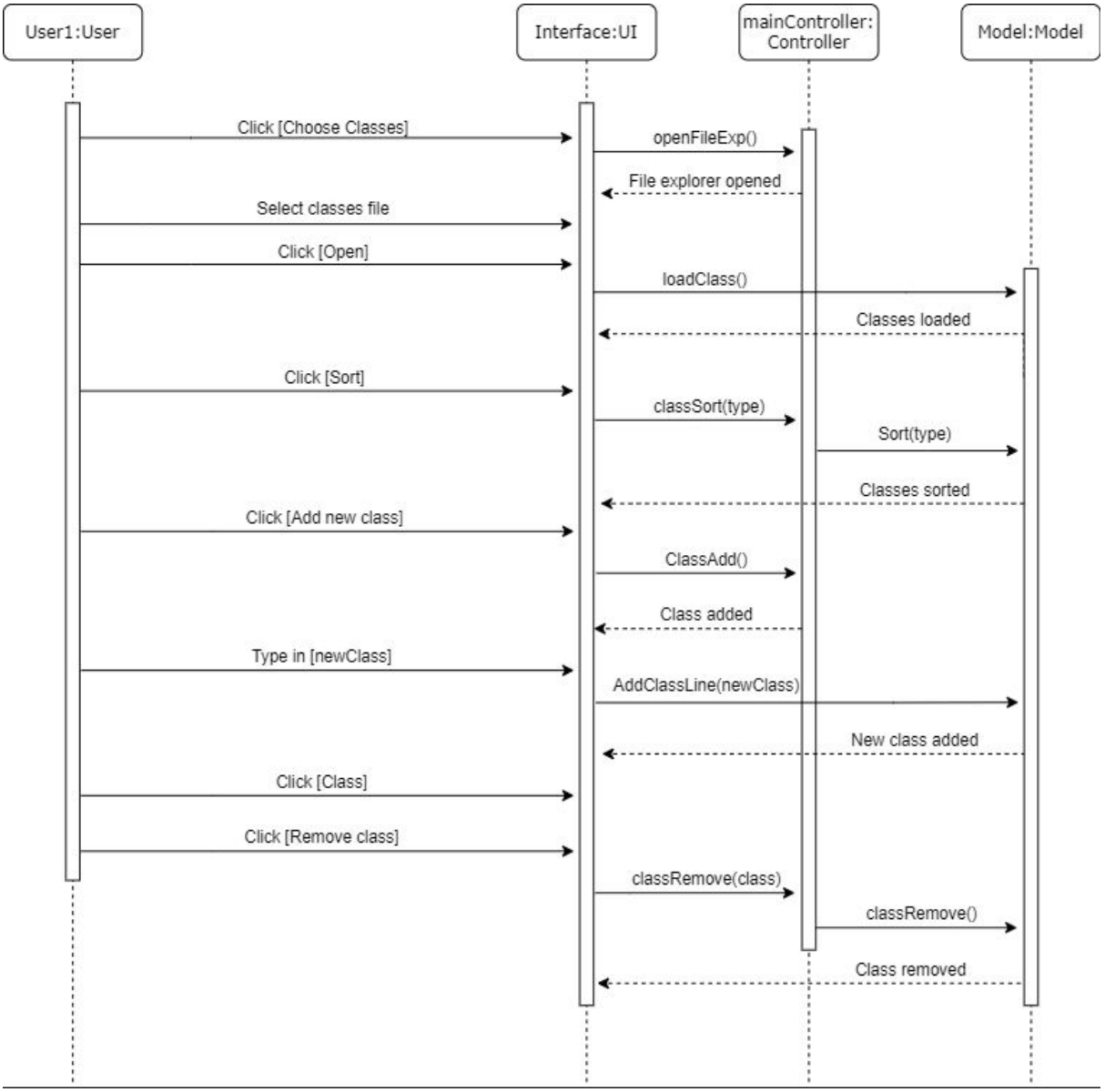


(Figure 6
Image Diagram)

This diagram shows the process with loading images, the user can select the images which is where the controller loads up a file dialog that allows the user to select and open files, the images are then loaded into the application. The user can then search for specific images by typing in the search box in the application, this makes the model search and display the relevant images. The user can sort the images by telling the controller to sort them which

gets the model to sort and display them back to the user. Finally the user can click on an image which tells the model to display it on the screen. The use cases complete requirements 1 and 2 for file handling and 2 in the data structures requirements with the ability to display common images and display compatible images in a folder that can be sorted in different ways.

9.4.2 Classes

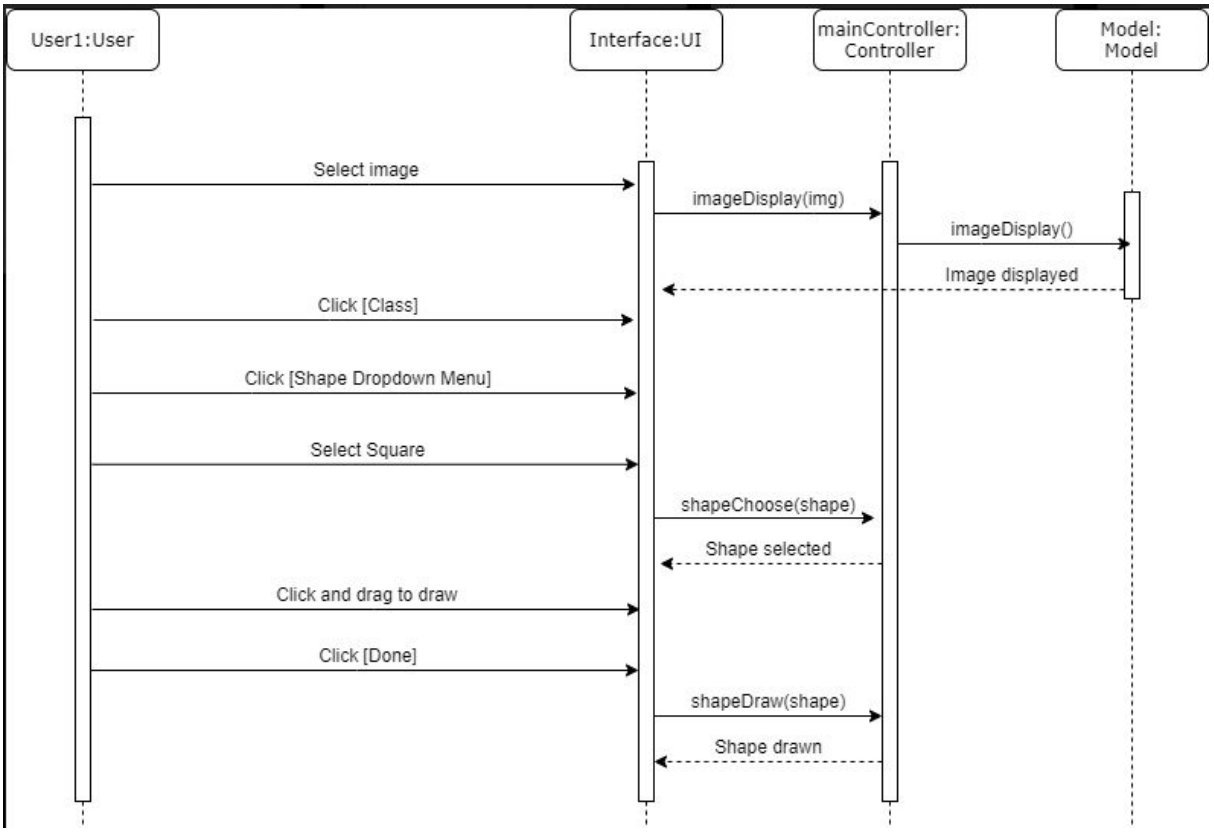


(Figure 7
Class diagram)

This diagram shows the interaction with classes. Firstly the user can choose a class file from the computer, this launches up an open file dialog by the controller where the user is able to select their selected classes and click open. These classes are loaded directly into the main application model. Once the classes are loaded the user can sort them by a type either alphabetical or by date which goes from the controller to the main application to sort.

New classes can be added by the user which are then named and loaded into the application. Classes can be selected by the user and used to make labels or they can be removed which deleted them from the application. This use case completes requirements 3, 4, 5, 6, 7 of the data structures requirements which allows a user to select, remove, sort and display classes.This use case completes requirements 4,6,7,9,10 and 11 which allows a user to select, remove, sort and display classes.

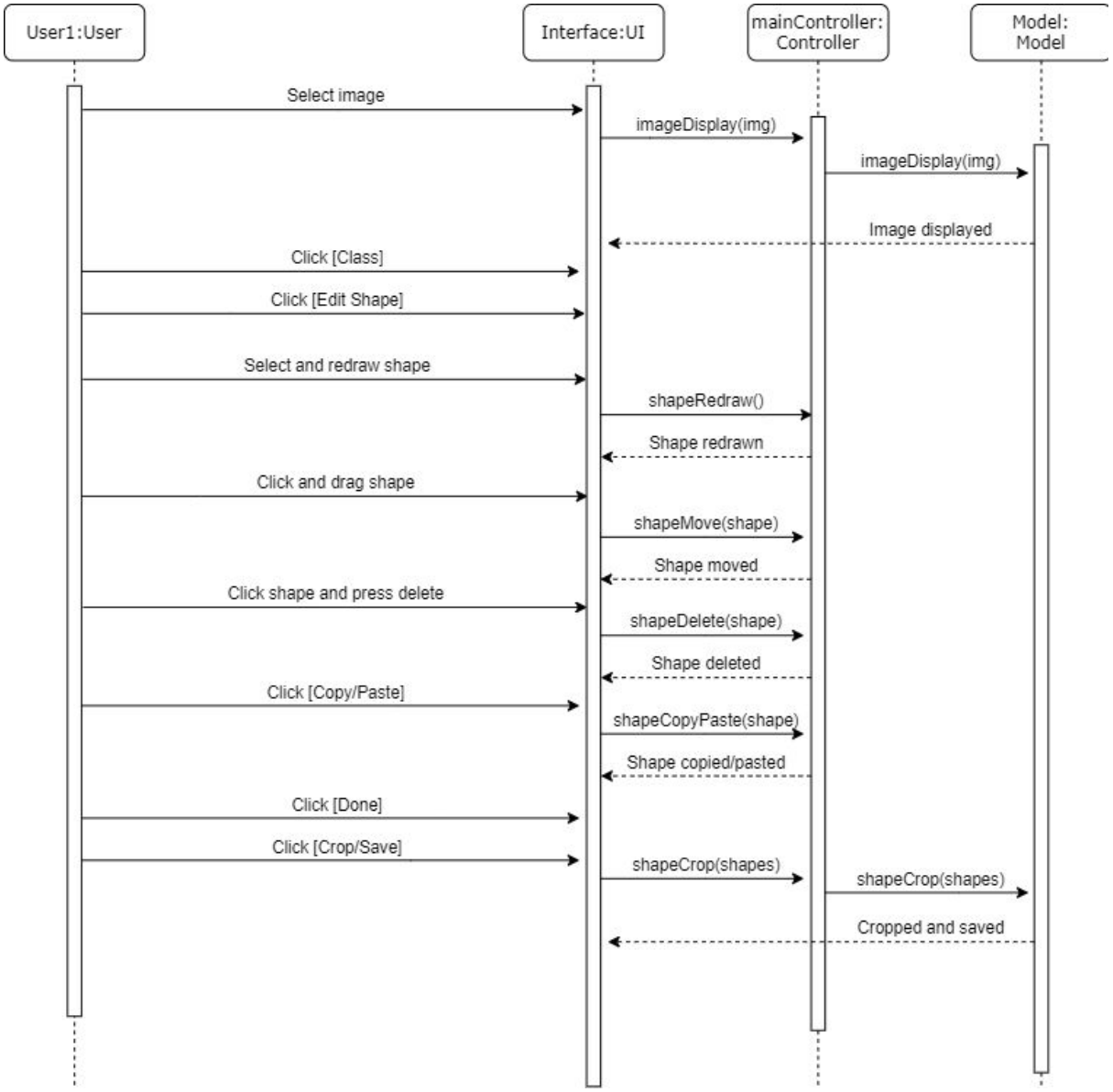
9.4.3 Drawing Shapes



(Figure 8
Drawing Diagram)

To draw shapes in this application the user must select an image which displays on screen, then they must choose a class to use to label with. Once both are selected the user can choose a shape from the drop down menu within the application and draw the shape onto the image. Once the shape is drawn the user can click done and the shape will be placed onto the image with the controller.
The user can also select a polygon and decide themselves how many vertices on the image to then be drawn onto the image. This use case completes requirement 3,4,5,6,8,10,12,13,14 in the graphics framework requirements and which allows the user to choose a shape and draw it.

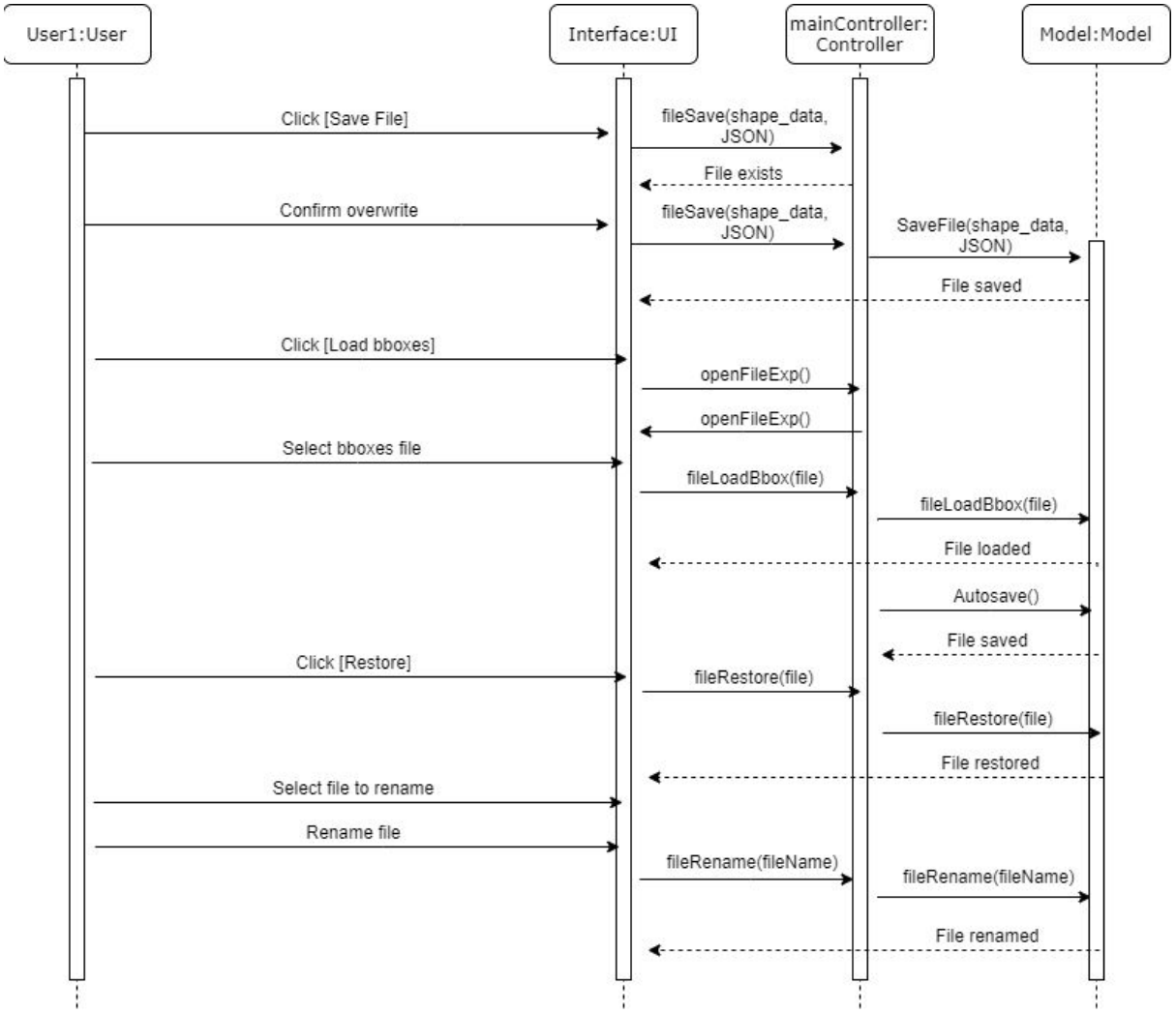
9.4.4 Editing Shapes



(Figure 9
Edit Diagram)

To edit a shape the user must be displaying the image the shape is on and the relevant shape within the class section. From here the user can interact with the UI to change, redraw the image, move the shape or delete the shape. These options will interact with the controller to change the shape, once finished the user must click done for the effects to save.
If the user wants to crop and save the controller gets the model to get the part the user wants cropped and saves it to the system. This completes requirements 7,8,10,12,13,14 from the graphics framework to be able to move the shape and change its size.

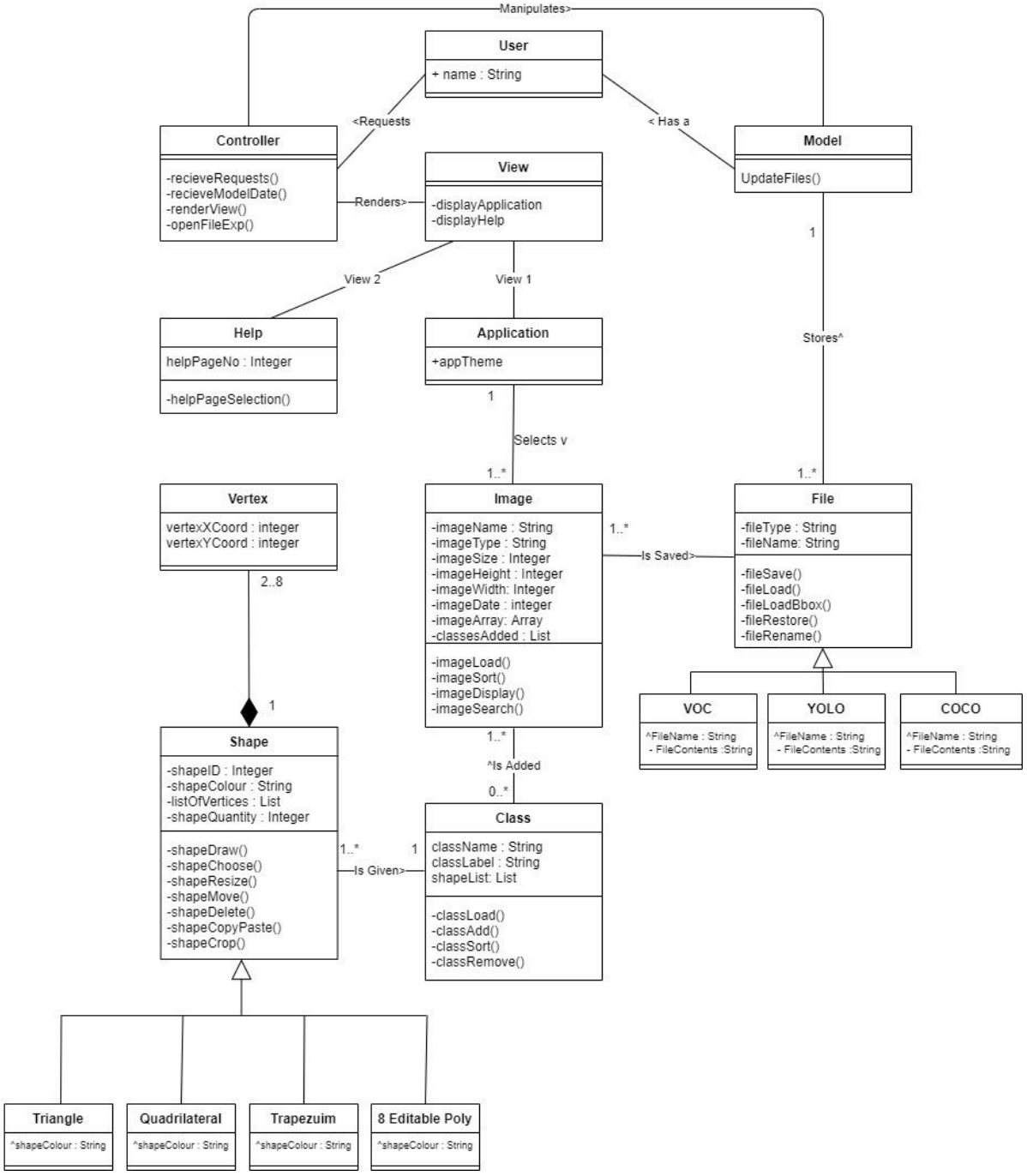
9.4.5 File Handling



(Figure 10
File Handling diagram)

The application saves the information into a JSON file that can be read back so that the annotations can be loaded back into the application for editing later on..
If the user wants to restore a file back to its original without the annotations, they can do so which restores the file back into the user's drive. The user can also rename the files to an appropriate name for viewing. This use case completes requirements 8 from the data structures requirements due to being able to load and save files with auto saving by using threads.

9.5 Class Diagram



(Figure 11
Class diagram)

Our class diagram identifies all major classes within our system along with relevant multiplicities and relationships and some aggregations and inheritances/generalisations

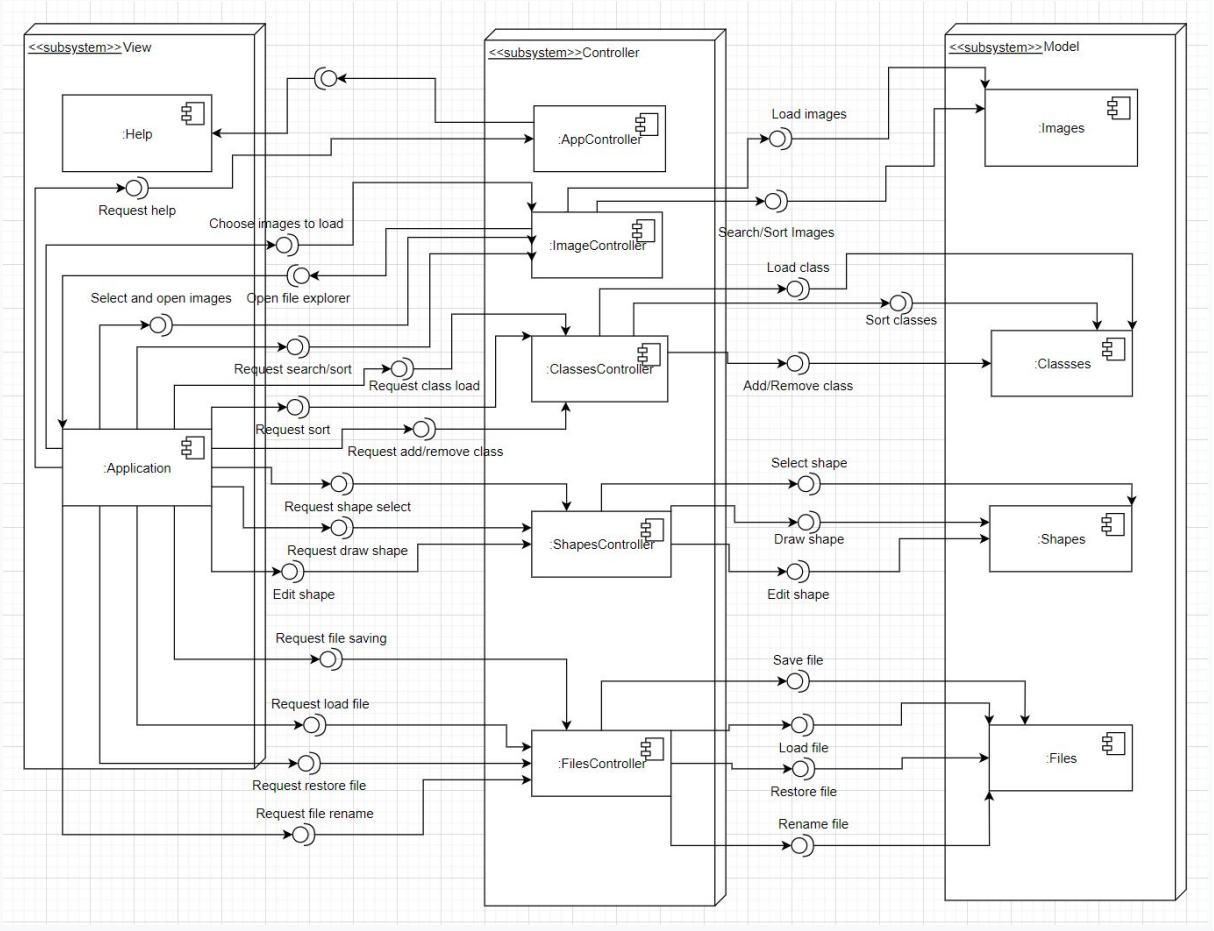
The **User** asks for the **Controller** to load and manipulate the **Model** so that the user can save and load the relevant files. The controller also displays the **View.** The view can display

2 views depending on user input. The main splash screen will be the **Application** but the user can select the **Help** screen at any time.
The **Shape** has a generalisation of its shape types along with an aggregation to the Vertex positions. The shape connects to the **Class** by linking a shape to a class name. This class shape is then added to an **Image** with associated class name as an annotation by the user.

Once complete the Image and its classes can be saved into the **File** using a JSON format. The file is then stored and managed by the **Model.**

This diagram covers a wide range of requirements that allow the user to have advanced functionality within the application.

9.6 Component Diagram



(Figure 12
Component diagram)

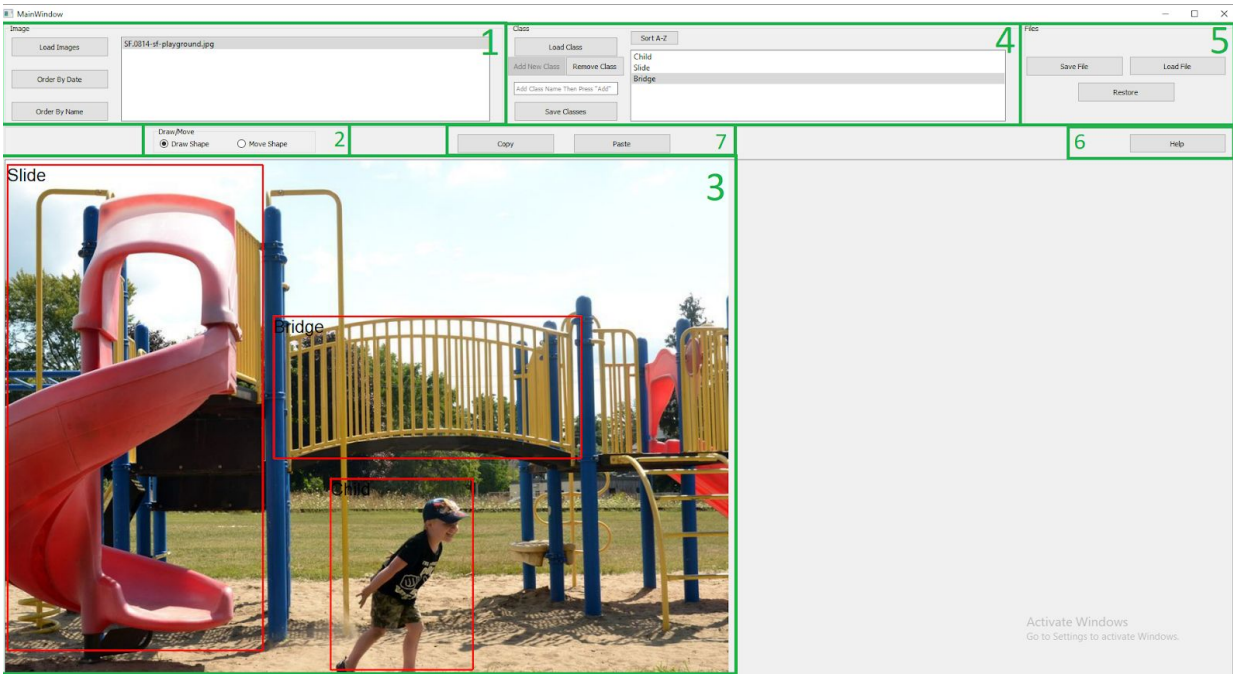
Our component diagram identifies the subsystems of View, Controller and Model, and maps their components and how they are connected via interfaces. Similarly to the class diagram, it shows the View involves the main application which allows the user to interact with the controller, and also a help screen. The application interfaces with various controllers to request file loading/saving, search/sort and shape manipulation. The Controller has a sub

controller that manages specific aspects of the application such as file handling or drawing/editing of shapes. The model then keeps track of any data being used by the application, and is consistently updated by the controllers.

11 - Results

Here are some images of the application running and describing how each part of the application works while making sure to handle any errors that might occur.
The application runs smoothly from start to finish allowing the user to insert images into the app and draw squares on the canvas with accompanying text to show what the square represents.
The user can add, save and load a JSON file with shape information in it to allow previously drawn shapes to be displayed again.

11.1 Main Application

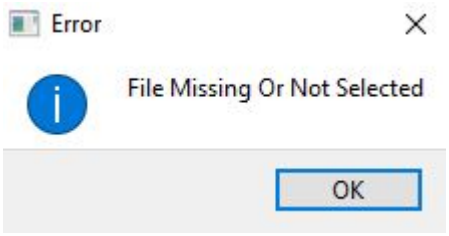


1. The image pane is where all images get loaded into to be viewed and ordered. Firstly the user can click load images which will bring up the file dialog to allow the user to select compatible image(s). The image filenames are put into a binary tree. Once the image is loaded it will be displayed in the image panel where, if multiple images are loaded at once, the user is then able to order by date and name using a bubble sort. The user can also double click the image, which then retrieves the respective file name from the binary tree and opens it, to display it on the canvas below.
2. This section allows the user to either draw shape or move shape, if draw shape is selected the user is able to click on the image that is loaded in the canvas to draw squares on with annotations in the top left of the shape depending on which

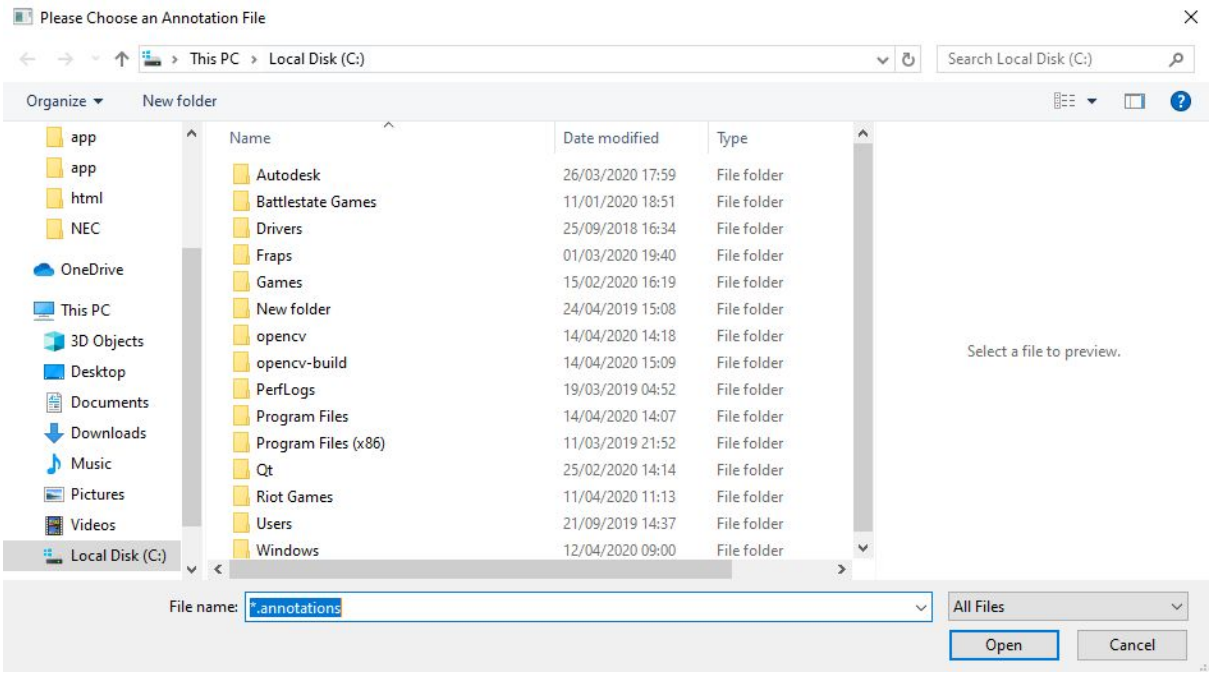
annotation they have chosen. If the user has selected, “move shape”, then the user will be able to move any pre existing shapes around the canvas.

3. This main section of the program is where the images and shapes get drawn. Multiple images can be loaded into the application and the user can switch between which image they want to annotate. Switching between images will clear the shapes to give the user a clean image so saving between changing images is a good idea.
4. The class selection is where all the classes can be loaded into and sorted for the users use. The user can load a class by clicking load class and choosing a .annotations file which will then show all of the classes inside the file into the class pane. From there the user can sort the classes using bubble sort into alphabetical order. If the user wanted to make their own classes they can by entering a class name into the text box and clicking add new class.. From here a new class will be made and shown in the class pane, any class within the pane can also be deleted by pressing delete class.
5. The user can load, save and restore files by using the relative buttons. If the user wants to save the annotations on the screen they can press save file which saves the location of the annotations to a JSON file. The user can also load previous annotation files which will load the shapes back onto the screen by reading the JSON files. Pressing restore will restore the image back to its original state by deleting all the shapes.
6. The help button will give the user key instructions on how to use the application.
7. The copy paste button will copy the selected shape that the user has chosen and paste the same shape at the same coordinates to be moved to a different location.

11.2 Error Handling

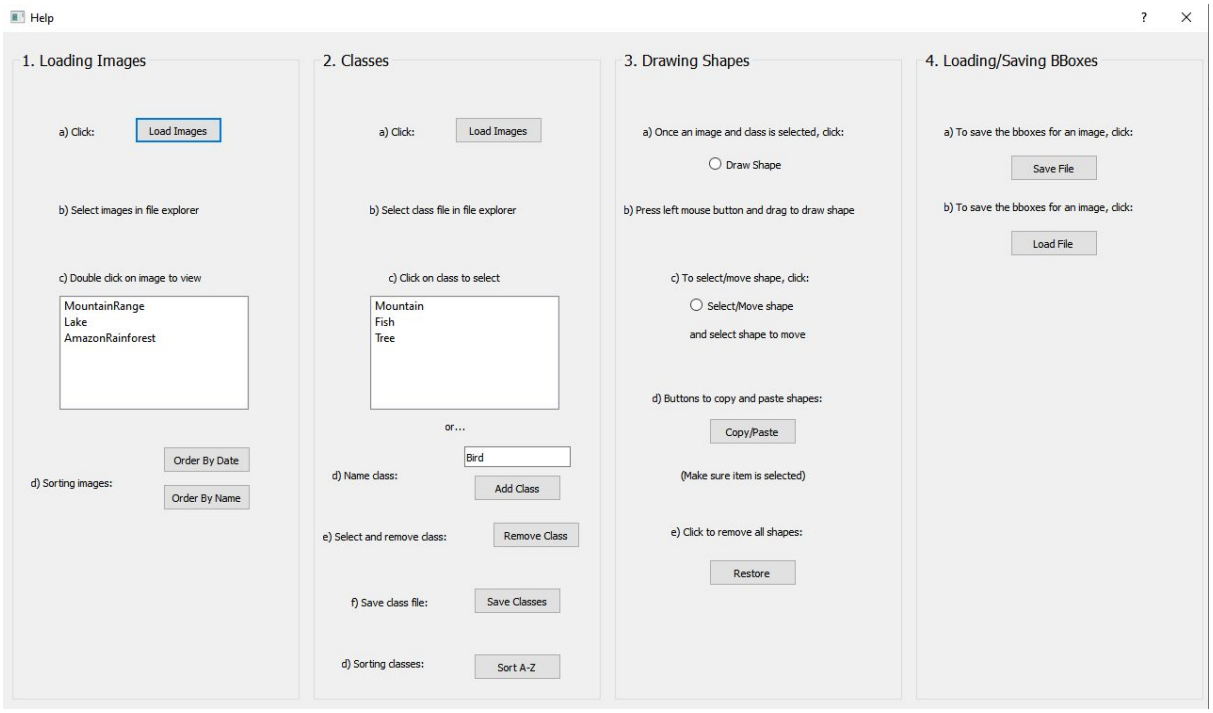


In our application when a user tries to select an image or class that is either not available or not the right type it will give them an error to let them know the file could not be loaded. This error handling is in place to stop the program from breaking when no file is chosen for editing.



The annotations that get loaded into the application must be a .annotations file, this is to stop any files that would not be accepted by the program blocked so the program does not break or crash .

11.3 Help Section



An additional help window was added to aid the user in using the application. We thought it prudent to add this as users less familiar with applications such as these may struggle

navigating it. It has a variety of widgets that can be clicked and edited to allow the user to get used to using the application.

12 - Conclusions And Future Work

12.1 Areas of improvement

Our system is not perfect and as such we have decided on some areas of improvement for future development to make a more effective and completed product.

- Due to our lack of a software tester, we have a few bugs in our code. Testing is an important part of development missed out on, so for future work we would have to bring in another external and compliant software tester to replace our current member as to establish proper tests into the future.

- Adding more shape options to allow the user will allow for better user options, as well as allowing colour customisation

- Improve error handling throughout the whole program. While current strategies are effective, more could be done to ensure errors are logged and the user is clear on the issues going ahead.

- Offering other file formats for .annotations, to allow a larger catch for resulting CNN programs to utilise.

- Scene options such as zooming, rotation and moving of the image to allow complete user customisation.

With these changes in the future we can make a more fulfilling product for an end user.

12.2 Conclusion

As previously stated throughout this report the application we have built is a labelling application for a Convolutional Neural Network (CNN) to learn from. The program works by loading in image and classes and drawing boxes over the relevant areas on the image to label them. For example, if the user was to use an image of a park they would be able to draw labels to show what part of the picture is what. The app is fully functional so that a user can get from start to finish within the app with all of the functions working as intended.

The project ran smoothly for the team apart from our software tester who made no effort to get in contact with the group and ended up not submitting or producing any work. The rest of the team made sure to have weekly meetings on a Monday to update each other on what each person has been working on and any updates that had been made to the report or application. Each group member par software tester also contributed a significant amount of work to the report and application, this means that work was completed on track of time so the team was not rushed to get parts finished. The application itself was easy to use and stylish giving it a more professional look.

The improvements that could be made to the project if more time was given was the ability to choose more types of shares or chosing a number of vertices to draw instead. This would increase the applications useability to be able to correctly identify the chosen part of the image. Another improvement would be the ability to edit the shape more fluently, this could be the ability to move individual vertices to fit the image more appropriately.

In conclusion we believe we have produced a suitable system that meets most if not all of our requirements. We learnt a lot about the professional and ethical mindsets needed for a large scale project like this. Dealing with issues such as a non compliant group member which put us at a disadvantage for the majority of the workflow and the Covid-19 pandemic changing our major working strategies almost overnight, causing increased stress on the team but we persevered and were able to complete the module.

13 - References

Doxygen.nl - Doxygen.nl. 2019. *Doxygen: Main Page*. [online] Available at: <http://www.doxygen.nl/> [Accessed 14 January 2020].

Git Hub - GitHub. 2019. *Build Software Better, Together*. [online] Available at: <https://github.com/> [Accessed 25 November 2019].

Draw.io - App.diagrams.net. 2020. *Flowchart Maker & Online Diagram Software*. [online] Available at: <https://app.diagrams.net/> [Accessed 17 February 2020].

Visio - Microsoft.com. 2020. *Flowchart Maker & Diagramming Software, Microsoft Visio*. [online] Available at: <https://www.microsoft.com/en-gb/microsoft-365/visio/flowchart-software> [Accessed 17 February 2020].

Visual Studio - Docs.microsoft.com. 2020. *Desktop Applications (Visual C++)*. [online] Available at:

<https://docs.microsoft.com/en-us/cpp/windows/desktop-applications-visual-cpp?view=vs-2019> [Accessed 20 April 2020].

Qt Documentation - Doc.qt.io. 2020. *Qt Reference Pages* | *Qt 5.14*. [online] Available at: <https://doc.qt.io/qt-5/reference-overview.html> [Accessed 24 April 2020].

QGraphicsItem Documentation - Doc.qt.io. 2020. *Qgraphicsitem Class* | *Qt Widgets 5.14.2*. [online] Available at: <https://doc.qt.io/qt-5/qgraphicsitem.html> [Accessed 25 April 2020].

QRect Class Documentation - Doc.qt.io. 2020. *Qrect Class* | *Qt Core 5.14.2*. [online] Available at: <https://doc.qt.io/qt-5/qrect.html> [Accessed 24 April 2020].

QGraphicsScene Documentation - Doc.qt.io. 2020. *Qgraphicsscene Class* | *Qt Widgets 5.14.2*. [online] Available at: <https://doc.qt.io/qt-5/qgraphicsscene.html> [Accessed 25 April 2020].

QFileDialog Documentation - Doc.qt.io. 2020. *Qfiledialog Class* | *Qt Widgets 5.14.2*. [online] Available at: <https://doc.qt.io/qt-5/qfiledialog.html> [Accessed 25 April 2020].

QMainWindow - Doc.qt.io. 2020. *Qmainwindow Class* | *Qt Widgets 5.14.2*. [online] Available at: <https://doc.qt.io/qt-5/qmainwindow.html> [Accessed 25 April 2020].

14 - Individual Contributions and Reflections

14.1 Individual Contributions

Jordan Rudge (Project Manager) - Main tasks were to set up a Git repository, gather a requirement list, project risks, and develop a mitigation plan. Contributed towards research of selected libraries and some UML diagrams. Also made contributions to the programming of the application. Would organise and attend every meeting, which occurred weekly on a Monday. Had a high group engagement and reliability. Completed all tasks assigned.

Liam Mullins (Software Architect) - Main tasks were to research data structures and sorting algorithms, producing UML Diagrams. Contributed towards gathering requirements and project risks. Also made contributions to the programming of the application. Would attend every meeting, had high engagement and reliability. Completed all tasks assigned to him.

Harry Dale (Software Developer) - Main task was developing the application, creating a code contribution guide and reference manual. Contributed towards gathering requirements and project risks and some UML Diagrams. Would attend every meeting and had high engagement and reliability besides the week in which he had extenuating circumstances through no fault of his own. Completed all tasks assigned to him.

Tanmay Poddar (Software Tester) - N/A

14.2 Individual Reflections

Harry N0803132 - I personally can take a lot away from this module, as it pertained to the industry standard practices I may be using in my future career. Utilising github effectively for the first time will of course allow me to collaborate with developers all over the world. My contributing team worked well to ensure personal deliverables were met to ensure individual marks were of a high quality as well as assisting with individual issues which were not within their deliverable to ensure no present members were left behind.

I have positive feelings towards this task, whilst challenges were posed along the way due to unforeseen events and a non committal group member that have changed my working methods for the better, It has positively impacted my resilience and resourcefulness which I hope to bring to future tasks in my university career.

Jordan N0833118 - I feel this module was great in teaching us key roles in the development of an application and assigning particular responsibilities to people with said roles. In this way it has given me some a great foundation that I can use to build upon after Uni when I enter the industry.

I have learnt valuable skills from this module, including the use of GitHub which will be endlessly useful in my future professional career. I also have gained experience on what it's like to manage a group, which is something I particularly felt wasn't one of my strongest attributes prior to this.

Liam N0801237 - This module has given me a lot of insight into how applications such as this are created in the real world. There is a process to how things are made and it has taught me that you cannot just build the application without putting in the research and design beforehand. Understanding the key programs that get used to create applications like this is going to be a big help to me when it comes to getting a job in the professional world. The team I was with ensured that work wass delegated and worked through with plenty of time to spare to make sure that the work was not rushed.

The challenges I have been presented with have given me ample time to research and commit my knowledge into the work.

Appendix 1 - Justification for leaving out testing

Baptista machado, Pedro
Wed 15/04/2020 19:39
Harry Dale 2018 (N0803132)

Hi Harry,
Please focus (PM, SA, SD) on the D5 report. I have the marking sheet being moderated/validated the marking calculation should be +/- as follows:

Individual Contribution [Subject Weight Factor: 0.5]: - Individual Deliverable (D1-D4) quality [Weight Factor: 0.2] - Tasks completion [WF: 0.05] - Weekly meetings attendance [WF: 0.01] - Group members' contribution form feedback [WF: 0.1] - Group engagement [WF: 0.05] - Reliability [WF: 0.05] - Individual tasks [WF: 0.01] - Advanced Features [WF: 0.03]
Comments:
Report [SWF: 0.3]: - Abstract [WF: 0.01] - Introduction [WF: 0.01] - Background Research [WF: 0.01] - Project Plan [WF: 0.05] - Project Design [WF: 0.05] - Implementation [WF: 0.05] - Results [WF: 0.05] - Conclusion and Future work [WF: 0.05] - Overall reflection [WF: 0.02]

Re: D5 Submission

Baptista machado, Pedro <pedro.baptistamachado@ntu.ac.uk>
18/04/2020 16:22

To: Jordan Rudge 2018 (N0833118)

Unit Test (i.e. linked-list, sort and search) is for the ST. Therefore, forcus on the D5.

Appendix 2 - GitHub Link

https://github.com/SadRavioli/SDI-Group-26

Appendix 2 - Coding style guide

Coding Styles Guide

C++ version
C++11 Is the version we utilise within QT Creator IDE

Header Files

Header files within the program will be self contained and always end in .h and will be stored at the top of the page utilising #include "xyz".

#Include libraries

#Include <xyz> is used to call specific classes and libraries for use within that page of code. Include will be held underneath header declarations to ensure ease of access if issues arise due to missing libraries.

Namespaces

Namespaces are used to avoid name conflicts within large projects and where possible
Using the term "using namespace x" will allow our code to remain clean and readable as well as automatically define names if appropriate . This statement will be placed just after the #include declarations.

Camel case

For the sake of consistency variable naming style will be the use of camel case.

exampleVariable

Tabbing Indentation

A consistent tabbing system will be enforced. This is for clarity in coding and consistency, it allows for easy readability for example.

A single tab key press is required per indent

```
Public Classname()
{
        Start line here
}
```

Variable and Array Initialization

You are permitted to use your personal preference of (),{} or =

String x = “hello”
String y(“hello”)
String z{“hello”}

Scope of variables

Global variables should be named with the addition of G at the front to clearly differentiate global variables

Meaningless variables such as iterator variables that are not used outside of objects such as loops must be given arbitrary names such as i,x or temp<name> to avoid confusion with key variables and clearly label them as meaningless. Additionally, any meaningless variable must be declared within the function they are used for example:

For (int i = 0; xyz ; xyz)

This will avoid confusion both within building and coding as these variables cannot be confused with another call somewhere else in the project.

Code Spacing

Use return to space code snippets to ensure easy readability, only use return to seperate, sub functions such as variable assignment to loop statements try to place aspects together. For example:

String y = (“XYZ”)
Int x = (2)

For (xyz)

Classes

Classes will be named as to match their contents such as a class for a linked list being called LinkedList this clear naming structure will ensure no confusion when switching between classes.

Naming

Variables will be named consistently to allow for clarity and easy readability when coding. Variables will be named in correspondence with the classes they are contained and used within.

Classes will be named consistently to what they will contain or what function they will process and works in tandem with the variables.

For examples for the class name and variable name for images:

```
Public image()
{
        imageSize
        imageShape
}
```

File names will utilise a similar clear structure being named appropriately and using _ for space to ensure implementation within code.

Braces

Braces or curly brackets are essential for containing functions and code. The code will utilise a returned style to break up code as much as possible to ensure clear form throughout.

```
Test Function()
{
        Qwerty
        qwertz
}
```

Standard Commenting

Utilising // structure, Commenting will be constant within our code to ensure that any further contributing users can clearly understand our code as to better integrate themselves with the product without needless testing. Comments will be used where deemed necessary to help with understanding of application describing each individual instance. Punctuation should be maintained and professional to ensure easy readability.

Line Length

Any line of code or comments must not exceed 80 characters long to ensure the lines can be read on the majority of working layouts.

Exclusions of this rule include
-Headers
-include statements
-comment that would harm the readability of the comment

Function Call

Function calls will be made on a single line unless exceed line length then an indented (Tab) new line can be utilised. But try to actively avoid excessive line use.

```
Function (Argument1, Argument2, Argument,3
        Argument4)
```

Operators

Operators will always have spaces around them to ensure easy readability

```
int x = 4
Int y = 4 + 6 - 2
```

DOXYGEN STYLING

We will be utilising DOXYGEN to auto generate documentation for our program, these stylings and formats will be used.

A comment can be read by DOXYGEN via its attributed styling. The plugin will automatically generate a series of input areas within the header files where the applications functions have been called. You can see these input areas in blue and should be formatted as such.

Use this format in the .h files to clearly reference and describe each function

```
/**
* @Brief
*
*/
Void FunctionToComment()
```

The Brief tag will display individually if the function has no input parameters
adding text after the brief will allow you to comment on the functions use. Please consider being detailed in your comments as it will allow for a higher quality of documentation.

@param can be used to highlight input parameters to classes
@return to highlight return values

Appendix 3 - Code Contribution Guide

Code Contribution Guide

This guide will clearly state the contribution rules so that any potential contributors to the project understand code of conduct and ensure we utilise a clear and consistent contribution flow.

Communication

Whenever a member would like to contribute to the repository direct communication should be made to the Project Manager or Project Developer via email, Slack or phone call, detailing the proposed changes and the outlying outcomes this change will produce.

Please regard the **coding style guide** when contributing to the source code and match the style methods the software developer has decided is appropriate. Changing styles goes against this code of conduct and can result in suspension from the repository

We utilise github for storing out repository, for the development of code tracking issues and for accepting pull requests.

Bug Reports

Whenever a new bug is found open an issue within the github with the following formatting. A clear bug report will result in more efficient solutions and repair.

- A Clear Title
- A summary of the bugs effect
- How to reproduce with specific step by step instructions
- What the process was expected to perform
- What the actual result is
- Additional notes

Updated Featureset

Whenever you update the repository with a new version or updated featureset please ensure when you are uploading with relevant explanation clearly list what was added. This can be a simple statement. Ensure it is clear and comprehensive to the new update so that other contributors understand what has been added.

Continuous Integration/Continuous Development Strategy

We will exclusively utilise Github for our project, maintaining a frequent level of commits of small changes to ensure that any contributor can remain up to date with changes and issues.

Each contributor whenever working on any aspect of the project are expected to maintain daily commits with clear comments for each addition (refer to updated featureset).

Ensure strong communication throughout so all contributors are maintained with a high level of understanding of tasks and changes the project has undertaken.

Code of Conduct

Our Pledge

We strive to create a welcoming environment for contributors, a conflict free workplace where a users issues can be dealt with no matter their personal background.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Use of inclusive language throughout project
- Be understanding of all contributors viewpoints
- Always submit constructive criticism.
- Understand other contributors feelings and be respectful of decisions

Examples of unacceptable behavior by contributors include:

- Utilising derogatory or racist terminology
- Harassment of any kind.
- Releasing private information of another individual without permission eg. Phone Number, Email Address
- Any other behavior deemed inappropriate for public group setting

Our Responsibilities

Contributors hold a responsibility of sharing rules and regulations of this project and are key in reporting any issues and problems that may negatively affect the project as a whole.

The Project Manager and Managing roles have the right to edit and reject any contributors contribution made to the project if they deem it is misaligned with the ethics or the code of conduct and can hand out punishment they deem appropriate.

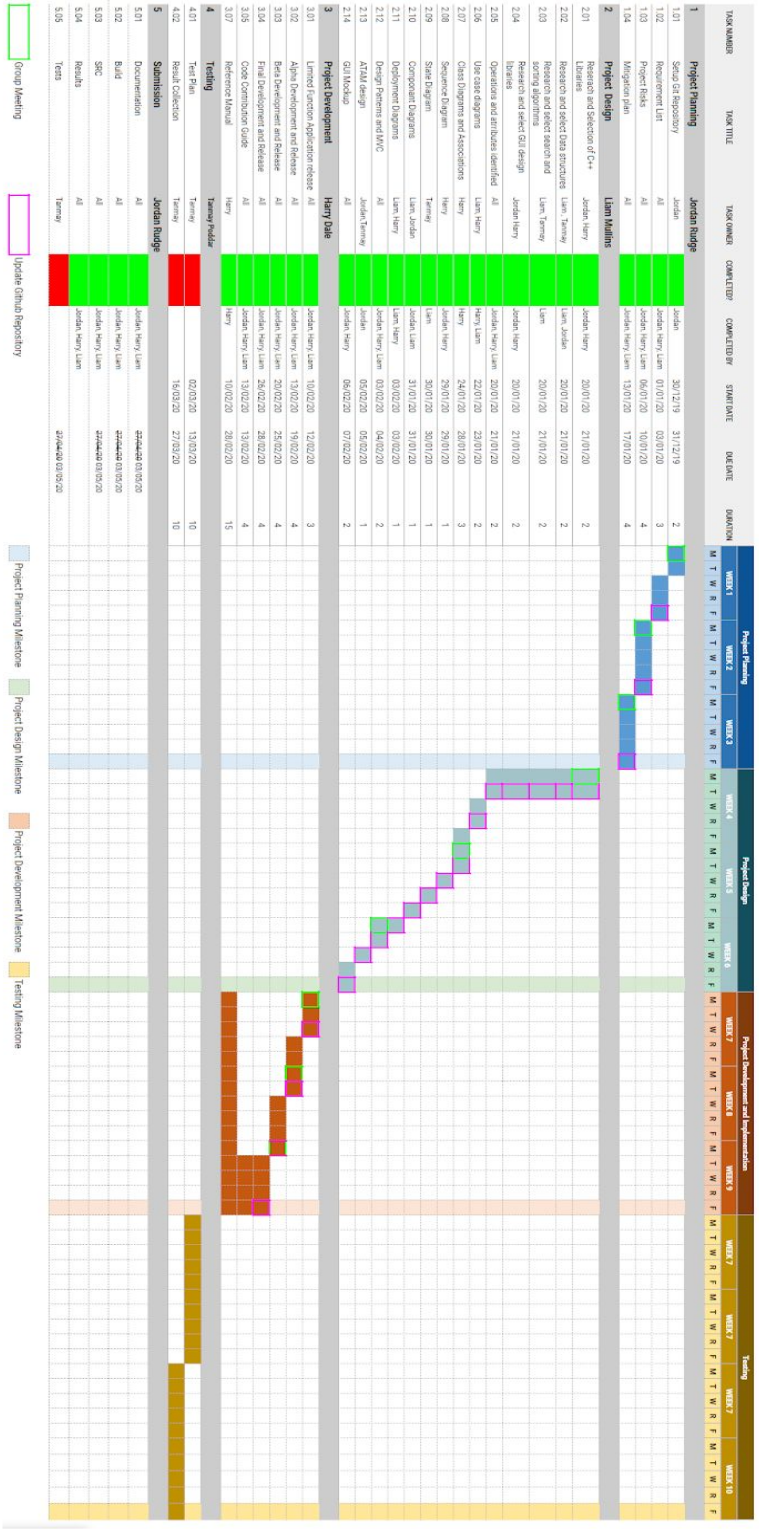
Scope

The Code of Conduct is applicable in the project spaces and in public or anywhere in which the individual would be representing the project. Email, Phone Calls and Instant Messaging are all within this scope and should be treated professionally at any time. Other aspects such as events in which a representative is sent must act within the code of conduct to show the project in positive light.

Enforcement

Anyone caught breaking the code of conduct may be reported by another member getting into contact with the project manager at Jordidan448@gmail.com all complaints will be formally reviewed to find the infraction. If the project manager believes an issue was created a penalty that is suitable for the individual, all reports will remain anonymous. Any serious infractions can result in project removal.

Appendix 4 - Gantt Chart



Appendix 4 - Reference Manual

Appendix 5 - Doxygen

Annotation Group 26

Generated by Doxygen 1.8.17

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 BinaryTree Class Reference

Binary tree file system.

`#include <binarytree.h>`

Classes

- struct TreeNode

Public Member Functions

- BinaryTree ()
  Constructor for BinaryTree class.
- void insertNode (QString, QString)
  Function to insert node into tree Each node stores filename and the time it was created.
- void remove (QString)
  Function to remove node from tree Takes filename and calls deleteNode function with root.
- void deleteNode (QString, TreeNode *&)
  Recursive function that goes through each node and then calls makeDeletion when the node is found.
- void makeDeletion (TreeNode *&)
  Function to delete nodes.
- void showNodeInOrder (void)
  Calls displayInOrder function.
- void showNodeInReverseOrder (void)
  Calls reverseInOrder function.
- void displayInOrder (TreeNode *)
  Recursive function that goes left as far as it can, before adding node to traversal list, then right.
- void reverseInOrder (TreeNode *)
  Recursive function like InOrder but reverse. Right, add to list, left.
- bool searchNode (QString)
  Function that searches for node of particular value. Returns true or false.
- QStringList returnTraversal (void)
  Function to return traversal list.

Public Attributes

- QStringList **traversal**
- TreeNode * **root**

3.1.1 Detailed Description

Binary tree file system.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BinaryTree()

```
BinaryTree::BinaryTree ( )
```

Constructor for BinaryTree class.

Parameters

parent	

3.1.3 Member Function Documentation

3.1.3.1 deleteNode()

```
void BinaryTree::deleteNode (
            QString string,
            TreeNode *& nodePtr )
```

Recursive function that goes through each node and then calls makeDeletion when the node is found.

Parameters

parent	

3.1.3.2 displayInOrder()

```
void BinaryTree::displayInOrder (
            TreeNode * nodePtr )
```

Recursive function that goes left as far as it can, before adding node to traversal list, then right.

Parameters

parent	

3.1.3.3 insertNode()

```
void BinaryTree::insertNode (
            QString string,
            QString dateTime )
```

Function to insert node into tree Each node stores filename and the time it was created.

Parameters

parent	

3.1.3.4 makeDeletion()

```
void BinaryTree::makeDeletion (
            TreeNode *& nodePtr )
```

Function to delete nodes.

Parameters

parent	

3.1.3.5 remove()

```
void BinaryTree::remove (
            QString string )
```

Function to remove node from tree Takes filename and calls deleteNode function with root.

Parameters

parent	

3.1.3.6 returnTraversal()

```
QStringList BinaryTree::returnTraversal (
            void  )
```

Function to return traversal list.

Parameters

parent	

3.1.3.7 reverseInOrder()

```
void BinaryTree::reverseInOrder (
            TreeNode * nodePtr )
```

Recursive function like InOrder but reverse. Right, add to list, left.

Parameters

parent	

3.1.3.8 searchNode()

```
bool BinaryTree::searchNode (
            QString string )
```

Function that searches for node of particular value. Returns true or false.

Parameters

parent	

3.1.3.9 showNodeInOrder()

```
void BinaryTree::showNodeInOrder (
            void  )
```

Calls displayInOrder function.

Parameters

parent	

3.1.3.10 showNodeInReverseOrder()

```
void BinaryTree::showNodeInReverseOrder (
            void  )
```

Calls reverseInOrder function.

Parameters

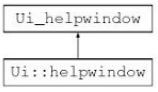
parent	

The documentation for this class was generated from the following files:

- app/binarytree.h
- app/binarytree.cpp

3.2 Ui::helpwindow Class Reference

Inheritance diagram for Ui::helpwindow:



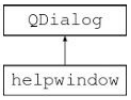
Additional Inherited Members

The documentation for this class was generated from the following file:

- app/ui_helpwindow.h

3.3 helpwindow Class Reference

Inheritance diagram for helpwindow:



Public Member Functions

- helpwindow (QWidget *parent=nullptr)
  Constructor for the helpwindow class Provides instructions on how to use the app.
- ~helpwindow ()
  Destructor for the helpwindow class.

Private Attributes

- Ui::helpwindow * **ui**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 helpwindow()

```
helpwindow::helpwindow (
            QWidget * parent = nullptr )  [explicit]
```

Constructor for the helpwindow class Provides instructions on how to use the app.

Parameters

parent	

3.3.1.2 ~helpwindow()

```
helpwindow::~helpwindow ( )
```

Destructor for the helpwindow class.

The documentation for this class was generated from the following files:

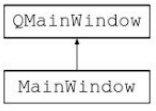
- app/helpwindow.h
- app/helpwindow.cpp

3.4 MainWindow Class Reference

Holds all UI elements and handles user input.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=nullptr)

  Constructor for the MainWindow class.
- ~MainWindow ()

  Destructor for the MainWindow class.

Private Slots

- void on_classLoad_clicked ()

  Load File explorer upon clicking button, Once user selects appropriate file the file is read line by line and loaded into listboxWidget. If user does not select or selects the wrong file a dialog will inform the user of the error.
- void on_classSave_clicked ()

  Open the class file user selected to load class names. Check if opened correctly else break. Get total of items in listbox and loop till this total using iterator and add each line into the class file, Output save dialog and close file.
- void on_classAdd_clicked ()

  User inputs a word into the lineEdit and aslong as not blank the Add button becomes available the new class name is appeneded to bottom of the listview.
- void on_classRemove_clicked ()

  From user selected row in the listview delete specific class name. This can be done at any time changes will not save in the class file unless save file is chosen.
- void on_classEnter_editingFinished ()

  When user has completed entering string triggered by pressing enter or clicking to an external component activate the aslong as the string entered is not blank.
- void on_classAsc_clicked ()

  load class into an QStringList using an iterating loop with a maximum from items run bubble sort function and order the items in alphabetical order. output a list and update the listview with the sorted values.
- void on_loadImagesButton_clicked ()

  open file explorer for the user to allow them to select multiple images to load into the list widget. Image names are stored in a linked list. Iterates through list to display image names. Will display error if user tries to load an image more than once.
- void on_imageList_itemDoubleClicked ()

  On double click of the image name in the list widget, the image will be displayed. Uses current row as index. Scales image to fit the window with smooth transformation function.
- void on_dateSortButton_clicked ()

  Will sort images by date downloaded.
- void on_saveButton_clicked ()

  Will sort images alphabetically.
- void on_nameSortButton_clicked ()

  Will save to json file.
- void on_drawShape_clicked (bool)

  Radio Button which when enabled allows the user to draw a shape to QgraphicsScene.
- void on_moveShape_clicked (bool)

  Radio Button which when enabled allows the user to move items around the QgraphicsScene with mouse.
- void on_classList_currentItemChanged (QListWidgetItem *current, QListWidgetItem *previous)

  Used to get the current class name the user has selected from the list.
- void on_loadButton_clicked ()

  Call REC Loading function that reads selected .annotations file and draws the contained shapes back onto the image.
- void on_restoreButton_clicked ()

  Button to clear all items on qgraphicsscene.
- void on_helpButton_clicked ()

  Produce the help screen.
- void on_copyButton_clicked ()

  Call copy paste function to copy and paste selected shape.

Private Member Functions

- void **swap** (QString *xp, QString *yp)

Private Attributes

- QGraphicsView * **view**
- QGraphicsScene * **scene**
- Rec * **rec**
- QString **GLocation**
- QStringList **fileNames**
- QStringList **baseNames**
- QStringList **classBaseNames**
- QString **baseName**
- QStringList **dateList**
- QStringList **dateNameList**
- QString **fileCreatedAt**
- bool **isImage** = false
- bool **nameClicked**
- bool **dateClicked**
- bool **buttonType**
- Ui::MainWindow * ui

  Initialises UI.

3.4.1 Detailed Description

Holds all UI elements and handles user input.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 MainWindow()

```
MainWindow::MainWindow (
            QWidget * parent = nullptr )
```

Constructor for the MainWindow class.

Parameters

parent	

3.4.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

Destructor for the MainWindow class.

3.4.3 Member Function Documentation

3.4.3.1 on_classAdd_clicked

```
void MainWindow::on_classAdd_clicked ( )  [private], [slot]
```

User inputs a word into the lineEdit and aslong as not blank the Add button becomes available the new class name is appeneded to bottom of the listview.

3.4.3.2 on_classList_currentItemChanged

```
void MainWindow::on_classList_currentItemChanged (
            QListWidgetItem * current,
            QListWidgetItem * previous )  [private], [slot]
```

Used to get the current class name the user has selected from the list.

3.4.3.3 on_classLoad_clicked

```
void MainWindow::on_classLoad_clicked ( )  [private], [slot]
```

Load File explorer upon clicking button, Once user selects appropriate file the file is read line by line and loaded into listboxWidget. If user does not select or selects the wrong file a dialog will inform the user of the error.

3.4.3.4 on_classRemove_clicked

```
void MainWindow::on_classRemove_clicked ( )  [private], [slot]
```

From user selected row in the listview delete specific class name. This can be done at any time changes will not save in the class file unless save file is chosen.

3.4.3.5 on_classSave_clicked

```
void MainWindow::on_classSave_clicked ( )  [private], [slot]
```

Open the class file user selected to load class names. Check if opened correctly else break. Get total of items in listbox and loop till this total using iterator and add each line into the class file, Output save dialog and close file.

3.4.3.6 on_copyButton_clicked

```
void MainWindow::on_copyButton_clicked ( )  [private], [slot]
```

Call copy paste function to copy and paste selected shape.

3.4.3.7 on_dateSortButton_clicked

```
void MainWindow::on_dateSortButton_clicked ( )  [private], [slot]
```

Will sort images by date downloaded.

3.4.3.8 on_drawShape_clicked

```
void MainWindow::on_drawShape_clicked (
            bool checked )  [private], [slot]
```

Radio Button which when enabled allows the user to draw a shape to QgraphicsScene.

3.4.3.9 on_helpButton_clicked

```
void MainWindow::on_helpButton_clicked ( )  [private], [slot]
```

Produce the help screen.

3.4.3.10 on_loadButton_clicked

```
void MainWindow::on_loadButton_clicked ( )  [private], [slot]
```

Call REC Loading function that reads selected .annotations file and draws the contained shapes back onto the image.

3.4.3.11 on_loadImagesButton_clicked

```
void MainWindow::on_loadImagesButton_clicked ( )  [private], [slot]
```

open file explorer for the user to allow them to select multiple images to load into the list widget. Image names are stored in a linked list. Iterates through list to display image names. Will display error if user tries to load an image more than once.

3.4.3.12 on_moveShape_clicked

```
void MainWindow::on_moveShape_clicked (
            bool checked )  [private], [slot]
```

Radio Button which when enabled allows the user to move items around the QgraphicsScene with mouse.

3.4.3.13 on_nameSortButton_clicked

```
void MainWindow::on_nameSortButton_clicked ( )  [private], [slot]
```

Will save to json file.

3.4.3.14 on_restoreButton_clicked

```
void MainWindow::on_restoreButton_clicked ( )  [private], [slot]
```

Button to clear all items on qgraphicsscene.

3.4.3.15 on_saveButton_clicked

```
void MainWindow::on_saveButton_clicked ( )  [private], [slot]
```

Will sort images alphabetically.

3.4.4 Member Data Documentation

3.4.4.1 ui

```
Ui::MainWindow* MainWindow::ui  [private]
```

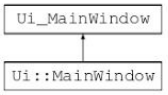
Initialises UI.

The documentation for this class was generated from the following files:

- app/mainwindow.h
- app/mainwindow.cpp

3.5 Ui::MainWindow Class Reference

Inheritance diagram for Ui::MainWindow:



Additional Inherited Members

The documentation for this class was generated from the following file:

- app/ui_mainwindow.h

3.6 qt_meta_stringdata_helpwindow_t Struct Reference

Public Attributes

- QByteArrayData **data** [1]
- char **stringdata0** [11]

The documentation for this struct was generated from the following file:

- app/moc_helpwindow.cpp

3.7 qt_meta_stringdata_MainWindow_t Struct Reference

Public Attributes

- QByteArrayData **data** [23]
- char **stringdata0** [479]

The documentation for this struct was generated from the following file:

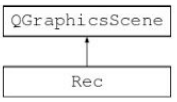
- app/moc_mainwindow.cpp

3.8 Rec Class Reference

Draw, Move, Save ,Load Boxes and annotations.

`#include <rec.h>`

Inheritance diagram for Rec:



Public Types

- enum **Mode** { **NoMode**, **SelectObject**, **DrawRec** }

Public Member Functions

- **Rec** (QObject *parent=0)
- void **setMode** (Mode mode)
- void clearRec ()

  function to remove selected rectangle
- void saveJson (QString file, QString ImageName)

  Create several JSON arrays and begin inserting each rectangles Image name, number of shapes x, y, Height, width and annotation names into a JSON file.
- void readJson (QString Load)

  Read Json file and pull outputs from the file by iterating through the number of shapes to assist in redrawing them to the scene.
- void copyPaste ()

  Copy selected item by storing data into variables and draw it at same postion of the selected box.

Public Attributes

- QString **ClassName**
- int **NoItems**

Protected Member Functions

- void mousePressEvent (QGraphicsSceneMouseEvent *event)

  get mouses starting position when clicked
- void mouseMoveEvent (QGraphicsSceneMouseEvent *event)

  move mouse set X,Y values for qrectitem in reference to the origin point set by mouse click
- void mouseReleaseEvent (QGraphicsSceneMouseEvent *event)

  Draw shape from origin point to mouses new set point.
- void keyPressEvent (QKeyEvent *event)

  Handler for button presses this will call ClearRec() function when the "Delete" key is pressed.
- void drawJson (int x, int y, int Height, int Width, QString annotation)

  Drawing system for the read JSON files, taking each aspect from the file and inserting them at relevent draw points.

Private Member Functions

- void **makeItemsControllable** (bool areControllable)

Private Attributes

- Mode **sceneMode**
- QPointF **origPoint**
- QGraphicsRectItem * **itemToDraw**
- QGraphicsSimpleTextItem * **simpleTextToDraw**

3.8.1 Detailed Description

Draw, Move, Save ,Load Boxes and annotations.

3.8.2 Member Function Documentation

3.8.2.1 clearRec()

```
void Rec::clearRec ( )
```

function to remove selected rectangle

3.8.2.2 copyPaste()

```
void Rec::copyPaste ( )
```

Copy selected item by storing data into variables and draw it at same postion of the selected box.

3.8.2.3 drawJson()

```
void Rec::drawJson (
            int x,
            int y,
            int Height,
            int Width,
            QString annotation )  [protected]
```

Drawing system for the read JSON files, taking each aspect from the file and inserting them at relevent draw points.

3.8.2.4 keyPressEvent()

```
void Rec::keyPressEvent (
            QKeyEvent * event )  [protected]
```

Handler for button presses this will call ClearRec() function when the "Delete" key is pressed.

3.8.2.5 mouseMoveEvent()

```
void Rec::mouseMoveEvent (
            QGraphicsSceneMouseEvent * event )  [protected]
```

move mouse set X,Y values for qrectitem in reference to the origin point set by mouse click

3.8.2.6 mousePressEvent()

```
void Rec::mousePressEvent (
            QGraphicsSceneMouseEvent * event )  [protected]
```

get mouses starting position when clicked

3.8.2.7 mouseReleaseEvent()

```
void Rec::mouseReleaseEvent (
            QGraphicsSceneMouseEvent * event )  [protected]
```

Draw shape from origin point to mouses new set point.

3.8.2.8 readJson()

```
void Rec::readJson (
            QString Load )
```

Read Json file and pull outputs from the file by iterating through the number of shapes to assist in redrawing them to the scene.

3.8.2.9 saveJson()

```
void Rec::saveJson (
            QString file,
            QString ImageName )
```

Create several JSON arrays and begin inserting each rectangles Image name, number of shapes x, y, Height, width and annotation names into a JSON file.

The documentation for this class was generated from the following files:

- app/rec.h
- app/rec.cpp

3.9 BinaryTree::TreeNode Struct Reference

Public Attributes

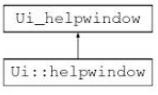
- QString **data**
- QString **date**
- struct TreeNode * **left**
- struct TreeNode * **right**

The documentation for this struct was generated from the following file:

- app/binarytree.h

3.10 Ui_helpwindow Class Reference

Inheritance diagram for Ui_helpwindow:



Public Member Functions

- void **setupUi** (QDialog *helpwindow)
- void **retranslateUi** (QDialog *helpwindow)

Public Attributes

- QGroupBox * **groupBox**
- QPushButton * **pushButton**
- QLabel * **label**
- QLabel * **label_2**
- QLabel * **label_3**
- QListWidget * **listWidget**
- QLabel * **label_15**
- QPushButton * **pushButton_6**
- QPushButton * **pushButton_12**
- QGroupBox * **groupBox_2**
- QPushButton * **pushButton_2**
- QLabel * **label_4**
- QLabel * **label_5**
- QListWidget * **listWidget_2**
- QLabel * **label_6**
- QLabel * **label_7**
- QLineEdit * **lineEdit**
- QLabel * **label_8**
- QPushButton * **pushButton_3**
- QPushButton * **pushButton_4**
- QLabel * **label_9**
- QLabel * **label_10**
- QPushButton * **pushButton_5**
- QLabel * **label_28**
- QPushButton * **pushButton_13**
- QGroupBox * **groupBox_3**
- QRadioButton * **radioButton**
- QLabel * **label_11**
- QLabel * **label_12**
- QLabel * **label_13**
- QRadioButton * **radioButton_2**
- QLabel * **label_14**
- QLabel * **label_16**
- QPushButton * **pushButton_7**
- QLabel * **label_19**
- QLabel * **label_20**
- QPushButton * **pushButton_11**
- QGroupBox * **groupBox_4**
- QLabel * **label_17**
- QPushButton * **pushButton_9**
- QPushButton * **pushButton_10**
- QLabel * **label_18**

The documentation for this class was generated from the following file:

- app/ui_helpwindow.h

3.11 Ui_MainWindow Class Reference

Inheritance diagram for Ui_MainWindow:

Public Member Functions

- void **setupUi** (QMainWindow *MainWindow)
- void **retranslateUi** (QMainWindow *MainWindow)

Public Attributes

- QWidget * **centralwidget**
- QGroupBox * **groupBox**
- QListWidget * **imageList**
- QPushButton * **loadImagesButton**
- QPushButton * **dateSortButton**
- QPushButton * **nameSortButton**
- QGroupBox * **groupBox_2**
- QPushButton * **classLoad**
- QListWidget * **classList**
- QLineEdit * **classEnter**
- QPushButton * **classRemove**
- QPushButton * **classAdd**
- QPushButton * **classSave**
- QPushButton * **classAsc**
- QGroupBox * **groupBox_3**
- QPushButton * **saveButton**
- QPushButton * **restoreButton**
- QPushButton * **loadButton**
- QGroupBox * **groupBox_4**
- QPushButton * **helpButton**
- QPushButton * **copyButton**
- QGroupBox * **groupBox_5**
- QRadioButton * **drawShape**
- QRadioButton * **moveShape**
- QLabel * **imageDisplay**
- QGraphicsView * **graphicsView**
- QStatusBar * **statusbar**

The documentation for this class was generated from the following file:

- app/ui_mainwindow.h

Index