Project Report

SOFT20091: Software Design & Implementation

Lab Tutor: David Adama

Group 33:

Jurini Alagoa

Lovro Krapic

Sami Ljimari

Arantes Alcue

**Abstract**

The assignment for Software Design & Implementation was to design and implement a labelling application (software) using C++ for annotating datasets used by CNNs. The chosen built platform for the project was QT creator due to the abundance of readily available tools that are accessible with in the application.

The decision was made to create this application that provides the ability to provide the appropriate labelling functionality. The basic premise of the app is to allow the user to the user to have full access to a GUI which was made in QT creator. Within this they are allowed to browse and choose image files. Which allows the user to sort by ascending name, descending name, ascending date and lastly descending date. They can also see the images in the image pane, which is created within the GUI, and have the ability to clear the image scene/the image list too. The user will also have the ability to browse classes, add/or remove classes and sort them accordingly. The user can also make annotations using within the images and have the ability to draw and label on the images which would be later on discussed in this report.

The program doesn’t have a database but has the ability/functionality of multi-threading to be able to save images/image annotations quickly.

**Change History:**

Table 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Issue Date** | **Stage** | **Changes** | **Authors** |
| 1 | 22/02/2020 | D3 | Deliverable  Edited | Sami |
| 2 | 29/03/2020 | D4 | Deliverable  Edited | Arantes |
| 3 | 20/04/2020 | GUI | Edited | Sami |
| 4 | 22/02/2020 | Reference Manual | Added | Sami |
| 5 | 22/02/2020 | Coding/Contribution Guide | Added | Sami |
| 6 | 22/02/2020 | Generated Source Code | Added | Sami |
| 7 | 22/01/2020 | Project manager  Gantt Chart  Requirements  Risk | Added | Jurini |
| 8 | 3/03/2020 | Software Architecture  All the diagrams | Edited | Jurini |

**Contents**

Introduction 5

Background research 17

Design 20

Implementation 24

Results/Test Table 30

Discussion/Future 65

Appendix A 66

Appendix B 67

**Introduction**

1.1 Background

As a group we have managed to create a functional labelling tool, using the QT creator software. We were able to implement all its functionality in its entirety and overcome and note every error we face using various test plans which will be covered throughout this report. Additionally, what we have been able to learn from this project is that is very possible to work as a team, even though we faced my adversities we were able to overcome them by splitting the work into bits and covered them using a structured Gantt chart.

The labelling tool application has been developed to address the annotation of Datasets for Computer Neural Networks (CNNs). We were not expected to implement or use a CNN for this specific application. However, we had to implement an application that provides the appropriate labelling functionality. This app is designed to allow users to browse and choose image files. Which allows the user to sort by ascending name, descending name, ascending date and lastly descending date. They can also see the images in the image pane, which is created within the GUI, and have the ability to clear the image scene/the image list too. The user will also have the ability to browse classes, add/or remove classes and sort them accordingly. The user can also make annotations using within the images and have the ability to draw and label on the images. The application was designed over a number of stages involving GUI design, various applications such as Team Gantt for the management of the teams, Facebook Messenger for getting in contact with the teams, Trelllo for setting certain tasks for the team to solve and Slack another way of communication with the team.

The main programming behind our application functionality is programmed in C++ using QT Creator.

There are a lot of programs that over a labelling tool functionality, however our program is different because we followed a specific structure which was given to us as deliverables which we had to follow inconclusion with specific task instructions. After conducting a lot of research online, learning how to use the QT creator app, we have been able to complete and execute all the tasks

1.2 Aim

To design and create a functioning labelling tool application, which allows users to browse and choose image files. Which allows the user to sort by ascending name, descending name, ascending date and lastly descending date. They can also see the images in the image pane, which is created within the GUI, and have the ability to clear the image scene/the image list too. The user will also have the ability to browse classes, add/or remove classes and sort them accordingly. The user can also make annotations using within the images and have the ability to draw and label on the images. The application must be user friendly and easy to use but look professional and provide its functions.

1.3 Objectives

The following lists of objectives includes the various tasks which we have to complete to ensure that the reequipments needed were met:

Table 1.2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Description** | **Implication** | **Tasks** |
| 1. | Must be able to target a folder containing photos, using buttons as a way to browse the folder in the Operating System | Making use of the buttons to obtain the contents in the folder and displaying the pictures in the GUI. | **Task 1**  Learn to operate and interpret values from QT. |
| 2. | Must be able to list the compatible image files (e.g \*.jpg, \*.png) in the folder created.   * Sort descending by file name; | Finding and creating the correct file path for the required images needed in the coursework. | **Task 2**  Developing the correct code that allows sorting algorithms for the necessary files. For example, sorting by descending the file by the name. |
|  | Must be able to list the compatible image files (e.g \*.jpg, \*.png) in the folder created.   * Sort ascending by file name; | Finding and creating the correct file path for the required images needed in the coursework. | Developing the correct code that allows sorting algorithms for the necessary files. For example, sorting by ascending the file by the name. |
|  | Must be able to list the compatible image files (e.g \*.jpg, \*.png) in the folder created.   * Sort descending by file date; | Finding and creating the correct file path for the required images needed in the coursework. | Developing the correct code that allows sorting algorithms for the necessary files. For example, sorting by descending the file by the date. |
|  | Must be able to list the compatible image files (e.g \*.jpg, \*.png) in the folder created.   * Sort ascending by file date; | Finding and creating the correct file path for the required images needed in the coursework. | Developing the correct code that allows sorting algorithms for the necessary files. For example, sorting by ascending the file by the date. |
| 3. | Classes with browser button, so user can navigate through the folder. | Allows the user to be able to move through the folders. | **Task 3**  The classes in the GUI should have buttons to allow the user prompt the folder navigation files. |
|  | They can select the class files (plain texts with extensions “\*. names”. It must correspond to the class. Types of class car, dog, cat, person | Allows the user to be able to move through the folders. | The classes in the GUI should have buttons to allow the user prompt the folder navigation files. These classes files in the folder should be saved as plain text. With the extension” .names. Mus also correspond with the types of class car, dog, cat person |
| 4. | Classes should be listed in a classes plane. | It should have structured file line numbers which are not easily changed in the classes file. So that they can’t be changed by the user | **Task 4**  Only have structured file line numbers which can’t change |
|  | Should be able to sort in ascending order.  Must be preserved in this order. | It should have structured file line numbers which are not easily changed in the classes file. So that they can’t be changed by the user | Classes must be listed in ascending order, it can be executed with the dates of the file names |
|  | Should be able to sort in descending order.  Must be preserved in this order. | It should have structured file line numbers which are not easily changed in the classes file. So that they can’t be changed by the user | Classes must be listed in descending order, it can be executed with the dates of the file names |
| 5. | Must be able to allow the user to add classes | The classes files can be added and removed by the user. This give the user the freedom to do this in the GUI. By doing this they can append them to their various classes’ files | **Task 5**  The GUI should allow the user to add classes. By creating specific algorithms to implement this function. |
|  | Must be able to allow the user to remove classes | The classes files can be added and removed by the user. This give the user the freedom to do this in the GUI. By doing this they can append them to their various classes’ files | The GUI should allow the user to remove classes. By creating specific algorithms to implement this function. |
|  | Must be able to allow the user to append them to the classes file. | The classes files can be added and removed by the user. This give the user the freedom to do this in the GUI. By doing this they can append them to their various classes’ files | The GUI should allow the user to append classes. By creating specific algorithms to implement this function. |
| 6. | Must be able to select and use one of the various shapes:   1. Triangle 2. Square/rectangle 3. Trapezium 4. Polygon (with up to 8 points); | Using the Open CV, we are able to draw and implicate the shapes various shapes into our code. | **Task 6**  Learn to operate and interpret values from OPEN CV. |
| 7. | Must only use shapes provided for annotating the given images. Users must also be able to select any images and draw a shape on the top of the image. Borders must be visible. | Making use of the Open CV, relevant information will be shared in OPEN CV. Doing so, this will allow code to be easily accessed by other platforms. | **Task 7**  Ensure that all the code within the system is shared in the OPEN CV. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirements** | **Description** | **Implication** | **Tasks** |
| 8. | The annotations(shapes) should be displayed on the image. | These shapes will be displayed on the GUI after the images have been implemented. | **Task 8**  Improve the code which enables the shapes to be implemented properly. |
| 9. | The annotation file/file name select must have these following options:   1. Open/load annotation files (\*. annotations) | The files should be able to load and open the annotations in the GUI. | **Task 9**  Developing the correct code that allows open/loading algorithms for the necessary files. For example, opening and loading the file by the name. |
|  | The annotation file/file name select must have these following options:  2. Save the annotation file. Warning must be displayed when the user chooses to overwrite the file. | The files should be able to save the annotations in the GUI and should be seen in the file extension (. \*annotations). It should throw an error message if the user chooses over writes a file | Developing the correct code that allows save algorithms for the necessary files. For example, opening and loading the file by the name. It should throw an error message if the user chooses over writes a file |
|  | The annotation file/file name select must have these following options:  3. Change the name of an existing file | The files should be able to change the annotations in the GUI and should be seen in the file extension. | Developing the correct code that allows change algorithms for the necessary files.  For example, creating code that can change an existing file name. |
|  | The annotation file/file name select must have these following options:   1. Must have a hierarchical data format 5 (HDF5 standard). | Hierarchical Data Format is a set of file formats designed to store and organize large amounts of data | Developing the correct code with allows us to allows us to store large datasets using HDF5. |
|  | The annotation file/file name select must have these following options:   1. The data must be stored in each of the annotation file:   Number of annotated images for each image | By using the right dataset this will enable us to be able to annotate the number of images in the GUI for each of the images | Developing the correct code that allows us to use correct datasets. |
|  | The annotation file/file name select must have these following options:  5b. The data must be stored in each of the annotation file:  Image file name | Using the correct file name would make it easier to locate and store images. | Developing the correct code that allows us to use correct datasets. |
|  | The annotation file/file name select must have these following options:  5c. The data must be stored in each of the annotation file:  Number of shapers per image  Shape type  Point\_1 (x,y);  Point\_2 (x,y);  Point\_n (x,y); | Make an annotation file. | Developing the correct code that allows us to use correct datasets. |
| 10. | The image selected should be displayed in an image pane. | The user will be able to select the image. In an image pane. This enables them to know what images have been selected. | **Task 10**  Ensure the correct file paths have been created. |
| 11. | Must perform shape operation using the move.   1. Increase size | The user will be able to manually control the GUI and see what happens in the GUI | **Task 11**  Develop a Navigational algorithm for the system. That allows the shape to increase in size. |
|  | Must perform shape operation using the move.  2. Move the vertices of the polygons | The user will be able to manually control the GUI and see what happens in the GUI | Develop a Navigational algorithm for the system. That allows the user to move the vertices of the polygons |
|  | Must perform shape operation using the move.  3. Delete shape | The user will be able to manually control the GUI and see what happens in the GUI | Develop a Navigational algorithm for the system. That allows the user to delete the shape |
|  | Must perform shape operation using the move.  4. Copy and paste shape | The user will be able to manually control the GUI and see what happens in the GUI | Develop a Navigational algorithm for the system. That allows the user to copy and paste the shape in the GUI |
|  | Must perform shape operation using the move.  5.Visualise the name of the class on the top of the shape | The user will be able to manually control the GUI and see what happens in the GUI | Develop a Navigational algorithm for the system. That allows the shape to display its class |
| 12. | Must use threads to autosave files. | This saves the user time for when they manipulate images in the GUI. | **Task 12**  Improve algorithms of the systems. |
| 13. | Must use the data structures developed in 1st term for storing data in the memory | By using the appropriate data structure this will enable the project to have a structure  E.g. binary Trees, linked lists and Hash tables. | **Task 13**  Look for the best/fastest data structure to use. |
| 14. | Must use a sort and a search algorithm | This allows a more structural approach to the project. It allows the user to find things within the GUI easily | **Task 14**  Developing the correct code that allows sorting/searching algorithms for the necessary files. |

1.4 Background research

Since we coded everything using QT creator, we didn’t use any external tools and libraries to implement in our application. Down below will be screenshots of the libraries we used to implement a functioning program;

Image 1.1

A screenshot of a social media post

Description automatically generated

This is an example of the libraries we used in our code. As you can see in the screenshot, we’ve added a ‘functions.h’ file where it contains the lists of libraries.

Image 1.2

A screenshot of a computer

Description automatically generated

Image 1.3

A screenshot of a computer

Description automatically generated

Image 1.4

A screenshot of a computer

Description automatically generated

Image 1.5

A screenshot of a cell phone

Description automatically generated

**Design**

2.1 List of requirements

Once we figured out how to design and creating the labelling application, our group continued to research about how to implement the specific functions. We additionally come up with a table of requirements which gave use a structured approach of how to implement the specific functions (see table 1.2). This table contains a Description, which explains to the team what needs to be done, an Implementation, which explains the necessary functionality that needs to be executed and tasks, which breaks down the description even further to give a better understanding of how to implement the requirement.

2.2 List of tasks

The tasks were created to help the group by giving us a better understanding of what the requirement is asking us to do (see table 1.2). By creating these tasks, we were able to meet what was required from us as a group and we were able to execute. Additionally, the tasks also gave us a road map of what potential we could use to implement specific requirements. For example, for implementing shapes/resizing we could have used OpenCV however QT creator has its own library that allows us to include this function for annotating the images.

2.3 Time plan

The planning was very important for our teams. As for any project, we needed a time scale so we would be able to hit targets on time. We came up with a plan to create a two GANTT charts. 1 for the whole date/duration of the project. 2 for the requirement/individual task for the group to execute.

Image 1.6

A screenshot of a social media post

Description automatically generated

This is an image showing us the whole duration of the project. This is to understand and keep in mind the time we had to complete the whole project.

Image 1.7

A screenshot of a cell phone

Description automatically generated

Image 1.8

A screenshot of a cell phone

Description automatically generated

Image 1.9

A screenshot of a cell phone

Description automatically generated

Image 2

A screenshot of a cell phone

Description automatically generated

These are screenshot showing us the 2nd Gantt chart. The second Gantt chart was made specifically for hitting requirement targets. This also gave us an idea of which team member is working on what requirement etc. This Gantt chart also allows us to know the date of which team member is going to finish their work soon, so there will be no overlapping work. We also updated the Gantt chart every week.

2.3 Adopted coding stands

Check folder (coding and contribution guide) to find adopted coding standards we used to program our work.

**Implementation**

3.1 Use case

Image 3

A picture containing text

Description automatically generated

The use case diagram depicts how a user would interact with the software. 2 main steps are to load an image / images and class / classes. Once this is done images and classes can be sorted by descending or ascending order in their respective pane according to name or file size. Additionally, classes can be added or removed. After this segment the user will annotate the selected number of images with shapes consisting from 3 to 8 vertexes. For each image an annotation file will be created which can be opened and loaded by the user. User can alter these files, in particular their content and file names + save any updates made.

3.2 GUI framework

Image 3.1

![A screenshot of a social media post

Description automatically generated]()

We created a GUI framework; the whole purpose of the framework was to give us an understanding of how our project was going to use. We used this example as you can see from the screen shot as a starting point. Which, later on has been improved upon

3.3 Class Diagram

Image 3.2

A close up of a map

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Diagram depicts the labelling software’s class diagram. 6 Class shapes have been added to represent the 6 classes identified by our team for the system. First Class the GUI contains no attributes and one function called displaysoftwareinterface():void and is used by the User Class. The user class contains id attribute as well as selectImage():void +annotateImage():void +selectShape():void +saveAnnotationFiles():void +loadAnnotationFiles():void +browseClasses():void functions. Classes Annotation, Shapes and Classes are used by the user along with their respective functions. Images class is annotated by the users

A close up of a logo

Description automatically generated

Diagram depicts the labelling software’s component diagram. Diagram contains depiction of 5 components which are user, GUI, Image , Class and shape component. Dependency symbol serves to demonstrate the user and GUI´s dependencies on both the image and the class component. Image component provides image manipulation interface while class component provides class manipulation interface. Shape component requires shape manipulation interface while the same interface is produced by the image component

3.4 Sequence Diagram

Image 3.3

A screen shot of a computer

Description automatically generated

The sequence diagram above gives us a better understanding of how we would map out our whole project in order to meet all the requirements. The diagram follows the MODEL, CONTROL, VIEW structure.

3.5 State Diagram

Image 3.4

A screenshot of a cell phone

Description automatically generated

The process starts with the user clicking on the Browse button in the image pane of GUI. The image pane browse button will trigger a binary tree function to open and browse files for desired image / images. Once the user is satisfied with the choice, the files get sent to the classes pane where the user can create any number of classes and edit them, these autosave on every iteration. If at least one class is created, and every class is created correctly, the data gets passed to the program for computation, where object recognition takes place. Once computing stage terminates the results are passed into a transition state, called “waiting” room. Objects computed is compared to the initial input of classes and if conditions check out, user is prompted with the results, data gets autosaved and program terminates. If this is not the case and an error occur, user is prompted with the error and taken back to the initial stage.

**Result/Testing Table**

**Test Table 1**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 1 | Description: | **Image Search**: Searching 3 Images from checking if |
| Test type | Quantity/Quality | Success Criteria: | Images displayed are equal to the selected images |
| Number of Attempts: | 3 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the Deliverable 4-Test-Search folders located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Contact Software Developer and redo the Image Selection code. | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successful | | |
| Date | 29/03/20 | | |

**Test Table 1.1**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 2 | Description: | **Linked List Test**: Add selected images to a Linked list and check the number to confirm in the list to confirm it’s the same as the number of selected images. |
| Test type | Quantity/Quality | Success Criteria: | Number of linked items equal to number of chosen images/integers |
| Number of Attempts: | 3 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the Deliverable 4-Test-LinkedList folders located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Notify Software developer and redo the Linked list. | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Sucessfull | | |
| Date | 29/03/20 | | |

**Test Table 1.2**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 3 | Description: | **Sort By Ascending name**: Sort selected three images by name in ascending order. |
| Test type | Quantity/Quality | Success Criteria: | New sort equal to ascending order |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Notify Software developer and redo the Ascending name Sort. | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result |  | | |
| Date | 29/03/20 | | |

**Test Table 1.3**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 4 | Description: | **Sort By Descending name**: Sort selected three images by name in Descending order. |
| Test type | Quantity/Quality | Success Criteria: | New sort equal to Descending name order |
| Number of Attempts: | 5m | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Notify Software developer and redo the Descending name Sort. | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test table 1.4**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 5 | Description: | **Sort By Ascending date**: Sort selected three images by date in Ascending order. |
| Test type | Quantity/Quality | Success Criteria: | New sort equal to Ascending date order |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Notify Software developer and redo the Ascending date Sort. | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 1.5**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 6 | Description: | **Sort by Descending date**: Sort selected three images by date in Descending order. |
| Test type | Quantity/Quality | Success Criteria: | New sort equal to Descending date order |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Notify Software developer and redo the Descending date Sort. | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 1.6**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 7 | Description: | **Class Selection**:  Store a text as a variable  Add stored test to the list view  Check if Class added to list view is equal to the selected class |
| Test type | Quantity/Quality | Success Criteria: | If class added to List view is equal to the selected class |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Contact the software tester and redo the class selection | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 1.7**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 8 | Description: | **Class sort by Ascending**:  Sort a list of class texts in ascending order |
| Test type | Quantity/Quality | Success Criteria: | If class added to List view is equal to the selected class |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Contact the software tester and redo the class ascending function | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 1.8**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 9 | Description: | **Class sort by Descending**:  Sort a list of class texts in descending order |
| Test type | Quantity/Quality | Success Criteria: | If class added to List view is equal to the selected class |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Contact the software tester and redo the class descending function | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 1.9**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 10 | Description: | **Class remove**:  Remove Class |
| Test type | Quantity/Quality | Success Criteria: | If class added to List view is equal to the selected class |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Contact the software tester and redo the class removal | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a cell phone  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 11 | Description: | **Quadrilaterous Annotation**:  Click on Triangle button  Check if shape showed on the scene is equal to the square. |
| Test type | Quantity/Quality | Success Criteria: | If shape shown on scene is equal to the selected shape |
| Number of Attempts: |  | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and do redo the Quadrilaterous annotation | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.1**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 12 | Description: | **Triangle Annotation**:  Click on triangle button  Check if shape showed on the scene is equal to the triangle. |
| Test type | Quantity/Quality | Success Criteria: | If shape shown on scene is equal to the selected shape |
| Number of Attempts: |  | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and do redo the triangle annotation | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.2**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 13 | Description: | **Pentagon Annotation**:  Click on Pentagon button  Check if shape showed on the scene is equal to the Pentagon. |
| Test type | Quantity/Quality | Success Criteria: | If shape shown on scene is equal to the selected shape |
| Number of Attempts: |  | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and do redo the pentagon annotation | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.3**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 14 | Description: | **Octagon Annotation**:  Click on Octagon button  Check if shape showed on the scene is equal to the Pentagon. |
| Test type | Quantity/Quality | Success Criteria: | If shape shown on scene is equal to the selected shape |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and do redo the pentagon annotation | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a dog  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.4**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 15 | Description: | **Annotation** file open and load: Load and open of previously saved annotation file |
| Test type | Quantity/Quality | Success Criteria: | If opened annotation file is same as saved |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the annotation file open and load | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a computer screen  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.5**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 16 | Description: | **Annotation** save: Save annotation of image including shape position used to annotate |
| Test type | Quantity/Quality | Success Criteria: | If saved annotation is equal to the made annotation |
| Number of Attempts: | 5 | Comments: | Sucessfull |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the annotation file save | | |
| Engineer(s)Technicians | Alceu Caterça, Sami Ljimari, Jay Alagoa and Lovro Krapic | | |
| Individual Results | A screenshot of a computer  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.6**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 17 | Description: | Annotation File name change: Change name of previously annotated file |
| Test type | Quantity/Quality | Success Criteria: | If new name of annotation file same as the name expected |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the annotation file name change. | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.7**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 18 | Description: | **Shape Increase**:  Increase number of the edges of the shape  Check if increased number is bigger than the original |
| Test type | Quantity/Quality | Success Criteria: | If increase number is greater than original number |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and add the shape increase | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 2.8**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 19 | Description: | **Vertices Move;**  Rotate the vertices of shapes |
| Test type | Quantity/Quality | Success Criteria: | If rotated number is equal to the coordinates of the new annotation. |
| Number of Attempts: | 0 | Comments: | Fail |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and add the vertices rotation | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results |  | | |
| Result | UnSuccessfull | | |
| Date | 29/03/20 | | |

**Test Table 2.9**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 20 | Description: | **Shape Deletion: Delete annotated shape** |
| Test type | Quantity/Quality | Success Criteria: | If original annotation shape is no longer on the image. |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the shape deletion function | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results |  | | |
| Result | successfull | | |
| Date | 29/03/20 | | |

**Test Table 3**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 21 | Description: | **Shape** Copy and Paste: Copy latest shape and paste it onto the image |
| Test type | Quantity/Quality | Success Criteria: | If original annotation shape is no longer on the image. |
| Number of Attempts: | 5 | Comments: | Successful |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the copy and paste function | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results | A screen shot of a city  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 3.1**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 22 | Description: | Class Visualization: Visualize the name of the class on top of the shape |
| Test type | Quantity/Quality | Success Criteria: | If the name of the class is visible |
| Number of Attempts: | 5 | Comments: | success |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the class visualization | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results | A screenshot of a social media post  Description automatically generated | | |
| Result | Successfull | | |
| Date | 29/03/20 | | |

**Test Table 3.2**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 23 | Description: | Annotation Save. Save the annotations every minute. |
| Test type | Quantity/Quality | Success Criteria: | If the file is saved after every minute |
| Number of Attempts: | 5 | Comments: | fail |
| List of Equipment/Requirements | Laptop: Lenovo Legion Y5  Operating System: Windows  File Directory: SDI/GUI and SDI/Deliverable 4  Widget toolkit: QT 4.11.1 | | |
| Setup Instructions | Open the QT version 4.11.1  On the file tab (located on the upper left of qt) click open file or project  Locate the SDI file  Open the GUI folder located in the SDI file  Open all files inside simultaneously onto QT | | |
| Failure correction Procedure | Alert the software developer and redo the every minute save feature | | |
| Engineer(s)Technicians | Alceu Caterça | | |
| Individual Results |  | | |
| Result | Unsuccessfull | | |
| Date | 29/03/20 | | |

**Conclusion/Future work**

As previously stated throughout this report, ‘Labelling Tools’ is a unique way of user getting a chance to see how a labelling tool works. This application works by allowing the user to choose whatever image they want and are able to draw on it. The application works off the interactive GUI that allows users to browse for images. For example, if a user chose an image, they are able to display it on the image pane, this is where the user is able to actively label on the image and see how the labelling tool works. The application contains some great features such as sort the chosen images in ascending (name), ascending (date), descending (name), descending (date). Clearing the image lists too. The labelling tool application is a very user-friendly application that appeals to everyone

As a team we have learnt and achieve a lot from this project, the one thing we learn was having a good GUI is an essential key to the project because that is what the users are going to be interacting with. So, by making it appealing and attractive to the users, they will be able to use properly. Also, the project has shown us how everything connects with the requirements to help make our application work and function properly. The main reason why our group was able to function well was because of how we structured our group meetings. The although the meetings were not carried out on a daily basis, it was tracked, and we executed the meetings every Mondays and Tuesdays. The group also used social media like Facebook to allow communication between each other. Since, we had such a good relationship with each other completing the work was quite easy. This already enabled us to secure good communication between each other which was the contributing factor of our success.

The app could be developed and further improved by implementing Convolutional Neural Networks (CNN) for performing online/offline object classification. The app could be more unique from other projects if we were able to perform online/offline object classification. This would be to give the user a better idea of how a professional labelling tool works. If we had more time, I believe our team could have been able to implement that but sadly we left it out. The group had some limitations, this was seen as not everyone is used to developing an application. This meant that we had to all be over achievers, which had a huge effect to our application. This however positively impacted our group because we were able to learn new skills, which could help us with future projects. During the project, the team used only QT creator to implement and create all the functionality of the GUI. QT creator was used for the coding of the application, our group members found this really hard because it involved using the programming language C++ which a lot of people have not had time to learn properly before.

Overall, our group performed well, we worked as a team. This required a lot of hard work and dedication to maintain the high level of work.

**Appendix A**

**Reference Manual**

**Look in Reference Manual Folder for more details**

**Appendix B**

**Testing Plan**

The Test Plan is designed to prescribe the scope, approach, resources, and schedule of all testing activities of our image annotator software

The plan is to identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

**1.Test Strategy**

1.Test Scope

* + 1. **Feature to be tested**

The features that are going to tested are all the requirements listed in the software requirements.

|  |  |  |
| --- | --- | --- |
| **Requirements** | **Description** | **Implication** |
| 1. Obtain Photos from folder | Must be able to target a folder containing photos, using buttons as a way to browse the folder in the Operating System | Making use of the buttons to obtain the contents in the folder and displaying the pictures in the GUI. |
| 1. List Compatible Images | Must be able to list the compatible image files (e.g \*.jpg, \*.png) in the folder created.   * Sort descending by file name; | Finding and creating the correct file path for the required images needed in the coursework. |
| 3. | Classes with browser button, so user can navigate through the folder. | Allows the user to be able to move through the folders. |
|  | They can select the class files (plain texts with extensions “\*. names”. It must correspond to the class. Types of class car, dog, cat, person | Allows the user to be able to move through the folders. |
| 4. | Classes should be listed in a classes plane.  Should be able to sort in ascending order. | It should have structured file line numbers which are not easily changed in the classes file. So that they can’t be changed by the user |
| 5. | Must be able to allow the user to add classes | The classes files can be added and removed by the user. This give the user the freedom to do this in the GUI. By doing this they can append them to their various classes’ files |
| 6. | Must be able to select and use one of the various shapes:   1. Triangle 2. Square/rectangle 3. Trapezium 4. Polygon (with up to 8 points); | Using the Open CV, we are able to draw and implicate the shapes various shapes into our code. |
| 7. | Must only use shapes provided for annotating the given images. Users must also be able to select any images and draw a shape on the top of the image. Borders must be visible. | Making use of the Open CV, relevant information will be shared in OPEN CV. Doing so, this will allow code to be easily accessed by other platforms. |
| **Requirements** | **Description** | **Implication** |
| 8. | The annotations(shapes) should be displayed on the image. | These shapes will be displayed on the GUI after the images have been implemented. |
| 9. | The annotation file/file name select must have these following options:   1. Open/load annotation files (\*. annotations) 2. 2. Save the annotation file. Warning must be displayed when the user chooses to overwrite the file. | The files should be able to load and open the annotations in the GUI. |
|  | The annotation file/file name select must have these following options:  2. Save the annotation file. Warning must be displayed when the user chooses to overwrite the file. | The files should be able to save the annotations in the GUI and should be seen in the file extension (. \*annotations). It should throw an error message if the user chooses overwrites a file |
|  | The annotation file/file name select must have these following options:  3. Change the name of an existing file | The files should be able to change the annotations in the GUI and should be seen in the file extension. |
|  | The annotation file/file name select must have these following options:   1. Must have a hierarchical data format 5 (HDF5 standard). | Hierarchical Data Format is a set of file formats designed to store and organize large amounts of data |
|  | The annotation file/file name select must have these following options:   1. The data must be stored in each of the annotation file:   Number of annotated images for each image | By using the right dataset this will enable us to be able to annotate the number of images in the GUI for each of the images |
|  | The annotation file/file name select must have these following options:  5b. The data must be stored in each of the annotation file:  Image file name | Using the correct file name would make it easier to locate and store images. |
|  | The annotation file/file name select must have these following options:  5c. The data must be stored in each of the annotation file:  Number of shapers per image  Shape type  Point\_1 (x,y);  Point\_2 (x,y);  Point\_n (x,y); | Make an annotation file. |
| 10. | The image selected should be displayed in an image pane. | The user will be able to select the image. In an image pane. This enables them to know what images have been selected. |
| 11. | Must perform shape operation using the move.   1. Increase size | The user will be able to manually control the GUI and see what happens in the GUI |
|  | Must perform shape operation using the move.  2. Move the vertices of the polygons | The user will be able to manually control the GUI and see what happens in the GUI |
|  | Must perform shape operation using the move.  3. Delete shape | The user will be able to manually control the GUI and see what happens in the GUI |
|  | Must perform shape operation using the move.  4. Copy and paste shape | The user will be able to manually control the GUI and see what happens in the GUI |
|  | Must perform shape operation using the move.  5.Visualise the name of the class on the top of the shape | The user will be able to manually control the GUI and see what happens in the GUI |
| 12. | Must use threads to autosave files. | This saves the user time for when they manipulate images in the GUI. |
| 13. | Must use the data structures developed in 1st term for storing data in the memory | By using the appropriate data structure this will enable the project to have a structure  E.g. binary Trees, linked lists and Hash tables. |
| 14. | Must use a sort and a search algorithm | This allows a more structural approach to the project. It allows the user to find things within the GUI easily |

* 1. **Feature not to be tested**

These features are not be tested because they are not included in the software requirement specs

-Performance

-Security

* 1. **Test Type**

There are 4 testing that will be conducted for the Image annotator

**Unit Testing**- is the execution of a complete class, routine, or small program that has been written by a single programmer or team of programmers, which is tested in isolation from the more complete system.

**Integration Testing**- the combined execution of two or more classes, packages, components, or subsystems that have been created by multiple programmers or programming teams. This kind of testing typically starts as soon as there are two classes to test and continues until the entire system is complete.

**Functional testing**-test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

**Acceptance Tests**-is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing are done.

**1.3.1 Document Risks and Issues**

|  |  |
| --- | --- |
| Risk | Mitigation |
| A team member doesn’t have the skills to do his job | Attend SDI surgery. |
| A Team member falls ill, thus no longer able to do his role | Ask Teacher for extension and have someone else form the group fill in the ill member. |
| The Software skips some key functionalities. | Re-Define the project requirement |
| The project schedule is too tight; it's hard to complete this project on time | Set Test Priority for each of the test activity |

**1.4 Test Logistics**

**1.4.1 Who will test?**

The Project will be tested by the software tester Alceu Caterça.

**1.4.2 When will test occur?**

The Tests will occur after the following Executions:

- After One working class created

- After Second working class created

- After All the functions of the Image Annotator are created

- After Project is completed

1.4.3. Use Case Scenarios

Class Diagram

A close up of text on a white background

Description automatically generated

Diagram depicts the labelling software’s class diagram. 6 Class shapes have been added to represent the 6 classes identified by our team for the system. First Class the Gui contains no attributes and one function called displaysoftwareinterface():void and is used by the User Class. The user class contains id attribute as well as selectImage():void +annotateImage():void +selectShape():void +saveAnnotationFiles():void +loadAnnotationFiles():void +browseClasses():void functions. Classes Annotation, Shapes and Classes are used by the user along with their respective functions. Images class is annotated by the users.

State DiagramA screenshot of a cell phone

Description automatically generated

The process starts with the user clicking on the Browse button in the image pane of GUI. The image pane browse button will trigger a binary tree function to open and browse files for desired image / images. Once the user is satisfied with the choice, the files get sent to the classes pane where the user can create any number of classes and edit them, these autosave on every iteration. If at least one class is created, and every class is created correctly, the data gets passed to the program for computation, where object recognition takes place. Once computing stage terminates the results are passed into a transition state, called “waiting” room. Objects computed is compared to the initial input of classes and if conditions check out, user is prompted with the results, data gets autosaved and program terminates. If this is not the case and a error occurs, user is prompted with the error and taken back to the initial stage.

A picture containing text

Description automatically generated

Use case diagram

The use case diagram depicts how a user would interact with the software. 2 main steps are to load an image / images and class / classes. Once this is done images and classes can be sorted by descending or ascending order in their respective pane according to name or file size. Additionally, classes can be added or removed. After this segment the user will annotate the selected number of images with shapes consisting from 3 to 8 vertexes. For each image an annotation file will be created which can be opened and loaded by the user. User can alter these files, in particular their content and file names + save any updates made. ´

Component Diagram

A close up of a logo

Description automatically generated

Diagram depicts the labelling software’s component diagram. Diagram contains depiction of 5 components which are user, GUI , Image , Class and shape component. Dependency symbol serves to demonstrate the user and GUI´s dependencies on both the image and the class component. Image component provides image manipulation interface while class component provides class manipulation interface. Shape component requires shape manipulation interface while the same interface is produced by the image component

**2 TEST OBJECTIVES**

The Objective of the tests is to prove the functionality of the image annotator. The tests will be focused on testing operations such as linked lists, sort and Image Search to ensure the quality of the Image Annotator.

**3 TEST CRITERIA**

**3.1 Suspension Criteria**

If the team members report that there are 40% of test cases failed, suspend testing until the development team fixes all the failed cases.

**3.2 Exit Criteria**

Specifies the criteria that denote a successful completion of a test phase - Run rate is mandatory to be 100% unless a clear reason is given. - Pass rate is 80%, achieving the pass rate is mandatory.

**4 RESOURCE PLANNING**

**4.1 Human Resource**

|  |  |  |
| --- | --- | --- |
| No. | Member | Tasks |
| 1. | Test Manager | Manage the whole project  Define project requirements |
| 2. | Software Architect | Project Design |
| 3. | Software Developer | Project Development |
| 4. | Software Tester | Execute the tests, Log results, Report the defects. |

**4.2 System Resource**

|  |  |  |
| --- | --- | --- |
| No. | Member | Tasks |
| 1. | Test Manager | Manage the whole project  Define project requirements |
| 2. | GUI | Learn Qt to develop the Image Annotator |
| 3. | Software Developer | Project Development |
| 4. | Software Tester | Execute the tests, Log results, Report the defects. |

1. **Test Environment**

* User
* GUI

1. **Schedule and Estimation**

**6.1 All project task and estimation**

|  |  |  |
| --- | --- | --- |
| Task | Members | Estimate Effort |
| Create the test plan | Tester | 150 Man-Hour |
| Perform Test Execution | Tester, Software Developer | 50 Man-Hour |
| Test Documentation | Tester | 10 Man-Hour |
| Total |  | 210 Man-Hour |

**6.2 Schedule to Complete these tasks**

A screenshot of a cell phone

Description automatically generated

**7 TEST DELIVERABLES Test deliverables are provided as below**

**7.1 Before testing phase**

- Test plan.

- Test cases documents

**7.2 During the testing**

- Test Tool - Simulators.

- Test Data

**7.3 After the testing cycles is over**

– Test Results/reports

- Installation/ Test procedures guidelines

- Release notes

1. **Bounds of Use.**

Long Period of Inactivity- Program run as expected regardless of long periods of inactivity such as When the application left opened for long period of time,

Loading of Multiple images, shapes and class- Program runs as expected regardless of the number of images, shapes and classes

9. **Stress Failure Conditions**

Shape annotator Pentagon, Hexagon, Octagon and Heptagon fail to run when the vertices of the shape all connect.