SOFT20181: SOFTWARE DESIGN AND IMPLEMENTATION (2019-2020)

Test Plan

by

Alceu Caterça

T0097093

INTRODUCTION .................................................................................................................... 3

TEST STRATEGY

Scope of Testing .....................................................................................................................3-7

1.2.1 Feature to be tested ...................................................................................................... 3-7

1.2.2 Feature not to be tested ....................................................................................... 7

1.3 Test Type ...............................................................................................................................8

1.3.1 Risk and Issues ........................................................................................................... 8-9

1.4 Test Logistics.............................................................................................................. 8-9

1.4.1 Who will test? ...................................................................................................... 9

1.4.2 When will test occur? .......................................................................................... 9

1.4.3 Use Case Scenarios……………………………………………………………………………………………………9-12

2 Test Objectives................................................................................................................ 13

3 TEST CRITERIA ................................................................................................................... 13

3.1 Suspension Criteria .................................................................................................... 13

3.2 Exit Criteria ................................................................................................................ 13

4 RESOURCE PLANNING ....................................................................................................... 13

4.1 System Resource ....................................................................................................... 13

4.2 Human Resource ....................................................................................................... 13

5 TEST ENVIRONMENT ......................................................................................................... 14

6 SCHEDULE & ESTIMATION ................................................................................................. 14

6.1 All project task and estimation ................................................................................. 14

6.2 Schedule to complete these tasks ............................................................................. 14

7 TEST DELIVERABLES ........................................................................................................... 14

7.1 Before testing phase ................................................................................................. 14

7.2 During the testing ...................................................................................................... 15

7.3 After the testing cycles is over .................................................................................. 15

8. Bounds of Use…………………………………………………………………………………………………………15

9. Stress failure condition……………………………………………………………………………………………….15

**Introduction**

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

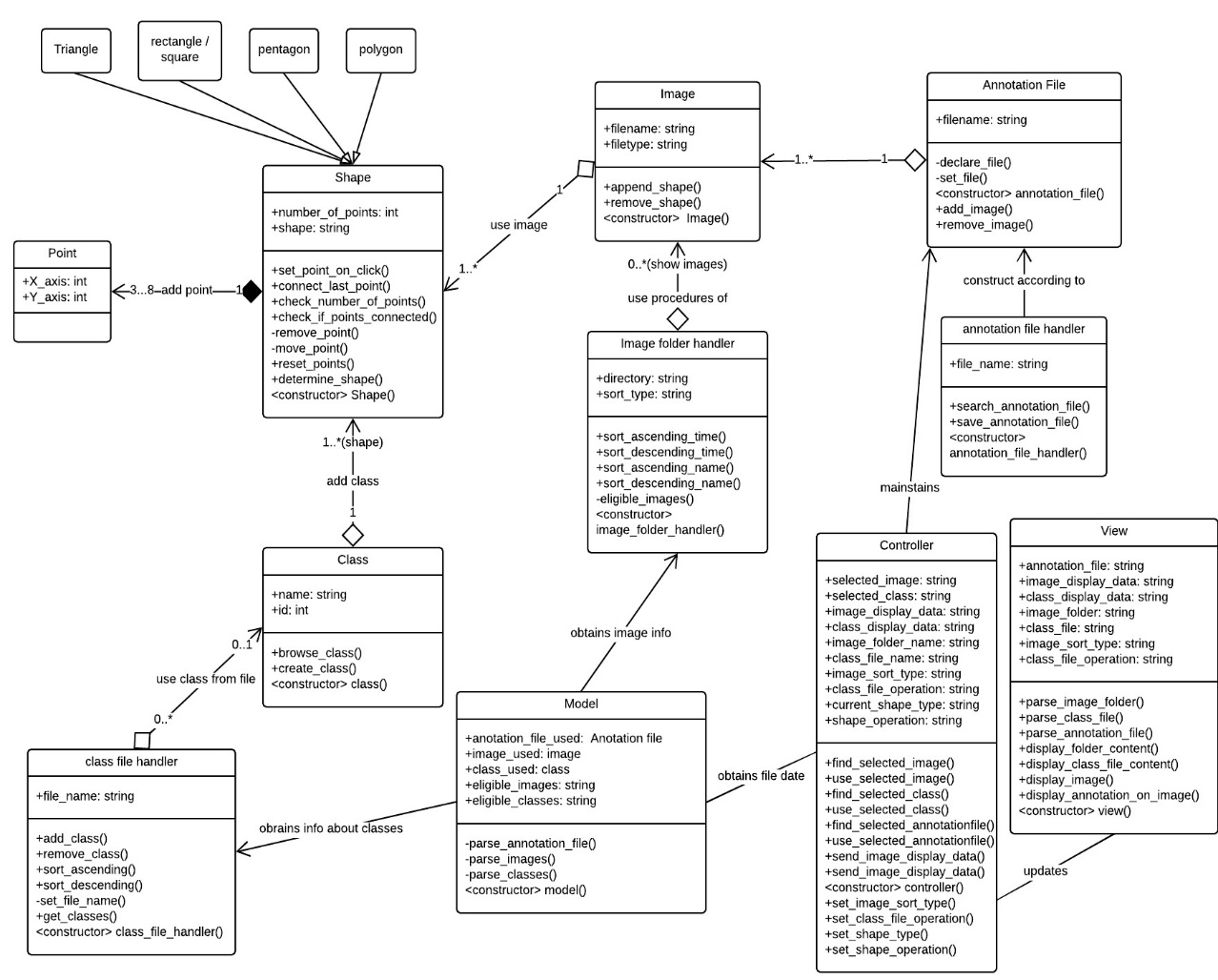
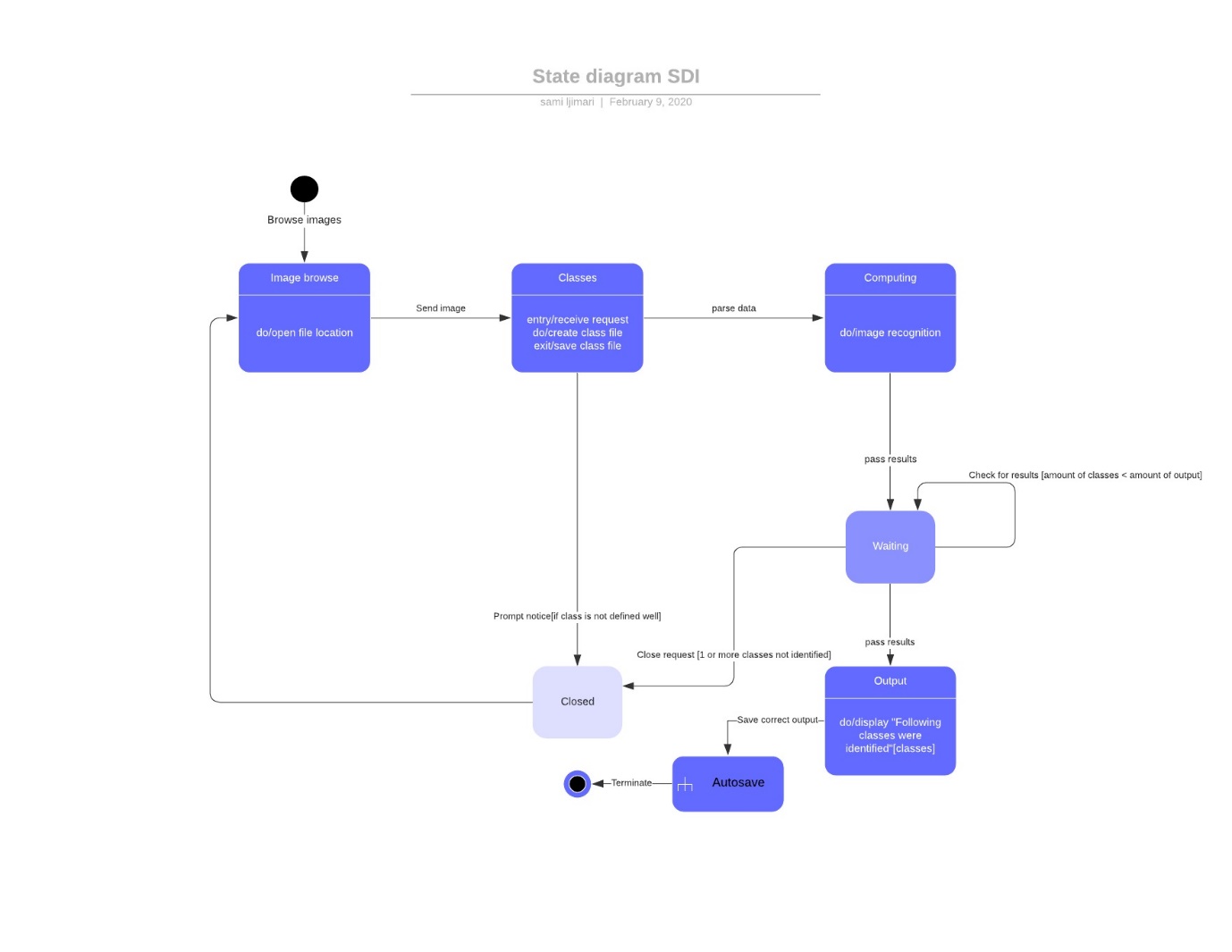
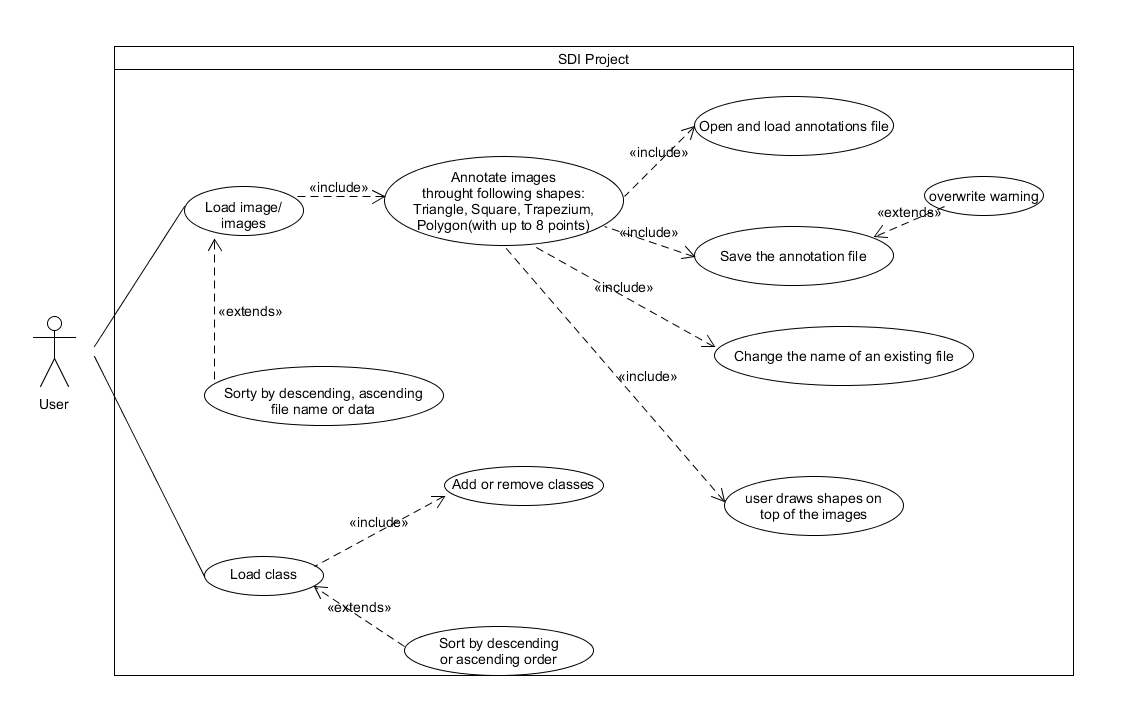


Diagram depicts the labelling software’s class diagram. 6 Class shapes have been added to represent the 6 classes identified by our team for the system. First Class the Gui contains no attributes and one function called displaysoftwareinterface():void and is used by the User Class. The user class contains 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 Diagram

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.



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

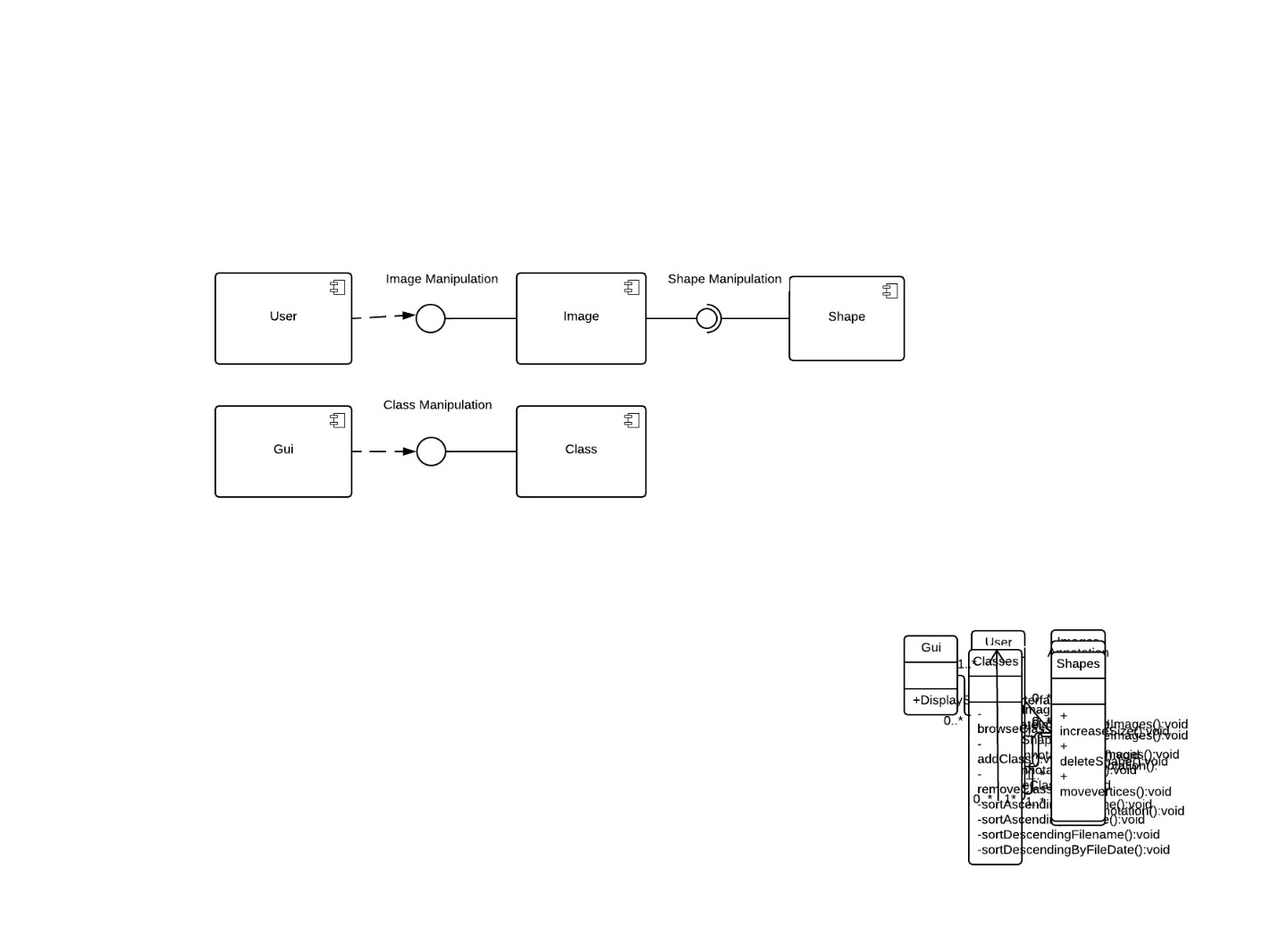


Diagram depicts the labelling software’s component diagram. Diagram contains depiction of 5 components which are user, GUI , Image , Class and shape component. Dependency symbol serves to demonstrate the user and GUI´s dependencies on both the image and the class component. Image component provides image 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**

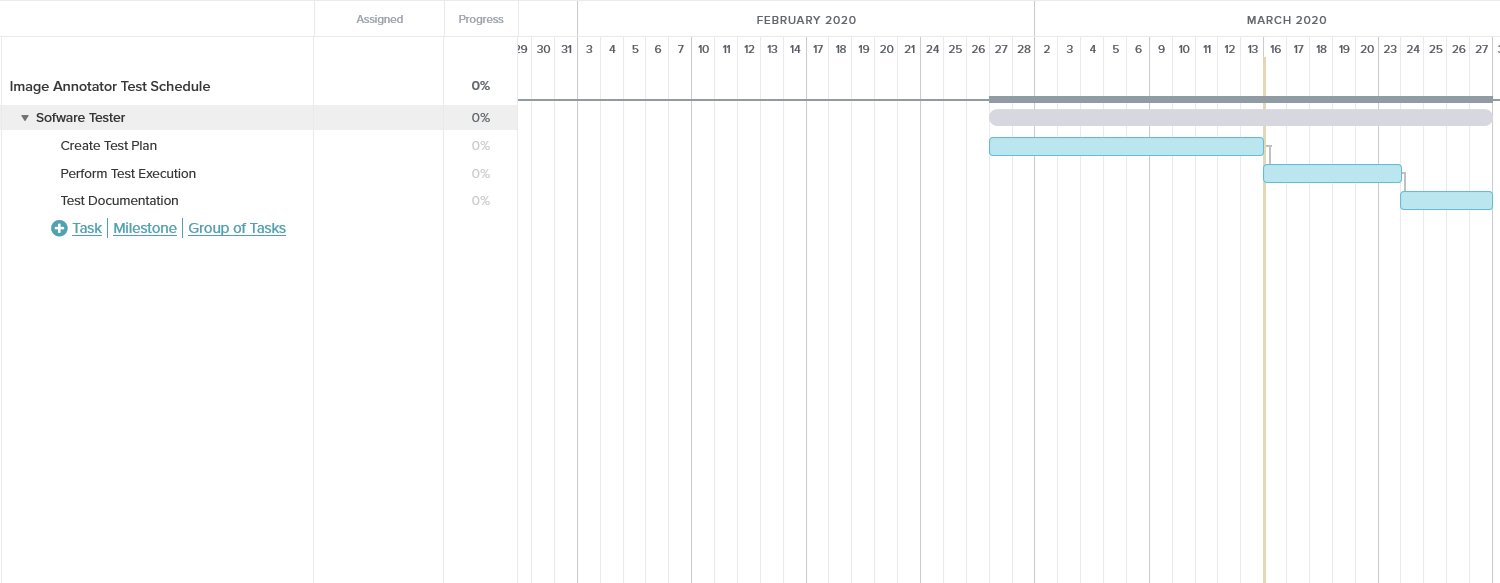
* User
* GUI

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**



**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.