Software Design and Implementation Test Plan: Neural Network Labelling Program.

Edward Whale 2018 (N0781255), Thomas Litherland (N0779393), Jamie Haywood (N0778608) , Ryan Ewart (N0727462)

# **Product Overview**

This test plan describes the testing approach and overall test framework that will be used to test the functionality and evaluate the requirements of the Neural Network Labelling application.

The Central Neural Network (CNN) Labelling application is a software application that aims to implement computer vision techniques in order to produce an efficient and streamlined image labelling program. Our software development project focuses on computer vision, the ability for a computer to interpret and understand the visual world in a similar way that human beings can interpret the objects they see. Our development team aim to utilise machine learning algorithms and convolutional neural network datasets in order to classify objects within digital images to provide an appropriate shape labelling functionality.

**Test Strategy**

Scope of Testing

In order to ensure the Convolutional Neural Networking labelling application performs the required processes, the scope of testing must be established. The test scope covers all components of the system that will be tested in order to ensure the team effectively follows the appropriate methodologies in order to keep up with the requirements within the given timeframe. The following processes and components will be evaluated within the scope of the tests.

|  |  |
| --- | --- |
| **Test case** | **Test description** |
| 1 | When the user runs the program, a simple interactive GUI is loaded within the application with an image panel, an image folder and buttons for navigating through shapes. |
| 2 | User can access the compatible image files available in a folder. |
| 3 | User can sort the compatible image files by file name in descending order. |
| 4 | User can sort the compatible image files by file name in ascending order. |
| 5 | User can sort the compatible image files by file date in descending order. |
| 6 | User can sort the compatible image files by file date is ascending order. |
| 7 | User can navigate through a class selector folder in order to select a class file where each line corresponds to a group of people or objects. |
| 8 | All classes should be listed in a panel with the option to sort in ascending order. |
| 9 | All classes should be listed in a panel with the option to sort in descending order. |
| 10 | A class must be able to be added and removed by appending the class file. |
| 11 | The user must be able to select and draw a triangle shape by using point and click. |
| 12 | The user must be able to select and draw a square/rectangle shape by using point and click. |
| 13 | The user must be able to select and draw a trapezium shape by using point and click. |
| 14 | The user must be able to select and draw a polygon shape within up to 8 points by using point and click |
| 15 | The user must only have the ability to draw the provided shapes for annotating the given images, the shape should have a visible border with no fill |
| 16 | User should be able to open and load the annotations file and filename selector |
| 17 | User should be able to save the annotation file when any changes are made. A warning should be displayed to the user and ask the user to confirm the overwrite if the file already exists |
| 18 | The following data is stored in each annotation file: Number of annotated images, image file name, number of shapes per image, shape type, point 1(x,y), point 2(x,y), point\_n(x,y) |
| 19 | The user should be able to increase the size of the shape using the mouse |
| 20 | The user should be able to move the vertices of polygons using the mouse |
| 21 | The user should be able to delete shapes using the mouse |
| 22 | The user should be able to copy and paste using the mouse |
| 23 | The user should be able to move the visualise the name of the class at the top of the shape |
| 24 | The application should autosave the annotation file every minute using threading |

Test Types

Each type of testing procedure has its own techniques, advantages and disadvantages therefore it is important to formulate several testing types in order to achieve early and efficient detections of defects and bugs continuously throughout the development of the Convolutional Neural Network labelling application. In our application, we will conduct the four following testing types in order to cover all bases.

* Unit Testing- Unit Testing will be carried out on the individual components that make up the integrated system. A unit is the smallest testable part of the software system, its purpose is to validate that each individual unit performs as it was designed according to the specification requirements.
* Integration Testing- Integration Testing will be performed after Unit Testing in order to combine individual components and test them as an integrated group to perform a specific functionality. The purpose of this stage in testing is to expose any faults and defects within the interaction between units. Our software testers will use an incremental approach to integration testing, this will allow our testers to isolate individual modules in order to establish any errors and testing can commence before all modules have been implemented into the system.
* System Testing- Once the implementation of the CNN Label ling application is completed, system testing will be carried out in order to validate the fully integrated software product. The application will be tested including the external peripherals in order to confirm all components interact with each other correctly and the program meets all the specified functional and non-functional requirements.
* Acceptance Testing- The final stage of testing will be user acceptance testing. During the acceptance testing, we will establish whether the system is suitable for operational use by the end-users and the testers will compare the application to its initial user requirements. We will perform an internal acceptance testing approach by members that were directly involved in the project.

**Risks & Issues**

Throughout the design, implementation and testing of the Convolutional Neural Networking Labelling application the development team may encounter a series of risks and issues that have the potential to endanger the objectives and schedule of the project. The following tables include potential risks and what will be done to mitigate these problems.

Product Risks

|  |  |
| --- | --- |
| **Potential Risks** | **Mitigation** |
| Software developers skip any key functions that were specified in the user requirements | Ensure all members of the development team are aware of all the user-requirements and perform effective and consistent design techniques. All team members should communicate regularly and any changes to the project should be appropriately logged |
| The software encounters problems with the interaction between the program and the hardware/software | The developers should make certain that the program is implemented for all appropriate operation systems (Windows, Mac OS, Linux). Appropriate testing should be carried out to guarantee does not clash with any peripherals |
| Complex features affecting multiple areas of the project, potentially conflicting with the functionality of the product | The design phase of product development should be clear and concise in order to ensure the programmers stay true to the user requirements and do not over-complicate the application with unnecessary features |
| Developers may lack experience therefore pose a higher risk to the project | All members of the project team should communicate their strengths and weaknesses effectively in order to allocate tasks in way that will maximise the efficiently and quality of the implementation. The project manager should regularly perform quality control in order to make sure the project is on schedule and all developers are working at an effective level. |

Project Risks

|  |  |
| --- | --- |
| **Potential Risks** | **Mitigation** |
| Project members falling behind on the given schedule may cause difficulties in meeting each deadline | The project manager should remain organised and maintain consistent and regular communication with all members of the group throughout the duration of the project. A clear and concise project schedule should be established, and all members should follow it effectively |
| Late delivery of the test items to the testing team | The software tester should ensure that they have designed a clear test plan and testing schedule so the developers have comprehensible guidelines to make certain that the developers effectively implement the features required by the testers by the given deadlines |
| Unexpected organisational problems such as a team member failing to apply themselves or sickness | The project manager should remain vigilant throughout the duration of the project in order to mitigate any potential unanticipated problems that may negatively affect the functionality of the program or the project schedule. If any unexpected problems to occur, it is the project managers responsibility to address the problems appropriately, either by handling the issue themselves of contacting the appropriate personnel (course leader) |

**Test Logistics**

Human Resource

Edward Whale – Project Manager

As the project manager, Edward Whale is responsible for leading the project and ensuring the development and testing team remain organised and communicate throughout the duration of the project. During the testing phase he will assist the software tester with compiling test cases, writing test scripts and overseeing the quality control of the testing.

Ryan Ewart – Software Tester

Ryan Ewart is the primary tester for the labelling application. His responsibilities include establishing a clear test plan that will be used to ensure the testing is clear and covers all the user specified requirements. He will prepare the test scenarios for unit, integration, system and acceptance testing. Ryan will also be responsible for writing the testing documentation.

Jamie Haywood – Software Developer

As the software developer, Jamie’s primary responsibilities are in the design and implementation of the code, therefore throughout the duration of development he will be responsible for keeping track of any bugs or issues with the program. He will be responsible for keeping the software tester up to date and informed with the status of the project so the team can stick to the project schedule.

Thomas Litherland – Software Architect

As the software architect, Thomas’ main responsibilities are designing the architecture of the project and establishing the modules and components for software development. He will assist Jamie in developing and implementing the program, therefore he will be responsible for incrementally running unit and integration tests throughout the duration of development in order to ensure the application covers all user requirements.

System Resource

|  |  |
| --- | --- |
| **Resource** | **Resource Description** |
| Computers | The testing team should utilise a variety of computer and laptop models in order to ensure compatibility across several devices (Dell, Macintosh, Linux machines) |
| Operating Systems | The system should be tested on a variety of operating systems in order to test compatibility (Windows, Mac OS, Linux) |
| Development Environment | QT Creator will be the chosen development environment for the development of the application. This will be used to design of the GUI, implementing the program and running the test scripts. |
| Testing Environment | The testing will be performed in QT creator using the C++ library, which allows the use of Boost UTF. Boost UTF is the chosen testing framework for unit testing and integration testing. |
| Collaborative development platform | GitHub will be used as a collaborative development platform for all members of the development team, we will use GitHub to access code repositories and ensure each user has the right version of the program. |

Schedules

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable** | **Member Responsibility** | **Deadline date** | **Dependencies** |
| Test plan | Ryan Ewart | 09/02/2020 | User requirements |
| Test scenarios | Ryan Ewart | 20/02/2020 | Test plan |
| Unit testing | Ryan Ewart  Edward Whale | 01/03/2020 | Implementation of beta version |
| Integration Testing | Ryan Ewart  Edward Whale | 20/03/2020 | Implementation of the beta version |
| System Testing | Ryan Ewart  Edward Whale  Jamie Haywood  Thomas Litherland | 20/04/2020 | Finalised version of the program |
| Acceptance Testing | Ryan Ewart  Edward Whale  Jamie Haywood  Thomas Litherland | 20/04/2020 | Finalised version of the program |
| Testing Documentation | Ryan Ewart | 26/04/2020 | Completion of all testing |

**Test Objectives**

The test objectives are to verify the functionality and ensure the Convolutional Neural Networking Labelling application is as bug-free as possible at its final version. Testing will be performed by the software tester based on the user specified requirements found in the Test Scope. Testing should focus on establishing whether the requirements have been met effectively. This will prevent defects and gain confidence by providing information about the level of quality.

**Test Criteria**

Suspension Criteria

The critical suspension criteria identify at what point the testing process should be suspended if a certain amount of test cases have failed. If this condition is met, the testing is halted, and the software tester will inform the developers of the issues and bugs so they can be rectified. The suspension criteria of the testing phase are if the team members report 40% of test cases have failed.

Exit Criteria

The exit criteria for the Convolutional Neural Networking labelling application denotes the successful completion of the testing phase. It establishes the targeted results that are necessary in order to complete the testing and move onto the next stage of development or release. The rune rate of this project is 100% and the pass criteria is 90%.

**Test Scenario structure**

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID: |  | Description: |  |
| Test Type: |  | Success criteria: |  |
| Number of attempts: |  | Comments: |  |
| List of equiptment/requirement: |  | | |
| Setup instructions: |  | | |
| Failure correction procedure: |  | | |
| Engineers/Technicians: |  | | |
| Individual results: |  | | |