Testing Document - Sprint 2

Version	Description	Author	Date (DD/MM/YYYY)
0.0	First draft of the Testing Document Sprint 2	AKHTAR KURNIAWAN	26/10/20

1. Introduction

1.1 Proposal

The purpose of this document is to define and present the test cases for project Robomaster AI Challenge by team RM-Koala, covering the test cases for the system use cases.

1.2 Target Users

This document is mainly designed for those responsible for executing the test cases in this project, namely the team members, client, and COMP90082 teaching team.

1.3 Conventions, terms and abbreviations

This section explains the concept of some important terms that will be used throughout this document. These terms are described in the following table, presented in alphabetical order.

Term	Description
Accurate armour identification	The program correctly identifies the type of nearest armour.
Accurate location prediction	Bounding box covers both lights of the nearest armour. It should be small enough so that it does not include the light of another armour.

2. Covered Requirements

This section lists the system requirements covered in the test cases.

User Story Id	User Story
3	As a member of the robotic team, I want my to have a Graphic User Interface (GUI) of the software so that I can evaluate the computer vision algorithms more easily.
4	As a member of the robotic team, I want the two algorithms to work in conjunction so that my robot can locate and identify the opponent's armour faster.
7	As a participant in the competition, I want my robot to locate multiple armours in the image so that my robot can have more information to make the next movement.
8	As a participant in the competition, I want my robot to my robot to identify the type of multiple armours appeared in the image so that my robot can have more information to make the next movement.

3. Test Cases

3.1 User Story 3

3.1.1 TC005: GUI Lets Users to Upload Images

Test Type:	Execution Type:
Functional	Manual

Objective: Verify if users can upload images to GUI.
Setup: • Prepare several image files.
Procedure: 1. Run the application. 2. Click Menu > Upload Images. 3. Check if the chosen image is displayed in the display area. 4. If multiple images are chosen, user will be able to navigate through the images using the next and previous buttons at the top.
Notes: [1] Tests should be done both with single and multiple images [2] Try uploading multiple times
Expected outcome: • The correct image is displayed on the display area • User can navigate through images using next and previous buttons
Time constraint: Maximum: 1 second per image

3.1.2 TC006: Uploading Weights, Names, and Config Files

Test Type: Functional	**Execution Type:** Manual
Objective: Verify that user can only upload weight, name, and config files if they are in the right formats.	
Setup: • Name, config, and weight files in one zip file.	
Procedure: 1. Click *Menu* > *Upload weight, name, and config files as a zip.* 2. Choose the prepared zip file.	
Notes: [1] Try with a zip file that is missing at least one of the required files. [2] Try uploading multiple times.	
Expected outcome: • The zip file is accepted if it contains all the required files. Otherwise, a dialogue box appears ("Please include names, cfg, and weights files in the zip") • If user uploads multiple times, the most recent files should be used.	
Time constraint: Maximum: 1 second	

3.1.3 TC007: The *Label* Button Works by Displaying Bounding Boxes

Test Type: Functional	**Execution Type:** Manual

Objective: Verify that the Label Button Works
Setup: • One or multiple images have been uploaded to GUI (see TC005) • Name, config, and weight files have been uploaded
Procedure: 1. Click *Label*
Notes: -
Expected outcome: • Bounding boxes are drawn on the robot's nearest armour, along with the label that indicates the colour (red, blue) and position (front, side, back).
Time constraint: Maximum: 1 second per uploaded image.

3.1.4 TC008: The Play Button Works by Playing a Slideshow

Test Type: Functional	**Execution Type:** Manual
Objective: Verify that the slideshow function works.	
Setup: • Multiple images have been uploaded to the GUI (see TC005)	
Procedure: 1. Click the play button at the top to start the slideshow	
Notes: -	
Expected outcome: • The displayed image automatically changes to the next uploaded image every few seconds.	
Time constraint: Maximum: 1 second	

3.1.5 TC009: *Export* Button

Test Type: Functional	**Execution Type:** Manual
Objective: Verify that the *Export* button works	
Setup: • One or multiple images have been uploaded to GUI (see TC005). • Name, config, and weight files have been uploaded (see TC007).	

Procedure: 1. Click *Export* after images and weight, name, and config files are uploaded.
Notes: -
Expected outcome: • Image annotations will be saved as txt files with same filenames as the images.
Time constraint: Maximum: 1 second per uploaded image.

3.2 User Story 4

3.2.1 TC010: Armour Location and Identification work in Conjunction

Test Type: Functional	**Execution Type:** Manual
Objective: Verify if armour localisation and identification are consistent with each other.	
Setup: • Same as TC007	
Procedure: Same as TC007	
Notes: [1] Test image should have at least 2 visible armours on 1 robot.	
Expected outcome: • The armour that is located by the bounding box should be correctly labelled (e.g. if front and side armours are visible and bounding box is on the front armour, it should be labelled 'front').	
Time constraint: Same as TC007.	

3.3 User Story 7

3.3.1 TC011: Detecting Multiple Armours

Test Type: Functional	**Execution Type:** Manual
Objective: Verify if program can detect armours of multiple robots.	
Setup: • Prepare a test image that contains multiple visible armours. • Choose the relevant mode on the GUI.	

Procedure: Same as TC007
Notes: -
Expected outcome: • The program should detect all visible armours and draw the bounding boxes.
Time constraint: Same as TC007.

3.4 User Story 8

3.4.1 TC012: Detecting Armours on Multiple Robots

Test Type: Functional	**Execution Type:** Manual
Objective: Verify if program can identify armours on multiple robots.	
Setup: • Prepare a test image that contains multiple robots in one image. • Name, config, and weight files in one zip file.	
Procedure: Same as TC007	
Notes: -	
Expected outcome: • The program should detect and label exactly one armour for each robot.	
Time constraint: Same as TC007.	