CSSR for

Culturally Sensitive Social Robotics
for Africa

D3.5 System Integration and Quality Assurance

Due date: **30/06/2024**
Submission Date: **25/07/2024**
Revision Date: **14/11/2025**

Start date of project: **01/07/2023** Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa**

Responsible Person: **D. Vernon** Revision: **1.3**

Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including Afretec Administration)	
RE	Restricted to a group specified by the consortium (including Afretec Administration)	
CO	Confidential, only for members of the consortium (including Afretec Administration)	

Executive Summary

Deliverable D3.5 represents the outcome of Task 3.5 and highlights the progress in integrating the software modules critical to the system's development. This deliverable comprises the integration results for all submitted software modules, as well as their corresponding unit test outcomes. The results are based on a checklist designed to assess functionality, performance, and compliance with the system's overall requirements.
Each software's integration process and unit test results are documented in dedicated subsections within the report, where the corresponding checklist details what was tested for and the results achieved. The following software modules successfully passed the integration tests, meeting all functional requirements and proving their readiness for system deployment:

D4.1 Sensor Tests

D4.2.1 Person Detection and Localization

D4.2.2 Face & Mutual Gaze Detection and Localization

D4.2.3 Sound Detection and Localization

D4.2.4 Robot Localization

D4.3.2 Speech Event

D5.1 Actuator Tests

D5.2 Animate Behavior Subsystem

D5.3 Attention Subsystem

D5.4.3 Robot Mission Interpreter

D5.5.1.1 Gesture Execution

D5.5.2.4 Integrated Text to Speech Conversion

D5.5.3 Environment Map Generation

D5.5.4 Robot Navigation

The checklist used in the integration process evaluates the modules against predefined rubrics for functionality and alignment with the software engineering standards set out for the CSSR4Africa project. Each test is marked as passed (☑), failed (☒), or not applicable (⊟) based on the specific characteristics of the software.
Software modules which pass the integration process have all tests either marked as passed (☑) or not applicable (⊟), depending on the applicability The current integration process prioritizes functionality of the software on the physical robot over the simulator robot.

Contents

1 D4.1 Sensor Tests

1.1 Sensor Tests

1.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **pepper_interface_tests** outside of the `cssr_system` and `unit_tests` package directories. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **pepper_interface_tests** directory five sub-directories: `config`, `data`, `include/pepper_interface_tests`, `launch`, and `src`.

☑ The `config` directory contains two files, named as follows.

- ☑ `actuatorTestConfiguration.json` and `sensorTestConfiguration.json`
- ☑ The configuration files `actuatorTestConfiguration.ini` and `sensorTestConfiguration.json` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains four input fules and one output file.

- The input files are named as follows.
  - ☑ `actuatorTestInput.dat`
  - ☑ `pepperTopics.dat`
  - ☑ `sensorTestInput.dat`
  - ☑ `simulatorTopics.dat`
  - ☑ The topics files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
  - ☑ Each key-value pair of the `pepperTopics.dat` and `sensorTopics.dat` files is written on a separate line.
- The output files are named as follows.
  - ☑ `sensorTestOutput.dat`

☑ The `include/pepper_interface_tests` directory contains two files, named as follows.

- ☑ `actuatorTestInterface.h`
- ☑ `sensorTestInterface.h`

☑ The `launch` directory contains three files, named as follows.

- ☑ `actuatorTestLaunchRobot.launch`
- ☑ `interfaceTestLaunchSimulator.launch`
- ☑ `sensorTestLaunchRobot.launch`

☑ The `src` directory contains four source files, named as follows.

- ☑ `actuatorTestApplication.cpp`
- ☑ `actuatorTestImplementation.cpp`
- ☑ `sensorTestApplication.cpp`
- ☑ `sensorTestImplementation.cpp`

☑ The `pepper_interface_tests` directory or node subdirectory contains a `README.md` file with instructions on how to run the node.

☑ The `pepper_interface_tests` directory contains a `CMakeLists.txt` build file.

☑ The `pepper_interface_tests` directory contains a `package.xml` manifest file since it is a package directory.

1.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `sensorTestApplication.cpp`

```
/* sensorTestApplication.cpp Application to test the sensors of the Pepper robot
*        using ROS interface.
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
```

☑ `sensorTestImplementation.cpp`

```
/* sensorTestImplementation.cpp Implementation code to test the sensors of the Pepper robot
*        using ROS interface.
*
* Author: Yohannes Tadesse Haile and Mihirteab Taye Hordofa
* Date: October 13, 2025
* Version: v1.1
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ `sensorTestInterface.h`

```
/* sensorTestInterface.h - Header file for the sensorTest module to test the sensors
*        of the Pepper robot using ROS interface.
 *
 * Author:  Yohannes Tadesse Haile and Mihirteab Taye Hordofa, Carnegie Mellon University Africa
 * Email:   yohanneh@andrew.cmu.edu
 * Date:    September 25, 2025
 * Version: v1.1
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The sensorTestApplication.cpp file contains a documentation comment with the following sections:

```
☑ /* sensorTestApplication.cpp Application to test the sensors of the
   *        Pepper robot using ROS interface.
   *
   * The component test the functionality of the sensor of the robot
   * using the ROS interface. The test is performed by subscribing to
   * the sensor topics and checking if the robot sends the expected data.
   * The test is performed in two modes: sequential and parallel. In
   * the sequential mode, the tests are performed one after the other.
   * In the parallel mode, the tests are performed simultaneously.
   *

☑ * Libraries
   * Standard libraries
   - std::string, std::vector, std::thread, std::fstream, std::cout,
       std::endl, std::fabs, std::time_t, std::tm, std::localtime,
       std::strftime
   * ROS libraries
   - ros/ros.h, ros/package.h, image_transport/image_transport.h,
       sensor_msgs/CameraInfo.h, sensor_msgs/Range.h,
       sensor_msgs/JointState.h, sensor_msgs/LaserScan.h,
       cv_bridge/cv_bridge.h, opencv2/highgui/highgui.hpp

☑ * Parameters
   *
   * Command-line Parameters

   * Configuration File Parameters

   * Key | Value
   * --- | ---
   * platform         | robot
   * simulatorTopics  | simulatorTopics.dat
   * robotTopics      | pepperTopics.dat
   * mode             | sequential
```

```
     * Key | Value
     * --- | ---
     * BackSonar             |   true
     * FrontSonar            |   true
     * BottomCamera          |   true
     * FrontCamera           |   true
     * realsenseRGBDCamera   |   true
     * realsenseDepthCamera  |   true
     * DepthCamera           |   true
     * StereoCamera          |   true
     * LaserSensor           |   true
     * Microphone            |   true
     * JointState            |   true
     * Odometry              |   true
     * IMU                   |   true
     * Speech                |   true

☑  * Subscribed Topics and Message Types
     *
     * /naoqi_driver/sonar/back                      sensor_msgs/Range
     * /naoqi_driver/sonar/front                     sensor_msgs/Range
     * /naoqi_driver/camera/front/image_raw          sensor_msgs/Image
     * /camera/color/image_raw                       sensor_msgs/Image
     * /camera/aligned_depth_to_color/image_raw      sensor_msgs/Image
     * /naoqi_driver/camera/bottom/image_raw         sensor_msgs/Image
     * /naoqi_driver/camera/depth/image_raw          sensor_msgs/Image
     * /naoqi_driver/laser                           sensor_msgs/LaserScan
     * /naoqi_driver/audio                           naoqi_driver/AudioCustomMsg
     * /naoqi_driver/joint_states                    sensor_msgs/JointState
     * /naoqi_driver/odom                            nav_msgs/Odometry
     * /naoqi_driver/imu                             sensor_msgs/Imu


     * /pepper/sonar_back                            sensor_msgs/Range
     * /pepper/sonar_front                           sensor_msgs/Range
     * /pepper/camera/front/image_raw                sensor_msgs/Image
     * /pepper/camera/bottom/image_raw               sensor_msgs/Image
     * /pepper/camera/depth/image_raw                sensor_msgs/Image
     * /pepper/laser_2                               sensor_msgs/LaserScan
     * /joint_states                                 sensor_msgs/JointState
     * /pepper/odom                                  nav_msgs/Odometry

☑  * Published Topics and Message Types
     * /naoqi_driver/speech                          std_msgs/String

☑  * Services Invoked
     *
     * None

☑  * Services Advertised and Request Message
     *
     * None

☑  * Input Data Files
```

```
    *
    * pepperTopics.dat
    * simulatorTopics.dat
    * sensorTestInput.dat

☑ * Output Data Files
    *
    * sensorTestOutput.dat,
    * frontCameraOutput.mp4,
    * realsenseRGBOutput.mp4,
    * realsenseDepthOutput.mp4,
    * bottomCameraOutput.mp4,
    * depthCameraOutput.mp4,
    * stereoCameraOutput.mp4,
    * microphoneOutput.wav

☑ * Configuration Files
    *
    * sensorTestConfiguration.ini

☑ * Example Instantiation of the Module
    *
    * rosrun pepper_interface_tests sensorTest

☑ * Author: Yohannes Tadesse Haile and Mihirteab Taye Hordofa,
       Carnegie Mellon University Africa

☑ * Email: yohanneh@andrew.cmu.edu

☑ * Date: October 13, 2025

☑ * Version: v1.1
```

1.1.3 Component Unit Testing

☑ A unit test application named `sensorTestLaunchRobot.launch` is provided in the `launch` directory.

☑ A unit test application named `interfaceTestLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `sensorTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `sensorTestLaunchRobot.launch` file launches the component being tested.

☑ The `interfaceTestLaunchSimulator.launch` file launches the component being tested.

⊟ The `sensorTestLaunchTestHarness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
sensorTest: v1.1

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
sensorTest: start-up.
sensorTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
sensorTest: running.
```

☑ The `sensorTestLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ The `interfaceTestLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `sensorTestLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `pepper_interface_tests` directory.

- ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`sensorTestConfiguration.ini`) file are altered.

2 D4.2.1 Person Detection and Localization

2.1 Person Detection

2.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **person_detection**.

☑ The **person_detection** directory has five sub-directories: `config`, `data`, `models`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ person_detection_configuration.json
- ☑ The configuration file person_detection_configuration.json contains the key-value pairs that set the component parameters. (The config file do not match what is stated in the workplan/system architecture. May be worth either revising those documents).
- ☑ Each key-value pair is written on a separate line.

☑ The data directory contains one file, named as follows.

- ☑ pepper_topics.dat

☑ The msg directory contains one file, named as follows.

- ☑ person_detection_msg_file.msg

☑ The src directory contains three source files, named as follows.

- ☑ person_detection_application.py
- ☑ person_detection_implementation.py
- ☑ person_detection_tracking.py

☑ The person_detection directory contains a README.md file with instructions on how to run the software

☑ The person_detection directory contains no CMakeLists.txt build file.

☑ The person_detection directory contains no package.xml manifest file since it is a package within the cssr_system ROS node.

☑ The person_detection directory contains a person_detection_requirements_x86.txt file with the dependencies (and their versions) required to be installed for proper functionality of the node.

2.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ person_detection_application.py

```
"""
person_detection_application.py Application code to run the
Person Detection and Localization ROS node.

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ person_detection_implementation.py

```
""""
person_detection_implementation.py Implementation code for running the
Person Detection and Localization ROS node.

Author: Yohannes Tadesse Haile
Date: April 21, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ person_detection_tracking.py

```
""""
person_detection_tracking.py Functionality for tracking people using
SORT (simple online and realtime tracking).

Author: Yohannes Tadesse Haile
Date: April 21, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The person_detection_application.py file contains a documentation comment:

☑
```
"""
 The person detection is implemented using the ROS image topic that could be
 configured to be the intel realsense camera or pepper robot camera.
 It uses OpenCV to visualize the detected persons. The code utilizes YOLOv8
 for person detection.This code contains the main function that initializes
 the person detection node and starts the person detection algorithm.
 It subscribes to the intel realsense camera or pepper robot camera topics for
 the RGB and depth images.It publishes one topic: /personDetection/data
 that contains the person label ID, the centroid of the person,
 width and height of the bounding box.
```

☑
```
Libraries
     - cv2
     - numpy
     - rospy
     - rospkg
     - os
     - onnxruntime
     - multiprocessing
     - json
     - random
     - threading
     - sensor_msgs.msg (Image, CompressedImage)
     - filterpy.kalman (KalmanFilter)
     - itertools (count)
     - cv_bridge (CvBridge, CvBridgeError)
     - message_filters (ApproximateTimeSynchronizer, Subscriber)
     - geometry_msgs.msg (Point)
     - cssr_system.msg (person_detection_msg_file)
     - person_detection_tracking (Sort)
```

☑ Parameters

None

☑ Configuration File Parameters

Key	Value
useCompressed	true
confidence_iou_threshold	0.8
sortMaxDisappeared	30
sortMinHits	20
sortIouThreshold	0.3
verboseMode	true

☑ Subscribed Topics

Topic Name	Message Type
/camera/color/image_raw	sensor_msgs/ Image
/camera/color/image_raw/compressed	sensor_msgs/ CompressedImage
/camera/aligned_depth_to_color/image_raw	sensor_msgs/ Image
/camera/aligned_depth_to_color/image_raw/compressed	sensor_msgs/ CompressedImage
/naoqi_driver/camera/front/image_raw	sensor_msgs/ Image
/naoqi_driver/camera/depth/image_raw	sensor_msgs/ Image

☑ Published Topics

Topic Name	Message Type
/personDetection/data	person_detection/ person_detection_msg_file.msg

☑ Input Data Files

- pepperTopics.dat: Data file for Pepper robot camera topics

☑ Model Files

- person_detection_yolov8s.onnx: YOLOv8 model for object detection tailored for person detection

☑ Output Data Files

None

☑ Configuration File

person_detection_configuration.json

☑ Example of instantiation of the module

```
roslauch cssr_system person_detection_robot.launch camera:=realsense

# Activate the python environment
source ~/workspace/pepper_rob_ws/cssr4africa_face_person_detection_env/
        bin/activate

(In a new terminal)
rosrun cssr_system person_detection_application.py
```

☑ `Author: Yohannes Tadesse Haile, Carnegie Mellon University Africa`

☑ `Email: yohanneh@andrew.cmu.edu`

☑ `Date: April 21, 2025`

☑ `Version: v1.0`

2.1.3 Component Unit Testing

⊟ A unit test application named `person_detection_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `person_detection_launch_simulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `person_detection_launch_test_harness.launch` is provided in the `launch` directory.

⊟ The `person_detection_launch_robot.launch` file launches the component being tested.

⊟ The `person_detection_launch_simulator.launch` file launches the component being tested.

⊟ The `person_detection_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
personDetection v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
personDetection: start-up.
personDetection: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
personDetection: running.
```

☑ The `person_detection_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `person_detection_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

⊟ The `person_detection_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ Unit test instructions are provided in a file named `README.md` in the `person_detection` directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`person_detection_configuration.ini`) file are altered.

2.2 Person Detection Unit Test

2.2.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **person_detection_test**.

☑ The **person_detection_test** directory has five sub-directories: `config`, `data`, `launch`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `person_detection_test_configuration.json`
- ☑ The configuration file `person_detection_test_configuration.json` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains five bag files, named as follows.

- ☑ `person_detection_test_input_realsense_lighting_1.bag`
- ☑ `person_detection_test_input_realsense_lighting_2.bag`
- ☑ `person_detection_test_input_realsense_multiple_people.bag`
- ☑ `person_detection_test_input_realsense_single_person.bag`

☑ The `launch` directory contains two files, named as follows.

- ☑ `person_detection_test_launch_robot.launch`
- ☑ `person_detection_test_launch_test_harness.launch`

☑ The msg directory contains one file, named as follows.

  ☑ person_detection_test_msg_file.msg

☑ The src directory contains two source files, named as follows.

  ☑ person_detection_test_application.py

  ☑ person_detection_test_implementation.py

☑ The person_detection directory contains a README.md file with instructions on how to run the software

☑ The person_detection directory contains no CMakeLists.txt build file.

☑ The person_detection directory contains no package.xml manifest file since it is a package within the unit_tests ROS node.

⊟ The person_detection directory contains a person_detection_requirements.txt file with the dependencies (and their versions) required to be installed for proper functionality of the node.

2.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ person_detection_test_application.py

```
"""
person_detection_test_application.py Application code to run the person
detection and localization unit test.

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ person_detection_test_implementation.py

```
""""
person_detection_test_implementation.py Implementation code for running the
Person Detection and Localization unit test.

Author: Yohannes Tadesse Haile
Date: April 25, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The `person_detection_test_application.py` file contains a documentation comment:

☑
```
"""
person_detection_test_application.py  Application code to run person detection
and localization unit test.

This person_detection_test is a unit test application code to test the person
detection and localization algorithm. This code contains the main function that
initializes the correct configuration parameters and tests whether person is
detected and localized. It has also utility functions to save video and images
with the bounding boxes.
```

☑ Libraries

```
    - rospkg
    - rospy
    - os
    - json
    - numpy
    - cv2
    - time
    - threading
    - colorsys
    - sensor_msgs.msg (Image)
    - cv_bridge (CvBridge)
    - message_filters (ApproximateTimeSynchronizer, Subscriber)
    - unit_tests.msg (person_detection_test_msg_file)
```

☑ Parameters

Launch File Parameters:

```
roslaunch unit_tests person_detection_test_launch_robot.launch
                                             camera:=realsense
    camera: Camera type or video file (realsense or pepper or video)
    bag_file: ROS bag file for testing
    pepper_robot_ip: Pepper robot IP address
    network_interperson: Network interperson for Pepper robot connection

roslaunch unit_tests person_detection_test_launch_test_harness.launch
```

☑ Configuration File Parameters

Key	Value
saveVideo	false
saveImage	false
videoDuration	10
imageInterval	5
recordingDelay	5
maxFramesBuffer	300
verboseMode	false

☑ Subscribed Topics

Topic Name	Message Type
/camera/color/image_raw	sensor_msgs/Image
/camera/aligned_depth_to_color/image_raw	sensor_msgs/Image
/naoqi_driver/camera/front/image_raw	sensor_msgs/Image
/naoqi_driver/camera/depth/image_raw	sensor_msgs/Image
/personDetection/data	person_detection/ person_detection_test_msg_file.msg

☑ Published Topics

None

☑ Input Data Files

- pepperTopics.dat: Data file for Pepper robot camera topics
- person_detection_test_input_realsense_single_person.bag
- person_detection_test_input_realsense_multiple_people.bag
- person_detection_test_input_realsense_lighting_1.bag
- person_detection_test_input_realsense_lighting_2.bag

☑ Output Data Files

- person_detection_test_rgb_video_{start_time}.mp4
- person_detection_test_depth_video_{start_time}.mp4
- person_detection_test_rgb_image_{start_time}.png
- person_detection_test_depth_image_{start_time}.png

☑ Configuration File

person_detection_test_configuration.json

☑ Example of instantiation of the module

```
roslaunch unit_tests person_detection_test_launch_robot.launch
                 camera:=video bag_file:=single_person

(In a new terminal)
roslaunch unit_tests person_detection_test_launch_testHarness.launch
```

☑ `Author: Yohannes Tadesse Haile, Carnegie Mellon University Africa`

☑ `Email: yohanneh@andrew.cmu.edu`

☑ `Date: March 21, 2025`

☑ `Version: v1.0`

2.2.3 Component Unit Testing

☑ A unit test application named `person_detection_test_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `person_detection_launch_simulator.launch` is provided in the `launch` directory.

☑ A unit test application named `person_detection_test_launch_test_harness.launch` is provided in the `launch` directory.

☑ The `person_detection_launch_robot.launch` file launches the component being tested.

⊟ The `person_detection_launch_simulator.launch` file launches the component being tested.

☑ The `person_detection_test_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
personDetectionTest v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
personDetectionTest: start-up.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
personDetectionTest: running.
```

☑ The `person_detection_test_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

- ⊟ The `person_detection_test_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

- ☑ The `person_detection_test_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

- ☑ Unit test instructions are provided in a file named `README.md` in the `person_detection` directory.

  - ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
  - ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`person_detection_configuration.ini`) file are altered.

3 D4.2.2 Face & Mutual Gaze Detection and Localization

3.1 Face Detection

3.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **face_detection**.

☑ The **face_detection** directory has five sub-directories: `config`, `data`, `models`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `face_detection_configuration.json`
- ☑ The configuration file `face_detection_configuration.json` contains the key-value pairs that set the component parameters. (The config file do not match what is stated in the workplan/system architecture. May be worth either revising those documents).
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains one file, named as follows.

- ☑ `pepper_topics.dat`

☑ The `msg` directory contains one file, named as follows.

- ☑ `face_detection_msg_file.msg`

☑ The `src` directory contains three source files, named as follows.

- ☑ `face_detection_application.py`
- ☑ `face_detection_implementation.py`
- ☑ `face_detection_tracking.py`

☑ The `face_detection` directory contains a `README.md` file with instructions on how to run the software

☑ The `face_detection` directory contains no `CMakeLists.txt` build file.

☑ The `face_detection` directory contains no `package.xml` manifest file since it is a package within the `cssr_system` ROS node.

☑ The `face_detection` directory contains a `face_detection_requirements_x86.txt` file with the dependencies (and their versions) required to be installed for proper functionality of the node.

3.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `face_detection_application.py`

```
"""
face_detection_application.py Application code to run the Face and Mutual
Gaze Detection and Localization ROS node.

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ `face_detection_implementation.py`

```
""""
face_detection_implementation.py Implementation code for running the Face and
Mutual Gaze Detection and Localization ROS node.

Author: Yohannes Tadesse Haile
Date: March 21, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ face_detection_tracking.py

```
""""
face_detection_tracking.py Functionality for tracking faces using
SORT(simple online and realtime tracking) and Centroid Tracking.

Author: Yohannes Tadesse Haile
Date: March 21, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The face_detection_application.py file contains a documentation comment:

☑

```
"""
face_detection_application.py   Application code to run the face and mutual gaze
detection algorithm.

The face detection is implemented using the ROS image topic that could be
configured to be the intel realsense camera or pepper robot camera. It uses
OpenCV to visualize the detected faces and gaze direction. The gaze direction is
calculated using face mesh landmarks which uses Google's MediaPipe library. The
media pipe utilizes CPU for face detection and gaze direction. The SixDrepNet
uses YOLOONNX for face detection and SixDrepNet for gaze direction.
The SixDrepNet utilizes GPU for faster inference and better performance.
This code contains the main function that initializes the face detection node
and starts the face detection algorithm. The face detection algorithm can be
either MediaPipe Face Detection or SixDrepNet that can be configured from the
configuration file. It is also responsible for detecting the head pose esimation
of the detected face. It subscribes to the intel realsense camera or pepper robot
camera topics for the RGB and depth images. It publishes three one topic:
/faceDetection/data that contains the face label ID, the centroid of the face,
mutual gaze direction.
```

☑ Libraries

- cv2
- mediapipe
- numpy
- rospy
- rospkg
- os
- onnxruntime
- multiprocessing
- json
- random
- math (cos, sin, pi)
- sensor_msgs.msg (Image, CompressedImage)
- cv_bridge (CvBridge, CvBridgeError)
- message_filters (ApproximateTimeSynchronizer, Subscriber)
- geometry_msgs.msg (Point)
- typing (Tuple, List)
- face_detection.msg (msg_file)
- face_detection_tracking (Sort, CentroidTracker)

☑ Parameters

None

☑ Configuration File Parameters

Key	Value
algorithm	sixdrep
use_compressed	true
mp_facedet_confidence	0.5
mp_headpose_angle	5
centroid_max_distance	15
centroid_max_disappeared	100
sixdrepnet_confidence	0.65
sixdrepnet_headpose_angle	10
sort_max_disappeared	30
sort_min_hits	20
sort_iou_threshold	0.3
verbose_mode	true

☑ Subscribed Topics

Topic Name	Message Type
/camera/color/image_raw	sensor_msgs/Image
/camera/aligned_depth_to_color/image_raw	sensor_msgs/Image
/naoqi_driver/camera/front/image_raw	sensor_msgs/Image
/naoqi_driver/camera/depth/image_raw	sensor_msgs/Image

☑ Published Topics

Topic Name	Message Type
/faceDetection/data	face_detection/face_detection.msg

☑ Input Data Files

- pepperTopics.dat: Data file for Pepper robot camera topics

☑ Model Files

- face_detection_YOLO.onnx: YOLOONNX model for face detection

```
        - face_detection_sixdrepnet360.onnx: SixDrepNet model for gaze direction
```

☑ `Output Data Files`

```
     None
```

☑ `Configuration File`

```
     face_detection_configuration.json
```

☑ `Example of instantiation of the module`

```
     roslauch cssr_system face_detection_robot.launch camera:=realsense

     (In a new terminal)
     rosrun cssr_system face_detection_application.py
```

☑ `Author: Yohannes Tadesse Haile, Carnegie Mellon University Africa`

☑ `Email: yohanneh@andrew.cmu.edu`

☑ `Date: March 21, 2025`

☑ `Version: v1.0`

3.1.3 Component Unit Testing

⊟ A unit test application named `face_detection_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `face_detection_launch_simulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `face_detection_launch_test_harness.launch` is provided in the `launch` directory.

⊟ The `face_detection_launch_robot.launch` file launches the component being tested.

⊟ The `face_detection_launch_simulator.launch` file launches the component being tested.

⊟ The `face_detection_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
faceDetection v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
faceDetection: start-up.
faceDetection: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
faceDetection: running.
```

⊟ The `face_detection_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `face_detection_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

⊟ The `face_detection_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ Unit test instructions are provided in a file named `README.md` in the `face_detection` directory.

  ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`face_detection_configuration.ini`) file are altered.

3.2 Face Detection Unit Test

3.2.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **face_detection_test**.

☑ The **face_detection_test** directory has five sub-directories: `config`, `data`, `launch`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

  ☑ `face_detection_test_configuration.json`

  ☑ The configuration file `face_detection_test_configuration.json` contains the key-value pairs that set the component parameters.

  ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains five bag files, named as follows.

- ☑ `face_detection_test_input_realsense_lighting_1.bag`
- ☑ `face_detection_test_input_realsense_lighting_2.bag`
- ☑ `face_detection_test_input_realsense_multiple_faces.bag`
- ☑ `face_detection_test_input_realsense_mutual_gaze.bag`
- ☑ `face_detection_test_input_realsense_single_face.bag`

☑ The `launch` directory contains two files, named as follows.

- ☑ `face_detection_test_launch_robot.launch`
- ☑ `face_detection_test_launch_test_harness.launch`

☑ The `msg` directory contains one file, named as follows.

- ☑ `face_detection_test_msg_file.msg`

☑ The `src` directory contains two source files, named as follows.

- ☑ `face_detection_test_application.py`
- ☑ `face_detection_test_implementation.py`

☑ The `face_detection` directory contains a `README.md` file with instructions on how to run the software

☑ The `face_detection` directory contains no `CMakeLists.txt` build file.

☑ The `face_detection` directory contains no `package.xml` manifest file since it is a package within the `unit_tests` ROS node.

⊟ The `face_detection` directory contains a `face_detection_requirements.txt` file with the dependencies (and their versions) required to be installed for proper functionality of the node.

3.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ face_detection_test_application.py

```
"""
face_detection_test_application.py Application code to run the Face and
Mutual Gaze Detection and Localization Unit test.

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ face_detection_test_implementation.py

```
""""
face_detection_test_implementation.py Implementation code for running the
Face and Mutual Gaze Detection and Localization unit test.

Author: Yohannes Tadesse Haile
Date: March 21, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

The face_detection_test_application.py file contains a documentation comment:

☑
```
"""
 face_detection_test_application.py  Application code to run the Face and
 Mutual Gaze Detection and Localization unit test.

 This face_detection_test is a unit test application code to test the face and
 mutual gaze detection algorithm. This code contains the main function that
 initializes the correct configuration parameters and tests face detection
```

algorthim. It has also utility functions to save video and images with the bounding boxes and gaze detection. The face detection algorithm can be either MediaPipe Face Detection or SixDrepNet that can be configured from the configuration file.

☑ Libraries

- rospkg
- rospy
- os
- json
- numpy
- cv2
- time
- threading
- colorsys
- sensor_msgs.msg (Image)
- cv_bridge (CvBridge)
- message_filters (ApproximateTimeSynchronizer, Subscriber)
- face_detection_test.msg (msg_file)

☑ Parameters

Launch File Parameters:

```
roslaunch unit_test face_detection_test_launch_robot.launch
                    camera:=realsense
    camera: Camera type or video file (realsense or pepper or video)
    bag_file: ROS bag file for testing (singleFace, multipleFaces,
                 faceTracking, mutualGaze, occlusion, lighting)
    pepper_robot_ip: Pepper robot IP address
    network_interface: Network interface for Pepper robot connection
```

☑ Configuration File Parameters

Key	Value
algorithm	sixdrep
save_video	false
save_image	false
video_duration	10
image_interval	5
recording_delay	5
max_frames_buffer	300
verbose_mode	false

☑ Subscribed Topics

Topic Name	Message Type
/camera/color/image_raw	sensor_msgs/Image
/camera/aligned_depth_to_color/image_raw	sensor_msgs/Image
/naoqi_driver/camera/front/image_raw	sensor_msgs/Image
/naoqi_driver/camera/depth/image_raw	sensor_msgs/Image
/faceDetection/data	face_detection/ face_detection.msg

☑ Published Topics

None

☑ Input Data Files

```
        - pepperTopics.dat: Data file for Pepper robot camera topics
        - face_detection_test_input_realsense_singleFace.bag
        - face_detection_test_input_realsense_multipleFaces.bag
        - face_detection_test_input_realsense_faceTracking.bag
        - face_detection_test_input_realsense_mutualGaze.bag
        - face_detection_test_input_realsense_occlusion.bag
        - face_detection_test_input_realsense_lighting.bag
```

☑ Output Data Files

```
        - face_detection_rgb_video_{start_time}.mp4
        - face_detection_depth_video_{start_time}.mp4
        - face_detection_rgb_image_{start_time}.png
        - face_detection_depth_image_{start_time}.png
```

☑ Configuration File

```
        face_detection_test_configuration.json
```

☑ Example of instantiation of the module

```
        roslaunch unit_test face_detection_test_launch_robot.launch
                              camera:=video bag_file:=singleFace

        (In a new terminal)
        roslaunch unit_test face_detection_test_launch_testHarness.launch
```

☑ Author: Yohannes Tadesse Haile, Carnegie Mellon University Africa

☑ Email: yohanneh@andrew.cmu.edu

☑ Date: March 21, 2025

☑ Version: v1.0

3.2.3 Component Unit Testing

☑ A unit test application named `face_detection_test_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `face_detection_launch_simulator.launch` is provided in the `launch` directory.

☑ A unit test application named `face_detection_test_launch_test_harness.launch` is provided in the `launch` directory.

☑ The `face_detection_launch_robot.launch` file launches the component being tested.

⊟ The `face_detection_launch_simulator.launch` file launches the component being tested.

☑ The `face_detection_test_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
faceDetectionTest v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
faceDetectionTest: start-up.
faceDetectionTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
faceDetectionTest: running.
```

☑ The `face_detection_test_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `face_detection_test_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ The `face_detection_test_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ Unit test instructions are provided in a file named `README.md` in the `face_detection` directory.

  ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`face_detection_configuration.ini`) file are altered.

4 D4.2.3 Sound Detection and Localization

4.1 Sound Detection

4.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **sound_detection**.

☑ The **sound_detection** directory has four sub-directories: `config`, `data`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `sound_detection_configuration.json`
- ☑ The configuration file `sound_detection_configuration.json` contains the key-value pairs that set the component parameters. (The config file do not match what is stated in the workplan/system architecture. May be worth either revising those documents).
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains one file, named as follows.

- ☑ `pepper_topics.dat`

☑ The `msg` directory contains one file, named as follows.

- ☑ `sound_detection_microphone_msg_file.msg`

☑ The `src` directory contains two source files, named as follows.

- ☑ `sound_detection_application.py`
- ☑ `sound_detection_implementation.py`

☑ The `sound_detection` directory contains a `README.md` file with instructions on how to run the software

☑ The `sound_detection` directory contains no `CMakeLists.txt` build file.

☑ The `sound_detection` directory contains no `package.xml` manifest file since it is a package within the `cssr_system` ROS node.

☑ The `sound_detection` directory contains a `sound_detection_requirements.txt` file with the dependencies (and their versions) required to be installed for proper functionality of the node.

4.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `sound_detection_application.py`

```
"""
sound_detection_application.py Application code to run the sound detection
and localization algorithm.

Author: Yohannes Tadesse Haile
Date: April 13, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ `sound_detection_implementation.py`

```
"""
sound_detection_implementation.py Implementation code for running the sound
detection and localization algorithm

Author: Yohannes Tadesse Haile
Date: April 13, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The `sound_detection_application.py` file contains a documentation comment:

```
☑ """
   sound_detection_application.py Application code to run the sound detection
   and localization algorithm.

   The sound localization algorithm is implemented using a ROS audio topic that
   can be configured to receive audio from a robot or an external microphone.
   It processes the incoming audio signal to detect sound events and localize the
   sound source by computing the interaural time difference (ITD) using the
   GCC-PHAT method. The ITD is then converted into an angle of arrival using
   physical parameters such as the speed of sound and the distance between the
   microphones. This code contains the main function that initializes the sound
   localization node, loads the configuration, and starts the algorithm.
   The algorithm is designed to accumulate fixed-size audio samples (e.g., 4096
   per callback) in a rolling buffer until sufficient data is collected for
   processing.

☑ Libraries:
       - math
       - numpy
       - rospy
       - rospkg
       - os
       - json
       - webrtcvad
       - std_msgs
       - threading
       - noisereduce
       - soundfile
       - datetime

☑ Parameters:
       Command line arguments: None

☑ Configuration File Parameters:
           Key                                     Value
           intenstiyThreshold                      [float]     e.g., 0.0039
           distanceBetweenEars                     [float]     e.g., 0.07
           localizationBufferSize                  [int]       e.g., 8192
           vadAggressiveness                       [int]       e.g., 1
           contextDuration                         [float]     e.g., 1.0
           useNoiseReduction                       [bool]      e.g., true
           verboseMode                             [bool]      e.g., true

☑ Subscribed Topics:
       Topic Name                                  Message Type
       /naoqi_driver/audio                         sound_detection/
                           sound_detection_microphone_msg_file.msg

☑ Published Topics:
       Topic Name                                  Message Type
       /soundDetection/signal                      std_msgs/Float32MultiArray
       /soundDetection/direction                   std_msgs/Float32
```

☑ Input Data Files:
    - pepper_topics.dat: Data file containing topic names for the robot's audio sources.

☑ Output Data Files:
    None

☑ Configuration File:
    sound_detection_configuration.json

☑ Example of instantiation of the module:
    rosrun cssr_system sound_detection_application.py

☑ Author: Yohannes Tadesse Haile, Carnegie Mellon University Africa

☑ Email: yohanneh@andrew.cmu.edu

☑ Date: April 13, 2025

☑ Version: v1.0

4.1.3 Component Unit Testing

⊟ A unit test application named `sound_detection_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `sound_detection_launch_simulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `sound_detection_launch_test_harness.launch` is provided in the `launch` directory.

⊟ The `sound_detection_launch_robot.launch` file launches the component being tested.

⊟ The `sound_detection_launch_simulator.launch` file launches the component being tested.

⊟ The `sound_detection_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
soundDetection v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
soundDetection: start-up.
soundDetection: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
soundDetection: running.
```

⊟ The `sound_detection_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `sound_detection_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

⊟ The `sound_detection_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ Unit test instructions are provided in a file named `README.md` in the `sound_detection` directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`sound_detection_configuration.ini`) file are altered.

4.2 Sound Detection Unit Test

4.2.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **sound_detection_test**.

☑ The **sound_detection_test** directory has five sub-directories: `config`, `data`, `launch`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `sound_detection_test_configuration.json`
- ☑ The configuration file `sound_detection_test_configuration.json` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

- ☑ The data directory contains four files, named as follows.
  - ☑ pepper_topics.dat
  - ☑ sound_detection_test_input_sound_angle.bag
  - ☑ sound_detection_test_input_sound_distance.bag
  - ☑ sound_detection_test_input_sound_noises.bag
- ☑ The launch directory contains two files, named as follows.
  - ☑ sound_detection_test_launch_robot.launch
  - ☑ sound_detection_test_launch_test_harness.launch
- ☑ The msg directory contains one file, named as follows.
  - ☑ sound_detection_test_microphone_msg_file.msg
- ☑ The src directory contains two source files, named as follows.
  - ☑ sound_detection_test_application.py
  - ☑ sound_detection_test_implementation.py
- ☑ The sound_detection directory contains a README.md file with instructions on how to run the software
- ☑ The sound_detection directory contains no CMakeLists.txt build file.
- ☑ The sound_detection directory contains no package.xml manifest file since it is a package within the unit_tests ROS node.
- ⊟ The sound_detection directory contains a sound_detection_requirements.txt file with the dependencies (and their versions) required to be installed for proper functionality of the node.

4.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ sound_detection_test_application.py

```
"""
sound_detection_test_application.py Application code to run the Sound
Detection and Processing Unit test.

Author: Yohannes Tadesse Haile
Date: April 06, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ sound_detection_test_implementation.py

```
"""
sound_detection_test_implementation.py Implementation code for running the
Sound Detection and Processing unit test.

Author: Yohannes Tadesse Haile
Date: April 13, 2025
Version: v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The `sound_detection_test_application.py` file contains a documentation comment:

```
☑ """
   sound_detection_test_application.py  Application code to run the Sound
   Detection and Processing unit test.

   This sound_detection_test is a unit test application code to test the sound
   detection algorithm. It can record audio both from the filtered output and
   the original unfiltered microphone input, generate plots of audio signals
   and sound direction data, and save them for analysis. The test helps validate
   the correctness of audio signal processing and source localization.


☑ Libraries
       - rospkg
       - rospy
       - os
       - json
       - numpy
       - soundfile
       - matplotlib
       - datetime
       - threading
       - std_msgs.msg (Float32MultiArray, Float32)
       - sound_detection.msg (sound_detection)

☑ Parameters
       Launch File Parameters:
           roslaunch unit_tests sound_detection_test_launch.launch
               robot_ip: Pepper robot IP address
                          (e.g 172.29.111.230 or 172.29.111.240)

☑ Configuration File Parameters
           Key                         Value
           saveDirectory               /path/to/save/directory
           sampleRate                  48000
           recordFiltered              true
           recordUnfiltered            true
           generatePlots               true
           recordDuration              10
           plotInterval                10
           plotDpi                     150
           maxDirectionPoints          100
           directionPlotYlimit         90
           verboseMode                 true

☑ Subscribed Topics
       Topic Name                                  Message Type
       /naoqi_driver/audio                         sound_detection/sound_detection
       /soundDetection/signal                      std_msgs/Float32MultiArray
       /soundDetection/direction                   std_msgs/Float32

☑ Published Topics
       None
```

☑ Input Data Files
- pepper_topics.dat: Data file for Pepper robot audio topics

☑ Output Data Files
- filtered_{timestamp}.wav: Filtered audio recordings
- unfiltered_{timestamp}.wav: Unfiltered audio recordings
- audio_signals_{timestamp}.png: Plot showing filtered and unfiltered signals
- direction_data_{timestamp}.png: Plot showing sound direction over time

☑ Configuration File
sound_detection_test_configuration.json

☑

Example of instantiation of the module
```
roslaunch unit_tests sound_detection_test_launch.launch
```

☑ Author: Yohannes Tadesse Haile, Carnegie Mellon University Africa

☑ Email: yohanneh@andrew.cmu.edu

☑ Date: April 06, 2025

☑ Version: v1.0

4.2.3 Component Unit Testing

☑ A unit test application named `sound_detection_test_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `sound_detection_launch_simulator.launch` is provided in the `launch` directory.

☑ A unit test application named `sound_detection_launch_test_harness.launch` is provided in the `launch` directory.

☑ The `sound_detection>_launch_robot.launch` file launches the component being tested.

⊟ The `sound_detection>_launch_simulator.launch` file launches the component being tested.

☑ The `sound_detection>_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
soundDetectionTest v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
soundDetectionTest: start-up.
soundDetectionTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
soundDetectionTest: running.
```

☑ The `sound_detection_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `sound_detection_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ The `sound_detection_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ Unit test instructions are provided in a file named `README.md` in the `sound_detection` directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`sound_detection_test_configuration.json`) file are altered.

5 D4.2.4 Robot Localization

5.1 Robot Localization

5.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **robotLocalization**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **robotLocalization** directory has six sub-directories: `config`, `data`, `include/robotLocalization`, `launch`, `src`, and `srv`.

☑ The `config` directory contains one file, named as follows.

- ☑ `robotLocalizationConfiguration.json`

  - ☑ The configuration file robotLocalizationConfiguration.json contains the key-value pairs that set the component parameters.
  - ☑ Each key-value pair is written on a separate line.

- ☑ The data directory contains at three of three files.
  - ☑ arucoLandmarks.json
  - ☑ cameraInfo.json
  - ☑ pepperTopics.dat
  - ☑ The topics files pepperTopics.dat contains the key-value pairs that set the topic names required by the component.
  - ☑ Each key-value pair is written on a separate line.

- ☑ The include/robotLocalization directory contains one file, named as follows.
  - ☑ robotLocalizationInterface.h

- ☑ The launch directory contains one file, named as follows.
  - ☑ robotLocalizationLaunchRobot.launch

- ☑ The src directory contains two source files, named as follows.
  - ☑ robotLocalizationApplication.cpp
  - ☑ robotLocalizationImplementation.cpp

- ☑ The robotLocalization directory or node subdirectory contains a README.md file with instructions on how to run the node.

- ☑ The robotLocalization directory contains a CMakeLists.txt build file.

- ☑ The robotLocalization directory contains no package.xml since it is a ROS node within the cssr_system package.

5.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ robotLocalizationApplication.cpp

```
/* robotLocalizationApplication.cpp    Program initialization and main function execution
*
* Author:  Ibrahim Olaide Jimoh, Carnegie Mellon University Africa
* Email:   ioj@andrew.cmu.edu
* Date:    June 25, 2025
* Version: v1.0
```

```
 *
 * Copyright (C) 2025 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
```

☑ robotLocalizationImplementation.cpp

```
/* robotLocalizationImplementation.cpp    Function definitions and implementation
 *
 * Author:   Ibrahim Olaide Jimoh, Carnegie Mellon University Africa
 * Email:    ioj@andrew.cmu.edu
 * Date:     June 25, 2025
 * Version:  v1.0
 *
 * Copyright (C) 2025 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ robotLocalizationInterface.h

```
/* robotLocalizationInterface.h      Function, class, and variable declarations
 *
 * Author:   Ibrahim Olaide Jimoh, Carnegie Mellon University Africa
 * Email:    ioj@andrew.cmu.edu
 * Date:     June 25, 2025
 * Version:  v1.0
 *
 * Copyright (C) 2025 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The robotLocalizationApplication.cpp file contains a documentation comment with the following sections:

☑

```
/* robotLocalizationApplication.cpp    Program initialization and main
 *        function execution
 *
 * This node is responsible for determining the robot's absolute position
 * and orientation in the environment using visual landmark detection and
 * sensor fusion. The node combines ArUco marker detection from RGB and
 * depth cameras with odometry data to provide accurate 6-DOF pose
 * estimation. The system uses triangulation (RGB-only) or trilateration
 * (with depth) algorithms to compute absolute poses from detected landmarks,
 * then maintains relative positioning through odometry integration. The
 * node supports both periodic automatic pose correction and on-demand pose
 * reset services for robust localization in dynamic environments.
 *

☑ * Libraries
 *     Standard libraries
 *        std::string, std::vector, std::map, std::fstream, std::algorithm,
 *        std::numeric, std::sqrt, std::abs, std::atan2, std::cos, std::sin
 *     ROS libraries
 *        ros/ros.h, nav_msgs/Odometry.h, sensor_msgs/Imu.h,
 *        sensor_msgs/Image.h, sensor_msgs/JointState.h,
 *        sensor_msgs/CameraInfo.h, geometry_msgs/Pose2D.h,
 *        geometry_msgs/TransformStamped.h
 *     ROS TF libraries
 *        tf2/LinearMath/Quaternion.h,
 *        tf2_geometry_msgs/tf2_geometry_msgs.h,
 *        tf2_ros/transform_listener.h, tf2_ros/buffer.h
 *     OpenCV libraries
 *        opencv2/opencv.hpp, opencv2/aruco.hpp, cv_bridge/cv_bridge.h
 *     Image transport
 *        image_transport/image_transport.h
 *     Utility libraries
 *        angles/angles.h, yaml-cpp/yaml.h, json/json.h

☑ * Parameters
 *     Command-line Parameters
 *        None
 *     Configuration File Parameters
 *        Key                   |     Value
 *        --------------------- |     --------------------
 *        verboseMode           |     false
 *        camera                |     FrontCamera
 *        depthCamera           |     DepthRealSense
 *        useDepth              |     false
 *        resetInterval         |     30.0
 *        absolutePoseTimeout   |     300.0
 *        cameraInfoTimeout     |     10.0
 *        useHeadYaw            |     false
 *        headYawJointName      |     HeadYaw
 *        mapFrame              |     map
 *        odomFrame             |     odom
 *        landmarkFile          |     /robotLocalization/data/arucoLandmarks.json
 *        topicsFile            |     /robotLocalization/data/pepperTopics.dat
 *        cameraInfoFile        |     /robotLocalization/data/cameraInfo.yaml

☑ * Subscribed Topics and Message Types
 *     /naoqi_driver/camera/front/image_raw          sensor_msgs/Image
 *     /naoqi_driver/camera/stereo/image_raw         sensor_msgs/Image
 *     /camera/color/image_raw                       sensor_msgs/Image
 *     /camera/aligned_depth_to_color/image_raw      sensor_msgs/Image
 *     /camera/color/camera_info                     sensor_msgs/CameraInfo
 *     /naoqi_driver/odom                            nav_msgs/Odometry
```

```
     *     /naoqi_driver/imu/base                          sensor_msgs/Imu
     *     /joint_states                                   sensor_msgs/JointState

☑  * Published Topics and Message Types
     *     /robotLocalization/pose                         geometry_msgs/Pose2D
     *     /robotLocalization/marker_image                 sensor_msgs/Image

☑  * Services Invoked
     *     None

☑  * Services Advertised and Message Types
     *     /robotLocalization/reset_pose                   cssr_system/ResetPose
     *     /robotLocalization/set_pose                     cssr_system/SetPose

☑  * Input Data Files
     *     pepperTopics.dat        Contains topic names for robot sensors and actuators
     *     arucoLandmarks.json     3D coordinates of ArUco markers in the environment
     *     cameraInfo.json         Camera intrinsic parameters for fallback (fx, fy, cx

☑  * Output Data Files
     *     None (publishes pose data via ROS topics)

☑  * Configuration Files
     *     robotLocalizationConfiguration.json - Main configuration parameters file

☑  * Example Instantiation of the Module
     *     roslaunch cssr_system robotLocalizationLaunchRobot.launch

☑  * Author:   Ibrahim Olaide Jimoh, Carnegie Mellon University Africa

☑  * Email:    ioj@andrew.cmu.edu

☑  * Date:     June 25, 2025

☑  * Version:  v1.0
```

5.1.3 Component Unit Testing

⊟ A unit test application named `robotLocalizationLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `robotLocalizationLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `robotLocalizationLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `robotLocalizationLaunchRobot.launch` file launches the component being tested.

⊟ The `robotLocalizationLaunchSimulator.launch` file launches the component being tested.

⊟ The `robotLocalizationLaunchTestHarness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
robotLocalization: v1.0

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
robotLocalization: start-up.
robotLocalization: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
robotLocalization: running.
```

⊟ The `robotLocalizationLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `robotLocalizationLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `robotLocalizationLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `robotLocalization` directory.

- ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`robotLocalizationConfiguration.json`) file are altered.

5.2 Robot Localization Unit Test

5.2.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **robotLocalizationTest**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **robotLocalization** directory has six sub-directories: `config`, `data`, `include/robotLocalizationTest`, `launch`, `src`, and `srv`.

☑ The `config` directory contains one file, named as follows.

  - ☑ `robotLocalizationTestConfiguration.ini`
  - ☑ The configuration file `robotLocalizationTestConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains at three input files and one output file.

  - ☑ `arucoLandmarks.json`, an input file
  - ☑ `cameraInfo.json`, an input file
  - ☑ `pepperTopics.dat`, an input file
  - ☑ `robotLocalizationTestOutput.dat`, an output file
  - ☑ The topics files `pepperTopics.dat` contains the key-value pairs that set the topic names required by the component.
  - ☑ Each key-value pair is written on a separate line.

☑ The `include/robotLocalizationTest` directory contains one file, named as follows.

  - ☑ `robotLocalizationTestInterface.h`

☑ The `launch` directory contains two files, named as follows.

  - ☑ `robotLocalizationTestLaunchRobot.launch`
  - ☑ `robotLocalizationTestLaunchTestHarness.launch`

☑ The `src` directory contains three source files, named as follows.

  - ☑ `robotLocalizationTestApplication.cpp`
  - ☑ `robotLocalizationTestDriver.cpp`
  - ☑ `robotLocalizationTestImplementation.cpp`

☑ The `robotLocalizationTest` directory contains a `README.md` file with instructions on how to run the test.

☑ The `robotLocalizationTest` directory contains a `CMakeLists.txt` build file.

☑ The `robotLocalizationTest` directory contains no `package.xml` since it is a ROS node within the `unit_test` package.

5.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `robotLocalizationTestApplication.cpp`

```
/* robotLocalizationTestApplication.cpp - Robot Localization Unit Test Application
 *
 * Author:   Ibrahim Olaide Jimoh, Carnegie Mellon University Africa
 * Email:    ioj@andrew.cmu.edu
 * Date:     June 25, 2025
 * Version:  v1.0
 *
 * Copyright (C) 2025 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `robotLocalizationTestImplementation.cpp`

```
/* robotLocalizationTestImplementation.cpp - Robot Localization Unit Test Implementation
 *
 * Author:   Ibrahim Olaide Jimoh, Carnegie Mellon University Africa
 * Email:    ioj@andrew.cmu.edu
 * Date:     June 25, 2025
 * Version:  v1.0
 *
 * Copyright (C) 2025 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `robotLocalizationTestDriver.cpp`

```
/* robotLocalizationTestDriver.cpp - Simple driver for testing
 *
 * Author:   Ibrahim Olaide Jimoh, Carnegie Mellon University Africa
 * Email:    ioj@andrew.cmu.edu
 * Date:     June 25, 2025
 * Version:  v1.0
 *
 * Copyright (C) 2025 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `robotLocalizationTestApplication.cpp` file contains a documentation comment with the following sections:

```
☑ /* robotLocalizationTestApplication.cpp - Application code to run the
       Robot Localization Unit tests.
   *
   * This module is responsible for running comprehensive tests on the robot
   * localization module. The tests are run using Google Test and the results
   * are written to a file. The module tests pose setting, pose reset, marker
   * detection, accuracy, service functionality, and system stability.

☑ * Libraries
   *       Standard libraries
   -           std::string, std::vector, std::fstream, std::chrono
   *       ROS libraries
   -           ros/ros.h, ros/package.h, geometry_msgs/Pose2D.h
   *       OpenCV libraries
   -           opencv2/opencv.hpp, opencv2/aruco.hpp

☑ * Parameters
   *       Command-line Parameters
   *           None
   *
   *       Configuration File Parameters
   *           Key                      | Value
   *           ------------------------ | -------------------
   *           poseSetTests             | true
   *           poseResetTests           | true
   *           markerDetectionTests     | true
   *           accuracyTests            | true
   *           serviceTests             | true
   *           stabilityTests           | true
   *           verboseMode              | true

☑ * Subscribed Topics and Message Types
   *       /robotLocalization/pose                 geometry_msgs/Pose2D
   *       /naoqi_driver/odom                      nav_msgs/Odometry

☑ * Published Topics and Message Types
   *       None

☑ * Services Used
   *       /robotLocalization/set_pose             cssr_system/setPose
   *       /robotLocalization/reset_pose           cssr_system/resetPose

☑ * Services Advertised and Message Types
   *       None

☑ * Input Data Files
   *       arucoLandmarks.json (test landmarks)
   *       pepperTopics.dat
   *       cameraInfo.json

☑ * Output Data Files
   *       robotLocalizationTestOutput.dat
```

☑ ⋆ Configuration Files
    ⋆   robotLocalizationTestConfiguration.ini

☑ ⋆ Example Instantiation of the Module
    ⋆   rosrun unit_tests robotLocalizationTest

☑ ⋆ Author: Ibrahim Olaide Jimoh, Carnegie Mellon University Africa

☑ ⋆ Email: ioj@andrew.cmu.edu

☑ ⋆ Date: June 25, 2025

☑ ⋆ Version: v1.0

5.2.3 Component Unit Testing

☑ A unit test application named `robotLocalizationTestLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `robotLocalizationTestLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `robotLocalizationTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `robotLocalizationTestLaunchRobot.launch` file launches the component being tested.

⊟ The `robotLocalizationTestLaunchSimulator.launch` file launches the component being tested.

☑ The `robotLocalizationTestLaunchTestHarness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
robotLocalizationTest: v1.0

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
robotLocalizationTest: start-up.
robotLocalizationTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
robotLocalizationTest: running.
```

☑ The robotLocalizationTestLaunchRobot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The robotLocalizationTestLaunchSimulator.launch file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

☑ The robotLocalizationTestLaunchTestHarness.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named README.md in the robotLocalizationTest directory.

  ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (robotLocalizationTestConfiguration.ini) file are altered.

6 D4.3.2 Speech Event

6.1 Speech Event

The **Speech Event** software was submitted for integration on April 7th, 2025. The results indicate the software has successfully passed the integration process.

6.1.1 Files and Directories

- ☑ Files for a single component are stored in a subdirectory named **speech_event**.
- ☑ The **speech_event** directory has five sub-directories: `config`, `data`, `models`, `src`, and `srv`.
- ☑ The `config` directory contains one file, named.
  - ☑ `speech_event_configuration.ini`
  - ☑ The configuration file `speech_event_configuration.ini` contains the key-value pairs that set the component parameters.
  - ☑ Each key-value pair is written on a separate line.
- ☑ The `data` directory contains one file, named as follows.
  - ☑ `pepper_topics.dat`
- ☑ The `model` directory contains two files, named as follows.
  - ☑ `stt_en_conformer_transducer_large.nemo`
  - ☑ `stt_rw_comformer_transducer_large.nemo`
- ☑ The `src` directory contains two source files, named as follows.
  - ☑ `speech_event_application.py`
  - ☑ `speech_event_implementation.py`
- ☑ The `srv` directory contains two files, named as follows.
  - ☑ `set_enabled.srv`
  - ☑ `set_language.srv`
- ☑ The `speech_event` directory contains a `README.md` file with instructions on how to run the software.
- ☑ The `speech_event` directory contains no `CMakeLists.txt` build file.
- ☑ The `speech_event` directory contains no `package.xml` manifest file since it is a package within the `cssr_system` ROS node.
- ☑ The `speech_event` directory contains a `speech_event_requirements.txt` file with the dependencies (and their versions) required to be installed for proper functionality of the node.

6.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `speech_event_application.py`

```
"""
speech_event_application.py - speechEvent ROS node definition

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ `speech_event_implementation.py`

```
""""
speech_event_implementation.py - audio manipulation functions that support
speechEvent

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The `speech_event_application.py` file contains a documentation comment:

☑
```
"""
 speech_event_application.py - speechEvent ROS node definition

     This program defines the speechEvent ROS node. The speechEvent ROS node
     transcribes Kinyarwanda and English speech utterances in an audio signal
     published by the soundDetection ROS node on the /soundDetection/signal ROS
     topic, and publishes the transcribed text on the /speechEvent/text ROS topic.
```

☑
```
Libraries:
           - Ubuntu libraries: cython3 ffmpeg gfortran libopenblas-dev
               libopenblas64-dev patchelf pkg-config python3-testresources
               python3-typing-extensions sox
           - Python libraries: nemo, numpy, rospy, scipy, std_msgs, torch
```

```
☑ Parameters:
          - None

☑ Command-line Parameters:
          - None

☑ Configuration File Parameters:
          - language                         Kinyarwanda | English
          - verboseMode                      true | false
          - cuda                             true | false
          - sampleRate                       48000
          - heartbeatMsgPeriod               10

☑ Subscribed Topics and Message Types:
          - /soundDetection/signal           std_msgs/Float32MultiArray

☑ Published Topics and Message Types:
          - /speechEvent/text                std_msgs/String

☑ Services Invoked:
          - None

☑ Services Advertised and Request Message:
          - /speechEvent/set_language         kinyarwanda | english

☑ Input Data Files:
          - speech_event_input.dat
          - pepper_topics.dat

☑ Output Data Files:
          - None

☑ Configuration Files:
          - speech_event_configuration.ini

☑ Example Instantiation of the Module:
          - rosrun cssr_system_speech_event speech_event_application.py

☑ Author: Clifford Onyonka, Carnegie Mellon University Africa

☑ Email: cliffor2@andrew.cmu.edu

☑ Date: 2025-02-23

☑ Version: v1.0
```

6.1.3 Component Unit Testing

⊟ A unit test application named `speech_event_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `speech_event_launch_simulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `speech_event_launch_test_harness.launch` is provided in the `launch` directory.

⊟ The `speech_event>_launch_robot.launch` file launches the component being tested.

⊟ The `speech_event>_launch_simulator.launch` file launches the component being tested.

⊟ The `speech_event>_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup: (The output is there, but could use better formatting (check additional comments)

```
 speechEvent v1.0

 This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.

 Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node: (The output is there, but could use better formatting (check additional comments)

```
speechEvent: start-up.
speechEvent: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
speechEvent: running.
```

⊟ The `speech_event_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `speech_event_launch_simulator.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

⊟ The `speech_event_launch_test_harness.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

- ☑ Unit test instructions are provided in a file named `README.md` in the `speech_event` directory. (The unit test instructions are there, but need some more information and better structure for easy usage)
  - ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
  - ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`speech_event_configuration.ini`) file are altered.

6.2 Speech Event Unit Test

6.2.1 Files and Directories

- ☑ Files for a single component are stored in a subdirectory named **speech_event_test**.
- ☑ The **speech_event_test** directory has five sub-directories: `config`, `data`, `launch`, `src`, and `srv`.
- ☑ The `config` directory contains two files, named.
  - ☑ `speech_event_driver_configuration.ini`
  - ☑ `speech_event_test_configuration.ini`
  - ☑ The configuration file `speech_event_test_configuration.json` contains the key-value pairs that set the component parameters. (Changing the language or the verbose mode has no visible effects)
  - ☑ Each key-value pair is written on a separate line.
- ☑ The `data` directory contains two data files and seven audio files, named as follows.
  - ☑ `pepper_topics.dat`
  - ☑ `speech_event_test_output.dat`
  - ☑ `en-he_now_resides_in_monte_carlo.wav`
  - ☑ `en-the_council_was_held.wav`
  - ☑ `en-turner_construction_was_the_construction_manager_for_the_project.wav`
  - ☑ `rw-atandatu.wav`
  - ☑ `rw-ibikenewe.wav`
  - ☑ `rw-ingendo.wav`
  - ☑ `rw-kabiri.wav`
- ☑ The `launch` directory contains two files, named as follows.
  - ☑ `speech_event_launch_robot.launch`

  - ☑ `speech_event_launch_test_harness.launch`

- ☑ The `src` directory contains three source files, named as follows.

  - ☑ `speech_event_driver.py`
  - ☑ `speech_event_test_application.py`
  - ☑ `speech_event_test_implementation.py`

- ☑ The `srv` directory contains three source files, named as follows.

  - ☑ `set_next_test_file.srv`

- ☑ The `speech_event` directory contains a `README.md` file with instructions on how to run the software. (README file not completely intuitive).

- ☑ The `speech_event` directory contains no `CMakeLists.txt` build file. (It is not necessary if you use the build setup in additional comments below).

- ☑ The `speech_event` directory contains no `package.xml` manifest file since it is a package within the `unit_tests` ROS node.

- ⊟ The `speech_event` directory contains a `speech_event_requirements.txt` file with the dependencies (and their versions) required to be installed for proper functionality of the node.

6.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

- ☑ `speech_event_driver.py`

```
"""
speech_event_driver.py - program that emulates a soundDetection ROS node

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

- ☑ `speech_event_test_application.py`

```
"""
speech_event_test_application.py - speechEvent test ROS node definition

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ `speech_event_test_implementation.py`

```
""""
speech_event_test_implementation.py - functions to be used by the test ROS node


Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

The `speech_event_test_application.py` file contains a documentation comment:

```
☑ """
  speech_event_test_application.py - speechEvent test ROS node definition

      This program defines a ROS node that is used to test speechEvent. It
      subscribes to the /speechEvent/text ROS topic.


☑ Libraries:
          - Ubuntu libraries: None
          - Python libraries: nemo, rospy, scipy

☑ Parameters:
          - None

      Command-line Parameters:
          - None

☑ Command-line Parameters:
          - None

☑ Configuration File Parameters:
          - language                          Kinyarwanda | English
          - verboseMode                       true | false
          - cuda                              true | false
          - sampleRate                        48000
          - heartbeatMsgPeriod                10

☑ Subscribed Topics and Message Types:
          - /speechEvent/text                 std_msgs/String

☑ Published Topics and Message Types:
          - None

☑ Services Invoked:
          - /speechEvent/set_language

☑ Services Advertised and Request Message:
          - None

☑ Input Data Files:
          - speech_event_test_input.dat
          - pepper_topics.dat

☑ Output Data Files:
          - speech_event_test_output.dat

☑ Configuration Files:
          - speech_event_test_configuration.ini

☑ Example Instantiation of the Module:
          - rosrun unit_tests_speech_event speech_event_test_application.py

☑ Author:     Clifford Onyonka, Carnegie Mellon University Africa

☑ Email:      cliffor2@andrew.cmu.edu

☑ Date:       2025-02-23

☑ Version: v1.0
```

6.2.3 Component Unit Testing

☑ A unit test application named `speech_event_test_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `speech_event_launch_simulator.launch` is provided in the `launch` directory.

☑ A unit test application named `speech_event_launch_test_harness.launch` is provided in the `launch` directory.

☑ The `speech_event_launch_robot.launch` file launches the component being tested. (It did not run. Please inspect)

⊟ The `speech_event_launch_simulator.launch` file launches the component being tested.

☑ The speech_event_launch_test_harness.launch file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
speechEventTest v1.0

This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
speechEventTest: start-up.
speechEventTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
speechEventTest: running.
```

☑ The speech_event_launch_robot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The speech_event_launch_simulator.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ The speech_event_launch_test_harness.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required. (No infoirmation in the readme file explaining what tests)

☑ Unit test instructions are provided in a file named README.md in the speech_event_test directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (speech_event_test_configuration.ini) file are altered.

7 D5.1 Actuator Tests

7.1 Actuator Tests

7.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **pepper_interface_tests** outside of the `cssr_system` and `unit_tests` package directories. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **pepper_interface_tests** directory five sub-directories: `config`, `data`, `include/pepper_interface_tests`, `launch`, and `src`.

☑ The `config` directory contains two files, named as follows.

- ☑ `actuatorTestConfiguration.json` and `sensorTestConfiguration.json`
- ☑ The configuration files `actuatorTestConfiguration.ini` and `sensorTestConfiguration.json` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains four input fules and one output file.

- The input files are named as follows.
  - ☑ `actuatorTestInput.dat`
  - ☑ `pepperTopics.dat`
  - ☑ `sensorTestInput.dat`
  - ☑ `simulatorTopics.dat`
  - ☑ The topics files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
  - ☑ Each key-value pair of the `pepperTopics.dat` and `sensorTopics.dat` files is written on a separate line.
- The output files are named as follows.
  - ☑ `sensorTestOutput.dat`

☑ The `include/pepper_interface_tests` directory contains two files, named as follows.

- ☑ `actuatorTestInterface.h`
- ☑ `sensorTestInterface.h`

☑ The `launch` directory contains three files, named as follows.

- ☑ `actuatorTestLaunchRobot.launch`
- ☑ `interfaceTestLaunchSimulator.launch`
- ☑ `sensorTestLaunchRobot.launch`

☑ The `src` directory contains four source files, named as follows.

- ☑ `actuatorTestApplication.cpp`
- ☑ `actuatorTestImplementation.cpp`
- ☑ `sensorTestApplication.cpp`
- ☑ `sensorTestImplementation.cpp`

☑ The `pepper_interface_tests` directory or node subdirectory contains a `README.md` file with instructions on how to run the node.

☑ The `pepper_interface_tests` directory contains a `CMakeLists.txt` build file.

☑ The `pepper_interface_tests` directory contains a `package.xml` manifest file since it is a package directory.

7.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `actuatorTestApplication.cpp`

```
/* actuatorTestApplication.cpp Application code for running the actuator tests on
*        Pepper robot
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
```

☑ `actuatorTestImplementation.cpp`

```
/* actuatorTestImplementation.cpp Implementation code for running the actuator tests on
*        Pepper robot
*
* Author: Yohannes Tadesse Haile and Mihirteab Taye Hordofa
* Date: September 25, 2025
* Version: v1.1
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ `actuatorTestInterface.h`

```
/* actuatorTestInterface.h - Header file for the actuatorTest module to test the
 *       actuators of the Pepper robot using ROS interface.
 *
 * Author:  Yohannes Tadesse Haile, Carnegie Mellon University Africa
 * Email:   yohanneh@andrew.cmu.edu
 * Date:    September 25, 2025
 * Version: v1.1
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `actuatorTestApplication.cpp` file contains a documentation comment with the following sections:

☑
```
/* actuatorTestApplication.cpp Application code for running the actuator
 *        tests on Pepper robot
 *
 * The component test the functionality of the actuator of the robot using
 * the ROS interface. The test is performed by sending commands to the
 * robot and checking if the robot performs the expected action. The test
 * is performed in two modes: sequential and parallel. In the sequential
 * mode, the tests are performed one after the other. In the parallel
 * mode, the tests are performed simultaneously.
 *
```

☑
```
* Libraries
 * Standard libraries
 - std::string, std::vector, std::thread, std::fstream, std::cout,
        std::endl, std::cin, std::pow, std::sqrt, std::abs
 * ROS libraries
 - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
        control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
```

☑
```
* Parameters
 *
 * Command-line Parameters
 *
 * None

 * Configuration File Parameters

 * Key | Value
 * --- | ---
 * platform        | robot
 * simulatorTopics | simulatorTopics.dat
 * robotTopics     | pepperTopics.dat
 * mode            | sequential
 *
 * Key | Value
```

```
    * --- | ---
    * Head   | true
    * RArm   | true
    * LArm   | true
    * RHand  | true
    * LHand  | true
    * Leg    | true
    * Wheels | false

☑ * Subscribed Topics and Message Types
    *
    * None

☑ * Published Topics and Message Types
    *
    * /pepper_dcm/Head_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper_dcm/RightArm_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper_dcm/LeftArm_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper_dcm/RightHand_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper_dcm/LeftHand_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper_dcm/Pelvis_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /cmd_vel
            geometry_msgs/Twist

    * /pepper/Head_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper/RightArm_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper/LeftArm_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper/Pelvis_controller/follow_joint_trajectory
            trajectory_msgs/JointTrajectory
    * /pepper/cmd_vel
            geometry_msgs/Twist

☑ * Services Invoked
    *
    * None

☑ * Services Advertised and Request Message
    *
    * None

☑ * Input Data Files
    *
    * pepperTopics.dat
    * simulatorTopics.dat
    * actuatorTestInput.dat
```

☑
```
* Output Data Files
 *
 * None
```

☑
```
* Configuration Files
 *
 * actuatorTestConfiguration.ini
```

☑
```
* Example Instantiation of the Module
 *
 * rosrun pepper_interface_tests actuatorTest
```

☑
```
* Author: Yohannes Tadesse Haile and Mihirteab Taye Hordofa,
        Carnegie Mellon University Africa
 * Author: Yohannes Tadesse Haile and Mihirteab Taye Hordofa,
        Carnegie Mellon University Africa
```

☑
```
* Email: yohanneh@andrew.cmu.edu
```

☑
```
* Date: September 25, 2025
```

☑
```
* Version: v1.1
```

7.1.3 Component Unit Testing

☑ A unit test application named `actuatorTestLaunchRobot.launch` is provided in the `launch` directory.

☑ A unit test application named `interfaceTestLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `actuatorTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `actuatorTestLaunchRobot.launch` file launches the component being tested.

☑ The `interfaceTestLaunchSimulator.launch` file launches the component being tested.

⊟ The `actuatorTestLaunchTestHarness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
actuatorTest: v1.1

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
actuatorTest: start-up.
actuatorTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
actuatorTest: running.
```

☑ The `actuatorTestLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

☑ The `interfaceTestLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `actuatorTestLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `pepper_interface_tests` directory.

  - ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
  - ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`actuatorTestConfiguration.ini`) file are altered.

8 D5.2 Animate Behaviour Subsystem

8.1 Animate Behaviour

8.1.1 Files and Directories

- ☑ Files for a single component are stored in a subdirectory named **animateBehaviour**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

- ☑ The **animateBehaviour** directory has five sub-directories: `config`, `data`, `include/animateBehaviour`, `src`, and `srv`.

- ☑ The `config` directory contains one file, named.

  - ☑ `animateBehaviourConfiguration.ini`
  - ☑ The configuration file `animateBehaviourConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ☑ Each key-value pair is written on a separate line.

- ☑ The `data` directory contains three files, named as follows.

  - ☑ `animateBehaviourLogFile.log`
  - ☑ `pepperTopics.dat`
  - ☑ `simulatorTopics.dat`
  - ☑ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
  - ☑ Each key-value pair is written on a separate line.

- ☑ The `include/animateBehaviour` directory contains one file, named:

  - ☑ `animateBehaviourInterface.h`

- ☑ The `src` directory contains two source files, named as follows.

  - ☑ `animateBehaviourApplication.cpp`
  - ☑ `animateBehaviourImplementation.cpp`

- ☑ The `srv` directory contains one file, named:

  - ☑ `setActivation.srv`

- ☑ The `animateBehaviour` directory contains a `README.md` file with instructions on how to run the node

- ☑ The `animateBehaviour` directory contains a `CMakeLists.txt` build file.

- ☑ The `animateBehaviour` directory contains no `package.xml` manifest file since it is a node within the `cssr_system` package .

8.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ animateBehaviourApplication.cpp

```
/* animateBehaviourApplication.cpp
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ animateBehaviourImplementation.cpp

```
/* animateBehaviourImplementation.cpp
  *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `animateBehaviourInterface.h`

```
/* animateBehaviourInterface.h
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `animateBehaviourApplication.cpp` file contains a documentation comment:

☑

```
/*
 * animateBehaviourApplication.cpp
 * Animate behavior controller for creating lifelike robot movements
 *
 * Implements a ROS node that generates subtle autonomous movements to make the
 * robot appear more lifelike and animate. The system manages three types
 * of movements:
 * - Subtle body joint movements
 * - Hand flexing movements
 * - Small base rotations around the z-axis
 *
 * Key features:
 * - Maintains movements near home positions using randomized patterns
 * - Configurable movement ranges (as percentage of joint ranges)
 * - No head control
 * - Supports selective enabling of different movement types
 * - Can be enabled/disabled via ROS service for social interaction coordination
 * - Configurable via external topic mapping files for physical/simulated robots
 * - Optional verbose mode for movement debugging
 *
 * Debug Settings:
 *   - verboseMode: Enable/disable debug output (default: false)
 *
 *   Range Parameters:
 *   - rotMaximumRange: Maximum rotation range for base movement
 *   - selectedRange: Selected movement range as fraction of maximum
 *   - armMaximumRange: Maximum range for each arm joint [5 values]
 *   - handMaximumRange: Maximum range for hand movement
 *   - legMaximumRange: Maximum range for each leg joint [3 values]
 *
 *   Movement Parameters:
 *   - gestureDuration: Duration of each movement in seconds
```

```
     *   - numPoints: Number of points for arm and hand movements
     *   - numPointsLeg: Number of points for leg movements
     *   - legRepeatFactor: Number of repetitions for leg movements
     *
    *
☑   * Libraries:
     *   - ROS core libraries:
     *     - roscpp
     *     - ros/package.h
     *     - actionlib
     *     - control_msgs
     *     - trajectory_msgs
     *     - geometry_msgs
     *   - Standard C++ libraries:
     *     - iostream
     *     - fstream
     *     - thread
     *     - chrono
     *     - vector
     *     - map
     *     - string
     *     - atomic
     *     - random
     *
☑    * Parameters:
     *   ROS Parameters:
     *   - None
     *
☑   * Configuration File Parameters (animateBehaviourConfiguration.ini):
     *   - platform: Target platform ("robot" or "simulator")
     *   - behaviour: Type of animation behavior ("body", "hands", "rotation", "All")
     *   - simulatorTopics: Topic mapping file for simulator ("simulatorTopics.dat")
     *   - robotTopics: Topic mapping file for robot ("pepperTopics.dat")
     *   - verboseMode: Enable/disable debug output (default: false)
     *   - rotMaximumRange
     *   - selectedRange
     *   - armMaximumRange
         - handMaximumRange
         - legMaximumRange
         - gestureDuration
         - numPoints
         - numPointsLeg
         - legRepeatFactor
     *
☑  * Subscribed Topics and Message Types:
     *   - None
     *
```

```
☑  * Published Topics and Message Types:
    * - None (This node does not publish any topics directly)
    *
    * Topics Used By Node (But Not Published):
    *   - Action Client Topics (Node sends goals to these action servers):
    *      Physical Robot:
    *           - /pepper_dcm/RightHand_controller/follow_joint_trajectory
    *           - /pepper_dcm/LeftHand_controller/follow_joint_trajectory
    *           - /pepper_dcm/RightArm_controller/follow_joint_trajectory
    *           - /pepper_dcm/LeftArm_controller/follow_joint_trajectory
    *           - /pepper_dcm/Pelvis_controller/follow_joint_trajectory
    *
    *    Simulator:
    *           - /pepper/RightArm_controller/follow_joint_trajectory
    *           - /pepper/LeftArm_controller/follow_joint_trajectory
    *           - /pepper/Pelvis_controller/follow_joint_trajectory
    *
    * Topics Used via Publishers (Node sends messages but doesn’t publish topics):
    *    Physical Robot:
    *          - /pepper_dcm/cmd_moveto
    *
    *    Simulator:
    *          - /pepper/cmd_vel
    *
    * Note: It sends action goals and movement commands through established topics
    *  but does not create or publish any topics of its own.
    *

☑  * Services Invoked:
    *  - None
    *

☑  * Services Advertised:
    *   - animateBehaviour/set_activation (cssr_system/set_activation)
    *      Request: string state ("enabled" or "disabled")
    *      Response: bool success
    *

☑  * Input Data Files:
    *   - pepperTopics.dat:
    *   - simulatorTopics.dat:
    *
   *

☑  * Output Data Files:
    *  - animateBehaviourLogFile.log: Log file for runtime messages
    *

☑  * Configuration Files
    *   - animateBehaviourConfiguration.ini:
    *
```

☑
```
*  Example Instantiation of the Module
  *   - rosrun cssr_system animateBehaviour
  *   - rosservice call /animateBehaviour/set_activation "state: 'enabled'"
  *   - rosservice call /animateBehaviour/set_activation "state: 'disabled'"
  *
 *
```

☑ `* Author:  Eyerusalem Mamuye Birhan, Carnegie Mellon University Africa`

☑ `* Email:   ebirhan@andrew.cmu.edu`

☑ `* Date:  2025-01-10`

☑ `* Version: v1.0`

8.1.3 Component Unit Testing

⊟ A unit test application named `animateBehaviourLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `animateBehaviourLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `animateBehaviourLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `animateBehaviourLaunchRobot.launch` file launches the component being tested.

⊟ The `animateBehaviourLaunchSimulator.launch` file launches the component being tested.

⊟ The `animateBehaviourLaunchTestHarness.launch` file launches the component being tested.

☑ The `animateBehaviour` being tested outputs the copyright message on startup:

```
animateBehaviour: v1.0

This project is funded by the African Engineering and Technology Network
 (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The `animateBehaviour` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
animateBehaviour: start-up.
animateBehaviour: subscribed to /topicName.
```

☑ The `animateBehaviour` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
animateBehaviour: running.
```

⊟ The `animateBehaviourLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `animateBehaviourLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `animateBehaviourLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `animateBehaviourTest` directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

8.2 Animate Behaviour Unit test

8.2.1 Files and Directories

☑ Files are stored in a subdirectory named **animateBehaviourTest**.

☑ The **animateBehaviourTest** directory has five sub-directories: `config`, `data`, `include/animateBehaviourTest`, `launch`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `animateBehaviourTestConfiguration.ini`
- ☑ The configuration file `animateBehaviourTestConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains one file, named as follows.

- ☑ `animateBehaviourTestOutput.dat`

☑ The `include/animateBehaviourTest` directory contains one file, named:

  ☑ `animateBehaviourTestInterface.h`

☑ The `launch` directory contains three files, named as follows.

  ☑ `animateBehaviourLaunchRobot.launch`
  ☑ `animateBehaviourLaunchSimulator.launch`
  ☑ `animateBehaviourLaunchTestHarness.launch`

☑ The `src` directory contains two source files, named as follows.

  ☑ `animateBehaviourTestApplication.cpp`
  ☑ `animateBehaviourTestImplementation.cpp`

☑ The `animateBehaviourTest` directory contains a `README.md` file with instructions on how to run the test.

☑ The `animateBehaviourTest` directory contains a `CMakeLists.txt` build file.

☑ The `animateBehaviourTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

8.2.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `animateBehaviourTestApplication.cpp`

```
/* animateBehaviourTestApplication.cpp
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ animateBehaviourTestImplementation.cpp

```
/* gestureExecutionImplementation.cpp
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ animateBehaviourTestInterface.h

```
/* animateBehaviourTestInterface.h
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The animateBehaviourTestApplication.cpp file contains a documentation comment with the following subsections:

☑ /*

```
 * animateBehaviourTestApplication.cpp
 * This code implements a ROS-based test application that uses Google Test
 * to validate the animate behavior module, generates structured test reports,
 * and supports continuous testing with user-controlled iterations.
 *
 * Implements a ROS node that executes a comprehensive test suite for the animate
 * behavior system. The test validates four types of movements:
 * - Subtle body joint movements
 * - Hand flexing movements
 * - Small base rotations around the z-axis
```

```
   * - Combined movements of all types
   *
   * Key features:
   * - Continuous test execution capability
   * - Automated test report generation
   * - Configurable test behavior via external configuration
   * - Clean setup and teardown between test runs
   * - User-controlled test repetition
   *
☑   * Libraries
   * Libraries:
   *   - ROS core libraries:
   *     - roscpp
   *     - ros/package.h
   *   - Testing libraries:
   *     - gtest/gtest.h
   *   - Standard C++ libraries:
   *     - iostream
   *     - fstream
   *     - ctime
   *     - string
   *
☑ * Parameters:
   *   - argCount: Number of command line arguments
   *   - argValues: Array of command line argument strings
   *
   * Command-line Parameters:
   *   - Standard ROS parameters
   *   - Google Test command line options
   *
☑ * Configuration File Parameters:
   *   Test configuration file (animateBehaviourTestConfiguration.ini):
   *   - hands:     True/False    # Enable or disable hand movement tests
   *   - body:      True/False    # Enable or disable body movement tests
   *   - rotation:  True/False    # Enable or disable rotation tests
   *   - All:       True/False    # Enable or disable combined movement tests
   *   Note: Setting value to True runs the test, False skips the test
   *
☑ * Subscribed Topics and Message Types:
   *   - None directly (handled by test implementations)
   *
☑ * Published Topics and Message Types:
   *   - None directly (handled by test implementations)
   *
☑   * Services Invoked
     *
   *   - /animateBehaviour/set_activation
   *     Used to enable/disable animate behavior during tests
   *
```

☑
```
 * Services Advertised and Message Types
```

☑
```
 * Input Data Files
 *
 * None
 *
```

☑
```
* Output Data Files:
 *   - animateBehaviourTest/data/animateBehaviourTestOutput.dat:Test execution
   results and timing information
```

☑
```
* Configuration Files:
 * - unit_tests/animateBehaviourTest/config/animateBehaviourTestConfiguration.ini:
```

☑
```
 * Example Instantiation of the Module
 *
 * animateBehaviourTestLaunchTestHarness.launch.
 *
 * roslaunch unit_tests animateBehaviourTestLaunchTestHarness.launch
 *
```

☑
```
 * Author:  Eyerusalem Mamuye Birhan, Carnegie Mellon University Africa
```

☑
```
 * Email:   ebirhan@andrew.cmu.edu
```

☑
```
 * Date:  2025-01-10
```

☑
```
 * Version: v1.0
```

8.2.3 Component Unit Testing

☑ A unit test application named `animateBehaviourTestLaunchRobot.launch` is provided in the `launch` directory.

☑ A unit test application named `animateBehaviourTestLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `animateBehaviourTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `animateBehaviourTestLaunchRobot.launch` file launches the component being tested.

⊟ The `animateBehaviourTestLaunchSimulator.launch` file launches the component being tested.

☑ The `animateBehaviourTestLaunchTestHarness.launch` file launches the component being tested.

☑ The animateBehaviourTest being tested outputs the copyright message on startup:

```
animateBehaviourTest: v1.0

This project is funded by the African Engineering and Technology Network
  (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The animateBehaviourTest being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
animateBehaviourTest: start-up.
```

☑ The animateBehaviourTest being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
animateBehaviourTest: running.
```

☑ The animateBehaviourTestLaunchRobot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The animateBehaviourTestLaunchSimulator.launch file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

☑ The animateBehaviourTestLaunchTestHarness.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named README.md in the animateBehaviourTest directory.

  ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (.ini) file are altered.

9 D5.3 Attention Subsystem

9.1 Overt Attention

9.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **overtAttention**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **overtAttention** directory has six sub-directories: `config`, `data`, `include/overtAttention`, `msg`, `src`, and `srv`.

☑ The `config` directory contains one file, named.

- ☑ `overtAttentionConfiguration.ini`
- ☑ The configuration file `overtAttentionConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains two files, named as follows.

- ☑ `pepperTopics.dat`
- ☑ `simulatorTopics.dat`
- ☑ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ☑ Each key-value pair is written on a separate line.

☑ The `include/overtAttention` directory contains one file, named:

- ☑ `overtAttentionInterface.h`

☑ The `msg` directory contains two files, named:

- ☑ `Mode.msg`
- ☑ `Status.msg`

☑ The `src` directory contains two source files, named as follows.

- ☑ `overtAttentionApplication.cpp`
- ☑ `overtAttentionImplementation.cpp`

☑ The `srv` directory contains one file, named:

- ☑ `setMode.srv`

☑ The `overtAttention` directory contains a `README.md` file with instructions on how to run the node

☑ The `overtAttention` directory contains a `CMakeLists.txt` build file.

☑ The `overtAttention` directory contains no `package.xml` manifest file since it is a node within the `cssr_system` package .

9.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `overtAttentionApplication.cpp`

```
/* overtAttentionApplication.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `overtAttentionImplementation.cpp`

```
/* overtAttentionImplementation.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
* This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `overtAttentionInterface.h`

```
/* overtAttentionInterface.h
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `overtAttentionApplication.cpp` file contains a documentation comment:

☑
```
/* overtAttentionApplication.cpp
 *
 * This module module equips the robot with the ability to direct its gaze toward
 *  salient features in its environment  or focus on specific locations,
 * facilitating socially and contextually appropriate behaviors.
 * This capability is crucial for enhancing the robot’s ability to interact
 * effectively with people and adapt to dynamic environments.
 *
 * The module operates in five distinct modes, each tailored to a specific context
 *   - Social mode is activated during social interactions, allowing the robot to
        focus on human faces, and voices.
 *     In this mode, the robot prioritizes social cues to maintain engagement
        and responsiveness.
 *
 *   - Scanning mode, on the other hand, is used when the robot is not engaged in
        social interaction.
 *     In this state, the robot scans its surroundings for potential interaction
        opportunities,
 *     focusing on people and objects of interest while giving higher priority to
        faces.
 *     The robot periodically shifts its focus to new areas,
        ensuring comprehensive environmental coverage.
 *
 *   - In location mode, the robot gazes at a specific target in its environment.
 *     If the robot’s head cannot achieve the required pose to fixate
        on the target,
 *     the robot’s base rotates to realign its head and body.
 *
 *   - Seeking mode enables the robot to establish mutual gaze with a nearby
        person by searching for a face looking directly at it.
 *     If this process is unsuccessful within a given timeframe, the robot returns
        either a success or failure status.
 *
 *   - Lastly, in disabled mode, the robot’s head remains centered and stationary,
        effectively deactivating the attention mechanism.
 *
 * To determine the focus of attention, the module generates two types of saliency
    maps.
 *   - A social saliency map leverages data from face detection and sound
        localization to identify socially significant features.
 *   - A general saliency map, on the other hand, uses information-theoretic
        models to identify visually conspicuous elements
 *     in the robot’s environment. These maps form the basis for the robot’s
        attentional behavior across different modes.
 *
 * In scanning mode, three key processes work together to enhance
    attentional dynamics.
 *   - First, a winner-take-all (WTA) mechanism identifies a single focus of
        attention from the saliency map using a selective tuning model.
 *   - Second, an Inhibition-of-Return (IOR) mechanism ensures that previously
        attended locations are deprioritized, encouraging exploration of new areas.
 *   - Third, a habituation process gradually reduces the salience of the current
        focus, ensuring that attention does not remain fixated on a single point
       for an extended period.
 *
 * The robot’s gaze is directed by publishing control commands to the
     headYaw and headPitch joints,
 * which align the head toward the selected focus of attention.
 * For aural attention, the robot adjusts its headYaw angle based on the angle of
    arrival of the sound.
 * Calibration parameters ensure accurate mapping between visual offsets in the
    image and the corresponding head joint angles.
 * When the required headYaw rotation exceeds a predefined threshold, the module
    coordinates the movement of the robot’s base
 * and head to maintain focus while realigning the head and torso.
 *
 * The module’s functionality is supported by four key inputs.
 * Data from the face detection and sound detection nodes inform the saliency map
    in social mode.
 * An RGB image from the robot’s camera is used to compute the saliency map in
    scanning mode,
 * while the robot’s current pose is utilized for attending to specific locations.
 * These inputs enable the module to adapt its behavior dynamically based on
    environmental conditions.
 *
 * The module provides four outputs to facilitate its operation.
 * First, it publishes control commands to the robot’s headYaw and headPitch
    joints,
 * as well as to the wheels and angular velocity when adjusting the robot’s pose.
 * Second, it generates an RGB image visualizing the saliency function and the
     current focus of attention,
 * which can be displayed in verbose mode for debugging purposes.
 * Third, the module continuously publishes the current active mode to the
    /overtAttention/mode topic,
 * enabling other system components to monitor the robot’s attentional state.
```

```
    * Finally, the module updates actuator topic names based on configuration files
      specific to either the physical robot or a simulation environment.
    *
    * The module’s operation is managed through dedicated ROS services. The module
      advertises services to allow the selection
    * of operational modes, such as social, scanning, or location mode.
    *
    * For ease of analysis, the module can also operate in verbose mode, where
      published data is printed to the terminal,
    * and output images are displayed in OpenCV windows.
    *
    *

✓   * Libraries
    * Standard libraries
    - std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
    * ROS libraries
    - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
      control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
    *

✓   * Parameters
    *
    * Command-line Parameters
    *
    * The attention mode to set the attention system to
    * The location in the world to pay attention to in x, y, z coordinates
    *

✓  * Configuration File Parameters

    * Key                   |     Value
    * --------------------- |     -------------------
    * platform                    simulator
    * camera                      FrontCamera
    * realignmentThreshold        5
    * xOffsetToHeadYaw            25
    * yOffsetToHeadPitch          20
    * simulatorTopics             simulatorTopics.dat
    * robotTopics                 pepperTopics.dat
    * verboseMode                 true

✓   * Subscribed Topics and Message Types
    *
    * /faceDetection/direction              faceDetection.msg
    * /robotLocalization/pose               sensor_msgs::JointState
    * /soundDetection/data                  std_msgs::Float64
    * /naoqi_driver/camera/front/image_raw  sensor_msgs::ImageConstPtr

✓  * Published Topics and Message Types
    *
    * /pepper_dcm/Head_controller/follow_joint_trajectory
    * /cmd_vel
    * /overtAttention/mode
```

```
☑  * Services Invoked
   *
   * None
   *

☑  * Services Advertised and Message Types
    *
   * /overtAttention/set_mode
   *

☑  * Input Data Files
   *
   * pepperTopics.dat
   * simulatorTopics.dat
   *

☑ * Output Data Files
   *
   * None
   *

☑  * Configuration Files
   *
   * overtAttentionConfiguration.ini
   *

☑  * Example Instantiation of the Module
   *
   * rosrun cssr_system overtAttention
   *
   * The clients can call the service by providing the attention mode and the
      location to pay attention to in the world.
   * The service will execute the attention mode selected and attend to the location
      provided if mode being set is location mode.
   * An example of calling the service is shown below:
   * ----- rosservice call /overAttention/set_mode -- location 3.0 2.0 1.0
   * This will set the attention mode to location and the location to pay attention
       to is (3.0, 2.0, 1.0)
   *

☑  * Author:  Muhammed Danso and Adedayo Akinade, Carnegie Mellon University Africa

☑  * Email:  mdanso@andrew.cmu.edu, aakinade@andrew.cmu.edu

☑  * Date:  January 10, 2025

☑  * Version: v1.0
```

9.1.3 Component Unit Testing

⊟ A unit test application named `overtAttentionLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `overtAttentionLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `overtAttentionLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `overtAttentionLaunchRobot.launch` file launches the component being tested.

⊟ The `overtAttentionLaunchSimulator.launch` file launches the component being tested.

⊟ The `overtAttentionLaunchTestHarness.launch` file launches the component being tested.

☑ The `overtAttention` being tested outputs the copyright message on startup:

```
overtAttention: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The `overtAttention` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
overtAttention: start-up.
overtAttention: subscribed to /topicName.
```

☑ The `overtAttention` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
overtAttention: running.
```

⊟ The `overtAttentionLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `overtAttentionLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `overtAttentionLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `overtAttentionTest` directory.

  ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

9.2 Overt Attention Unit test

9.2.1 Files and Directories

☑ Files are stored in a subdirectory named **overtAttentionTest**.

☑ The **overtAttentionTest** directory has six sub-directories: `config`, `data`, `include/overtAttentionTest`, `launch`, `msg`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `overtAttentionTestConfiguration.ini`
- ☑ The configuration file `overtAttentionTestConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains three files, named as follows.

- ☑ `overtAttentionTestOutput.dat`
- ☑ `pepperTopics.dat`
- ☑ `simulatorTopics.dat`
- ☑ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ☑ Each key-value pair is written on a separate line.

☑ The `include/overtAttentionTest` directory contains one file, named:

- ☑ `overtAttentionTestInterface.h`

☑ The `launch` directory contains three files, named as follows.

- ☑ `overtAttentionTestLaunchRobot.launch`
- ☑ `overtAttentionTestLaunchSimulator.launch`
- ☑ `overtAttentionTestLaunchTestHarness.launch`

☑ The `msg` directory contains one file, named:

- ☑ `faceDetection.msg`

☑ The `src` directory contains three source files, named as follows.

- ☑ `overtAttentionTestApplication.cpp`
- ☑ `overtAttentionTestDriver.cpp`
- ☑ `overtAttentionTestImplementation.cpp`

☑ The `overtAttentionTest` directory contains a `README.md` file with instructions on how to run the test.

☑ The `overtAttentionTest` directory contains a `CMakeLists.txt` build file.

☑ The `overtAttentionTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

9.2.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `overtAttentionTestApplication.cpp`

```
/* overtAttentionTestApplication.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `overtAttentionTestImplementation.cpp`

```
/* overtAttentionImplementation.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ overtAttentionTestInterface.h

```
/* overtAttentionTestInterface.h
 *
 * Author: AMohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ overtAttentionTestDriver.cpp

```
/* overtAttentionTestDriver.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinadee
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `overtAttentionTestApplication.cpp` file contains a documentation comment with the following subsections:

☑
```
/* overtAttentionTestApplication.cpp
 *
 * This module is responsible for running the tests on the overt attention module.
 * The tests are run using Google Test and the results are written to a file.
 * The module tests the scanning, social, seeking, location and disabled modes of
    the overtAttention node
 *
```

☑
```
 * Libraries
 * Standard libraries
 - std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
```

```
     * ROS libraries
     - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
       control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
     *

☑    * Parameters
     *
     * Command-line Parameters
     *
     * None
     *

☑   * Configuration File Parameters

     * Key                    |     Value
     * ---------------------- |     -------------------
     * platform                       robot
     * scanning                       true
     * social                         true
     * seeking                        true
     * location                       true
     * disabled                       true
     * verboseMode                    true
     *

☑    * Subscribed Topics and Message Types
      *
      * /faceDetection/direction              faceDetection.msg
     * /robotLocalization/pose                sensor_msgs::JointState
     * /soundDetection/data                   std_msgs::Float64
     * /naoqi_driver/camera/front/image_raw  sensor_msgs::ImageConstPtr
     * /overtAttention/mode                   Status.msg

☑   * Published Topics and Message Types
     *
     * None
     *

☑    * Services Invoked
       *
     * /overtAttention/set_mode
     *

☑    * Services Advertised and Message Types
      *
      * None

☑    * Input Data Files
     *
     * pepperTopics.dat
     * simulatorTopics.dat
     *
```

```
☑  * Output Data Files
   *
   * overtAttentionTestOutput.dat
   *

☑  * Configuration Files
   *
   * overtAttentionTestConfiguration.ini
   *

☑  * Example Instantiation of the Module
   *
   * roslaunch unit_tests overtAttentionTestLaunchTestHarness.launch
   *

☑  * Author:  Muhammed Danso and Adedayo Akinade, Carnegie Mellon University

☑  * Email:   mdanso@andrew.cmu.edu, aakinade@andrew.cmu.edu

☑  * Date:  January 10, 2025

☑  * Version: v1.0
```

9.2.3 Component Unit Testing

☑ A unit test application named `overtAttentionTestLaunchRobot.launch` is provided in the `launch` directory.

☑ A unit test application named `overtAttentionTestLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `overtAttentionTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `overtAttentionTestLaunchRobot.launch` file launches the component being tested.

⊟ The `overtAttentionTestLaunchSimulator.launch` file launches the component being tested.

☑ The `overtAttentionTestLaunchTestHarness.launch` file launches the component being tested.

☑ The `overtAttentionTest` being tested outputs the copyright message on startup:

```
overtAttentionTest: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The overtAttentionTest being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
overtAttentionTest: start-up.
```

☑ The overtAttentionTest being tested periodically (every ten seconds) writes a short heart-beat message to the terminal to indicate the state of the node:

```
overtAttentionTest: running.
```

☑ The overtAttentionTestLaunchRobot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The overtAttentionTestLaunchSimulator.launch file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

☑ The overtAttentionTestLaunchTestHarness.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named README.md in the overtAttentionTest directory.

  ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (.ini) file are altered.

10 D5.4.3 Robot Mission Interpreter

10.1 Behavior Controller

10.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **behaviourController**.

☑ The **behaviourController** directory has six sub-directories: `config`, `data`, `include/behaviourController`, `msg`, `src`, and `srv`.

☑ The `config` directory contains three files, named.

- ☑ `behaviourControllerConfiguration.ini`
- ☑ `cultureKnowledgeBaseConfiguration.ini`
- ☑ `environmentKnowledgeBaseConfiguration.ini`
- ☑ The configuration file `behaviourControllerConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ The configuration file `cultureKnowledgeBaseConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ The configuration file `environmentKnowledgeBaseConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains four files, named as follows.

- ☑ `cultureKnowledgeBaseInput.dat`
- ☑ `cultureKnowledgeValueTypesInput.dat`
- ☑ `environmentKnowledgeBaseInput.dat`
- ☑ `labTour.xml`
- ☑ Each key-value pair is written on a separate line.

☑ The `include/behaviourController` directory contains three files, named:

- ☑ `behaviourControllerInterface.h`
- ☑ `cultureKnowledgeBaseInterface.h`
- ☑ `environmentKnowledgeBaseInterface.h`

☑ The `msg` directory contains one file, named:

- ☑ `overtAttentionMode.msg`

☑ The src directory contains four source files, named as follows.

- ☑ behaviourControllerApplication.cpp
- ☑ behaviourControllerImplementation.cpp
- ☑ cultureKnowledgeBaseImplementation.cpp
- ☑ environmentKnowledgeBaseImplementation.cpp

☑ The srv directory contains nine files, named:

- ☑ animateBehaviorSetActivation.srv
- ☑ gestureExecutionPerformGesture.srv
- ☑ overtAttentionSetMode.srv
- ☑ robotLocalizationSetPose.srv
- ☑ robotNavigationSetGoal.srv
- ☑ speechEventSetLanguage.srv
- ☑ speechEventSetStatus.srv
- ☑ tabletEventPromptAndGetResponse.srv
- ☑ textToSpeechSayText.srv

☑ The behaviourController directory contains a README.md file with instructions on how to run the node

☑ The behaviourController directory contains a CMakeLists.txt build file.

☑ The behaviourController directory contains no package.xml manifest file since it is a node within the cssr_system package .

10.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ behaviourControllerApplication.cpp

```
/* behaviourControllerApplication.cpp
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ behaviourControllerImplementation.cpp

```
/* behaviorControllerImplementation.cpp   Source code for the implementation of
the robot mission node classes and other utility functions
 *
 * Author: Tsegazeab Taye Tefferi
 * Date: April 25, 2025
 * Version: 1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ cultureKnowledgeBaseImplementation.cpp

```
/* cultureKnowledgeBaseImplementation.cpp   Source code for the implementation of
the culture knowledge base helper class: CultureKnowledgeBase
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ environmentKnowledgeBaseImplementation.cpp

```
/* environmentKnowledgeBaseImplementation.cpp   Source code for the
implementation of the environment knowledge base helper class:
EnvironmentKnowledgeBase
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ behaviourControllerInterface.h

```
/* behaviorControllerInterface.h - interface file for
behaviorControllerApplication and behaviorControllerImplementation.
*
* Author: Tsegazeab Taye Tefferi
* Date: April 08, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ cultureKnowledgeBaseInterface.h

```
/* cultureKnowledgeBaseInterface.h   Interface source code for the culture
knowledge base helper class: CultureKnowledgeBase
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ environmentKnowledgeBaseInterface.h

```
/* environmentKnowledgeBaseInterface.h   Interface source code for the
environment knowledge base helper class: EnvironmentKnowledgeBase
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afret
 * Inclusive Digital Transformation Research Grant Programme.
 *
```

```
    * Website: www.cssr4africa.org
    *
    * This program comes with ABSOLUTELY NO WARRANTY.
    *
    */
```

The behaviourControllerApplication.cpp file contains a documentation comment:

```
✓ /* behaviorControllerApplication.cpp
   *
   * <detailed functional description>
   * The component starts the 'Robot Mission Interpreter' ROS Node.
   This node is the starting point for the robot to execute the mission selected.
   * Though, this component exists as a standalone and can be started as such,
   without the set of ROS nodes that are part of the CSSR4Afica system architecture
   (see D3.1 System Architecture), it will not be able function.
   ...

✓ * Libraries
   * Standard libraries
   - std::string, std::fstream
   * ROS libraries
   - ros/ros.h, ros/package.h, std_msgs
   * BehaviorTree.Cpp libraries
   - behaviortree_cpp/bt_factory.h, behaviortree_cpp/loggers/groot2_publisher.h
   ...

✓  * Parameters
   *
   * Command-line Parameters
   *
   * None
   *

✓ * Configuration File Parameters
   * Key | Value
   * ----|------
   * scenarioSpecification | <the mission scenario to be interpreted>
   * verboseMode           | <true/false - enables/disables the display of
                                 diagnostic messages>
   * asrEnabled            | <true/false> - enables/disables the Automatic Speech
                               Recognition. If diabled, pepper's tablet will be
                                            primary input method
   * audioDebugMode        | <true/false> - enables/disables the audio for debugging
   ...

✓ * Subscribed Topics and Message Types
   **
   - /overtAttention/mode      overtAttentionMode.msg
   - /speechEvent/text         std_msgs::String

✓  * Published Topics and Message Types
    *
    * None
```

☒
```
* Services Invoked
 * *
 * /animateBehaviour/setActivation
 * /gestureExecution/perform_gesture
 * /overtAttention/set_mode
 * /robotLocalization/reset_pose
 * /robotNavigation/set_goal
 * /speechEvent/set_language
 * /speechEvent/set_enabled
 * /tabletEvent/prompt_and_get_response
 * /textToSpeech/say_text
```

☒
```
 * Services Advertised and Message Types
  *
  * None
  *
```

☑
```
* Input Data Files
 *
 * lab_tour.xml
```

☑
```
* Output Data Files
 *
 * None
```

☑
```
 * Configuration Files
  *
  * behaviourControllerConfiguration.ini
  *
```

☑
```
 * Example Instantiation of the Module
  *
  * rosrun cssr_system behaviourController
  *
```

☑
```
* Author: Tsegazeab Taye Tefferi, Carnegie Mellon University Africa
```

☑
```
* Email: ttefferi@andrew.cmu.edu
```

☑
```
* Date: April 08, 2025
```

☑
```
* Version: v1.0
```

10.1.3 Component Unit Testing

⊟ A unit test application named `behaviourControllerLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `behaviourControllerLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `behaviourControllerLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `behaviourControllerLaunchRobot.launch` file launches the component being tested.

⊟ The `behaviourControllerLaunchSimulator.launch` file launches the component being tested.

⊟ The `behaviourControllerLaunchTestHarness.launch` file launches the component being tested.

☑ The `behaviourController` being tested outputs the copyright message on startup:

```
behaviourController: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The `behaviourController` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
behaviourController: start-up.
```

☑ The `behaviourController` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
behaviourController: running.
```

⊟ The `behaviourControllerLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `behaviourControllerLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `behaviourControllerLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `behaviourControllerTest` directory.

  ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

10.2 Behavior Controller Unit test

10.2.1 Files and Directories

- ☑ Files are stored in a subdirectory named **behaviourControllerTest**.
- ☑ The **behaviourControllerTest** directory has seven sub-directories: `config`, `data`, `include/behaviourControllerTest`, `launch`, `msg`, `src`, and `srv`.
- ☑ The `config` directory contains one file, named.
  - ☑ `behaviourControllerTestConfiguration.ini`
  - ☑ The configuration file `behaviourControllerTestConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ☑ Each key-value pair is written on a separate line.
- ☑ The `data` directory contains one file, named as follows.
  - ☑ `behaviorControllerTestOutput.dat`
- ☑ The `include/behaviourControllerTest` directory contains one file, named:
  - ☑ `behaviourControllerTestInterface.h`
- ☑ The `launch` directory contains one file, named as follows.
  - ⊟ `behaviourControllerTestLaunchRobot.launch`
  - ☑ `behaviourControllerTestLaunchTestHarness.launch`
- ☑ The `msg` directory contains one file, named as follows.
  - ☑ `overtAttentionMonde.msg`
- ☑ The `src` directory contains three source files, named as follows.
  - ☑ `behaviourControllerTestApplication.cpp`
  - ☑ `behaviourControllerTestDriver.cpp`
  - ☑ `behaviourControllerTestImplementation.cpp`
  - ☑ `behaviourControllerTestStub.cpp`
- ☑ The `srv` directory contains nine files, named:
  - ☑ `animateBehaviorSetActivation.srv`
  - ☑ `gestureExecutionPerformGesture.srv`
  - ☑ `overtAttentionSetMode.srv`
  - ☑ `robotLocalizationSetPose.srv`

- ☑ `robotNavigationSetGoal.srv`
- ☑ `speechEventSetLanguage.srv`
- ☑ `speechEventSetStatus.srv`
- ☑ `tabletEventPromptAndGetResponse.srv`
- ☑ `textToSpeechSayText.srv`

☑ The `behaviourControllerTest` directory contains a `README.md` file with instructions on how to run the test.

☑ The `behaviourControllerTest` directory contains a `CMakeLists.txt` build file.

☑ The `behaviourControllerTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

10.2.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `behaviourControllerTestApplication.cpp`

```
/* behaviourControllerTestApplication.cpp
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `behaviourControllerTestDriver.cpp`

```
/* behaviorControllerTestDriver.cpp   Source code for the simulated topics
(drivers)
 *
 * Author: Tsegazeab Taye Tefferi
 * Date: April 25, 2025
 * Version: 1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ behaviourControllerTestImplementation.cpp

```
/* behaviorControllerTestImplementation.cpp   Source code for the methods used
by behaviiorControllerTestApplication
 *
 * Author: Tsegazeab Taye Tefferi
 * Date: April 25, 2025
 * Version: 1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ behaviourControllerTestStub.cpp

```
/* behaviorControllerTestStub.cpp   Source code for the simulated services
(stubs)
 *
 * Author: Tsegazeab Taye Tefferi
 * Date: April 25, 2025
 * Version: 1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
 (Afretec) Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 *
 */
```

☑ behaviourControllerTestInterface.h

```
/* behaviorControllerTestInterface.h - interface file for
behaviorControllerTestApplication and behaivorControllerTestImplementation
*
* Author: Tsegazeab Taye Tefferi
* Date: April 20, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network
```

```
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

The `behaviourControllerTestApplication.cpp` file contains a documentation comment with the following subsections:

```
☑ /* behaviourControllerTestApplication.cpp
   * <detailed functional description>
   * This module is responsible for running the tests on the behaviorController
   ROS node
   * The tests will check if all the action and condition nodes of the
    behaviorController are communicating
   * and processing the data they receive as expected
   *
```

```
☑ * Libraries
   * Standard libraries
   - std::string, std::vector, std::fstream
   * ROS libraries
   - ros/ros.h, ros/package.h
```

```
☑  * Parameters
    *
    * Command-line Parameters
    *
    * None
```

```
☑ * Configuration File Parameters

   * Key                    |     Value
   * ---------------------- |     -------------------
   * verboseMode  `         |      <true/false - enables/disables the display of
                                              diagnostic messages>
   * failureRate            |      the rate at which service calls and topics
                                 will provide a failed or not successful response
   * arrivalRate            |      the rate at wich an event occurs (valid only
                                for the driver functions)
```

```
☑  * Subscribed Topics and Message Types
    *
    * None
```

```
☑  * Published Topics and Message Types
    *
    * None
```

```
☒  * Services Invoked
     *
   * None
   *
```

☒ `* Services Advertised and Message Types`
`*`
`* None`
`*`

☑ `* Input Data Files`
`*`
`* None`

☑ `* Output Data Files`
`*`
`* behaviorControllerTestOutput.dat`

☑ `* Configuration Files`
`*`
`* behaviourControllerTestConfiguration.ini`

☑ `* Example Instantiation of the Module`
`*`
`* roslaunch unit_tests behaviorControllerLaunchTestHarness`

☑ `* Author: Tsegazeab Taye Tefferi, Carnegie Mellon University Africa`

☑ `* Email: ttefferi@andrew.cmu.edu`

☑ `* Date: April 20, 2025`

☑ `* Version: v1.0`

10.2.3 Component Unit Testing

☑ A unit test application named `behaviourControllerTestLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `behaviourControllerTestLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `behaviourControllerTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `behaviourControllerTestLaunchRobot.launch` file launches the component being tested.

⊟ The `behaviourControllerTestLaunchSimulator.launch` file launches the component being tested.

☑ The `behaviourControllerTestLaunchTestHarness.launch` file launches the component being tested.

- ☑ The behaviourControllerTest being used for test outputs the copyright message on startup:

```
behaviourControllerTest: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY
```

- ☑ The behaviourControllerTest being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
behaviourControllerTest: start-up.
```

- ⊟ The behaviourController being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
behaviourControllerTest: running.
```

- ☑ The behaviourControllerTestLaunchRobot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

- ⊟ The behaviourControllerLaunchSimulator.launch file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

- ☑ The behaviourControllerTestLaunchTestHarness.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

- ☑ Unit test instructions are provided in a file named README.md in the behaviourControllerTest directory.

  - ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.

  - ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (.ini) file are altered.

11 D5.5.1.1 Gesture Execution

11.1 Gesture Execution

11.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **gestureExecution**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **gestureExecution** directory has six sub-directories: `config`, `data`, `include/gestureExecution`, `msg`, `src`, and `srv`.

☑ The `config` directory contains one file, named.

- ☑ `gestureExecutionConfiguration.ini`
- ☑ The configuration file `gestureExecutionConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains eight files, named as follows.

- ☑ `gestureDescriptors.dat`
- ☑ `lArmShakeGestureDescriptors.dat`
- ☑ `lArmWelcomeGestureDescriptors.dat`
- ☑ `pepperTopics.dat`
- ☑ `rArmShakeGestureDescriptors.dat`
- ☑ `rArmWelcomeGestureDescriptors.dat`
- ☑ `simulatorTopics.dat`
- ☑ `waveGestureDescriptors.dat`
- ☑ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ☑ Each key-value pair is written on a separate line.

☑ The `include/gestureExecution` directory contains two files, named:

- ☑ `gestureExecutionInterface.h`
- ☑ `pepperKinematicsUtilitiesInterface.h`

☑ The `msg` directory contains one file, named:

- ☑ `Gesture.msg`

☑ The src directory contains three source files, named as follows.

  ☑ gestureExecutionApplication.cpp

  ☑ gestureExecutionImplementation.cpp

  ☑ pepperKinematicsUtilitiesImplementation.cpp

☑ The srv directory contains one files, named:

  ☑ performGesture.srv

☑ The gestureExecution directory contains a README.md file with instructions on how to run the node

☑ The gestureExecution directory contains a CMakeLists.txt build file.

☑ The gestureExecution directory contains no package.xml manifest file since it is a node within the cssr_system package .

11.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ gestureExecutionApplication.cpp

```
/* gestureExecutionApplication.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ gestureExecutionImplementation.cpp

```
/* gestureExecutionImplementation.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
* This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ pepperKinematicsUtilitiesImplementation.cpp

```
/* pepperKinematicsUtilitiesImplementation.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
* This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ gestureExecutionInterface.h

```
/* gestureExecutionInterface.h
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ pepperKinematicsUtilitiesInterface.h

```
/* pepperKinematicsUtilitiesInterface.h
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `gestureExecutionApplication.cpp` file contains a documentation comment:

☑

```
/* gestureExecutionApplication.cpp
 *
 * This module is responsible for hosting the service that executes the gestures on the robot.
 * The module receives the gesture type, gesture ID, gesture duration, bow/nod angle, and the location in the world to pay attention/point to in x, y, z coordinates.
 * The module then executes the gesture based on the received parameters.
 *
 * The module supports the execution of deictic, iconic, symbolic, bow, and nod gestures.
 * The iconic gestures currently supported are welcome and wave (goodbye) gestures.
```

```
   *
   * The module also supports the selection of the implementation platform
      (simulator or robot) and the interpolation type (linear or biological motion).
   *
   * The module is implemented as a ROS service that receives the gesture parameters
   * and returns the status of the gesture execution.
   *
   * The gestures could either be executed using linear velocity interpolation or
     a model of biological motion (minimum-jerk model).
   *
   * The module subscribes to the /sensor_msgs/joint_states topic to receive
     the joint states of the robot.
   * The module also subscribes to the /robotLocalization/pose topic to receive
     the coordinates of the robot in the world.
   *
   * The module is implemented in C++ and uses the ROS libraries for communication
     with the robot.
   * The module is part of the CSSR4A package and is used to execute gestures
      on the robot.
   *
   *
☑  * Libraries
   * Standard libraries
   - std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
   * ROS libraries
   - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
     control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
   *
☑  * Parameters
   *
   * Command-line Parameters
   *
   * None
   *
☑ * Configuration File Parameters

   * Key                   |     Value
   * --------------------- |     --------------------
   * platform                    robot
   * interpolation               biological
   * gestureDescriptors          gestureDescriptors.dat
   * simulatorTopics             simulatorTopics.dat
   * robotTopics                 pepperTopics.dat
   * verboseMode                 true
☑  * Subscribed Topics and Message Types
   *
   * /sensor_msgs/joint_states
   * /robotLocalization/pose
```

```
☑ * Published Topics and Message Types
    *
    * None

☑  * Services Invoked
     *
    * /overtAttention/set_mode
    *

☑  * Services Advertised and Message Types
     *
    * /gestureExecution/perform_gesture
    *

☑  * Input Data Files
    *
    * pepperTopics.dat
    * simulatorTopics.dat
    * gestureDescriptors.dat
    *

☑ * Output Data Files
    *
    * None
    *

☑  * Configuration Files
    *
    * gestureExecutionConfiguration.ini
    *

☑  * Example Instantiation of the Module
    *
    * rosrun gestureExecution perform_gesture
    *
    * The clients can invoke the service by providing the gesture type, gesture ID,
      gesture duration, bow_nod angle,
    * and the location in the world to pay attention/point to in x, y, z coordinates.
    * The service will execute the gesture based on the received parameters and
      return the status of the gesture execution.
    *
    * Examples of calling the service is shown below:
    * ----- rosservice call /perform_gesture -- deictic 01 3000 25 3.6 2.5 0.82
    * This will execute a pointing gesture with a duration of 3000 ms, and the
      location in the world to point to in x, y, z coordinates.
    *
    * ----- rosservice call /perform_gesture -- bow 01 3000 25 3.6 2.5 0.82
    * This will execute a pointing gesture with a duration of 3000 ms, and bow at an
      angle of 45 degrees.
    *

☑  * Author:  Adedayo Akinade, Carnegie Mellon University Africa

☑  * Email:   aakinade@andrew.cmu.edu
```

☑ * Date: January 10, 2025

☑ * Version: v1.0

11.1.3 Component Unit Testing

⊟ A unit test application named gestureExecutionLaunchRobot.launch is provided in the launch directory.

⊟ A unit test application named gestureExecutionLaunchSimulator.launch is provided in the launch directory.

⊟ A unit test application named gestureExecutionLaunchTestHarness.launch is provided in the launch directory.

⊟ The gestureExecutionLaunchRobot.launch file launches the component being tested.

⊟ The gestureExecutionLaunchSimulator.launch file launches the component being tested.

⊟ The gestureExecutionLaunchTestHarness.launch file launches the component being tested.

☑ The gestureExecution being tested outputs the copyright message on startup:

```
gestureExecution: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The gestureExecution being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
gestureExecution: start-up.
gestureExecution: subscribed to /topicName.
```

☑ The gestureExecution being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
gestureExecution: running.
```

⊟ The gestureExecutionLaunchRobot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The gestureExecutionLaunchSimulator.launch file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The gestureExecutionLaunchTestHarness.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named README.md in the gestureExecutionTest directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (.ini) file are altered.

11.2 Gesture Execution Unit test

11.2.1 Files and Directories

☑ Files are stored in a subdirectory named **gestureExecutionTest**.

☑ The **gestureExecutionTest** directory has six sub-directories: config, data, include/gestureExecutionTest, launch, src, and srv.

☑ The config directory contains one file, named.

- ☑ gestureExecutionTestConfiguration.ini
- ☑ The configuration file gestureExecutionTestConfiguration.ini contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The data directory contains three files, named as follows.

- ☑ gestureExecutionTestOutput.dat
- ☑ pepperTopics.dat
- ☑ simulatorTopics.dat
- ☑ The topic files pepperTopics.dat and simulatorTopics.dat contain the key-value pairs that set the topic names required by the component.
- ☑ Each key-value pair is written on a separate line.

☑ The include/gestureExecutionTest directory contains one file, named:

- ☑ gestureExecutionTestInterface.h

☑ The launch directory contains three files, named as follows.

- ☑ gestureExecutionLaunchRobot.launch

- ☑ gestureExecutionLaunchSimulator.launch
- ☑ gestureExecutionLaunchTestHarness.launch

☑ The src directory contains four source files, named as follows.

- ☑ gestureExecutionTestApplication.cpp
- ☑ gestureExecutionTestDriver.cpp
- ☑ gestureExecutionTestImplementation.cpp
- ☑ gestureExecutionTestStub.cpp

☑ The srv directory contains two files, named:

- ☑ setMode.srv
- ☑ setPose.srv

☑ The gestureExecutionTest directory contains a README.md file with instructions on how to run the test.

☑ The gestureExecutionTest directory contains a CMakeLists.txt build file.

☑ The gestureExecutionTest directory contains no package.xml manifest file since it is a node within the unit_tests package .

11.2.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ gestureExecutionTestApplication.cpp

```
/* gestureExecutionTestApplication.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ gestureExecutionTestImplementation.cpp

```
/* gestureExecutionImplementation.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ gestureExecutionTestInterface.h

```
/* gestureExecutionTestInterface.h
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ gestureExecutionTestDriver.cpp

```
/* gestureExecutionTestDriver.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ gestureExecutionTestStub.cpp

```
/* gestureExecutionTestStub.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `gestureExecutionTestApplication.cpp` file contains a documentation comment with the following subsections:

☑
```
/* gestureExecutionTestApplication.cpp
 *
 * This module is responsible for running the tests on the gesture execution node.
 * The tests are run using Google Test and the results are written to a file.
 * The module tests the iconic, deictic, bow, nod, and symbolic gestures.
 *
```

```
☑  * Libraries
   * Standard libraries
   - std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
   * ROS libraries
   - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
     control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
   *

☑  * Parameters
   *
   * Command-line Parameters
   *
   * None
   *

☑ * Configuration File Parameters

   * Key                    |     Value
   * ---------------------- |     -------------------
   * platform               |     robot
   * iconic                       true
   * deictic                      true
   * bow                          true
   * nod                          true
   * symbolic                     false
   *

☑  * Subscribed Topics and Message Types

☑ * Published Topics and Message Types
   *
   * /pepper/cmd_vel                 geometry_msgs/Twist
   *

☑  * Services Invoked
     *
   * /gestureExecution/perform_gesture
   *

☑  * Services Advertised and Message Types

☑  * Input Data Files
   *
   * None
   *

☑  * Output Data Files
   *
   * gestureExecutionTestOutput.dat
   *

☑  * Configuration Files
   *
   * gestureExecutionTestConfiguration.ini
   *
```

```
☑  * Example Instantiation of the Module
   *
   * rosrun unit_tests gestureExecutionTest
   *
   * The launch file for the gesture execution unit tests is
      gestureExecutionTestLaunchTestHarness.launch.
   *
   * roslaunch unit_tests gestureExecutionTestLaunchTestHarness.launch
   *

☑  * Author:  Adedayo Akinade, Carnegie Mellon University Africa

☑  * Email:   aakinade@andrew.cmu.edu

☑  * Date:  January 10, 2025

☑  * Version: v1.0
```

11.2.3 Component Unit Testing

☑ A unit test application named `gestureExecutionTestLaunchRobot.launch` is provided in the `launch` directory.

☑ A unit test application named `gestureExecutionTestLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `gestureExecutionTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `gestureExecutionTestLaunchRobot.launch` file launches the component being tested.

⊟ The `gestureExecutionTestLaunchSimulator.launch` file launches the component being tested.

☑ The `gestureExecutionTestLaunchTestHarness.launch` file launches the component being tested.

☑ The `gestureExecutionTest` being tested outputs the copyright message on startup:

```
gestureExecutionTest: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

- ☑ The `gestureExecutionTest` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

  ```
  gestureExecution: start-up.
  ```

- ☑ The `gestureExecutionTest` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

  ```
  gestureExecutionTest: running.
  ```

- ☑ The `gestureExecutionTestLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

- ⊟ The `gestureExecutionTestLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

- ☑ The `gestureExecutionTestLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

- ☑ Unit test instructions are provided in a file named `README.md` in the `gestureExecutionTest` directory.

  - ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
  - ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

12 D5.5.2.4 Integrated Text to Speech Conversion

12.1 Integrated Text to Speech Conversion

12.1.1 Files and Directories

- ☑ Files for a single component are stored in a subdirectory named **text_to_speech**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

- ☑ The **text_to_speech** directory has four sub-directories: `config`, `data`, `src`, and `srv`.

- ☑ The `config` directory contains one file, named as follows.

  - ☑ `text_to_speech_configuration.ini`
  - ☑ The configuration file `text_to_speech_configuration.ini` contains the key-value pairs that set the component parameters.

  - ☑ Each key-value pair is written on a separate line.

- ☑ The data directory contains one input file.

  - ☑ pepper_topics.dat

- ☑ The src directory contains three source files, named as follows.

  - ☑ send_and_play_audio.py
  - ☑ text_to_speech_application.py
  - ☑ text_to_speech_implementation.py

- ☑ The text_to_speech directory contains a README.md file with instructions on how to run the node.

- ☑ The text_to_speech directory contains no CMakeLists.txt build file.

- ☑ The text_to_speech directory contains no package.xml since it is a ROS node within the cssr_system package.

- ☑ The text_to_speech directory contains a text_to_speech_requirements.txt file with the dependencies (and their versions) required to be installed for proper functionality of the node.

12.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

- ☑ send_and_play_audio.py

```
"""
send_and_play_audio.py - functionality to send and play audio on the robot for the
transcribed kinyarwanda text

Author:     Muhirwa Richard
Date:       2025-04-18
Version:    v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

- ☑ text_to_speech_application.py

```
"""
text_to_speech_application.py - ROS node for integrated multilingual text-to-speech
functionality

Author:     Muhirwa Richard
Date:       2025-04-18
Version:    v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ `text_to_speech_implementation.py`

```
"""
text-to-speech_Implementation.py - implementation of the text-to-speech functionality

Author:     Muhirwa Richard
Date:       2025-04-18
Version:    v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The `text_to_speech_application.py` file contains a documentation comment with the following sections:

"""

☑
```
> text_to_speech_application.py - ROS node for integrated multilingual
  text-to-speech functionality

  This ROS application node provides text-to-speech (TTS) services
  supporting both Kinyarwanda and English languages. The node integrates
  with Pepper robot's audio system for English TTS and uses a custom
  TTS model for Kinyarwanda. It exposes a ROS service interface that
  accepts text messages and language specifications, then generates and
  plays the corresponding speech audio.
```

☑
```
> Libraries
      - rospy
      - os
      - sys
      - threading
      - time
      - text_to_speech_implementation
      - cssr_system.srv: (TTS, TTSResponse)
```

```
☑ > Parameters
       > Command line parameters:
           None

       > Configuration File Parameters
           - language: (english/kinyarwanda)
           - verboseMode: (True/False)
           - ip: Network IP address of the Pepper robot
           - port: Communication port for robot connection
           - useCuda: (True/False)

☑ > Subscribed Topics
       - None

☑
   > Published Topics and Message Types
       - /speech (std_msgs/String): English text messages for Pepper
               robot's built-in TTS

☑ > Services Invoked
       - None

☑ > Services Advertised and Request Message
       - /textToSpeech/say_text (cssr_system/TTS)
         Request: {string message, string language}
         Response: {bool success}

☑ > Input Data Files
       - None

☑ > Output Data Files
       - Temporary .wav files: Generated audio files for Kinyarwanda
               speech (auto-deleted)
       - ROS log files: Node operation logs in ~/.ros/log/

☑ > Configuration Files
       - text_to_speech_configuration.ini

☑ > Example Instantiation of the Module
       - rosrun cssr_system text_to_speech_application.py
       - roslaunch cssr_system text_to_speech_launch_robot.launch

☑ - Author:  Muhirwa Richard, Carnegie Mellon University Africa

☑ - Email:   muhirwarichard1@gmail.com

☑ - Date:    2025-04-18

☑ - Version: v1.0

"""
```

12.1.3 Component Unit Testing

⊟ A unit test application named `text_to_speech_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `text_to_speech_launch_simulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `text_to_speech_launch_test_harness.launch` is provided in the `launch` directory.

⊟ The `text_to_speech_launch_robot.launch` file launches the component being tested.

⊟ The `text_to_speech_launch_simulator.launch` file launches the component being tested.

⊟ The `text_to_speech_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
textToSpeech version v1.0

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
textToSpeech: start-up.
textToSpeech: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
textToSpeech: running.
```

⊟ The `text_to_speech_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `text_to_speech_launch_simulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `text_to_speech_launch_test_harness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `text_to_speech` directory.

  - ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
  - ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`text_to_speech_configuration.ini`) file are altered.

12.2 Integrated Text to Speech Conversion Unit Test

12.2.1 Files and Directories

- ☑ Files for a single component are stored in a subdirectory named **text_to_speech_test**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.
- ☑ The **text_to_speech_test** directory has three sub-directories: `data`, `launch`, and `src`.
- ⊟ The `config` directory contains one file, named as follows.
  - ⊟ `text_to_speech_test_configuration.ini`
  - ⊟ The configuration file `text_to_speech_test_configuration.ini` contains the key-value pairs that set the component parameters.
  - ⊟ Each key-value pair is written on a separate line.
- ☑ The `data` directory contains one output file.
  - ☑ `text_to_speech_test_output.dat`
- ☑ The `launch` directory contains two launch files, named as follows.
  - ☑ `text_to_speech_test_launch_robot.launch`
  - ☑ `text_to_speech_test_launch_test_harness.launch`
- ☑ The `src` directory contains two source files, named as follows.
  - ☑ `text_to_speech_test_application.py`
  - ☑ `text_to_speech_test_implementation.py`
- ☑ The `text_to_speech_test` directory contains a `README.md` file with instructions on how to run the test.
- ☑ The `text_to_speech_test` directory contains no `CMakeLists.txt` build file.
- ☑ The `text_to_speech_test` directory contains no `package.xml` since it is a ROS node within the `unit_test` package.
- ⊟ The `text_to_speech_test` directory contains a `text_to_speech_test_requirements.txt` file with the dependencies (and their versions) required to be installed for proper functionality of the node.

12.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `text_to_speech_test_application.py`

```
"""
text_to_speech_test_application.py - ROS-based integration test application for
    validating Text-to-Speech (TTS) service implementation

Author:     Muhirwa Richard
Date:       2025-04-18
Version:    v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network (AfretecInclusive
Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

☑ `text_to_speech_test_implementation.py`

```
"""
text_to_speech_test_implementation.py - text-to-speech test cases

Author:     Muhirwa Richard
Date:       2025-04-18
Version:    v1.0

Copyright (C) 2023 CSSR4Africa Consortium

This project is funded by the African Engineering and Technology Network (AfretecInclusive
Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
"""
```

The `text_to_speech_test_application.py` file contains a documentation comment with the following sections:

`"""`

☑
```
> text_to_speech_test_application.py - ROS-based integration test application
    for validating Text-to-Speech (TTS) service implementation.

This application serves as a comprehensive testing framework for validating
the functionality and integration of a Text-to-Speech (TTS) service within
a ROS environment.
The primary purpose is to ensure that the TTS implementation correctly
processes text input in both English and Kinyarwanda languages and
produces appropriate audio output on pepper robot.
```

```
☑ > Libraries
      - rospy: ROS Python client library for node initialization and service
              handling
      - sys: System-specific parameters and functions for exit codes
      - text_to_speech_test_implementation: Custom module containing:
          - TTSImplementationIntegrationTest
          - TestOutputLogger
          - print_success, print_warning, print_error, print_info, print_header:
                  Colored output functions

☑ > Parameters
      > Command-line Parameters
          - None (no command-line arguments required)

      > Configuration File Parameters
          - None (no configuration file used)

☑ > Subscribed Topics and Message Types
      - None (no topics subscribed)

☑ > Published Topics and Message Types
      - None (no topics published)

☑ > Services Invoked
      - /textToSpeech/say_text (cssr_system.srv.TTS): Text-to-speech service
              for converting text to audio output

☑ > Services Advertised and Request Message
      - None (no services advertised)

☑ > Input Data Files
      - None (no input data files required)

☑ > Output Data Files
      - text_to_speech_test_output.dat: Test results and logging output file
              stored in unit_tests/text_to_speech_test/data/

☑ > Configuration Files
      - None (no configuration files used)

☑ > Example Instantiation of the Module
      rosrun unit_tests text_to_speech_test_application.py

☑ - Author:  Muhirwa Richard, Carnegie Mellon University Africa

☑ - Email:   muhirwarichard1@gmail.com

☑ - Date:    2025-04-18

☑ - Version: v1.0

"""
```

12.2.3 Component Unit Testing

☑ A unit test application named `text_to_speech_test_launch_robot.launch` is provided in the `launch` directory.

⊟ A unit test application named `text_to_speech_test_launch_simulator.launch` is provided in the `launch` directory.

☑ A unit test application named `text_to_speech_test_launch_test_harness.launch` is provided in the `launch` directory.

☑ The `text_to_speech_test_launch_robot.launch` file launches the component being tested.

⊟ The `text_to_speech_test_launch_simulator.launch` file launches the component being tested.

☑ The `text_to_speech_test_launch_test_harness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
textToSpeechTest version v1.0

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
textToSpeechTest: start-up.
textToSpeechTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
textToSpeechTest: running.
```

☑ The `text_to_speech_test_launch_robot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `text_to_speech_test_launch_simulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

☑ The `text_to_speech_test_launch_test_harness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `text_to_speech_test` directory.

- ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
- ⊟ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`text_to_speech_test_configuration.ini`) file are altered.

13 D5.5.3 Environment Map Generation

13.1 Map Generation

13.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **mapGeneration**.

☑ The **mapGeneration** directory has four sub-directories: `config`, `data`, `include/mapGeneration`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `mapGenerationConfiguration.ini`
- ☑ The configuration file `mapGenerationConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains four files, named as follows.

- ☑ `configurationSpaceMap.png`
- ☑ `environmentMap.png`
- ☑ `mapGenerationInput.dat`
- ☑ `obstacles.dat`
- ⊟ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ☑ Each key-value pair is written on a separate line.

☑ The `include/mapGeneration` directory contains one file, named:

- ☑ `mapGenerationInterface.h`

☑ The `src` directory contains two source files, named as follows.

- ☑ `mapGenerationApplication.cpp`
- ☑ `mapGenerationImplementation.cpp`

☑ The `mapGeneration` directory contains a `README.md` file with instructions on how to run the node

☑ The `mapGeneration` directory contains a `CMakeLists.txt` build file.

☑ The `mapGeneration` directory contains no `package.xml` manifest file since it is a node within the `cssr_system` package .

13.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ mapGenerationApplication.cpp

```
/* mapGenerationApplication.cpp
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ mapGenerationImplementation.cpp

```
/* mapGenerationImplementation.cpp
*
* Author: Birhanu Shimelis Girma
* Date: April 08, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ `mapGenerationInterface.h`

```
/* mapGeneration.h
*
* Author: Birhanu Shimelis Girma
* Date: April 08, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

The `mapGenerationApplication.cpp` file contains a documentation comment:

☑
```
/* mapGenerationApplication.cpp
 *
  * This node is responsible for running the environment map generation node.
  * The node is responsible for creation of empty maps, maps with obstacles,
  * and configuration space generation with robot radius value.
 *
```

☑
```
 * Libraries
  * Standard libraries - std::string, std::vector, std::fstream, std::chrono,
  std::ctime
  * ROS libraries - ros/ros.h, ros/package.h
  * OpenCV libraries - opencv2/opencv.hpp, opencv2/highgui.hpp,
  opencv2/imgproc.hpp
  * Google Test - gtest/gtest.h
```

☑
```
 * Parameters
 *
 * Command-line Parameters
 *
 * None
 *
```

☑
```
 * Configuration File Parameters
  * Key                        |     Value
  * ----------------------     |     --------------------
  *   mode                     |  CAD
  *   verboseMode              | true
  *   resolution               | 0.05
  *   robotRadius              | 0.3
  *   inputFile                | mapGenerationInput.dat
  *
```

☑
```
 * Subscribed Topics and Message Types
  *
  * None
```

☑
```
* Published Topics and Message Types
 *
 * None
```

☑
```
* Services Invoked
 *
 * None
 *
```

☑
```
* Services Advertised and Message Types
 *
 * None
 *
```

☑
```
* Input Data Files
 *
 * mapGenerationInput.txt - Contains map dimensions and filenames
 * obstacles.txt - Contains obstacle definitions
```

☑
```
* Output Data Files
 *
 * environmentMap.png - Generated workspace map
 * configurationSpaceMap.png - Generated configuration space map
 *
```

☑
```
* Configuration Files
 *
 * mapGenerationConfiguration.ini
 *
```

☑
```
* Example Instantiation of the Module
 *
 * rosrun cssr_system mapGeneration
 *
```

☑ `* Author: Biruh Girmash, Carnegie Mellon University Africa`

☑ `* Email: bgirmash@andrew.cmu.edu`

☑ `* Date: June 05, 2025`

☑ `* Version: v1.0`

13.1.3 Component Unit Testing

⊟ A unit test application named `mapGenerationLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `mapGenerationLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `mapGenerationLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `mapGenerationLaunchRobot.launch` file launches the component being tested.

⊟ The mapGenerationLaunchSimulator.launch file launches the component being tested.

⊟ The mapGenerationLaunchTestHarness.launch file launches the component being tested.

☑ The mapGeneration being tested outputs the copyright message on startup:

```
mapGeneration: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

☑ The mapGeneration being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
mapGeneration: start-up.
```

☑ The mapGeneration being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
mapGeneration: running.
```

⊟ The mapGenerationLaunchRobot.launch file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the README.md file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The mapGenerationLaunchSimulator.launch file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The mapGenerationLaunchTestHarness.launch file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named README.md in the mapGenerationTest directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (.ini) file are altered.

13.2 Map Generation Unit test

13.2.1 Files and Directories

☑ Files are stored in a subdirectory named **mapGenerationTest**.

☑ The **mapGenerationTest** directory has five sub-directories: `config`, `data`, `include/mapGenerationTest`, `launch`, and `src`.

☑ The `config` directory contains one file, named.

- ☑ `mapGenerationTestConfiguration.ini`
- ☑ The configuration file `mapGenerationTestConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains four files and one sub-directory, named as follows.

- ☑ `/testOutput/`
- ☑ `environmentMap.png`
- ☑ `mapGenerationInput.dat`
- ☑ `testObstacles.dat`
- ☑ `testRobotRadius.dat`
- ⊟ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ☑ Each key-value pair is written on a separate line.

☑ The `include/mapGenerationTest` directory contains one file, named:

- ☑ `mapGenerationTestInterface.h`

☑ The `launch` directory contains one file, named as follows.

- ⊟ `mapGenerationLaunchRobot.launch`
- ⊟ `mapGenerationLaunchSimulator.launch`
- ☑ `mapGenerationLaunchTestHarness.launch`

☑ The `src` directory contains three source files, named as follows.

- ☑ `mapGenerationTestApplication.cpp`
- ☑ `mapGenerationTestDriver.cpp`
- ☑ `mapGenerationTestImplementation.cpp`

☑ The `mapGenerationTest` directory contains a `README.md` file with instructions on how to run the test.

☑ The `mapGenerationTest` directory contains a `CMakeLists.txt` build file.

☑ The `mapGenerationTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

13.2.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ `mapGenerationTestApplication.cpp`

```
/* mapGenerationTestApplication.cpp
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `mapGenerationTestImplementation.cpp`

```
/* mapGenerationTestImplementation.cpp
*
* Author: Birhanu Shimelis Girma
* Date: April 08, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ `mapGenerationTestInterface.h`

```
/*  mapGenerationTestInterface.h
*
* Author: Birhanu Shimelis Girma
* Date: April 08, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
```

```
* This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

✓ mapGenerationTestDriver.cpp

```
/*  mapGenerationTestDriver.cpp
*
* Author: Birhanu Shimelis Girma
* Date: April 08, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network
(Afretec) Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

The `mapGenerationTestApplication.cpp` file contains a documentation comment with the following subsections:

✓
```
/* mapGenerationTestApplication.cpp
  * This node is responsible for running tests on the environment map generation
  node.
  * The tests are run using Google Test and the results are written to a log file.
  * The node tests the creation of empty maps, maps with obstacles, and
  * configuration space generation with different robot radii values. It verifies
  * that the map generation system correctly processes obstacle data, generates
  * workspace maps, and createsconfiguration space maps that account for the
  * robot’s physical dimensions.
```

✓
```
 * Libraries
  * Standard libraries - std::string, std::vector, std::fstream, std::chrono,
  std::ctime
  * ROS libraries - ros/ros.h, ros/package.h
  * OpenCV libraries - opencv2/opencv.hpp, opencv2/highgui.hpp,
  opencv2/imgproc.hpp
  * Google Test - gtest/gtest.h
  *
```

✓
```
 * Parameters
  *
  * Command-line Parameters
  *
  * None
```

```
☑  * Configuration File Parameters
    * Key                         |     Value
    * ---------------------       |     --------------------
    *   mode                      |  CAD
    *   verboseMode               | true
    *   resolution                | 0.05
    *   robotRadius               | 0.3
    *   inputFile                 | mapGenerationInput.txt
    *

☑  * Subscribed Topics and Message Types
    *
    * None

☑  * Published Topics and Message Types
    *
    * None

☑  * Services Invoked
     *
   * None
   *

☑  * Services Advertised and Message Types

☑  * Input Data Files
    *
    * mapGenerationInput.dat - Contains map dimensions and filenames
    * testObstacles.dat - Contains obstacle definitions
    * testRobotRadius.dat - Contains robot radius values for configuration space
    * tests

☑  * Output Data Files
    *
    * testOutput.logs - Log file containing test results
    * (Output maps are generated by the actual mapGeneration node in its data
    directory)

☑  * Configuration Files
    *
    * mapGenerationTestConfiguration.ini

☑  * Example Instantiation of the node
    *
    * rosrun unit_tests mapGenerationTest
    *
    * ...
    *
    * The launch file for the map generation unit tests is
    mapGenerationLaunchTestHarness.launch.
    *
    * roslaunch unit_tests mapGenerationLaunchTestHarness.launch
    *

☑  * Author: Biruh Girmash, Carnegie Mellon University Africa
```

☑ `* Email: bgirmash@andrew.cmu.edu`

☑ `* Date: June 05, 2025`

☑ `* Version: v1.0`

13.2.3 Component Unit Testing

⊟ A unit test application named `mapGenerationLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `mapGenerationLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `mapGenerationLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `mapGenerationLaunchRobot.launch` file launches the component being tested.

⊟ The `mapGenerationLaunchSimulator.launch` file launches the component being tested.

☑ The `mapGenerationLaunchTestHarness.launch` file launches the component being tested.

☑ The `mapGenerationTest` being used for test outputs the copyright message on startup:

```
mapGenerationTest: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY
```

☑ The `mapGeneration` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
mapGenerationTest: start-up.
```

☑ The `mapGeneration` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
mapGenerationTest: running.
```

⊟ The `mapGenerationLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `mapGenerationLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

☑ The `mapGenerationLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `mapGenerationTest` directory.

- ☑ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

14 D5.5.4 Robot Navigation

14.1 Robot Navigation

14.1.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **robotNavigation**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **robotNavigation** directory has six sub-directories: `config`, `data`, `include/robotNavigation`, `msg`, `src`, and `srv`.

☑ The `config` directory contains one file, named as follows.

- ☑ `robotNavigationConfiguration.ini`
- ☑ The configuration file `robotNavigationConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains five input files and three output files.

- The input files are named as follows.
  - ☑ `pepperTopics.dat`
  - ☑ `parameters230.dat`
  - ☑ `parameters240.dat`
  - ☑ `scenarioOneConfigMap.dat`
  - ☑ `scenarioOneEnvironmentMap.dat`

    - ☑ The topics file pepperTopics.dat contains the key-value pairs that set the topic names required by the component.
    - ☑ Each key-value pair of the pepperTopics.dat file is written on a separate line.
  - The output files are named as follows.
    - ☑ robotPose.dat
    - ☑ configMapWaypoints<algorithm>.png where <algorithm>.png can be one of AStar, BFS, DFS, or Dijkstra.
    - ☑ environmentMapWaypoints<algorithm>.png where <algorithm>.png can be one of AStar, BFS, DFS, or Dijkstra.

☑ The include/robotNavigation directory contains one file, named as follows.

  - ☑ robotNavigationInterface.h

☑ The msg directory contains one files, named as follows.

  - ☑ Goal.msg

☑ The src directory contains two source files, named as follows.

  - ☑ robotNavigationApplication.cpp
  - ☑ robotNavigationImplementation.cpp

☑ The robotNavigation directory contains a README.md file with instructions on how to run the software.

☑ The robotNavigation directory contains a CMakeLists.txt build file.

☑ The robotNavigation directory contains no package.xml since it is a ROS node within the cssr_system package.

14.1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ robotNavigationApplication.cpp

```
/* robotNavigationApplication.cpp - robot navigation ROS Node definition
 *
 * Author:  Birhanu Shimelis Girma, Carnegie Mellon University Africa
 * Email:   bgirmash@andrew.cmu.edu
 * Date:    June 05, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ `robotNavigationApplicationImplementation.cpp`

```
/* robotNavigationImplementation.cpp - robot navigation functions implementation.
*
* Author:   Birhanu Shimelis Girma, Carnegie Mellon University Africa
* Email:    bgirmash@andrew.cmu.edu
* Date:     June 05, 2025
* Version:  v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ `robotNavigationApplicationInterface.h`

```
/* robotNavigationInterface.h - robot navigation interfacing and imported libraries
 *
 * Author:  Birhanu Shimelis Girma, Carnegie Mellon University Africa
 * Email:   bgirmash@andrew.cmu.edu
 * Date:    June 05, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `robotNavigationApplication.cpp` file contains a documentation comment:

☑
```
/* robotNavigationApplication.cpp - Application code to run the robot Navigation.

 * This node is responsible for navigating the robot to a goal location in
 * the environment map using a path planning algorithm. The node reads the
 * robot pose input and the goal location from the service request and
 * navigates the robot to the goal location. The node uses the path
 * planning algorithm to find the shortest path to the goal location.
 *
```

☑
```
* Libraries
 *       Standard libraries
 *           std::string, std::vector, std::thread, std::fstream, std::cout,
 *          std::endl, std::cin, std::pow, std::sqrt, std::abs
 *      ROS libraries
 *          ros/ros.h, ros/package.h,
 *          actionlib/client/simple_action_client.h,
 *          control_msgs/FollowJointTrajectoryAction.h,
 *          geometry_msgs/Twist.h
 *      OpenCV libraries
 *          opencv2/opencv.hpp
 *

☑  * Parameters
 *      Command-line Parameters
 *          None
 *      Configuration File Parameters
 *          Key                   |     Value
 *          --------------------- |     -------------------
 *          environmentMap        |     scenarioOneEnvironmentMap.dat
 *          configurationMap      |     scenarioOneConfigMap.dat
 *          pathPlanning          |     astar
 *          socialDistance        |     true
 *          robotTopics           |     pepperTopics.dat
 *          verboseMode           |     true
 *          robotType             |     old
 *

☑  * Subscribed Topics and Message Types
 *      /robotLocalization/pose     geometry_msgs/Pose2D
 *

☑  * Published Topics and Message Types
 *      /pepper_dcm/cmd_vel         geometry_msgs/Twist
 *      /joint_angles               naoqi_bridge_msgs/JointAnglesWithSpeed
 *

☑  * Services Invoked
 *      None
 *

☑  * Services Advertised and Message Types
 *      /robotNavigation/set_goal   cssr_system/setGoal
 *

☑  * Input Data Files
 *      pepperTopics.dat - Contains topic names for robot actuators
 *      scenarioOneEnvironmentMap.dat - Environment map data
 *      scenarioOneConfigMap.dat - Configuration space map data
 *      parameters230.dat - Locomotion parameters for old robot
 *      parameters240.dat - Locomotion parameters for new robot
 *

☑  * Output Data Files
 *      robotPose.dat - Updated robot position after navigation
 *      environmentMapWaypoints[Algorithm].png - Environment map with waypoints
 *      configMapWaypoints[Algorithm].png - Configuration map with waypoints
 *
```

```
☑  * Configuration Files
    *      robotNavigationConfiguration.ini
    *

☑  * Example Instantiation of the Module
    *      rosrun cssr_system robotNavigation
    *

☑  * Author:  Adedayo Akinade, Carnegie Mellon University Africa

☑  * Email:   aakinade@andrew.cmu.edu

☑  * Date:    January 7, 2025

☑  * Version: v1.0

☑  * Author:  Birhanu Shimelis Girma, Carnegie Mellon University Africa

☑  * Email:   bgirmash@andrew.cmu.edu

☑  * Date:    June 05, 2025

☑  * Version: v1.0
    */
```

14.1.3 Component Unit Testing

⊟ A unit test application named `robotNavagationLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `robotNavigationLaunchSimulator.launch` is provided in the `launch` directory.

⊟ A unit test application named `robotNavigationLaunchTestHarness.launch` is provided in the `launch` directory.

⊟ The `robotNavigationLaunchRobot.launch` file launches the component being tested.

⊟ The `robotNavigationLaunchSimulator.launch` file launches the component being tested.

⊟ The `robotNavigationLaunchTestHarness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
robotNavigation: v1.0

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
robotNavigation: start-up.
robotNavigation: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
robotNavigation: running.
```

⊟ The `robotNavigationLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `robotNavigationLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

⊟ The `robotNavigationLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `robotNavigation` directory.

- ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`robotNavigationConfiguration.ini`) file are altered.

14.2 Robot Navigation Unit Test

14.2.1 Files and Directories

☑ Files for a single component are stored in a subdirectory named **robotNavigationTest**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

☑ The **robotNavigationTest** directory has seven sub-directories: `config`, `data`, `include/robotNavigationTes` `launch`, `msg`, `src`, and `srv`.

☑ The `config` directory contains one file, named as follows.

- ☑ `robotNavigationTestConfiguration.ini`
- ☑ The configuration file `robotNavigationTestConfiguration.ini` contains the key-value pairs that set the component parameters.
- ☑ Each key-value pair is written on a separate line.

☑ The `data` directory contains six input files and one output file.

- The input files are named as follows.
  - ☑ `pepperTopics.dat`
  - ☑ `astarAlgorithmInput.dat`
  - ☑ `bfsAlgorithmInput.dat`
  - ☑ `dijkstraAlgorithmInput.dat`
  - ☑ `boundaryTestInput.dat`
  - ☑ `navigationServiceInput.dat`
  - ☑ The topics file `pepperTopics.dat` contains the key-value pairs that set the topic names required by the component.
  - ☑ Each key-value pair of the `pepperTopics.dat` file is written on a separate line.
- The output files are named as follows.
  - ☑ `robotNavigationTestOutput.dat`

☑ The `include/robotNavigationTest` directory contains one file, named as follows.

- ☑ `robotNavigationTestInterface.h`

☑ The `launch` directory contains two launch files, named as follows.

- ☑ `robotNavigationTestLaunchRobot.launch`
- ☑ `robotNavigationTestLaunchTestHarness.launch`

☑ The `msg` directory contains one files, named as follows.

- ☑ `Goal.msg`

☑ The `src` directory contains three source files, named as follows.

- ☑ `robotNavigationDriver.cpp`
- ☑ `robotNavigationTestImplementation.cpp`
- ☑ `robotNavigationTestImplementation.cpp`

☑ The `robotNavigationTest` directory contains a `README.md` file with instructions on how to run the test.

☑ The `robotNavigationTest` directory contains a `CMakeLists.txt` build file.

☑ The `robotNavigationTest` directory contains no `package.xml` since it is a ROS node within the `unit_test` package.

Files

14.2.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

☑ robotNavigationTestApplication.cpp

```
/* robotNavigationTestApplication.cpp - robot navigation unit test ROS Node definition
*
* Author:   Birhanu Shimelis Girma, Carnegie Mellon University Africa
* Email:    bgirmash@andrew.cmu.edu
* Date:     June 05, 2025
* Version:  v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ robotNavigationTestImplementation.cpp

```
/* robotNavigationTestImplementation.cpp - robot navigation functions that can be used
*        by the unit test ROS Node
*
* Author: Birhanu Shimelis Girma, Carnegie Mellon University Africa
* Email: bgirmash@andrew.cmu.edu
* Date: June 05, 2025
* Version: v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

☑ robotNavigationDriver.cpp

```
/* robotNavigationDriver.cpp - a driver software that simulates robot localization
*
* Author:   Birhanu Shimelis Girma, Carnegie Mellon University Africa
* Email:    bgirmash@andrew.cmu.edu
* Date:     June 05, 2025
* Version:  v1.0
*
* Copyright (C) 2023 CSSR4Africa Consortium
*
* This project is funded by the African Engineering and Technology Network (Afretec)
* Inclusive Digital Transformation Research Grant Programme.
*
* Website: www.cssr4africa.org
*
* This program comes with ABSOLUTELY NO WARRANTY.
*/
```

The `robotNavigationTestApplication.cpp` file contains a documentation comment:

```
☑ /* robotNavigationTestApplication.cpp - Application code to run the robot
   *       Navigation Unit test.
   *
   * This module is responsible for running the tests on the robot navigation
   *   module.
   * The tests are run using Google Test and the results are written to a file.
   * The module tests the path planning algorithms (BFS, Dijkstra, A*) via
   *    service calls, navigation service functionality, boundary conditions,
   *    and configuration management.

☑ * Libraries
   *       Standard libraries
   -           std::string, std::vector, std::fstream
   *       ROS libraries
   -           ros/ros.h, ros/package.h, cssr_system/setGoal.h

☑ * Parameters
   *       Command-line Parameters
   *           None
   *       Configuration File Parameters
   *           Key                    |     Value
   *           --------------------- |     -------------------
   *           pathPlanningBfs        |     true
   *           pathPlanningDijkstra   |     true
   *           pathPlanningAstar      |     true
   *           serviceTests           |     true
   *           boundaryTests          |     true
   *           configurationTests     |     true
   *           verboseMode            |     true

☑ * Subscribed Topics and Message Types
   *       None

☑ * Published Topics and Message Types
   *       None

☑ * Services Used
   *       /robotNavigation/set_goal            cssr_system/setGoal

☑ * Services Advertised and Message Types
   *       None

☑ * Input Data Files
   *       astarAlgorithmInput.dat
   *       bfsAlgorithmInput.dat
   *       boundaryTestInput.dat
   *       dijkstraAlgorithmInput.dat
   *       navigationServiceInput.dat
   *       pepperTopics.dat

☑ * Output Data Files
   *       robotNavigationTestOutput.dat
```

☑
```
* Configuration Files
 *       robotNavigationTestConfiguration.ini
```

☑
```
* Example Instantiation of the Module
 *       rosrun unit_tests robotNavigationTest
```

☑
```
 * Author:  Birhanu Shimelis Girma, Carnegie Mellon University Africa
```

☑
```
 * Email:   bgirmash@andrew.cmu.edu
```

☑
```
* Date:     January 10, 2025
```

☑
```
 * Version: v1.0
  */
```

14.2.3 Component Unit Testing

☑ A unit test application named `robotNavagationTestLaunchRobot.launch` is provided in the `launch` directory.

⊟ A unit test application named `robotNavigationTestLaunchSimulator.launch` is provided in the `launch` directory.

☑ A unit test application named `robotNavigationTestLaunchTestHarness.launch` is provided in the `launch` directory.

☑ The `robotNavigationTestLaunchRobot.launch` file launches the component being tested.

⊟ The `robotNavigationTestLaunchSimulator.launch` file launches the component being tested.

☑ The `robotNavigationTestLaunchTestHarness.launch` file launches the component being tested.

☑ The component being tested outputs the copyright message on startup:

```
robotNavigationTest: v1.0

This project is funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

This program comes with ABSOLUTELY NO WARRANTY.
```

☑ The component being tested write short messages to the terminal during the start-up phase to indicate the state of the node:

```
robotNavigationTest: start-up.
robotNavigationTest: subscribed to /topicName.
```

☑ The component being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
robotNavigationTest: running.
```

☑ The `robotNavigationTestLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

⊟ The `robotNavigationTestLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

☑ The `robotNavigationTestLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches there required drivers and stubs, and the node being tested.

☑ Unit test instructions are provided in a file named `README.md` in the `robotNavigationTest` directory.

- ☑ The instructions explain how the communication and computation functionality are validated by describing the (sink) output data that will be produced from the (source) input data.
- ☑ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`robotNavigationTestConfiguration.ini`) file are altered.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).
Adedayo Akinade, Carnegie Mellon University Africa.
Clifford Onyonka, Carnegie Mellon University Africa.
David Vernon, Carnegie Mellon University Africa.

Document History

Version 1.0

First draft.
David Vernon.
25 July 2024.

Version 1.1

Updated the integration results for gestureExecution (Success).
Updated the integration results for animateBehaviour (Success).
Updated the integration results for overtAttention (Success).
Adedayo Akinade.
27 January 2025.

Version 1.2

Updated the integration results for all accepted software – personDetection, faceDetection, soundDetection, speechEvent, behaviourController, and mapGeneration.
Adedayo Akinade.
30 June 2025.

Version 1.3

Updated the integration results for the following accepted software – actuatorTests, sensorTests, robotNavigation, robotLocalization, and textToSpeech.
Clifford Onyonka.
14 November 2025.