Test cases

This is the test cases if we want them in the appendix.

0.1 Test cases

Item	Description
Name	Command line interface (CLI) functionality
Test identifier	UNIT-01
Person responsible	Henrik Knutsen
Feature(s) to be tested	All possible commands when running the program with the CLI. That input with incompatible commands does not run.
Pre-conditions	Code for input handling for all possible commands. Error handling for invalid inputs.
Execution steps	1. Test all the commands one by one. 2. Test the combinations of invalid inputs.
Expected result	1. The program runs using the input variables. 2. Error warning: Invalid arguments. Abort startup.

Item	Description
Name	P3P parser
Test identifier	UNIT-02
Person responsible	Henrik Knutsen
Feature(s) to be tested	That the P3P parser correctly parses all the required fields and their values.
Pre-conditions	The P3P parser must be implemented. Need to have P3P policies with a wide range of cases.
Execution steps	1. Manually go through a P3P XML and obtain all the required fields and the values. 2. Run the same P3P XML in the P3P parser and print the parsed elements and their values to console. 3. Compare the results from the two parsing methods.
Expected result	The two parsing methods give identical output. They must both have the same fields, each containing the same values

Item	Description
Name	Local database
Test identifier	UNIT-03
Person responsible	Henrik Knutsen
Feature(s) to be tested	Writing to and reading from the local database. That the serialization of the database is working.
Pre-conditions	Code for writing to and reading from the database file must be implemented. Need to have two different P3P policies.
Execution steps	1. Write policy A to the local database. 2. Write policy B to the local database. 3. Read policy A from the local database. 4. Read policy B from the local database.
Expected result	The written policy A and the read policy A must be identical. The written policy B and the read policy B must be identical.

Item	Description
Name	Graphical user interface (GUI) functionality
Test identifier	UNIT-04
Person responsible	Henrik Knutsen
Feature(s) to be tested	That all the interactable elements, buttons, lists etc., is working as intended.
Pre-conditions	GUI with all the necessary listeners must be implemented. Code for running the program with the GUI must be implemented.
Execution steps	1. Run the program using the GUI. 2. Test all the interactable elements.
Expected result	All the interactable elements is triggering the right methods when used.

Item	Description
Name	Algorithm classification
Test identifier	UNIT-05
Person responsible	Henrik Knutsen
Feature(s) to be tested	That the k-nearest neighbor algorithm bases its decision on the k most similar policies
Pre-conditions	Code for reading from the weights file must be implemented. A working k-nearest neighbor algorithm that uses the weights must be implemented. Need one policy to test on, and a set of policies to be used as history.
Execution steps	1. Load a set of policies into the history. 2. Run the k-nn algorithm on the policy to be classified and the history. 3. Manually go through the policies and verify the output of the algorithm.
Expected result	The algorithm finds the most similar policy.

Item	Description
Name	Algorithm learning
Test identifier	UNIT-06
Person responsible	Henrik Knutsen
Feature(s) to be tested	That the weights file is updated when a new policy is added to history.
Pre-conditions	Code for reading from and writing to the weights file must be implemented. Algorithms for classification and learning must be implemented.
Execution steps	1. Get the contents of the weights file. 2. Load a set of policies into the history. 3. Run the classification algorithm on the single policy and the history. 4. Choose to store the new policy, the context and the action. 5. Get the contents of the weights file. 6. Compare the contents of the weights files obtained in steps 1. and 5.
Expected result	The two weights files obtained in steps 1. and 5. are different

Item	Description
Name	Packet passing through network to community database
Test identifier	UNIT-07
Person responsible	Henrik Knutsen
Feature(s) to be tested	That packets can be sent between the client program and the community database.
Pre-conditions	A running local client. A (virtual) server. Code for sending and receiving packets must be implemented.
Execution steps	1. Start the program locally. 2. Start the (virtual) server. 3. Send packet A from the local client. 4. Receive packet A at the (virtual) server. 5. Send packet B from the (virtual) server. 6. Receive packet B at the local client.
Expected result	The received packet A is identical to the sent packet A. The received packet B is identical to the sent packet B.