Testing Document

Fingerprint Biometric Research Tool

Contents

Testing Functional Requirements

Converting fingerprint template to image

Test Case 1: Converting valid template to image and ensuring a valid output that matches the minutiae in the template file.

Test Case 2: Converting invalid template to image and ensuring that there is no generated output.

Test Case 3: Converting multiple templates (directory) to images and ensuring that there is a generated fingerprint images directory that matches the corresponding templates directory data.

Converting fingerprint template to image

Test Case 1: Converting valid image (any format) to template and ensuring that the fingerprint template matches the image.

Test Case 2: Converting multiple images (directory) to templates and ensuring that there is a generated fingerprint templates directory that matches the corresponding images directory data.

Test Case 3: Converting invalid image to template and ensuring that there is no generated output.

Converting fingerprint image to printing object

Test Case 1: Converting valid image (any format) to printing object and ensuring that the fingerprint printing object surface matches the image.

Test Case 2: Converting multiple images (directory) to printing objects and ensuring that there is a generated fingerprint printing objects directory that matches the corresponding images directory data.

Test Case 3: Converting invalid image to template and ensuring that there is no generated output.

Fingerprint Matching

Test Case 1: Matching one template with one template and ensure that there is a score that is the expected score of those compared templates.

Test Case 2: Matching one template with many templates (directory) and ensure there is matching score matrix that is the expected matrix score for the templates.

Test Case 3: Matching many templates (directory) with many templates (directory) and ensure there is matching score matrix that is the expected matrix score of the templates.

Test Case 4: Matching invalid templates of any kind of matching, ensuring that there is no score.

Test Case 5: Export templates matching score as csv file and ensure its exported to the right location.

Experiments Management

Test Case 1: Viewing all experiments and making sure they appear all.

Test Case 2: Rename experiment and ensure that the experiment renamed to the new name.

Test Case 3: Set an experiment to be the current experiment and ensure it is changed.

Test Case 4: Set non-existing experiment to be the current experiment and ensure that the current experiment is not changed.

Test Case 5: Rename an experiment with invalid or existing name and ensure its not renamed.

Test Case 6: Delete an experiment and ensure that is no longer in the experiments list.

Test Case 7: Delete Operation inside an experiment and ensure its removed from the operations list under the experiment.

Test Case 8: Perform any conversion operation and ensure that a new operation is added under the current experiment.

Test Case 9: Perform any matching operation and ensure that a new operation is added under the current experiment.

Minutiae Map Visualization

Test Case 1: Import a template file (.min) format and ensure the visualization works and the minutiae appear on the map image.

Test Case 8: : Import an invalid template file (.min) format and ensure there is no generated image.

Testing Non-Functional Requirements

We did test that the exported assets from the system are exported in the desired location, we also manually test the visualization of the assets such as 3d object visualization and displaying the images.

Test Driven Development

The project timeline was so tight (this was discussed with our advisor) therefore we did not follow the TDD strategy. We did have a challenging process to provide convincing proof of concept so we were busy implementing the parts that needs to be integrated for the PoC.

Random & Automatically Generated Tests

N\A

Testing the user interface

We tested the GUI manually to ensure that page navigation works and the continuous frames destruction and reconstruction work, also we did other tests to ensure that the GUI does not freeze while the backend processing. This way we ensure user interface stability.

We meet with our customer continuously to ensure UI satisfaction, the customer received early versions of the application to test himself. This way we can receive feedback in order to change or fix or develop new UI features

We also tested features such as drag and drop on the Windows operating system to ensure smooth operation.

Testing build, integration & deployment

We wrote integration tests to that verify the work of the different integrated systems in our software tool. We tested pipelines from one system to another, for example the integrated machine learning model and the NBIS tool.

We have a requirements file in our GitHub repository to ensure all required libraries that our system needs are installed. Creating a virtual environment helps avoiding libraries and dependencies conflicts.

The customer has been provided a batch script to run the software with one click.

The filesystem of our software has been tested and the data is saved successfully in the customer's local machine.