Fingerprint Biometric Research Tool
Application Design Document

Luai Okasha, Yazan Abbas

2023 - 2024

Contents

1 Use Cases

1.1 Template to 2D Fingerprint Image Conversion:

- **Primary actor:** Researcher.
- **Description:** The Researcher imports a template (or directory) into the system and converts it to a 2D fingerprint image (or images directory).
- **Stakeholders and interests:**
  - The researcher wants to see the imported template and the converted 2D fingerprint image visualization in between different stages (after template importing and before image exporting).
  - The researcher may want to export the converted image/s to his local storage.
- **Pre-conditions:**
  - Imported template/s must be valid.
- **Post-conditions:**
  - The system displays the converted 2D fingerprint image and is ready for export.
- **Main success scenario:**
  1. The researcher imports a template (or templates directory).
  2. The system displays the template.
  3. The system displays the converted image.
  4. The researcher exports the converted image (or images directory).
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher with relevant info.
  - Importing invalid template: a message will be displayed to the researcher.

1.2 2D Fingerprint Image to 3D Printing Object Conversion:

- **Primary actor:** Researcher.
- **Description:** The Researcher imports a 2D fingerprint image (or images directory) into the system, then the system converts it to a 3D printing object (or 3D printing objects directory).
- **Stakeholders and interests:**
  - The researcher wants to see the enhanced fingerprint image and the converted 3D printing object visualization in between different stages (after image importing and before 3D object exporting).
  - The researcher may want to export the converted 3D printing object (or 3D printing objects directory) to his local storage.
- **Pre-conditions:**
  - Must import a fingerprint image (or images directory).
- **Post-conditions:**
  - The system displays the converted 3D printing object and is ready for export.
- **Main success scenario:**
  1. The researcher imports a fingerprint image (or images directory).
  2. The system displays the fingerprint image.
  3. The system displays the converted 3D printing object.
  4. The researcher exports the converted 3D printing object (or printing objects directory).
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.
  - Importing improper image: a message will be displayed to the researcher.
  - Some of the images in the directory are bad or low quality: the system will convert the good images and display to the researcher how many are converted successfully out of the total images.

1.3 2D Fingerprint Image to Template Conversion:

- **Primary actor:** Researcher.
- **Description:** The Researcher imports a 2D fingerprint image (or images directory), and the system converts it to the template (or templates directory).
- **Stakeholders and interests:**
  - The researcher wants to see the fingerprint template image and template visualization in between stages.
  - The researcher may want to export the converted template (or templates directory) to his local storage.
- **Pre-conditions:**
  - Must import a fingerprint image (or images directory).
- **Post-conditions:**
  - The system displays the converted template and is ready for export.
- **Main success scenario:**
  1. The researcher imports a fingerprint image (or images directory).
  2. The system displays the fingerprint template image.
  3. The researcher exports the converted template (or templates directory).
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.
  - Importing improper image: a message will be displayed to the researcher.

1.4 Templates Matching:

- **Primary actor:** Researcher.
- **Description:** The researcher imports 2 template sets (i.e. single template file or directory) and compares the templates with one another, the system displays statistics for the comparison.
- **Stakeholders and interests:**
  - The researcher wants to see the statistics for comparing templates with others and to be able to export the matching score *.csv* file to his local storage.
- **Pre-conditions:**
  - Imported templates are valid.
- **Post-conditions:**
  - The system displays corresponding statistics.
- **Main success scenario:**
  1. The researcher imports template sets.
  2. The system displays comparison statistics.
  3. The system allows the researcher to export *.csv* scores file.
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.
  - Importing improper template/s: a message will be displayed to the researcher.

1.5 Delete Experiment:

- **Primary actor:** Researcher.
- **Description:** The researcher deletes a specific experiment.
- **Pre-conditions:**
  - Existing experiment.
- **Post-conditions:**
  - The experiment and all of its operations are deleted.
- **Main success scenario:**
  1. The researcher deletes an experiment.
  2. The deletes the experiments with all of its operations after the research confirmation.
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.

1.6 Rename Experiment:

- **Primary actor:** Researcher.
- **Description:** The researcher renames a specific experiment.
- **Pre-conditions:**
  - Existing experiment.
  - Valid new experiment name and is not already used.
- **Post-conditions:**
  - The experiment will be renamed to the new name.
- **Main success scenario:**
  1. The researcher renames an experiment.
  2. The experiment will be renamed while notifying the researcher.
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.
  - Renaming to invalid or existing name: a message will be displayed to the researcher and the experiment will not be renamed.

1.7 Delete Operation:

- **Primary actor:** Researcher.
- **Description:** The researcher deletes an operation from a specific experiment.
- **Pre-conditions:**
  - Existing operation inside an experiment.
- **Post-conditions:**
  - The operation will be deleted.
- **Main success scenario:**
  1. The researcher deletes an operation.
  2. The operation will be deleted.
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.

1.8 Viewing All Experiments and Operations:

- **Primary actor:** Researcher.
- **Description:** The researcher may want to see all the experiments and operations that he performed and to keep track of them (history).
- **Pre-conditions:**
  - Existing experiment.
- **Post-conditions:**
  - The system views all experiments and all of its operations.
- **Main success scenario:**
  1. The researcher requests to see all experiments.
  2. The system displays all experiments with all operations packed each one to its related experiment.
- **Alternative flows:**
  - On system failure: a message will be displayed to the researcher.

2 System Architecture

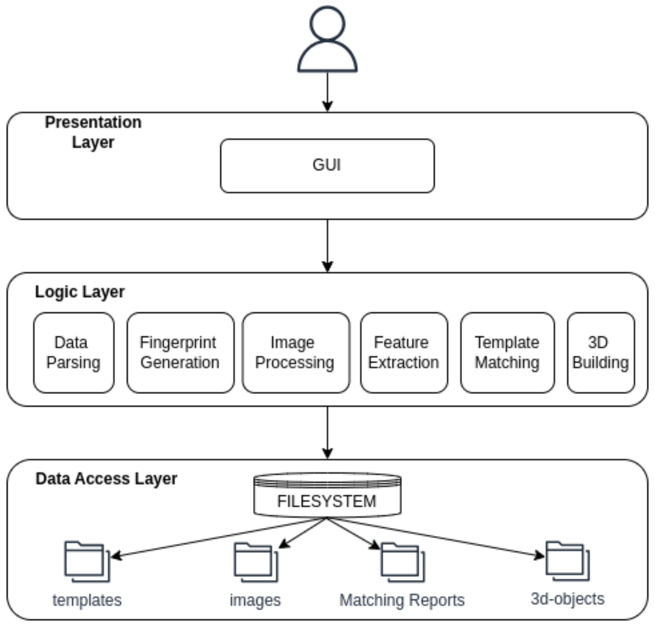


Figure 1: System Architecture Diagram

- **Presentation Layer:** The GUI component is responsible for rendering and visualizing the main system components such as templates, images, and 3d printing objects.
- **Logic Layer:** The logic layer contains the logic of the following main components of the system:
  - **Data Parsing:** This component is responsible for parsing files into objects and is mainly used for parsing text template files.
  - **Fingerprint Generation:** This integrated machine learning model is responsible for generating the fingerprint image from a given template.
  - **Image Processing:** The system is required to perform image processing-related operations such as pre-processing, image enhancement, etc. This kind of operation is performed by using *opencv-python* library and *NBIS* tool.
  - **Features Extraction:** The system is adopting an external library *NBIS* that is responsible for extracting minutiae from the fingerprint images.

  - **Templates Matching:** The template Matcher is responsible for receiving fingerprint templates and trying to match and compare them either with stored templates or given templates and returning a report that points out to the similarity rate, identical/similar templates with comprehension related details, etc.
  - **3D Building:** The system is adopting an external library *csdt_stl_converter* for creating 3D printing objects from a given image, this model can be used also for editing such objects.

- **Data Access Layer:** The system maintains a local file system *FILESYSTEM* that stores derived data from the locally saved assets (Templates, Images, 3d Objects, Matching reports) and more relevant data to the user activity in the system alongside mapping paths of assets that are stored locally on the disk.

3 Data Model

3.1 Description of Data Objects

The experiment contains at least one operation, and operation can be only in one experiment and it should have exactly 2 assets, one as an input and the other as an output.

1. **Experiment:**
   - **experiment_name:** experiment name.
   - **experiment_date:** represents the experiment creation date.
   - **experiment_last_update_date:** represents the date of the last edit such as renaming the experiment.
   - **operations:** represents the operations of the experiment.
2. **Operation:**
   - **operation_id:** operation id
   - **operation_datetime:** represents the operation date and time.
   - **operation_type:** represents the type of operation, which can be converting template to image, image to template, image to 3D object, or matching report.
   - **operation_output:** represents the output of the operation.
   - **operation_input:** represents the input of the operation.
   - **operation_optional_extra_input:** represents the second input for matching operations.
3. **Asset:**
   - **path:** asset path.
   - **is_dir:** indicates if the given path is a directory path.

3.2 Data Objects Relationships

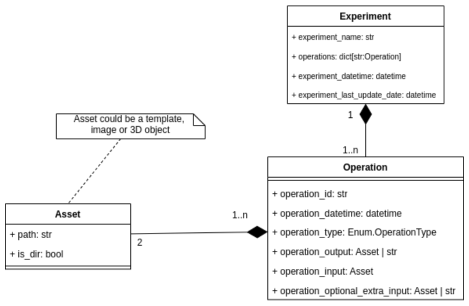

Experiment
+ experiment_name: str
+ operations: dict[str:Operation]
+ experiment_datetime: datetime
+ experiment_last_update_date: datetime
1
1..n
Asset could be a template, image or 3D object
Operation
+ operation_id: str
+ operation_datetime: datetime
+ operation_type: Enum.OperationType
+ operation_output: Asset | str
+ operation_input: Asset
+ operation_optional_extra_input: Asset | str
Asset
+ path: str
+ is_dir: bool
1..n
2

Figure 2: Data Objects Relationships

3.3 FILESYSTEM

The system maintains a *FILESYSTEM* that saves logic data and paths of system assets such as templates, images, 3d objects, and matching reports that navigate to the actual files on the local storage. All data paths and details are saved in a *metadata.json* file under the corresponding directory.

4 Behavioral Analysis

4.1 Sequence Diagrams

4.1.1 Template to 2D Fingerprint Image Conversion

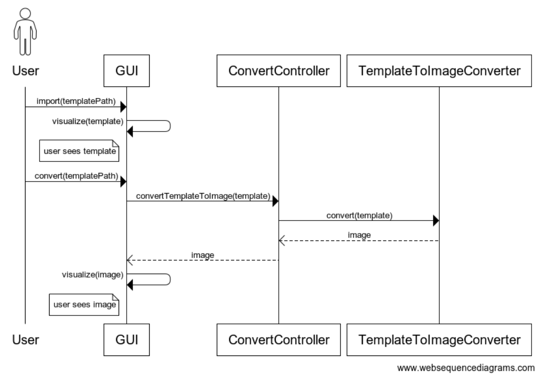


Figure 3: Template to 2D Fingerprint Image Conversion Sequence Diagram

4.1.2 Fingerprint Image to 3D Printing Object Conversion

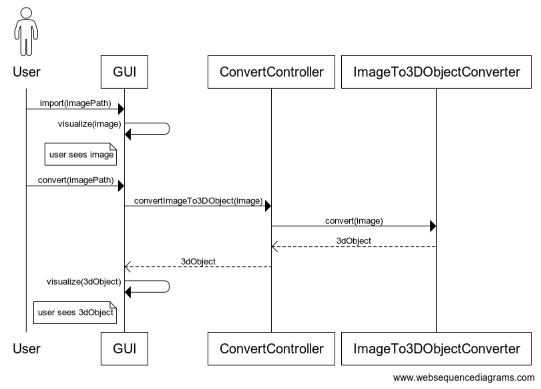


Figure 4: Fingerprint Image to 3D Printing Object Conversion Sequence Diagram

4.1.3 2D Fingerprint Image to Template Conversion

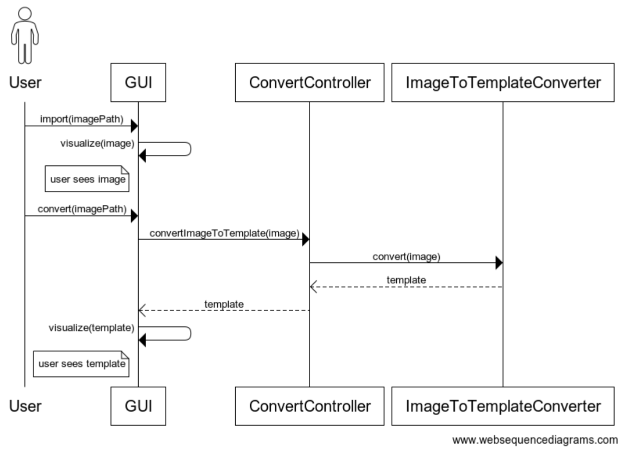


Figure 5: 2D Fingerprint Image to Template Conversion Sequence Diagram

4.1.4 Templates Matching

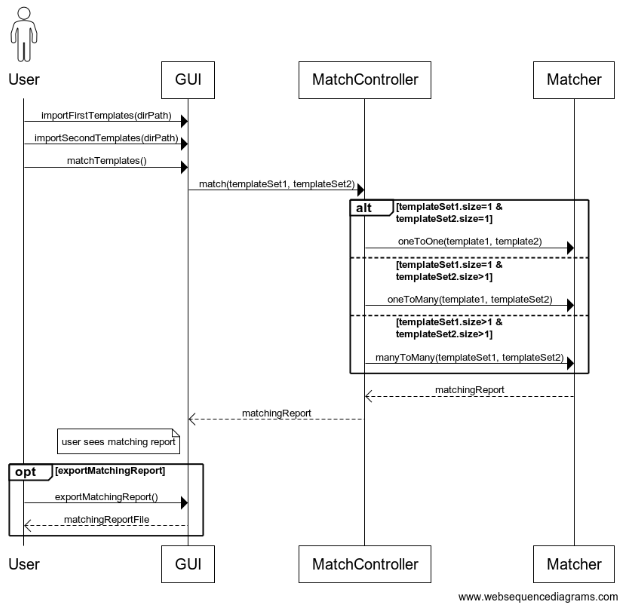

User
GUI
MatchController
Matcher
importFirstTemplates(dirPath)
importSecondTemplates(dirPath)
matchTemplates()
match(templateSet1, templateSet2)
alt
[templateSet1.size=1 & templateSet2.size=1]
oneToOne(template1, template2)
[templateSet1.size=1 & templateSet2.size>1]
oneToMany(template1, templateSet2)
[templateSet1.size>1 & templateSet2.size>1]
manyToMany(templateSet1, templateSet2)
matchingReport
matchingReport
user sees matching report
opt
[exportMatchingReport]
exportMatchingReport()
matchingReportFile
www.websequencediagrams.com

Figure 6: Templates Matching Sequence Diagram

4.1.5 Delete Experiment

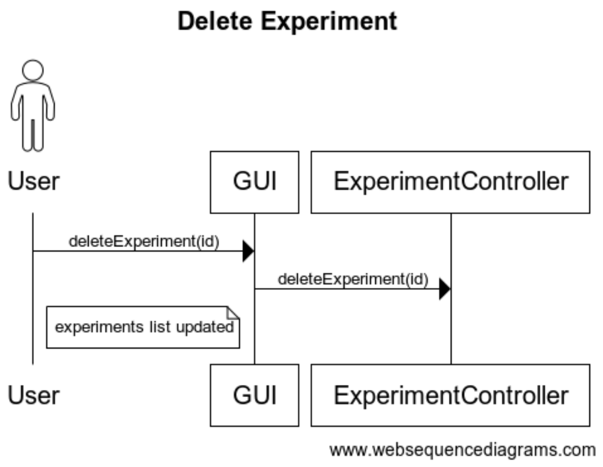


Figure 7: Delete Experiment Sequence Diagram

4.1.6 Rename Experiment

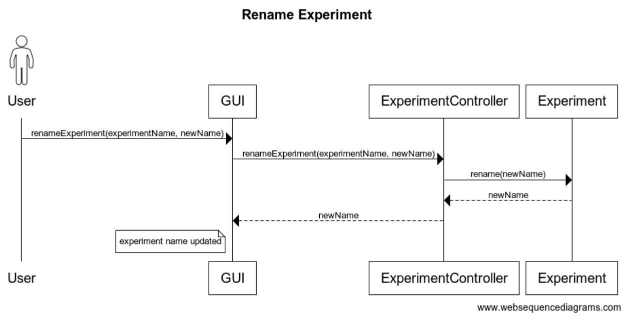


Figure 8: Rename Experiment Sequence Diagram

4.1.7 Delete Operation

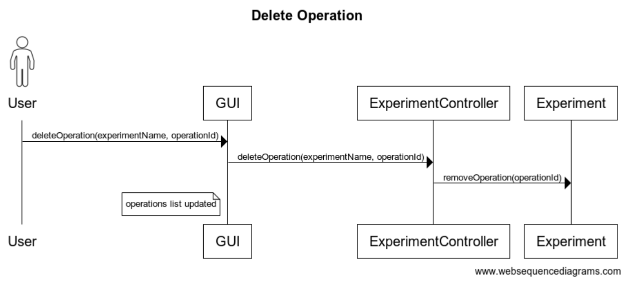

Delete Operation
User
GUI
ExperimentController
Experiment
deleteOperation(experimentName, operationId)
deleteOperation(experimentName, operationId)
removeOperation(operationId)
operations list updated
User
GUI
ExperimentController
Experiment
www.websequencediagrams.com

Figure 9: Delete Operation Sequence Diagram

4.1.8 View All Experiments and Operations

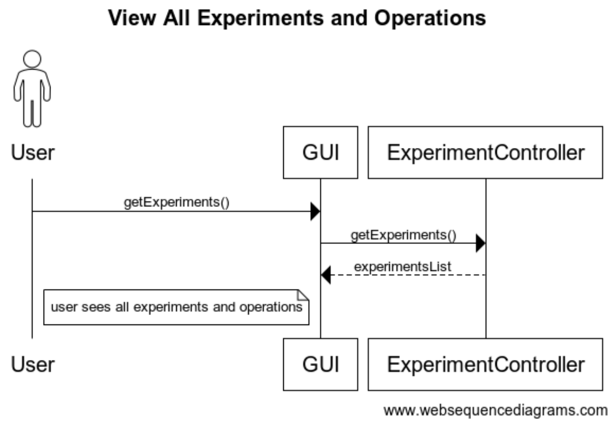

View All Experiments and Operations
User
GUI
ExperimentController
getExperiments()
getExperiments()
experimentsList
user sees all experiments and operations
User
GUI
ExperimentController
www.websequencediagrams.com

Figure 10: View All Experiments and Operations Sequence Diagram

5 Object-Oriented Analysis

5.1 Class Diagrams

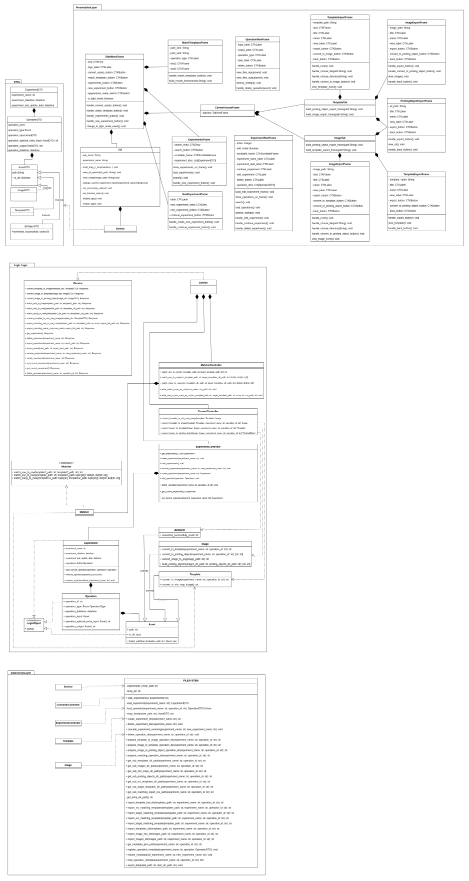
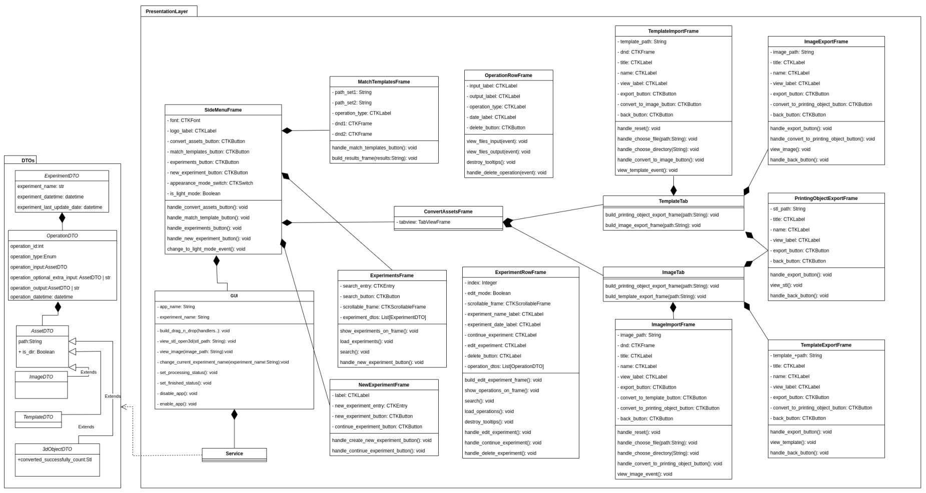


Figure 11: Class Diagram

5.2 Class Description

DTO classes are data transfer objects classes that are designed to be shared objects between the *Presentation Layer* and *Logic Layer* that are reflecting the logic layer's objects with remaining consistent with 3-tier architecture.



DTOs
ExperimentDTO
experiment_name: str
experiment_datetime: datetime
experiment_last_update_date: datetime
OperationDTO
operation_id:int
operation_type:Enum
operation_input:AssetDTO
operation_optional_extra_input: AssetDTO | str
operation_output:AssetDTO | str
operation_datetime: datetime
AssetDTO
path:String
+ is_dir: Boolean
ImageDTO
TemplateDTO
3dObjectDTO
+converted_successfully_count:Stl
Extends
Extends
Extends
PresentationLayer
SideMenuFrame
- font: CTKFont
- logo_label: CTKLabel
- convert_assets_button: CTKButton
- match_templates_button: CTKButton
- experiments_button: CTKButton
- new_experiment_button: CTKButton
- appearance_mode_switch: CTKSwitch
- is_light_mode: Boolean
handle_convert_assets_button(): void
handle_match_template_button(): void
handle_experiments_button(): void
handle_new_experiment_button(): void
change_to_light_mode_event(): void
GUI
- app_name: String
- experiment_name: String
- build_drag_n_drop(handlers..): void
- view_stl_open3d(stl_path: String): void
- view_image(image_path: String):void
- change_current_experiment_name(experiment_name:String):void
- set_processing_status(): void
- set_finished_status(): void
- disable_app(): void
- enable_app(): void
Service
MatchTemplatesFrame
- path_set1: String
- path_set2: String
- operation_type: CTKLabel
- dnd1: CTKFrame
- dnd2: CTKFrame
handle_match_templates_button(): void
build_results_frame(results:String): void
OperationRowFrame
- input_label: CTKLabel
- output_label: CTKLabel
- operation_type: CTKLabel
- date_label: CTKLabel
- delete_button: CTKButton
view_files_input(event): void
view_files_output(event): void
destroy_tooltips(): void
handle_delete_operation(event): void
ConvertAssetsFrame
- tabview: TabViewFrame
ExperimentsFrame
- search_entry: CTKEntry
- search_button: CTKButton
- scrollable_frame: CTKScrollableFrame
- experiment_dtos: List[ExperimentDTO]
show_experiments_on_frame(): void
load_experiments(): void
search(): void
handle_new_experiment_button(): void
NewExperimentFrame
- label: CTKLabel
- new_experiment_entry: CTKEntry
- new_experiment_button: CTKButton
- continue_experiment_button: CTKButton
handle_create_new_experiment_button(): void
handle_continue_experiment_button(): void
ExperimentRowFrame
- index: Integer
- edit_mode: Boolean
- scrollable_frame: CTKScrollableFrame
- experiment_name_label: CTKLabel
- experiment_date_label: CTKLabel
- continue_experiment: CTKLabel
- edit_experiment: CTKLabel
- delete_button: CTKLabel
- operation_dtos: List[OperationDTO]
build_edit_experiment_frame(): void
show_operations_on_frame(): void
search(): void
load_operations(): void
destroy_tooltips(): void
handle_edit_experiment(): void
handle_continue_experiment(): void
handle_delete_experiment(): void
TemplateImportFrame
- template_path: String
- dnd: CTKFrame
- title: CTKLabel
- name: CTKLabel
- view_label: CTKLabel
- export_button: CTKButton
- convert_to_image_button: CTKButton
- back_button: CTKButton
handle_reset(): void
handle_choose_file(path:String): void
handle_choose_directory(String): void
handle_convert_to_image_button(): void
view_template_event(): void
TemplateTab
build_printing_object_export_frame(path:String): void
build_image_export_frame(path:String): void
ImageTab
build_printing_object_export_frame(path:String): void
build_template_export_frame(path:String): void
ImageImportFrame
- image_path: String
- dnd: CTKFrame
- title: CTKLabel
- name: CTKLabel
- view_label: CTKLabel
- export_button: CTKButton
- convert_to_template_button: CTKButton
- convert_to_printing_object_button: CTKButton
- back_button: CTKButton
handle_reset(): void
handle_choose_file(path:String): void
handle_choose_directory(String): void
handle_convert_to_printing_object_button(): void
view_image_event(): void
ImageExportFrame
- image_path: String
- title: CTKLabel
- name: CTKLabel
- view_label: CTKLabel
- export_button: CTKButton
- convert_to_printing_object_button: CTKButton
- back_button: CTKButton
handle_export_button(): void
handle_convert_to_printing_object_button(): void
view_image(): void
handle_back_button(): void
PrintingObjectExportFrame
- stl_path: String
- title: CTKLabel
- name: CTKLabel
- view_label: CTKLabel
- export_button: CTKButton
- back_button: CTKButton
handle_export_button(): void
view_stl(): void
handle_back_button(): void
TemplateExportFrame
- template_+path: String
- title: CTKLabel
- name: CTKLabel
- view_label: CTKLabel
- export_button: CTKButton
- convert_to_printing_object_button: CTKButton
- back_button: CTKButton
handle_export_button(): void
view_template(): void
handle_back_button(): void

Figure 12: Presentation Layer Diagram

Presentation layer main classes: GUI: This class is the main class for the UI application which contains all the UI windows and manages the navigation between them, this class has the app service as an instance.

- **SideMenuFrame:** This class displays the pages of the software and allows it to navigate from one another, also to allow to change the theme of the UI.
- **ConvertAssetFrame:** This class contains 2 main classes: the TemplateTab and the ImageTab.
- **TemplateTab:** This class contains 3 main classes: the TemplateImportFrame, ImageExportFrame, and PrintingObjectExportFrame.
- **ImageTab:** This class contains 3 main classes: the ImageImportFrame, TemplateExportFrame, and PrintingObjectExportFrame.
- **TemplateImportFrame:** This class allows the user to import a template from his local storage to the system and displays the template as a minutiae map.
- **TemplateExportFrame:** This class allows you to Export the generated template and visualize it as a minutiae map.

- **ImageImportFrame:** This class allows the user to import an image from his local storage to the system and displays the imported image.
- **ImageExportFrame:** This class allows the user to view and export the generated image to his local storage.
- **PrintingObjectFrame:** This class allows the user to view the 3D printing object and export it to his local storage.
- **MatchingTemplateFrame:** This class allows the user to import 2 template sets (single templates or directory of templates) to match between them and to receive a score and allows him to export the matching scores as *.csv* file to his local storage.
- **ExperimentsFrame:** This class is responsible for displaying all the experiments that the user created and all the operations that the user performed in a specific experiment.
- **ExperimentRowFrame:** This class is responsible for viewing experiment fields and attributes in the experiments list.
- **OperationRowFrame:** This class is responsible for viewing operation fields and attributes in the operations list.
- **NewExperimentFrame:** This class allows the user to create a new experiment by entering the experiment's name.

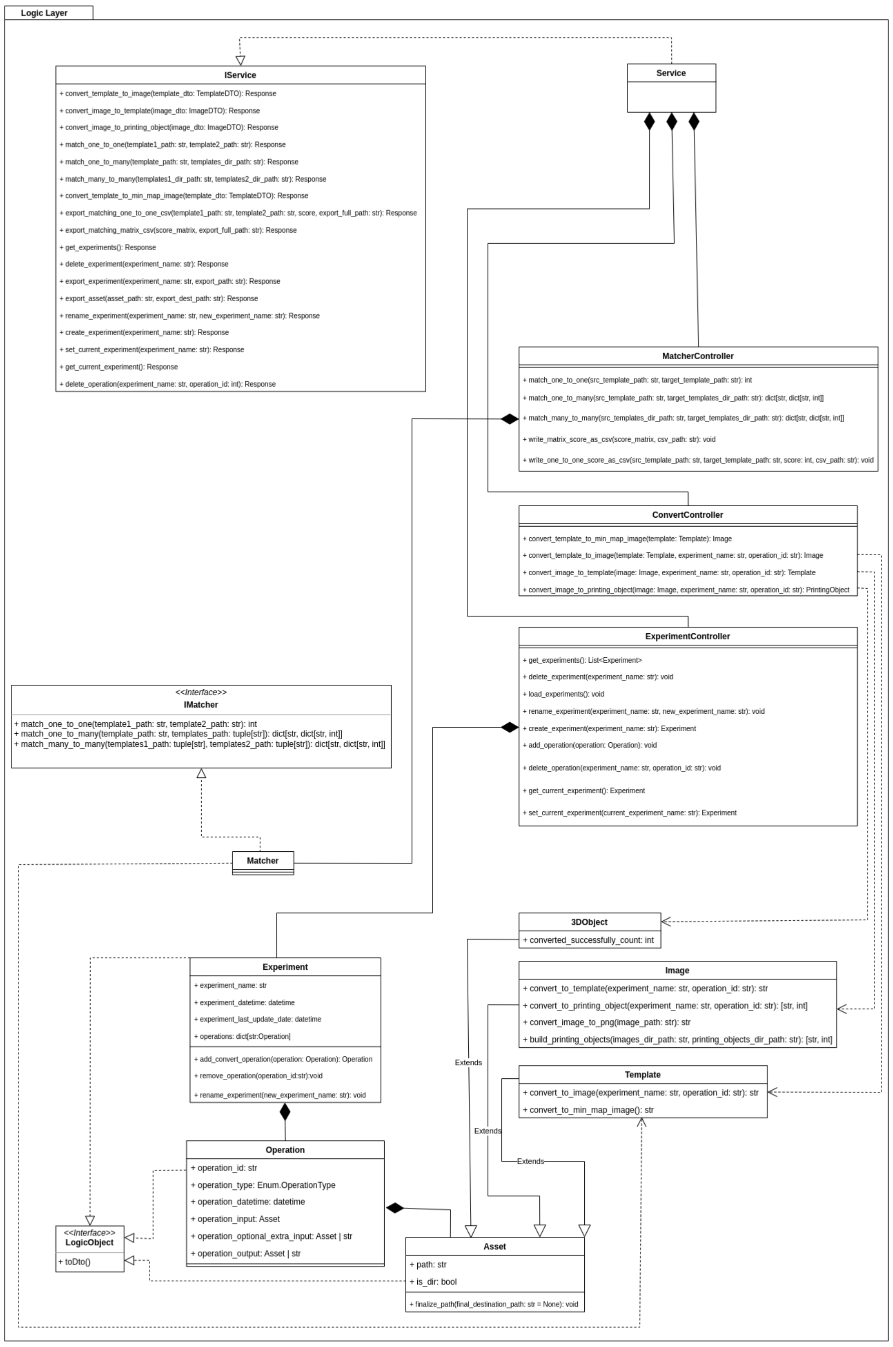

Logic Layer
IService
+ convert_template_to_image(template_dto: TemplateDTO): Response
+ convert_image_to_template(image_dto: ImageDTO): Response
+ convert_image_to_printing_object(image_dto: ImageDTO): Response
+ match_one_to_one(template1_path: str, template2_path: str): Response
+ match_one_to_many(template_path: str, templates_dir_path: str): Response
+ match_many_to_many(templates1_dir_path: str, templates2_dir_path: str): Response
+ convert_template_to_min_map_image(template_dto: TemplateDTO): Response
+ export_matching_one_to_one_csv(template1_path: str, template2_path: str, score, export_full_path: str): Response
+ export_matching_matrix_csv(score_matrix, export_full_path: str): Response
+ get_experiments(): Response
+ delete_experiment(experiment_name: str): Response
+ export_experiment(experiment_name: str, export_path: str): Response
+ export_asset(asset_path: str, export_dest_path: str): Response
+ rename_experiment(experiment_name: str, new_experiment_name: str): Response
+ create_experiment(experiment_name: str): Response
+ set_current_experiment(experiment_name: str): Response
+ get_current_experiment(): Response
+ delete_operation(experiment_name: str, operation_id: int): Response
Service
MatcherController
+ match_one_to_one(src_template_path: str, target_template_path: str): int
+ match_one_to_many(src_template_path: str, target_templates_dir_path: str): dict[str, dict[str, int]]
+ match_many_to_many(src_templates_dir_path: str, target_templates_dir_path: str): dict[str, dict[str, int]]
+ write_matrix_score_as_csv(score_matrix, csv_path: str): void
+ write_one_to_one_score_as_csv(src_template_path: str, target_template_path: str, score: int, csv_path: str): void
ConvertController
+ convert_template_to_min_map_image(template: Template): Image
+ convert_template_to_image(template: Template, experiment_name: str, operation_id: str): Image
+ convert_image_to_template(image: Image, experiment_name: str, operation_id: str): Template
+ convert_image_to_printing_object(image: Image, experiment_name: str, operation_id: str): PrintingObject
ExperimentController
+ get_experiments(): List<Experiment>
+ delete_experiment(experiment_name: str): void
+ load_experiments(): void
+ rename_experiment(experiment_name: str, new_experiment_name: str): void
+ create_experiment(experiment_name: str): Experiment
+ add_operation(operation: Operation): void
+ delete_operation(experiment_name: str, operation_id: str): void
+ get_current_experiment(): Experiment
+ set_current_experiment(current_experiment_name: str): Experiment
<<Interface>>
IMatcher
+ match_one_to_one(template1_path: str, template2_path: str): int
+ match_one_to_many(template_path: str, templates_path: tuple[str]): dict[str, dict[str, int]]
+ match_many_to_many(templates1_path: tuple[str], templates2_path: tuple[str]): dict[str, dict[str, int]]
Matcher
3DObject
+ converted_successfully_count: int
Image
+ convert_to_template(experiment_name: str, operation_id: str): str
+ convert_to_printing_object(experiment_name: str, operation_id: str): [str, int]
+ convert_image_to_png(image_path: str): str
+ build_printing_objects(images_dir_path: str, printing_objects_dir_path: str): [str, int]
Template
+ convert_to_image(experiment_name: str, operation_id: str): str
+ convert_to_min_map_image(): str
Experiment
+ experiment_name: str
+ experiment_datetime: datetime
+ experiment_last_update_date: datetime
+ operations: dict[str:Operation]
+ add_convert_operation(operation: Operation): Operation
+ remove_operation(operation_id:str):void
+ rename_experiment(new_experiment_name: str): void
Operation
+ operation_id: str
+ operation_type: Enum.OperationType
+ operation_datetime: datetime
+ operation_input: Asset
+ operation_optional_extra_input: Asset | str
+ operation_output: Asset | str
Extends
Extends
Extends
<<Interface>>
LogicObject
+ toDto()
Asset
+ path: str
+ is_dir: bool
+ finalize_path(final_destination_path: str = None): void

Figure 13: Logic Layer Diagram

Logic layer main classes:

- **Service** class is designed as a singleton class that is considered as the "API" of the *Logic Layer*'s functionality.
- **ConvertController** class is a controller that is responsible for navigating the conversion requests to the suitable handling converter method.
- **Asset** class is a *Logic* class that represents a system asset and there are the following sub-assets in the system:
    1. **Template** class represents the fingerprint template in the *Logic Layer*.
    2. **Image** class represents the fingerprint image in the *Logic Layer*.
    3. **3DObject** class represents the fingerprint 3d object in the *Logic Layer*.
- **Operation** class represents a conversion operation in the system by input *Asset* and output *Asset*.
- **Experiment** class represents an experiment as an aggregation of operations.
- **ExperimentController** class is a controller class that is responsible for handling experiment-related requests such as: opening a new experiment, deleting an experiment, etc...
- **IMatcher** interface is a contract that is enforcing the concrete *Matcher* to implement the following different matching methods:
    1. *oneToOne* templates matching.
    2. *oneToMany* templates matching.
    3. *manyToMany* templates matching.
- **Matcher** class is the concrete matcher that will implement the *IMatcher* interface.
- **MatcherController** class is a controller class that is responsible for managing template matching requests.
- **LogicObject** interface is a contract that enforce all the following *Logic* classes:
    1. *Template*
    2. *Image*
    3. *3DObject*
    4. *Operation*
    5. *Experiment*

  to implement the "toDto()" method which is responsible for building the *DtO* objects of the logic object respectively.

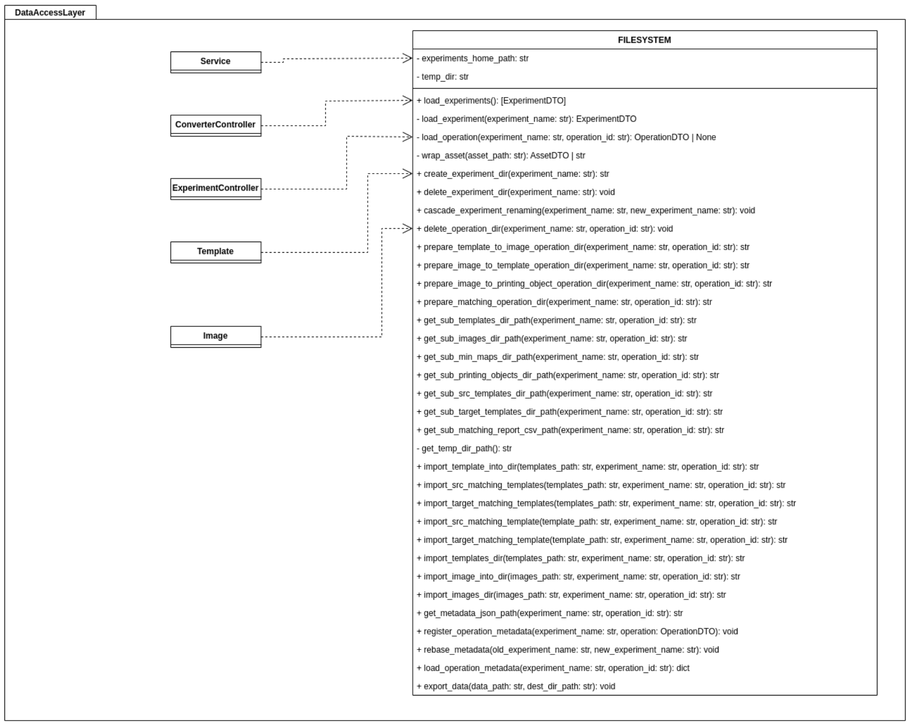

DataAccessLayer
Service
ConverterController
ExperimentController
Template
Image
FILESYSTEM
- experiments_home_path: str
- temp_dir: str
+ load_experiments(): [ExperimentDTO]
- load_experiment(experiment_name: str): ExperimentDTO
- load_operation(experiment_name: str, operation_id: str): OperationDTO | None
- wrap_asset(asset_path: str): AssetDTO | str
+ create_experiment_dir(experiment_name: str): str
+ delete_experiment_dir(experiment_name: str): void
+ cascade_experiment_renaming(experiment_name: str, new_experiment_name: str): void
+ delete_operation_dir(experiment_name: str, operation_id: str): void
+ prepare_template_to_image_operation_dir(experiment_name: str, operation_id: str): str
+ prepare_image_to_template_operation_dir(experiment_name: str, operation_id: str): str
+ prepare_image_to_printing_object_operation_dir(experiment_name: str, operation_id: str): str
+ prepare_matching_operation_dir(experiment_name: str, operation_id: str): str
+ get_sub_templates_dir_path(experiment_name: str, operation_id: str): str
+ get_sub_images_dir_path(experiment_name: str, operation_id: str): str
+ get_sub_min_maps_dir_path(experiment_name: str, operation_id: str): str
+ get_sub_printing_objects_dir_path(experiment_name: str, operation_id: str): str
+ get_sub_src_templates_dir_path(experiment_name: str, operation_id: str): str
+ get_sub_target_templates_dir_path(experiment_name: str, operation_id: str): str
+ get_sub_matching_report_csv_path(experiment_name: str, operation_id: str): str
- get_temp_dir_path(): str
+ import_template_into_dir(templates_path: str, experiment_name: str, operation_id: str): str
+ import_src_matching_templates(templates_path: str, experiment_name: str, operation_id: str): str
+ import_target_matching_templates(templates_path: str, experiment_name: str, operation_id: str): str
+ import_src_matching_template(template_path: str, experiment_name: str, operation_id: str): str
+ import_target_matching_template(template_path: str, experiment_name: str, operation_id: str): str
+ import_templates_dir(templates_path: str, experiment_name: str, operation_id: str): str
+ import_image_into_dir(images_path: str, experiment_name: str, operation_id: str): str
+ import_images_dir(images_path: str, experiment_name: str, operation_id: str): str
+ get_metadata_json_path(experiment_name: str, operation_id: str): str
+ register_operation_metadata(experiment_name: str, operation: OperationDTO): void
+ rebase_metadata(old_experiment_name: str, new_experiment_name: str): void
+ load_operation_metadata(experiment_name: str, operation_id: str): dict
+ export_data(data_path: str, dest_dir_path: str): void

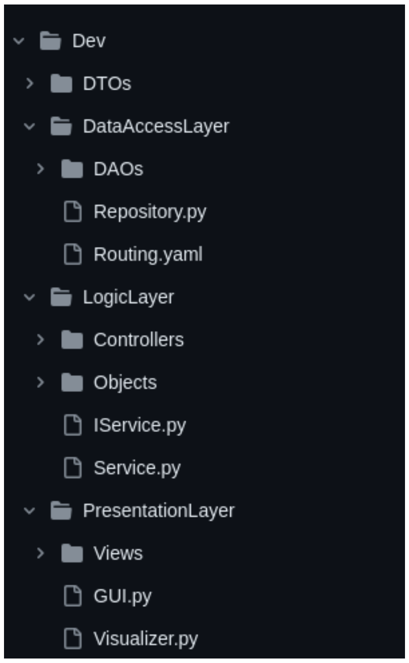
Figure 14: Data Access Layer Diagram

Data Access layer main classes:

- **FILESYSTEM** class is a singleton class that navigates requests and establishes the connection between the *Logic Layer* and the *Data Access Layer* and it has direct access to the assets on the local storage.
- **Operation's Assets** are stored on local storage with a *metadata.json* class that stores the paths of the assets on the local file system.

5.3 Packages

The system classes are organized into a hierarchy of packages based on their concept, responsibilities, and functionality, see figure below (* Visualized is not there anymore as it is integrated into the ConverterController functionality):



```
Dev
  DTOs
  DataAccessLayer
    DAOs
    Repository.py
    Routing.yaml
  LogicLayer
    Controllers
    Objects
    IService.py
    Service.py
  PresentationLayer
    Views
    GUI.py
    Visualizer.py
```

Figure 15: System Packages

6 User Interface Draft

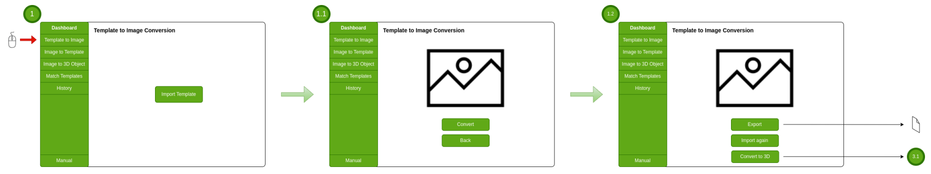


Figure 16: Template to Image Conversion Views

Template to Image Conversion Views describe the fingerprint template to fingerprint image conversion workflow, starting with importing a fingerprint template and displaying it then displaying the generated fingerprint image - output from ML model in logic layer- and giving the user the option to export it.

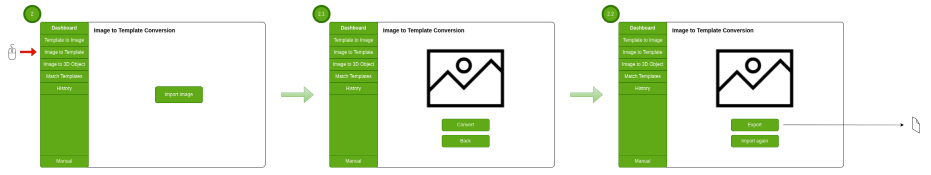


Figure 17: Image to Template Conversion Views

Image to Template Conversion Views describe the fingerprint image to fingerprint template - minutiae points - conversion workflow, starting with importing a fingerprint image and displaying it with the option to perform image enhancement on demand then displaying the extracted fingerprint template - output from *NBIS Tool* in a logic layer- and giving the user the option to export it.

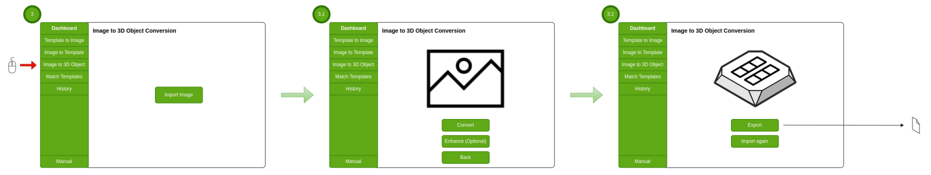
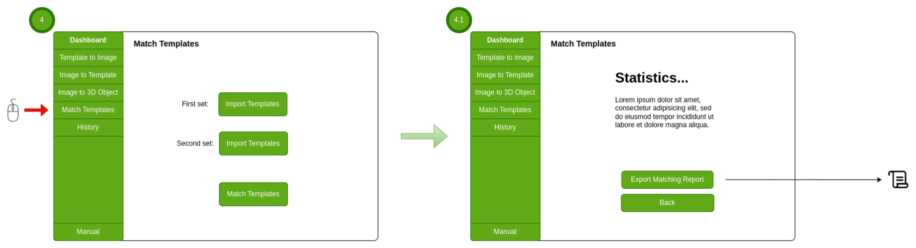


Figure 18: Image to 3d Object Conversion Views

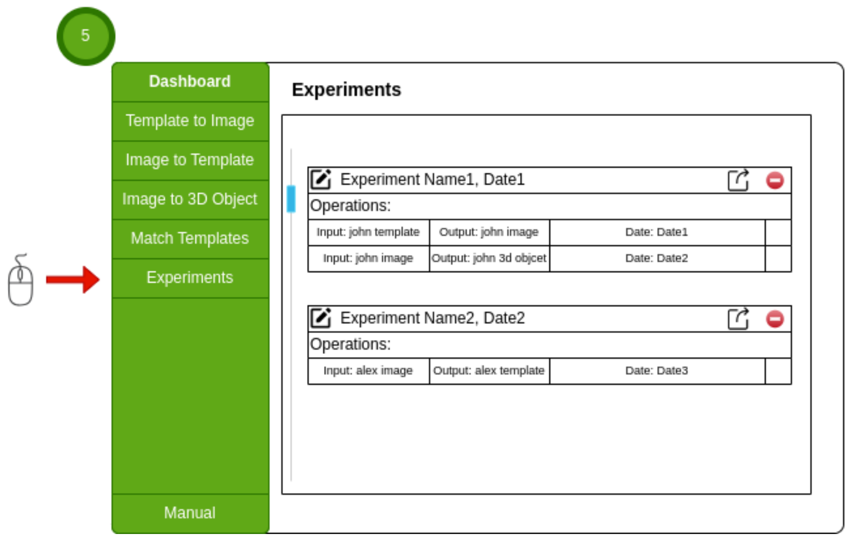
Image to 3d Object Conversion Views describe the fingerprint image to 3d Object - *.stl* - conversion workflow, starting with importing a fingerprint image and displaying it with the option to perform image enhancement on demand then displaying the 3d object of the fingerprint - output from in the logic layer- and giving the user the option to export it.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Figure 19: Templates Matching Views

Templates Matching Views describe the templates matching workflow, starting with importing the templates to be matched and then importing the templates to get matched with templates that have been imported in the first view finally displaying the matching scores - output from the logic layer - with the option to export a matching report with details regarding the matching.



Experiment Name1, Date1

Operations:

Input: john template	Output: john image	Date: Date1
Input: john image	Output: john 3d objcet	Date: Date2

Experiment Name2, Date2

Operations:

Input: alex image	Output: alex template	Date: Date3

Figure 20: Experiments History View

Experiments History View describes the feature of displaying the history of previous experiments and their operations and data.

7 Testing

7.1 Conversion:

1. - **Description:** Converting valid template to image.
   - **Input:** Valid template
   - **Expected Result:** Fingerprint image that matches the template data.
   - **Actual Result:** Success
2. - **Description:** Converting invalid template to image.
   - **Input:** Invalid template
   - **Expected Result:** Conversion failure.
   - **Actual Result:** Success
3. - **Description:** Converting multiple templates (directory) to images.
   - **Input:** Valid templates directory
   - **Expected Result:** Fingerprint images directory that matches the corresponding templates directory data.
   - **Actual Result:** Success
4. - **Description:** Converting valid image to template.
   - **Input:** Valid image (any format)
   - **Expected Result:** Fingerprint template that matches the image.
   - **Actual Result:** Success
5. - **Description:** Converting multiple images (directory) to templates.
   - **Input:** Valid images directory
   - **Expected Result:** Fingerprint templates directory that matches the corresponding images directory data.
   - **Actual Result:** Success
6. - **Description:** Converting invalid image to template.
   - **Input:** Invalid image
   - **Expected Result:** Conversion failure.
   - **Actual Result:** Success
7. - **Description:** Converting valid image to 3d object.
   - **Input:** Valid image (any format)
   - **Expected Result:** 3d fingerprint object surface that matches the fingerprint image.
   - **Actual Result:** Success

8. - **Description:** Converting multiple images (directory) to 3d objects.
   - **Input:** Valid images directory
   - **Expected Result:** Fingerprint 3d objects directory that matches the corresponding images directory data.
   - **Actual Result:** Success

9. - **Description:** Converting invalid image to 3d object.
   - **Input:** Invalid image
   - **Expected Result:** Conversion failure.
   - **Actual Result:** Success

7.2 Matching:

1. - **Description:** Matching one template with one template.
   - **Input:** 2 valid templates
   - **Expected Result:** Matching score for the 2 templates.
   - **Actual Result:** Success

2. - **Description:** Matching one template with many templates (directory).
   - **Input:** Valid template and templates directory
   - **Expected Result:** Matching score for the single template with each of the other templates (Matrix score).
   - **Actual Result:** Success

3. - **Description:** Matching many templates (directory) with many templates (directory).
   - **Input:** 2 valid template directories
   - **Expected Result:** Matching score matrix that contains each template name and the score with each one of the other templates in the other directory.
   - **Actual Result:** Success

4. - **Description:** Matching invalid templates of any matching.
   - **Input:** one template or templates directory.
   - **Expected Result:** Failure, no matching score.
   - **Actual Result:** Success

5. - **Description:** Export templates matching score.
   - **Input:** 2 valid templates or template sets
   - **Expected Result:** Matching score *.csv* file.
   - **Actual Result:** Success

7.3 Experiments Manipulation:

1. - **Description:** viewing all experiments.
   - **Pre-condition:** Existing experiment.
   - **Input:** None
   - **Expected Result:** List of experiments.
   - **Actual Result:** Success
2. - **Description:** Rename experiment.
   - **Pre-condition:** Existing experiment.
   - **Input:** Valid name
   - **Expected Result:** Experiment is renamed to the new name.
   - **Actual Result:** Success
3. - **Description:** Set current experiment.
   - **Pre-condition:** Existing experiment.
   - **Input:** Valid name
   - **Expected Result:** The Experiment is set to be the current experiment.
   - **Actual Result:** Success
4. - **Description:** Set the non-existent experiment name to be the current experiment.
   - **Pre-condition:** Existing experiment.
   - **Input:** Non existing experiment name.
   - **Expected Result:** Failure, the old experiment is still current.
   - **Actual Result:** Success
5. - **Description:** Rename invalid experiment name.
   - **Pre-condition:** Existing experiment.
   - **Input:** Invalid name
   - **Expected Result:** Failure, experiment not renamed.
   - **Actual Result:** Success
6. - **Description:** Rename valid but existing experiment name.
   - **Pre-condition:** Existing experiment.
   - **Input:** Valid and existing experiment name.
   - **Expected Result:** Failure, experiment not renamed.
   - **Actual Result:** Success
7. - **Description:** Delete experiment.

- **Pre-condition:** Existing experiment.
- **Input:** Valid experiment name
- **Expected Result:** Experiment is deleted.
- **Actual Result:** Success

8. - **Description:** Delete operation.
   - **Pre-condition:** Existing operation.
   - **Input:** Valid operation id
   - **Expected Result:** Operation is deleted.
   - **Actual Result:** Success

9. - **Description:** Operation is added after convert.
   - **Pre-condition:** Perform conversion operation.
   - **Input:** Valid convertible asset.
   - **Expected Result:** Operation is added to its parent experiment.
   - **Actual Result:** Success

10. - **Description:** Operation is added after matching.
    - **Pre-condition:** Perform conversion operation.
    - **Input:** Valid template or templates directory.
    - **Expected Result:** Operation is added to its parent experiment.
    - **Actual Result:** Success

7.4 Experiments Manipulation:

1. - **Description:** template file (.min) minutiae map creation.
   - **Input:** Valid template (.min) file.
   - **Expected Result:** minutiae map image created (.png).
   - **Actual Result:** Success

2. - **Description:** invalid template file (.min) minutiae map creation.
   - **Input:** Invalid template (.min) file.
   - **Expected Result:** Failure, no generated minutiae map image.
   - **Actual Result:** Success