Module Interface Specification for Retinal Vessel Segmentation System (RVSS)

Xinyu Ma

March 18, 2024

1 Revision History

Date	Version	Notes
03/14/2024	1.0	Initial Release
03/18/2024	2.0	Add some flow charts

2 Symbols, Abbreviations and Acronyms

See SRS Documentation for symbols, abbreviations and acronyms.

symbol	description
CNN	Convolution Neural Network
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
RVSS	Retinal Vessel Segmentation System
SRS	Software Requirements Specification

Contents

List of Tables

3 Introduction

The following document details the Module Interface Specifications for Retinal Vessel Segmentation System (RVSS). This document specifies how every module is interfacing with every other parts.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at Github repository for RVSS.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (1991). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SCEC.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
Boolean	$\mathbb{B}$	a value of either True or False

The specification of RVSS uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, RVSS uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Upload and Validate Module
	Image Management Module
	Image Preprocessing Module
	Image Segmentation Module
	Output Format Module
	Report Generation Module
	User Interface Module
	Main Function Module
Software Decision Module	Algorithm Optimization Module
	Plotting Result Module
	Logging Module
	Image patching Module
	Image Dataset Reader Module
	Network Model Reader Module

Table 1: Module Hierarchy

6 MIS of Input Upload and Validate Module

6.1 Module

UploadValidate

6.2 Uses

- Image Management Module

6.3 Syntax

6.3.1 Exported Constants

- Accpted_Image_Formats: list of image file formats that the system can process (e.g., [".jpg", ".jpeg", ".png"]).
- Max_Image_Size: maximum file size for uploaded images, specified in megabytes.
- Min_Image_Resolution: minimum image resolution required for processing, specified as (width, height) in pixels.

6.3.2 Exported Access Programs

1. UploadImage

- Description: receives and stores an image file uploaded by the user.
- Input: an image file representing the retinal image.
- Output: upload success or fail (data type: boolean); ImageID (string if uploading successfully).
- Exceptions: "UploadFailureException" if the upload process fails due to system issues.

2. ValidateImage

- Description: validates the uploaded image to ensure it meets the system's criteria for format, size, and quality.
- Input: ImageID (unique identifier for the uploaded image).
- Output: Validation Result (object containing validation status and any issues found).
- Exceptions: "ImageNotFoundException" if the image corresponding to the ImageID cannot be found, and "ValidationException" if the image does not match the predefined criteria.

6.4 Semantics

6.4.1 State Variables

Uploaded_Images_Data: a collection or database table reference storing information about uploaded images, including their temporary IDs, upload status, and validation status.

6.4.2 Environment Variables

Upload_Path: the file system path to the directory where uploaded images are temporarily stored.

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

1. UploadImage

   - Transition: accepts an image file ("imageFile") uploaded by the user. The routine stores the image temporarily in the system and assigns it a unique "ImageID", then updates the "UploadedImagesData" state variable to include the new image and its upload status. The image is saved to the temporary storage area specified by the "Upload_Path" environment variable.

   - Exceptions: "UploadFailureException" raised if the upload process fails due to system issues.

2. ValidateImage

   - Transition: takes a "ImageID" as input and retrieves the corresponding image from "Uploaded_Images_Data". Then validates the image against predefined criteria such as format, size, and resolution. The "UploadedImagesData" state variable is updated to reflect the validation status of the image.

   - Exceptions: "ImageNotFoundException" raised if no image matches the provided "ImageID". "ValidationException" raised if an error occurs during the validation process.

6.4.5 Local Functions

None

7 MIS of Image Management Module

For each retina image uploaded by users, a unique ImageID will be assigned. The ImageID, original image, image upload verification status, image preprocessing status, image segmentation status, and image segmentation result are stored as one item, and all image status can be accessed using only the ImageID.

7.1 Module

ImageManagement

7.2 Uses

None

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

1. StoreImage

- Input: image data (type: Tensor)
- Output: ImageID which is a unique identifier for the stored image.
- Exceptions: "StorageException" raised if failure occurs during the image storage process.

2. RetrieveImage

- Input: ImageID
- Output: data of the retrieved image.
- Exceptions: "ImageNotFoundException" raised if no image matches the ImageID, and "RetrievalException" raised if failure happens during the image retrieval process.

7.4 Semantics

7.4.1 State Variables

ImageDatabase: A data structure that keeps track of all stored images, their state, and ImageID. This could be an in-memory structure or a connection to an external database system.

7.4.2 Environment Variables

Storage_Path: specifies the filesystem path to the directory where images are stored.

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

1. StoreImage

- Transition:Saves the provided image to the storage system, then generates a unique ImageID for the image, which is used for future retrieval or reference. Updates ImageDatabase by adding a new entry for the stored image, including its storageID and associated state.
- Exceptions: "StorageException" raised if failure occurs during the image storage process.

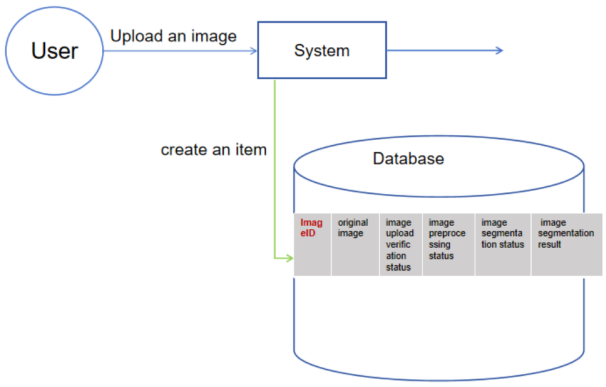

Figure 1: Store Image

2. RetrieveImage

- Transition: ImageID is provided, then retrieves the specific image and its state associated with the ImageID.
- Exceptions: "ImageNotFoundException" raised if no image matches the ImageID, and "RetrievalException" raised in case of failures during the retrieval process.

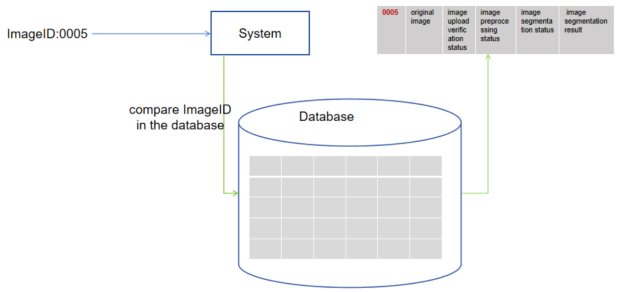

Figure 2: Retrieve an image

7.4.5 Local Functions

None

8 MIS of Image Preprocessing Module

8.1 Module

ImagePreprocessing

8.2 Uses

– Image Management Module

8.3 Syntax

8.3.1 Exported Constants

– Normalization_Range: defines the target range for image intensity normalization, e.g., [0, 1] or [0, 255].

– Denoising_Strength: a predefined level or parameter settings for the denoising algorithm.

– Contrast _Enhancement _Parameters: a parameter that controls the degree of contrast enhancement applied to images.

8.3.2 Exported Access Programs

1. NormalizeImage

   – Inputs: ImageID

   – Outputs: the image with intensity values normalized.

   – Exceptions: "NormalizationException" raised if normalization fails, and "ImageNotFoundException" raised if no image matches the ImageID.

2. DenoiseImage

   – Inputs: ImageID

   – Outputs: the image with reduced noise.

   – Exceptions: "DenoisingException" raised if denoising fails.

3. EnhanceImageContrast

   – Inputs: ImageID

   – Outputs: the image with with enhanced contrast.

   – Exceptions: "ContrastEnhancementException" raised if enhancement fails.

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

- Consistent preprocessing parameters: there's an assumption that preprocessing parameters (e.g., normalization range, denoising strength) are chosen to be effective across the spectrum of images processed by the system. While parameters might be adjustable, the defaults are assumed to be generally applicable.
- Image preprocessing steps: For an image, first normalize the image, then denoise, and finally enhance the contrast.

8.4.4 Access Routine Semantics

1. NormalizeImage
   - Transition: use the "ImageID" to retrieve images in the Image Database, then adjusts the intensity values of the raw image to fall within a specified range, enhancing uniformity across images for improved segmentation performance. Store the normalized image in the Image Database.
   - Exceptions:"NormalizationException" if normalization fails due to unexpected image properties, and "ImageNotFoundException" raised if no image matches the ImageID.
2. DenoiseImage
   - Transition: use the "ImageID" to retrieve images in the Image Database, then check whether the image has been normalized. If the image has a normalized version, then applies a denoising algorithm to it to reduce noise while preserving essential details, crucial for accurate segmentation. Store the normalized image in the Image Database.
   - Exceptions:"DenoisingException" raised if denoising is unsuccessful or leads to significant loss of image detail.
3. EnhanceImageContrast

– Transition: use the “ImageID” to retrieve images in the Image Database, then check whether the image has been denoised. If the image has a denoised version, then using methods suited to highlight features relevant to vessel segmentation, potentially making vessels more distinct. Store the contrast enhanced image in the Image Database.

– Outputs: The image with with enhanced contrast.

– Exceptions: “ContrastEnhancementException” raised if the process fails or results in an unusable image.

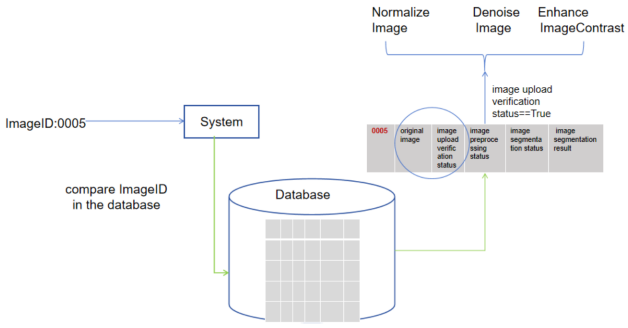

Figure 3: Image Preprocessing Module

8.4.5 Local Functions

None

9 MIS of Image Segmentation Module

9.1 Module

ImageSegmentation

9.2 Uses

- Image Management Module
- Image Preprocessing Module

9.3 Syntax

9.3.1 Exported Constants

Segmentation_Algorithm: Specifies the default algorithm or model used for image segmentation. It could be a string identifier or algorithm-specific parameters.

9.3.2 Exported Access Programs

1. SegmentImage

- Description: performs segmentation on a preprocessed image to identify and delineate retinal vessels.
- Input: ImageID
- Output: prompt message that image segmentation is completed
- Exceptions: "PreprocessedImageNotFoundException" raised if the preprocessed image corresponding to the ImageID cannot be found, and "SegmentationException raised if the segmentation process encounters errors.

9.4 Semantics

9.4.1 State Variables

Current_Algorithm_Settings: Stores the current settings or parameters used by the segmentation algorithm, allowing for adjustments if needed.

9.4.2 Environment Variables

Model_Storage_Path: The filesystem path pointing to the location where segmentation models or algorithm parameters are stored, enabling the module to dynamically load different models based on system configuration or needs.

9.4.3 Assumptions

- The input images are preprocessed and normalized, ensuring consistency in image quality and format that is conducive to effective segmentation.
- The segmentation algorithms or models are pre-trained, capable of delivering high accuracy in vessel delineation across diverse image datasets.

9.4.4 Access Routine Semantics

1. SegmentImage

- Transition: use the"ImageID" to retrieve images in the Image Database, then check whether the image has been preprocessed. Takes the preprocessed image and applies the segmentation algorithm defined by Segmentation_Algorithm and Current_Algorithm
  _Settings, producing a prompt message when the image segmentation is completed. Stores the segmented image in the Image Database and logs related to segmentation performance.
- Exception: "PreprocessedImageNotFoundException" raised if the preprocessed image corresponding to the ImageID cannot be found, and "SegmentationException raised if the segmentation process encounters errors.

9.4.5 Local Functions

1. LoadSegmentationModel

- Description: loads the segmentation model or algorithm parameters from storage and it is primarily called within SegmentImage before applying the segmentation algorithm.
- Input: None
- Output: Model or algorithm parameters ready for use in segmentation.

10 MIS of Output Format Module

10.1 Module

OutputFormat

10.2 Uses

- Image Management Module

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

1.OutputSegmentationImage

- Description: Output retinal image segmentation results.
- Input: ImageID
- Output: the corresponding segmented image
- Exceptions: "SegmentationImageNotFoundException" raised if the segmentation result corresponding to the ImageID cannot be found.

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Environment Variables

Default_Output_Path: Specifies the default filesystem path where output images are saved if no specific path is provided in the request. This ensures that generated images are stored in a consistent location.

10.4.3 Assumptions

None

10.4.4 Access Routine Semantics

1. output

- Transition: This module is able to output the segmentation result. Use the"ImageID" to retrieve images in the Image Database, then check whether the image has been segmented. Then shows the segmentation image on the screen and store the image in Default_Output_Path.
- Exception: "SegmentationImageNotFoundException" raised if the segmented image corresponding to the ImageID cannot be found.

10.4.5 Local Functions

None

11 MIS of Report Generation Module

11.1 Module

ReportGeneration

11.2 Uses

- Image Management Module
- Output Format Module

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

1. ReportGeneration

- Description: Creates a comprehensive report that includes the segmentation results along with metrics relevant to the segmentation process.
- Input: ImageID
- Output: a document file that contains the generated report,
- Exceptions: "ImageNotFoundException" raised if the image corresponding to the ImageID cannot be found. "ReportGenerationException": raised if there is an error during the report generation process, including issues with input data integrity or problems creating the document file.

11.4 Semantics

11.4.1 State Variables

- Current_Report_Data: This variable holds the data currently being processed for report generation. It could be structured to include segmentation results, and any additional information that will be included in the final report
- Generated_Reports: A collection or history of reports that have been generated during a session or over a specified period. This could be used for tracking, auditing, or re-generation purposes.

11.4.2 Environment Variables

- Report_Output_Path: specifies the default filesystem path where generated report files are saved. This ensures that reports are stored in a consistent and retrievable location.
- Default_Report_Format: indicates the default format for reports generated by the module, used if no specific format is requested. This variable ensures consistency in report presentation and can be set based on user or system preferences.

11.4.3 Assumptions

- The input segmentation result is complete and correctly formatted, ensuring that all necessary data for report generation is available and accurate.
- A suitable report template is available and correctly formatted, allowing for the dynamic insertion of analysis results.
- The system has access to the necessary resources and permissions to create and save the report document in the specified format.

11.4.4 Access Routine Semantics

1. ReportGeneration

- Transition: Utilizing the "ImageID" to retrieve images in the Image Database. It should include the original image, and the segmentation results along with metrics relevant to the segmentation process. Then use the Default_Report_Format to generate the report and save it according to the Report_Output_Path.
- "UnsupportedFormatException": triggered if the requested report format is not among those supported by the system. "ImageNotFoundException" raised if the image corresponding to the ImageID cannot be found. "ReportGenerationException": raised in cases where the report generation process encounters errors, such as template processing issues, data format problems, or file saving errors.

11.4.5 Local Functions

None

12 MIS of User Interface Module

12.1 Module

UI

12.2 Uses

- Input Upload and Validate Module
- Image Management Module
- Image Preprocessing Module
- Image Segmentation Module
- Output Format Module
- Report Generation Module

12.3 Syntax

12.3.1 Exported Constants

- Supported_Image_Formats: a list of image file formats that can be uploaded.
- Default_Report_Format:the default format for generated reports.

12.3.2 Exported Access Programs

1. Display Main Interface

- Description: Presents the main interface screen to the user, offering access to various functionalities like image upload, segmentation, and report generation.
- Input: None
- Output: displays the main interface to the user.
- Exceptions: None

2. Upload Image Interface

- Description: provides an interface for users to upload images for segmentation.
- Input: user-selected image files.
- Output: confirmation of successful upload or feedback on any issues encountered, and a unique ImageID if uploading successfully.

- Exceptions: "UploadFailureException" raised if there's an issue with uploading the image.

3. Segmentation Initiation

- Description: triggers the segmentation process for uploaded images and displaying progress.
- Input: identifiers for images uploaded ("ImageID") and ready for segmentation.
- Output: status feedback during segmentation. Notification upon completion or if issues arise.
- Exceptions: "SegmentationInitiationException": raised if the segmentation process cannot be initiated .

4. View Segmentation Results

- Description: displays segmentation results for selected images.
- Input: identifiers for images ("ImageID") whose segmentation results are to be viewed.
- Output: a display of segmentation results.
- Exceptions: "ResultsDisplayException": raised if there's an issue retrieving or displaying results.

5. Generate Report

- Description: facilitates the generation of reports based on segmentation results.
- Input: identifiers for images ("ImageID") and segmentation results to be included in the report.
- Output: a report file.
- Exceptions: "ReportGenerationException": raised if report generation fails.

12.4 Semantics

12.4.1 State Variables

- Current_Screen: Tracks which interface screen is currently being displayed.
- Uploaded_Images: A list of images that have been uploaded and are available for processing.

12.4.2 Environment Variables

- Max_Upload_Size: defines the maximum size allowed for image uploads.

12.4.3 Assumptions

- Users have a basic understanding of how to interact with web or desktop applications.
- The system is running on a platform capable of supporting graphical user interfaces (GUIs).

12.4.4 Access Routine Semantics

1. Display Main Screen

- Purpose: To present the primary interface through which users interact with the system, offering navigation to various functionalities.
- Output: The main UI screen is displayed to the user and ensures that all primary functionalities (e.g., image upload, segmentation, report generation) are accessible.
- Exceptions: None

2. Upload Image Interface

- Purpose: provides a mechanism for uploading images to be processed, including file selection dialogs.
- Transitions: validates the uploaded files and give a unique ImageID if uploading successfully. Displays progress and confirms once uploads are complete or provides error feedback.
- Exceptions: "UploadFailureException" which is triggered by issues like file size exceeding limits or unsupported formats.

3. Segmentation Initiation

- Purpose: allows users to start the segmentation process for uploaded images.
- Transitions: initiates the process, and provides feedback on progress. Notifies the user upon completion or in case of errors.
- Exceptions: "SegmentationInitiationException": raised if segmentation cannot be initiated.

4. View Segmentation Results

- Purpose: displays the results of image segmentation.
- Transitions: retrieves segmentation results according to the "ImageID" and provide a display interface showing segmentation results..

- Exceptions: "ResultsDisplayException": raised if there's a failure in retrieving or displaying the segmentation results, such as missing data or rendering issues

5. Generate Report

- Purpose: enables users to generate comprehensive reports based on segmentation results.
- Transitions: retrieves original image and segmentation results according to the "ImageID", then compiles the image data, and specified analyses into a formatted report.
- Exceptions: "ReportGenerationException": raised in case of failures during report compilation.

12.4.5 Local Functions

1. Display_Progress

- Description: updates the UI to show progress during each operation.
- Input: None
- Output: Updated progress indicator on the UI.

2. Error_Message

- Description: displays error messages or alerts to the user in case of failures or exceptions in any process.
- Input: error message or exception details.
- Output: displayed message or alert in the UI.

13 MIS of Main Function Module

For each retina image uploaded by a user, a unique ID will be assigned. The image ID, image, image upload verification status, image preprocessing status, image segmentation status, and image segmentation result are stored as one item, and the image status can be accessed using only the ID. We mainly focus on the user usage, rather than developers retraining the model, so the main function here does not involve the model training part. We will train an image segmentation model and use it directly in the software.

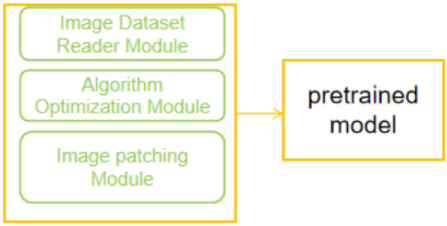

Figure 4: Model Pretrain Process

13.1 Module

MainFunction

13.2 Uses

- Hardware-Hiding Module
- Input Upload and Validate Module
- Image Management Module
- Image Preprocessing Module
- Image Segmentation Module
- Output Format Module
- Report Generation Module
- User Interface Module

13.3 Syntax

13.3.1 Exported Constants

- Max_Upload_Size: The maximum size allowed for image uploads.
- Supported_Image_Formats: A list of image formats that the system can process.

13.3.2 Exported Access Programs

1. System Initialization

- Description: prepares the RVSS for operation, initializing subsystems, loading necessary resources, and setting up the environment for user interactions.
- Inputs: None
- Outputs: System status.
- Exceptions: "InitializationException": raised if the system fails to initialize properly, possibly due to configuration errors or unavailable resources.

2. Start a new Session

- Description: open a new user session, presenting the user interface and enabling access to the system's functionalities.
- Inputs: None
- Outputs: session identifier or status indicating a successful start.
- Exceptions: "SessionStartException" raised if there's an issue initiating a new session.

3. Image Upload Process

- Description: coordinates the upload of images by users, ensuring proper validation and storage for subsequent processing steps.
- Inputs: image files selected by the user.
- Outputs: confirmation of successful uploads, including identifiers (ImageID) for the uploaded images.
- Exceptions: "ImageUploadException" Raised if an error occurs during the image upload process.

4. Execute Segmentation

- Description: triggers the image segmentation process for the uploaded images, utilizing the Image Preprocessing and Image Segmentation Modules.
- Input: identifiers for images (ImageID) ready for segmentation.
- Output: status feedback during segmentation. Notification upon completion or if issues arise.
- Exceptions: "SegmentationException": raised if the segmentation process encounters errors.

5. Display Segmentation Results

- Description: display the image segmentation result for the uploaded images.
- Input: identifiers for images (ImageID) already segmented.
- Output: Segmentation results, including any relevant data and metrics.
- Exceptions: "ResultsDisplayException": raised if the display result process encounters errors.

6. Generate Reports

- Description: initiates the generation of comprehensive reports based on segmentation results, incorporating analytical insights.
- Inputs: identifiers for segmentation results to be included in reports.
- Outputs: the generated reports files.
- Exceptions: "ReportGenerationException": raised if there are issues generating the reports.

13.4 Semantics

13.4.1 State Variables

- Current_Session: tracks information about the active user session.
- Uploaded_Images: maintains a list of images uploaded during the current session.

13.4.2 Environment Variables

- System_Config_Path: specifies the path to system configuration files.

13.4.3 Assumptions

- The system is operated in an environment with adequate computational resources.
- External services and modules (e.g., for image processing or report generation) are available and functional.

13.4.4 Access Routine Semantics

main(): 1. Transition: Control the order of execution of different modules as follow:

- Upload Image
  - Module Used: Input Upload and Validate Module (M2, Section 6), Image Management Module (M3, Section 7), User Interface Module (M8, Section 12)
  - Process: User selects image files for upload (M8, Section 12). The Input Upload and Validate Module (M2, Section 6) validates the file format and size. Images are uploaded and stored (M3, Section 7), ready for preprocessing.

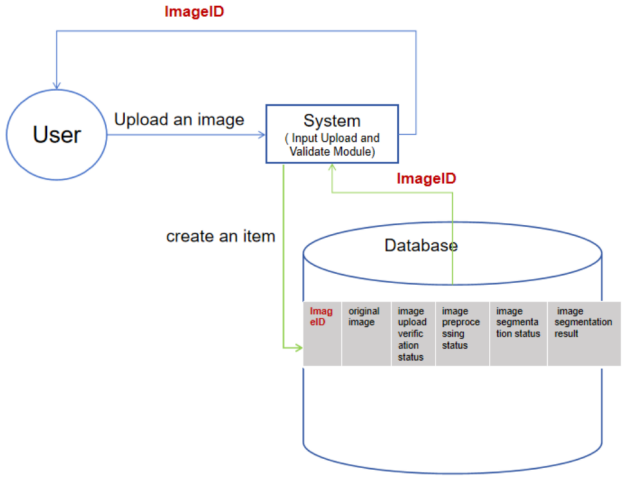

Figure 5: Upload Image

- Preprocess Image
  - Module Used: Image Management Module (M3, Section 7), Image Preprocessing Module (M4, Section 8)
  - Process: The Main Function Module sends uploaded images to the Image Preprocessing Module (M4, Section 8). The images undergo preprocessing steps to enhance quality. Preprocessed images are stored with the ImageID (M3, Section 7), ready for segmentation.

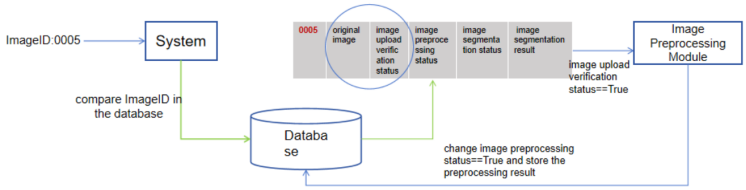

Figure 6: Preprocess Image

- Segment Image
  - Module Used: Image Management Module (M3, Section 7), Image Segmentation Module (M5, Section 9)
  - Process: Preprocessed images are then forwarded to the Image Segmentation Module (M5, Section 9), where segmentation model has pre-trained. Segmentation model can identify and delineate retinal vessels.Preprocessed images are stored with the ImageID (M3, Section 7).

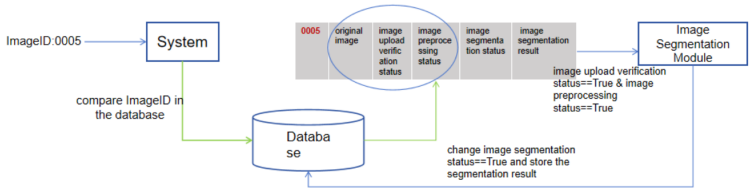

Figure 7: Segment Image

- Display Segmentation Result
  - Module Used: Image Management Module (M3, Section 7), Output Format Module (M6, Section 10), User Interface Module (M8, Section 12)
  - Process: use the ImageID to retrieve the segmented images and the main function module instructs the user interface module to display the segmentation results.

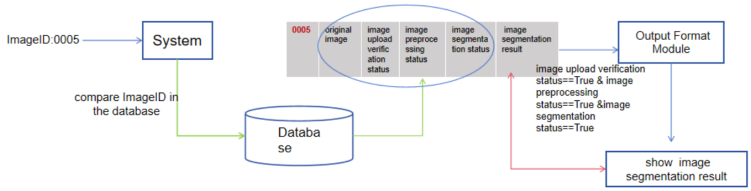

Figure 8: Display Segmentation Result

- Generate Report
  - Module Used: Image Management Module (M3, Section 7), User Interface Module (M8, Section 12), Report Generation Module (M7, Section 11)
  - Process: use the ImageID to retrieve the segmented images and the main function module instructs the user interface module to display the segmentation results.
  - Segmentation results are collected according to the ImageID (M3, Section 7). The Report Generation Module compiles these into a structured report (M7, Section 11).
- Exception: Potential exceptions raises are from different sub-modules only.

13.4.5 Local Functions

None

14 MIS of Plotting Result Module

14.1 Module

plot

14.2 Uses

- Hardware Hiding Module

14.3 Syntax

14.3.1 Exported Constants

None

14.3.2 Exported Access Programs

1. Plot Segmentation Overlay

- Description: generates a plot displaying the segmentation overlay on the original image, highlighting the segmented vessels.
- Inputs: the original fundus image “OriginalImage” and the output from the segmentation process “SegmentationResult”, including the segmented vessel image.
- Outputs: an image “OverlayPlot” with the segmentation results overlaid on the original image, suitable for visualization and assessment.
- Exceptions: “PlottingException” raised if there is an issue generating the overlay plot.

2. Plot Segmentation Metrics

- Description: visualizes various segmentation metrics in a graphical format, such as accuracy, sensitivity, specificity, or other relevant performance measures.
- Inputs: a collection of metrics data generated from the segmentation process.
- Outputs: a graphical representation (e.g., bar chart, line graph) of the segmentation metrics, facilitating easy interpretation of performance.
- Exceptions:“MetricsPlottingException”: Raised if there is an error in plotting the metrics data.

14.4 Semantics

14.4.1 State Variables

- Current_Plot: Stores the most recently generated plot, which can be an overlay plot or a metrics plot.

14.4.2 Environment Variables

- Plot_Storage_Path: specifies the directory where generated plots should be saved.

14.4.3 Assumptions

- The segmentation results and metrics data provided to the module are accurate and in a format that can be processed for visualization.

14.4.4 Access Routine Semantics

1. Plot Segmentation Overlay

- Transitions: The module combines the "OriginalImage" and "SegmentationResult" to create an overlay image where segmented vessels are clearly visible against the original background.

- Exceptions: "PlottingException" raised if the overlay cannot be generated due to issues with input data formats, or incompatibilities.

2. Plot Segmentation Metrics

- Transitions: The module takes metrics data and converts it into a visual format, choosing appropriate chart types (e.g., bar charts, line graphs) based on the data characteristics.

- Exceptions: "MetricsPlottingException" raised if there are issues converting metrics data into graphical form, such as missing data points or unsupported data types.

14.4.5 Local Functions

None

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of software engineering*. Prentice-Hall, Inc., 1991.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.