Module Interface Specification for CXR

Team 27, Neuralyzers
Ayman Akhras
Nathan Luong
Patrick Zhou
Kelly Deng
Reza Jodeiri

January 7, 2025

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

3 Introduction

The following document details the Module Interface Specifications for [Fill in your project name and description —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at .... [provide the url for your repo —SS]

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

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. (2003). 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 CXR.

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 (-∞, ∞)

The specification of CXR 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, CXR 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	
Behaviour-Hiding	Input Parameters Output Format Output Verification Temperature ODEs Energy Equations Control Module Specification Parameters Module
Software Decision	Sequence Data Structure ODE Solver Plotting

Table 1: Module Hierarchy

6 MIS of Web Application Server Module

6.1 Module

Web Application Server

6.2 Uses

M2: HTTP Server Module
M5: Doctor Profile View Module
M6: Patient List View Module
Other view modules (M7-M10) for displaying data.

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
handleRequest	HTTP request	HTTP response	InvalidRequestException

6.4 Semantics

6.4.1 State Variables

- **sessionData**: Stores current session information.
- **activeUsers**: Keeps track of currently active users.

6.4.2 Environment Variables

- **serverPort**: The port on which the server listens for incoming connections.
- **hostAddress**: The server's host address.

6.4.3 Assumptions

- Assumes the HTTP Server Module (M2) is properly configured and running.
- Assumes valid HTTP requests are received.

6.4.4 Access Routine Semantics

handleRequest(request)

- transition: Processes the incoming HTTP request and routes it to the appropriate view module.
- output: Returns the HTTP response based on the request.

6.4.5 Local Functions

- **parseRequest(request)**: Parses the incoming HTTP request to extract necessary information.
- **generateResponse(data)**: Constructs an HTTP response based on the processed data.
- **authenticateUser(credentials)**: Verifies the user's credentials before processing the request.

7 Web Application Server Module

7.1 Other Modules the Current Module Uses

- M1: Web Application Server Module
- M3: Disease Prediction Server Module
- M4: Disease Progression Tracking Server Module
- M5: Doctor Profile View Module
- M6: Patient List View Module
- Other view modules (M7-M10) for displaying data.

7.2 State Variables

- **requestQueue**: Holds incoming HTTP requests until they are processed.
- **responseQueue**: Holds outgoing HTTP responses that need to be sent back to clients.

7.3 Exported Constants and Access Programs

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
`startServer`	None	None	`ServerStartException`
`stopServer`	None	None	`ServerStopException`
`processRequest`	`HTTP request`	`HTTP response`	`InvalidRequestException`

7.3.2 Exported Constants

- **SERVER_PORT**: 8080
- **MAX_CONNECTIONS**: 100

7.4 Environment Variables

- **serverPort**: The port on which the server listens for incoming HTTP connections.
- **maxConnections**: Maximum number of simultaneous connections the server can handle.

7.5 Assumptions

- Assumes the Web Application Server Module (M1) is properly configured and running.
- Assumes incoming HTTP requests are formatted correctly.

7.6 Access Routine Semantics

7.6.1 startServer()

- **Transition**: Starts the HTTP server, initializes necessary resources, and begins listening for incoming requests.
- **Output**: No output, but may throw a `ServerStartException` if the server cannot be started.

7.6.2 stopServer()

- **Transition**: Stops the HTTP server, gracefully shuts down connections.
- **Output**: No output, but may throw a `ServerStopException` if the server cannot be stopped.

7.6.3 processRequest(request)

- **Transition**: Takes an incoming HTTP request and processes it, routing it to the appropriate server module or view module.
- **Output**: Returns an HTTP response based on the processed request.

7.7 Local Functions

- **parseRequest()**: Parses the incoming HTTP request to extract necessary information such as headers and parameters.
- **generateResponse()**: Constructs an HTTP response based on the processed data from the request.
- **handleError()**: Handles errors that arise during request processing and generates appropriate error responses.

8 Disease Prediction Server Module

8.1 Other Modules the Current Module Uses

- M1: Web Application Server Module

- M2: HTTP Server Module
- M4: Disease Progression Tracking Server Module
- M5: Doctor Profile View Module
- M6: Patient List View Module
- M7: Patient Diseases Progression View Module

8.2 State Variables

- **model**: The pre-trained model from `torchxrayvision` used for predicting lung diseases from X-ray images.
- **modelAccuracy**: Tracks the accuracy of the current model after training and validation.
- **predictionThreshold**: A constant threshold to determine the classification outcome (e.g., disease presence).
- **patientImageData**: Holds the chest X-ray image data used for prediction.

8.3 Exported Constants and Access Programs

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
`loadModel`	None	Loaded model	`ModelLoadException`
`predictDisease`	X-ray image data	Disease prediction	`InvalidImageException`

8.3.2 Exported Constants

- **PREDICTION_THRESHOLD**: 0.75 (threshold for classification of disease presence)
- **MODEL_PATH**: Path to the pre-trained model (e.g., `./models/chest_xray_model.pth`)
- **MAX_PREDICTIONS**: 1000 (maximum number of predictions to handle concurrently)

8.4 Environment Variables

- **modelPath**: The path where the `torchxrayvision` pre-trained model is saved or loaded from.
- **predictionEndpoint**: The endpoint for making predictions using chest X-ray images.

8.5 Assumptions

- Assumes the pre-trained `torchxrayvision` model is available and compatible with the data provided.
- Assumes valid X-ray image data is available for predictions.
- Assumes the Web Application Server Module (M1) and HTTP Server Module (M2) are properly configured and running.

8.6 Access Routine Semantics

8.6.1 loadModel()

- **Transition**: Loads the pre-trained disease prediction model from the specified path after image is uploaded using `torchxrayvision`.

8.6.2 predictDisease(patientImageData)

- **Transition**: Uses the loaded model to make predictions based on the provided X-ray image data.
- **Output**: Returns the disease prediction (e.g., probability of a disease being present) or throws an `InvalidImageException` if the image is invalid.

8.7 Local Functions

- **loadModel()**: Loads the pre-trained model from disk or cloud storage using `torchxrayvision`'s functionality.
- **evaluateModel()**: Evaluates the model's performance with a test dataset to calculate metrics like accuracy and sensitivity.
- **preprocessImage()**: Preprocesses incoming X-ray image data to fit the model's input requirements (e.g., resizing, normalization).
- **postprocessPrediction()**: Processes the raw output from the model (e.g., probabilities) into a human-readable format (e.g., disease labels).

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

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.

9 Appendix

[Extra information if required —SS]

Appendix — Reflection

[Not required for CAS 741 projects —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)