Module Interface Specification for Room8

Team 19
Mohammed Abed
Maged Armanios
Jinal Kasturiarachchi
Jane Klavir
Harshil Patel

January 17, 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 **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. 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 Room8.

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 Room8 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, Room8 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 [Module Name —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

6.1 Module

[Short name for the module —SS]

6.2 Uses

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

6.4 Semantics

6.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

6.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

6.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

6.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

6.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

7 MIS of Sensor Reading Module

7.1 Module

M1: Sensor Reading Module.

7.2 Uses

Used to gather data on user presence in shared space to determine when to take before and after pictures of the shared space.

7.3 Syntax

7.3.1 Exported Constants

motionPresent: boolean

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
timeMotion	timesOfMotion	timeMotionDetected	-
detector	sensorData	motionPresent	-
insertTime	-	-	-

7.4 Semantics

7.4.1 State Variables

timesOfMotion: List[datetime] - List of time when motion was detected by the sensor.

7.4.2 Environment Variables

sensorData: sensorDataType - Data acquired by the sensor with data type supported by sensor.

7.4.3 Access Routine Semantics

timeMotion(timesOfMotion: datetime) -> datetime:

- input: List of time stamps of when motion was detected.
- output: Last timestamp of when motion was detected.

detector(sensorData: sensorDataType) -> boolean:

- input: Data received from sensor.
- output: True of false value depending on if motion was detected.

7.4.4 Local Functions

insertTime() - Inserting time stamp into timesOfMotion variable.

8 MIS of Image Capture Module

8.1 Module

M2: Image Capture Module.

8.2 Uses

Capture image of shared space using camera system.

8.3 Syntax

8.3.1 Exported Constants

image: png

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
captureImage	-	image	-

8.4 Semantics

8.4.1 Access Routine Semantics

captureImage() -> 'png':

- output: Image taken.

8.4.2 Assumptions

- captureImage will only be called when picture is needed to be taken, that is before user starts using shared space and five minutes after user has left shared space.

9 MIS of Image Upload Module

9.1 Module

M3: Image Upload Module.

9.2 Uses

Uploads the captured image to Raspberry Pi for the cleanliness detection system to use for analysis.

9.3 Syntax

9.3.1 Exported Constants

imgUploadErrorMessage : str - Message for cause of upload error.

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
uploadImage	image	-	imgUploadErrorMessage

9.4 Semantics

9.4.1 Access Routine Semantics

uploadImage(image: png):

- input: Image taken from camera.
- exception:
  - ImageUploadInterrupted: Raised if error occurs during image upload.

9.4.2 Assumptions

- Network is stable and never causes error in image upload.
- Power source is stable and never causes error in image upload.

9.4.3 Local Functions

captureImage() -> png - Takes picture to be uploaded.

10 MIS of Request Listener Module

10.1 Module

M7: Request Listener Module.

10.2 Uses

Exposes cleanliness detector to camera and image by making it an application programming interface.

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
getImages	-	images	-

10.4 Semantics

10.4.1 Access Routine Semantics

getImages() -> List[png]:

- output: Two most recent images where one is the before and other is the after state of shared space after user is finished.

11 MIS of PreprocessingModule

11.1 Module

PreprocessingModule

11.2 Uses

The Preprocessing Module is used for preparing raw images (i.e. from the Image Upload module) to be submitted to subsequent modules (i.e. Object Detection and Scoring). A series of transformations is applied to the image.

11.3 Syntax

11.3.1 Exported Constants

- SUPPORTED_FORMATS: List[str] = [”JPEG”, ”PNG”]

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
process_image	image: Image	Image	-
apply_transformations	image: Image	Image	-

11.4 Semantics

11.4.1 State Variables

11.4.2 Environment Variables

- ImageUploadModule: Module from which input image is recieved

11.4.3 Assumptions

- The input image is in a valid format supported by the module (e.g. JPEG, PNG)
- Network connectivity is stable for receiving images through the cloud network

11.4.4 Access Routine Semantics

process_image(image: Image):

- transition: Applies a sequence of transformations to the input image
- output: Returns the transformed image ready for subsequent object detection
- exception:

  - InvalidImageFormatException: Raised if the input image format is unsupported
  - TransformationError: Raised if an error occurs during transformations

process_image(image: Image):

- transition: Applies pixel map transformations (e.g. resizing, cropping, filtering) to the input image
- output: Returns the transformed image
- exception:
  - TransformationError: Raised if an error occurs during transformations

11.4.5 Local Functions

- validate_image_format(image: Image) → bool: Checks if the input image format is supported
- resize_image(image: Image, dimensions: Tuple[int, int]) → Image: Resizes the image to the specified dimensions
- apply_filter(image: Image, filter_type: str) → Image: Applies a specified filter (e.g., Gaussian blur, edge detection) to the image
- normalize_image(image: Image) → Image: Normalizes pixel values for consistency

12 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.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

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)