Module Interface Specification for Software Engineering

Team 21, Visionaries
Angela Zeng
Ann Shi
Ibrahim Sahi
Manan Sharma
Stanley Chen

November 13, 2025

1 Revision History

Date	Version	Notes
Nov 13	1.0	Version 1 completed

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/mansha71/CapstoneProject/tree/main/docs/SRS

Contents

3 Introduction

The following document details the Module Interface Specifications for Visionaries, a system designed to analyze and visualize student engagement during lectures using eye-tracking technology. Each module is described in terms of its purpose, syntax, semantics, and interactions with other modules.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/mansha71/CapstoneProject.

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. (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 Software Engineering.

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 Software Engineering 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, Software Engineering 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	Pupil Labs Neon Glasses, Central Camera, Instructor Laptop/Desktop
Behaviour-Hiding	Data Ingestion Module Real-Time Streaming Module Data Preprocessing Module Privacy & Infrastructure Module Engagement Analytics Module Correlation & Visual Analysis Module Dashboard Visualization Module Reporting Module
Software Decision	Data Storage and API Layer Authentication and Session Management Visualization Frameworks and Libraries

Table 1: Module Hierarchy for Visionaries System

6 MIS of Data Ingestion Module

6.1 Module

This module is responsible for collecting eye-tracking and video data from multiple eye-tracking devices and also the central camera. It makes sure that the raw data streams are captured, synchronized, and stored in a structured format. Making it suitable for further processing. It also collects instructor audio recordings, slide content from the lecture, and pre-/post-lecture questionnaire responses.

6.2 Uses

Real-Time Streaming Module, Data Preprocessing Module, Privacy & Infrastructure Module

6.3 Syntax

6.3.1 Exported Constants

None.

6.3.2 Exported Access Programs

Name	Input	Output	Exceptions
collectData	deviceID, streamConfig	rawData	ConnectionError
collectAudio	deviceID	audioData	ConnectionError
collectSlides	slideSource	slideData	IOError
collectQuestionnaire	formInput	questionnaireData	FormatError
syncTimestamps	dataStreams	syncedData	SyncError
storeRawData	syncedData	status	IOError

Table 2: Exported Access Programs for Data Ingestion Module

6.4 Semantics

6.4.1 State Variables

storedData: list of captured data streams currently saved in local or cloud storage.

6.4.2 Environment Variables

Connection to Pupil Labs Neon devices, the central camera feed, audio recording devices, slide file sources, and questionnaire input sources.

6.4.3 Assumptions

The eye-tracking devices are properly connected and the network latency is within acceptable limits for real-time data transfer.

6.4.4 Access Routine Semantics

collectData():

- transition: Starts capturing data streams from connected devices.
- output: Returns the raw data collected.
- exception: Raises ConnectionError if device is unavailable.

collectAudio():

- transition: Captures raw instructor audio from the recording device.
- output: Returns raw audio data.
- exception: Raises ConnectionError if the audio device is unavailable.

collectSlides():

- transition: Retrieves slide files or slide metadata from the specified source.
- output: Returns slide data.
- exception: Raises IOError if slide files cannot be accessed.

collectQuestionnaire():

- transition: Collects pre- or post-lecture questionnaire responses.
- output: Returns questionnaire data.
- exception: Raises FormatError for incomplete or malformed questionnaire submissions.

syncTimestamps():

- transition: Aligns timestamps between multiple devices using NTP or local clock sync.
- output: Returns synchronized data streams.
- exception: Raises SyncError if timestamp alignment fails.

storeRawData():

- transition: Saves synchronized gaze, video, audio, slide, and questionnaire data to secure storage.
- output: Returns success status.
- exception: Raises IOError if data cannot be written.

6.4.5 Local Functions

validateStream(deviceID), checkIntegrity(dataChunk)

7 MIS of Real-Time Streaming Module

7.1 Module

This module handles the continuous transmission of eye-tracking and video data to the dashboard for the live visualization. It ensures low-latency data flow and manages temporary buffering to keep the stream stable.

7.2 Uses

Data Ingestion Module, Data Preprocessing Module, Dashboard Visualization Module

7.3 Syntax

7.3.1 Exported Constants

None.

7.3.2 Exported Access Programs

Name	Input	Output	Exceptions
initializeStream	configSettings	streamSession	ConnectionError
transmitData	processedData	status	StreamError
terminateStream	sessionID	status	None

Table 3: Exported Access Programs for Real-Time Streaming Module

7.4 Semantics

7.4.1 State Variables

activeStream: maintains the current streaming session status.

7.4.2 Environment Variables

Network socket for dashboard communication and connected video devices.

7.4.3 Assumptions

Stable network connection and active dashboard session are available.

7.4.4 Access Routine Semantics

initializeStream():

- transition: Opens a new connection to the dashboard and starts transmitting.
- output: Returns a session object.
- exception: Raises ConnectionError if the network fails.

transmitData():

- transition: Sends processed gaze and video frames in real time.
- output: Returns success confirmation.
- exception: Raises StreamError if data loss occurs.

terminateStream():

- transition: Ends the streaming session and clears buffers.
- output: Returns final session status.
- exception: None.

7.4.5 Local Functions

bufferData(), reconnectStream()

8 MIS of Data Preprocessing Module

8.1 Module

This module prepares the captured data for analysis by filtering noise, normalizing the coordinates, and structuring the gaze and video data into useable formats. It also cleans questionnaire responses by removing empty, duplicate, or invalid entries before analysis.

8.2 Uses

Data Ingestion Module, Real-Time Streaming Module, Privacy & Infrastructure Module

8.3 Syntax

8.3.1 Exported Constants

None.

8.3.2 Exported Access Programs

Name	**Input**	**Output**	**Exceptions**
filterNoise	rawData	cleanData	DataError
normalizeCoordinates	cleanData	normalizedData	MathError
formatForAnalysis	normalizedData	structuredData	FormatError

Table 4: Exported Access Programs for Data Preprocessing Module

8.4 Semantics

8.4.1 State Variables

None. Operates statelessly on provided input data.

8.4.2 Environment Variables

None.

8.4.3 Assumptions

Input data follows the expected format and has valid timestamps.

8.4.4 Access Routine Semantics

filterNoise():

- transition: Applies smoothing or low-pass filters to reduce jitter.
- output: Returns cleaned gaze and video data.
- exception: Raises DataError for corrupted or incomplete data.

normalizeCoordinates():

- transition: Converts device-specific gaze coordinates to a shared reference frame.
- output: Returns normalized gaze data.
- exception: Raises MathError if coordinate mapping fails.

formatForAnalysis():

- transition: Structures data into a consistent schema for downstream analytics.
- output: Returns data in the standardized format.
- exception: Raises FormatError for invalid schema generation.

cleanQuestionnaireResponses():

- transition: Removes empty responses, malformed entries, and invalid questionnaire submissions.
- output: Returns cleaned questionnaire data ready for engagement analytics.
- exception: Raises DataError if response data cannot be validated.

8.4.5 Local Functions

interpolateMissingFrames(), alignStreams(), cleanQuestionnaireResponses()

9 MIS of Privacy and Infrastructure Module

9.1 Module

This module ensures secure data handling and stability of the system. It manages anonymization, encryption, and access control. It also handles maintaining infrastructure-level configurations and compliance with privacy policies. This includes securing instructor audio recordings and questionnaire response data through encryption and anonymization.

9.2 Uses

Data Ingestion Module, Data Preprocessing Module, Real-Time Streaming Module

9.3 Syntax

9.3.1 Exported Constants

None.

9.3.2 Exported Access Programs

Name	**Input**	**Output**	**Exceptions**
encryptData	plainData	encryptedData	EncryptionError
anonymizeParticipant	userData	anonymizedData	PrivacyError
validateAccess	userRole, resource	permissionStatus	AuthError
monitorInfrastructure	metrics	systemReport	None

Table 5: Exported Access Programs for Privacy and Infrastructure Module

9.4 Semantics

9.4.1 State Variables

accessLog: records of authenticated access attempts and actions performed.

9.4.2 Environment Variables

System environment variables for encryption keys, network configurations, and role-based access rules.

9.4.3 Assumptions

All data passing through this module adheres to encryption and anonymization standards under PIPEDA.

Audio data and questionnaire responses must also follow the system's encryption and anonymization requirements.

9.4.4 Access Routine Semantics

encryptData():

- transition: Applies encryption to sensitive data before storage or transmission.
- output: Returns encrypted data.
- exception: Raises EncryptionError if encryption fails.

anonymizeParticipant():

- transition: Removes personally identifiable information from datasets.
- output: Returns anonymized data.
- exception: Raises PrivacyError if anonymization is incomplete.

validateAccess():

- transition: Checks user permissions based on their assigned role.
- output: Returns access approval or denial.
- exception: Raises AuthError for invalid credentials.

monitorInfrastructure():

- transition: Tracks system metrics and logs anomalies for administrative review.
- output: Returns current system status.
- exception: None.

9.4.5 Local Functions

generateKey(), rotateLogs(), alertAdmin()

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.

Appendix — Reflection

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?

   Stanley:

   Manan: The team was able to work well together and we were able to split the work effectively.

   Angela: We were able to meet with the supervisors this week and get a good grasp on what they're looking for in terms of the designing of our modules, dashboard, and eventually, POC.

   Ann:

   Ibrahim:

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

   Stanley:

   Manan: The main pain point we experienced was coordinating with our supervisors and coming up with a list of modules we though worked well with out system.

   Angela: While we were able to derive a sufficient list of modules for our system, it did take a lot of back-and-forth, as well as brainstorming to come up with the modules, the purpose for them, etc.

   Ann:

   Ibrahim:

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?

Stanley:

Manan: Decisions regarding the modules were made after speaking with our TA and supervisor to ensure that we were on the right track. We were suggested to add more modules to better break down the system and also focus on the processing aspect of the system.

Angela: We were able to speak with both supervisors, as well as another pHD student who has experience with attention, to infer our design decisions. It really helped as well that the supervisors are not stranger's to teaching, so they're sort of the primary audience we'd be creating our modules for.

Ann:

Ibrahim:

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?

   Stanley:

   Manan: Nothing significantly changed for over previous documents as we had a solid foundation from our SRS and other documents.

   Angela: There isn't really anything that needs to be changed stemming just from the design doc. However, that will likely change based on feedback from supervisors, TA, and peer reviews.

   Ann:

   Ibrahim:

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)

   Stanley:

   Manan: We are limited by the availability of eye-tracking devices and the accuracy of those devices. Being able to have these devices on hand at all times would make it easier to plan a system that works well with the hardware.

   Angela: We don't have free reign access over using the eye-tracking devices, so we'll have to plan with the supervisors accordingly to use them.

   Ann:

   Ibrahim:

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)

   Stanley:

Manan: We considered breaking down the modules further but we felt that the current breakdown was sufficient to cover all aspects of the system without overcomplicating it. The tradeoff with having too many modules is that it can make the system harder to manage and understand.

Angela: We briefly considered both a more monolithic design and a more fine-grained module breakdown, but chose the documented design because it provides clearer information hiding and better aligns with the SRS without adding unnecessary complexity.

Ann:

Ibrahim: