

Module Interface Specification for Software Engineering

Team #22, TeleHealth Insights

Mitchell Weingust

Parisha Nizam

Promish Kandel

Jasmine Sun-Hu

January 15, 2025

1 Revision History

Date	Version	Member(s)	Notes
1/13/2025	1.0	Promish	Added MIS Decomposition for Real-TimeFeedback, Auth, ReportGeneration, Result Storage
1/14/2025	1.1	Jasmine	Added MIS Decomposition for Question Bank, English, Mandarin, Matching, Repetition
1/15/2025	1.2	Mitchell	Added MIS Decomposition for Parent GUI, Clinician GUI, API Gateway and App Controller
1/15/2025	1.3	Parisha	Added MIS Decomposition for Logging, Media, Video and Audio Processing

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at <https://github.com/parishanizam/TeleHealth/blob/main/docs/SRS-Volere/SRS.pdf>

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	2
6	MIS of Clinician GUI	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Constants	3
6.3.2	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Environment Variables	3
6.4.3	Assumptions	3
6.4.4	Access Routine Semantics	3
6.4.5	Local Functions	5
7	MIS of Parent GUI	6
7.1	Module	6
7.2	Uses	6
7.3	Syntax	6
7.3.1	Exported Constants	6
7.3.2	Exported Access Programs	6
7.4	Semantics	6
7.4.1	State Variables	6
7.4.2	Environment Variables	6
7.4.3	Assumptions	6
7.4.4	Access Routine Semantics	7
7.4.5	Local Functions	8
8	MIS of App Controller	9
8.1	Module	9
8.2	Uses	9
8.3	Syntax	9
8.3.1	Exported Constants	9
8.3.2	Exported Access Programs	9

8.4	Semantics	9
8.4.1	State Variables	9
8.4.2	Environment Variables	9
8.4.3	Assumptions	9
8.4.4	Access Routine Semantics	9
8.4.5	Local Functions	10
9	MIS of API Gateway	11
9.1	Module	11
9.2	Uses	11
9.3	Syntax	11
9.3.1	Exported Constants	11
9.3.2	Exported Access Programs	11
9.4	Semantics	11
9.4.1	State Variables	11
9.4.2	Environment Variables	12
9.4.3	Assumptions	12
9.4.4	Access Routine Semantics	12
9.4.5	Local Functions	13
10	MIS of Authentication Module	13
10.1	Module	13
10.2	Uses	13
10.3	Syntax	13
10.3.1	Exported Constants	13
10.3.2	Exported Access Programs	13
10.4	Semantics	14
10.4.1	State Variables	14
10.4.2	Environment Variables	14
10.4.3	Assumptions	14
10.4.4	Access Routine Semantics	14
10.4.5	Local Functions	15
11	MIS of Result Storage Module	15
11.1	Module	15
11.2	Uses	15
11.3	Syntax	15
11.3.1	Exported Constants	15
11.3.2	Exported Access Programs	15
11.4	Semantics	15
11.4.1	State Variables	15
11.4.2	Environment Variables	15
11.4.3	Assumptions	15

11.4.4	Access Routine Semantics	16
11.4.5	Local Functions	16
12	MIS of Media Processing Module	16
12.1	Module	16
12.2	Uses	16
12.3	Syntax	16
12.3.1	Exported Constants	16
12.3.2	Exported Access Programs	17
12.4	Semantics	17
12.4.1	State Variables	17
12.4.2	Environment Variables	17
12.4.3	Assumptions	17
12.4.4	Access Routine Semantics	17
12.4.5	Local Functions	17
13	MIS of Logging Module	18
13.1	Module	18
13.2	Uses	18
13.3	Syntax	18
13.3.1	Exported Constants	18
13.3.2	Exported Access Programs	18
13.4	Semantics	18
13.4.1	State Variables	18
13.4.2	Environment Variables	18
13.4.3	Assumptions	18
13.4.4	Types of Logs	19
13.4.5	Access Routine Semantics	19
13.4.6	Local Functions	19
14	MIS of Question Bank Module	20
14.1	Module	20
14.2	Uses	20
14.3	Syntax	20
14.3.1	Exported Constants	20
14.3.2	Exported Access Programs	20
14.4	Semantics	20
14.4.1	State Variables	20
14.4.2	Environment Variables	20
14.4.3	Assumptions	20
14.4.4	Access Routine Semantics	20
14.4.5	Local Functions	21

15 MIS of Real-Time Feedback Module	21
15.1 Module	21
15.2 Uses	21
15.3 Syntax	21
15.3.1 Exported Constants	21
15.3.2 Exported Access Programs	21
15.4 Semantics	22
15.4.1 State Variables	22
15.4.2 Environment Variables	22
15.4.3 Assumptions	22
15.4.4 Access Routine Semantics	22
15.4.5 Local Functions	22
16 MIS of Report Generation Module	23
16.1 Module	23
16.2 Uses	23
16.3 Syntax	23
16.3.1 Exported Constants	23
16.3.2 Exported Access Programs	23
16.4 Semantics	23
16.4.1 State Variables	23
16.4.2 Environment Variables	23
16.4.3 Assumptions	23
16.4.4 Access Routine Semantics	24
16.4.5 Local Functions	24
17 MIS of Video Processing Module	24
17.1 Module	24
17.2 Uses	24
17.3 Syntax	24
17.3.1 Exported Constants	24
17.3.2 Exported Access Programs	25
17.4 Semantics	25
17.4.1 State Variables	25
17.4.2 Environment Variables	25
17.4.3 Assumptions	25
17.4.4 Access Routine Semantics	25
17.4.5 Local Functions	25
18 MIS of Audio Processing Module	25
18.1 Module	25
18.2 Uses	26
18.3 Syntax	26

18.3.1	Exported Constants	26
18.3.2	Exported Access Programs	26
18.4	Semantics	26
18.4.1	State Variables	26
18.4.2	Environment Variables	26
18.4.3	Assumptions	26
18.4.4	Access Routine Semantics	26
18.4.5	Local Functions	27
19	MIS of English Question Bank Module	27
19.1	Module	27
19.2	Uses	27
19.3	Syntax	27
19.3.1	Exported Constants	27
19.3.2	Exported Access Programs	27
19.4	Semantics	27
19.4.1	State Variables	27
19.4.2	Environment Variables	27
19.4.3	Assumptions	27
19.4.4	Access Routine Semantics	28
19.4.5	Local Functions	28
20	MIS of Mandarin Question Bank Module	28
20.1	Module	28
20.2	Uses	28
20.3	Syntax	28
20.3.1	Exported Constants	28
20.3.2	Exported Access Programs	28
20.4	Semantics	29
20.4.1	State Variables	29
20.4.2	Environment Variables	29
20.4.3	Assumptions	29
20.4.4	Access Routine Semantics	29
20.4.5	Local Functions	29
21	MIS of Matching Question Bank Module	29
21.1	Module	29
21.2	Uses	29
21.3	Syntax	30
21.3.1	Exported Constants	30
21.3.2	Exported Access Programs	30
21.4	Semantics	30
21.4.1	State Variables	30

21.4.2	Environment Variables	30
21.4.3	Assumptions	30
21.4.4	Access Routine Semantics	30
21.4.5	Local Functions	31
22	MIS of Repetition Question Bank Module	31
22.1	Module	31
22.2	Uses	31
22.3	Syntax	31
22.3.1	Exported Constants	31
22.3.2	Exported Access Programs	31
22.4	Semantics	31
22.4.1	State Variables	31
22.4.2	Environment Variables	31
22.4.3	Assumptions	31
22.4.4	Access Routine Semantics	32
22.4.5	Local Functions	32

3 Introduction

The following document details the Module Interface Specifications for TeleHealth Insights. It is an at-home bilingual speech assessment system with video and audio analysis features. The system is designed to provide clear guidance to parents when administering the assessment to their children, in an environment where speech-language pathologists (SLPs) are unavailable. By streamlining the assessment process, the project aims to provide a convenient and comprehensive solution for SLPs to assess and support their patients' speech and language development remotely.

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

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 $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

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 $(-\infty, \infty)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$

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	N/A
Behaviour-Hiding	Clinician GUI Parent GUI Authentication Module Result Storage Module Real-Time Feedback Module Report Generation Module Media Processing Module Video Processing Module Audio Processing Module Logging Module Question Bank Module Mandarin Question Bank English Question Bank Repetition Question Bank Module Matching Question Bacnk Module
Software Decision	APP Controller API Gateway

Table 1: Module Hierarchy

6 MIS of Clinician GUI

6.1 Module

clinicianGUI

6.2 Uses

- ApplicationController

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayLoginPage	-	-	-
displayClinicianDashboardPage	-	-	-
displayAddClientPage	-	-	-
displayPatientOverviewPage	-	-	-
displayPatientAssessmentResultsAnalysisPage	-	-	-
displayBiasReviewPage	-	-	-
displayFlagBiasPage	-	-	-

6.4 Semantics

6.4.1 State Variables

N/A

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

N/A

6.4.4 Access Routine Semantics

displayLoginPage():

- transition: Navigates to and displays the clinician login page for the system.

- output: N/A
- exception: N/A

displayClinicianDashboardPage():

- transition: Navigates to and displays the clinician dashboard page for accessing a clinician's list of clients.
- output: N/A
- exception: N/A

displayAddClientPage():

- transition: Navigates to and displays the add client page for adding a new client to a clinician's list.
- output: N/A
- exception: N/A

displayPatientOverviewPage():

- transition: Navigates to and displays the patient overview page for accessing all of the patient's previous assessments.
- output: N/A
- exception: N/A

displayPatientAssessmentResultsAnalysisPage():

- transition: Navigates to and displays the patient assessment results analysis page for accessing all of the results of a particular assessment.
- output: N/A
- exception: N/A

displayBiasReviewPage():

- transition: Navigates to and displays the bias review page for reviewing and removing bias from a particular question in an assessment.
- output: N/A
- exception: N/A

displayFlagBiasPage():

- transition: Navigates to and displays the flag bias page for reviewing and flagging bias on a particular question in an assessment.
- output: N/A
- exception: N/A

6.4.5 Local Functions

N/A

7 MIS of Parent GUI

7.1 Module

parentGUI

7.2 Uses

- ApplicationController

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayLoginPage	-	-	-
displayCreateAccountPage	-	-	-
displayHomePage	-	-	-
displayAssessmentSelectionPage	-	-	-
displayParentChecklistPage	-	-	-
displayInputCheckPage	microphoneInput, cameraInput	-	-
displayAssessmentInstructionsPage	-	-	-
displayAssessmentQuestionsPage	-	-	-
displayCompletionPage	-	-	-

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

- microphoneInput
- cameraInput

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

displayLoginPage():

- transition: Navigates to and displays the parent login page for the system.
- output: N/A
- exception: N/A

displayCreateAccountPage():

- transition: Navigates to and displays the parent account creation page for creating a new account for the system.
- output: N/A
- exception: N/A

displayHomePage():

- transition: Navigates to and displays the homepage for a parent account, with the ability to start a new assessment.
- output: N/A
- exception: N/A

displayAssessmentSelectionPage():

- transition: Navigates to and displays the assessment selection page for selecting the type of assessment for the user.
- output: N/A
- exception: N/A

displayParentChecklistPage():

- transition: Navigates to and displays the parent checklist page for informing parents about the requirements of the assessment.
- output: N/A
- exception: N/A

displayInputCheckPage():

- transition: Navigates to and displays the input check page for testing input devices.
- output: N/A

- exception: N/A

displayAssessmentInstructionsPage():

- transition: Navigates to and displays the assessment instructions page for the child to read and engage with to learn how to interact with the assessment interface.
- output: N/A
- exception: N/A

displayAssessmentQuestionsPage():

- transition: Navigates to and displays the assessment questions page for displaying a question and its corresponding answers for the user to select.
- output: N/A
- exception: N/A

displayCompletionPage():

- transition: Navigates to and displays the completion page to confirm to the user that the assessment is complete and their results have been saved.
- output: N/A
- exception: N/A

7.4.5 Local Functions

N/A

8 MIS of App Controller

8.1 Module

AppController

8.2 Uses

- APIGateway

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
accessAPIGateway	-	-	-

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

accessAPIGateway():

- transition: Controller accesses the API Gateway.
- output: N/A
- exception: N/A

8.4.5 Local Functions

N/A

9 MIS of API Gateway

9.1 Module

APIGateway

9.2 Uses

- Authentication
- ResultStorage
- MediaProcessing
- Logging
- QuestionBank
- RealTimeFeedback
- ReportGeneration

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
accessAuthentication	-	-	-
accessResultStorage	-	-	-
accessMediaProcessing	-	-	-
accessLogging	-	-	-
accessQuestionBank	-	-	-
accessRealTimeFeedback	-	-	-
accessReportGeneration	-	-	-

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

accessAuthentication():

- transition: Controller accesses the authentication module.
- output: N/A
- exception: N/A

accessResultStorage():

- transition: Controller accesses the results storage module.
- output: N/A
- exception: N/A

accessMediaProcessing():

- transition: Controller accesses the media processing module.
- output: N/A
- exception: N/A

accessLogging():

- transition: Controller accesses the logging module.
- output: N/A
- exception: N/A

accessQuestionBank():

- transition: Controller accesses the question bank module.
- output: N/A
- exception: N/A

accessRealTimeFeedback():

- transition: Controller accesses the real time feedback module.
- output: N/A
- exception: N/A

accessReportGeneration():

- transition: Controller accesses the report generation module.
- output: N/A
- exception: N/A

9.4.5 Local Functions

N/A

10 MIS of Authentication Module

10.1 Module

AuthenticationModule

10.2 Uses

N/A

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
signup	username: str, email: str, password: str, role: str	status: bool	UserAlreadyExistsException
login	username: str, password: str	sessionToken: str	InvalidCredentialsException
logout	sessionToken: str	status: bool	InvalidSessionException

10.4 Semantics

10.4.1 State Variables

- `userList`: `Set(User)` - maintains a set of all registered users.
- `activeSessions`: `Map(sessionToken, User)` - tracks active user sessions.

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

- Usernames and emails are unique.
- Sessions are managed using session tokens.
- Role can be one of `['parent', 'clinician', 'admin']`.
- Clinicians have given user their login token

10.4.4 Access Routine Semantics

`signup()`:

- `transition`: Adds a new user to `'userList'` if the username and email are unique.
- `output`: Returns `'true'` if the user is successfully created, otherwise throws `'UserAlreadyExistsException'`.
- `exception`: Throws `'UserAlreadyExistsException'` if the username or email already exists.

`login()`:

- `transition`: Adds a new session to `'activeSessions'` if the credentials are valid.
- `output`: Returns a `'sessionToken'` for the logged-in user.
- `exception`: Throws `'InvalidCredentialsException'` if the username or password is incorrect.

`logout()`:

- `transition`: Removes the `'sessionToken'` from `'activeSessions'`.
- `output`: Returns `'true'` if the session is successfully ended.
- `exception`: Throws `'InvalidSessionException'` if the session token does not exist.

10.4.5 Local Functions

N/A

11 MIS of Result Storage Module

11.1 Module

ResultStorageModule

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
storeResult	data: JSON object	status: bool	StorageException
retrieveResult	resultID: str	data: JSON object	NotFoundException
deleteResult	resultID: str	status: bool	NotFoundException

11.4 Semantics

11.4.1 State Variables

- resultStorage: Map(resultID, JSON object) - stores all processed results with unique IDs.

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

- Each result is assigned a unique resultID.
- Results are stored as JSON objects for flexibility.
- Data is stored in MongoDB or an equivalent NoSQL database.

11.4.4 Access Routine Semantics

storeResult():

- transition: Adds the ‘data’ to ‘resultStorage’ with a unique ‘resultID’.
- output: Returns ‘true’ if the result is successfully stored.
- exception: Throws ‘StorageException’ if there is an issue storing the data.

retrieveResult():

- transition: None
- output: Returns the result associated with the ‘resultID’.
- exception: Throws ‘NotFoundException’ if the ‘resultID’ does not exist.

deleteResult():

- transition: Removes the result associated with the ‘resultID’ from ‘resultStorage’.
- output: Returns ‘true’ if the result is successfully deleted.
- exception: Throws ‘NotFoundException’ if the ‘resultID’ does not exist.

11.4.5 Local Functions

N/A

12 MIS of Media Processing Module

12.1 Module

MediaProcessingModule

12.2 Uses

VideoProcessingModule, AudioProcessingModule

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
processMedia	mediaFile: str, mediaType: str, assessmentID: str	report: MediaAnalysisReport	MediaProcessingException

12.4 Semantics

12.4.1 State Variables

- processedMedia: Map(assessmentID, MediaAnalysisReport) - stores combined results from video and audio analysis.

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

- mediaType specifies whether the file is video or audio (e.g., "video", "audio").
- Delegates processing to VideoProcessingModule or AudioProcessingModule based on mediaType.

12.4.4 Access Routine Semantics

processMedia():

- transition:
 - If mediaType is "video", calls processVideo from VideoProcessingModule.
 - If mediaType is "audio", calls processAudio from AudioProcessingModule.
 - Combines results into a single MediaAnalysisReport per assessment completed.
- output: Returns a MediaAnalysisReport with details from video and audio analyses.
- exception: Throws MediaProcessingException if the file cannot be processed or delegated.

12.4.5 Local Functions

N/A

13 MIS of Logging Module

13.1 Module

LoggingModule

13.2 Uses

N/A

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
logEvent	eventType: str, message: str	status: bool	LoggingException
fetchLogs	logType: str, timeRange: TimeRange	logData: List(LogEntry)	LogFetchException
clearLogs	logType: str, timeRange: TimeRange	status: bool	LogClearException

13.4 Semantics

13.4.1 State Variables

- logs: Map(logType, List(LogEntry)) - stores all logs categorized by type.

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

- Logs are categorized by their type and timestamp for easy retrieval.
- Each log entry includes metadata such as the timestamp, severity, and module of origin.

13.4.4 Types of Logs

Event Logs

Tracks application activities, such as user logins, file uploads, and media processing events. Example: "User 'parent1' logged in successfully at 14:32:00."

Error Logs
Captures unexpected behaviors, exceptions, or failures in the application. Example: "VideoProcessingException: Unable to analyze video file 'session123.mp4' due to corrupted data."

Audit Logs
Records critical changes and actions for accountability, such as user role changes or log clearances. Example: "Admin user 'clinician1' updated parent access rights at 16:45:00."

Performance Logs
Monitors application performance metrics like response times, memory usage, and processing durations. Example: "Media processing for 'session456' completed in 3.2 seconds with 200MB memory usage."

Debug Logs
Includes detailed information for troubleshooting during development or maintenance. Example: "Entering function 'processVideo' with input file 'session789.mp4'."

Security Logs
Tracks security-related events such as failed logins, access violations, or token expirations. Example: "Security alert: Failed login attempt for user 'parent2' at 18:12:30."

13.4.5 Access Routine Semantics

logEvent():

- transition: Adds a new log entry to the logs map under the appropriate eventType.
- output: Returns true if the log is successfully recorded.
- exception: Throws LoggingException if the log entry cannot be added.

fetchLogs():

- transition: Retrieves all logs of the specified logType within the provided timeRange.
- output: Returns a list of LogEntry objects matching the criteria.
- exception: Throws LogFetchException if no logs are found or retrieval fails.

clearLogs():

- transition: Removes all logs of the specified logType within the provided timeRange.
- output: Returns true if logs are successfully cleared.
- exception: Throws LogClearException if logs cannot be cleared.

13.4.6 Local Functions

N/A

14 MIS of Question Bank Module

14.1 Module

QuestionBankModule

14.2 Uses

EnglishQuestionBankModule, MandarinQuestionBankModule

14.3 Syntax

14.3.1 Exported Constants

N/A

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
selectQuestionBank	language: str	questionBank: ADT	InvalidLanguageException
retrieveQuestion	language: str, questionID: str	question: JSON object	NotFoundException

14.4 Semantics

14.4.1 State Variables

- activeQuestionBanks: Map(language, ADT) - maps language to its respective question bank module.

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

- Supported languages include English and Mandarin.
- Each question bank module is preloaded with language-specific questions.

14.4.4 Access Routine Semantics

selectQuestionBank():

- transition: Selects the question bank module corresponding to the input language.

- output: Returns the selected question bank module.
- exception: Throws `InvalidLanguageException` if the input language is not supported.

`retrieveQuestion()`:

- transition: None
- output: Retrieves the question from the appropriate question bank module.
- exception: Throws `NotFoundException` if the questionID does not exist in the selected module.

14.4.5 Local Functions

N/A

15 MIS of Real-Time Feedback Module

15.1 Module

`RealTimeFeedbackModule`

15.2 Uses

- Media Processing Module

15.3 Syntax

15.3.1 Exported Constants

N/A

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
<code>provideFeedback</code>	<code>sessionID</code> : str, <code>liveFeed</code> : media stream	<code>feedback</code> : JSON object	<code>FeedbackException</code>
<code>logFeedback</code>	<code>sessionID</code> : str, <code>feedback</code> : JSON object	<code>status</code> : bool	<code>LoggingException</code>

15.4 Semantics

15.4.1 State Variables

- `feedbackLogs`: `Map(sessionID, List(feedback))` - stores real-time feedback for sessions.

15.4.2 Environment Variables

N/A

15.4.3 Assumptions

- The module receives a continuous media stream (audio or video) during a session.
- Feedback is generated by analyzing live media streams using the Media Processing Module.
- Feedback is stored for each session to provide session summaries if needed.
- The module operates within acceptable latency constraints to ensure real-time performance.

15.4.4 Access Routine Semantics

`provideFeedback()`:

- `transition`: Generates feedback from the `'liveFeed'` media stream and optionally logs it in `'feedbackLogs'`.
- `output`: Returns actionable feedback in a structured JSON format (e.g., `"Adjust microphone"`, `"Increase lighting"`).
- `exception`: Throws `'FeedbackException'` if there is an issue processing the live feed.

`logFeedback()`:

- `transition`: Adds the provided `'feedback'` to `'feedbackLogs'` for the corresponding `'sessionID'`.
- `output`: Returns `'true'` if the feedback is successfully logged.
- `exception`: Throws `'LoggingException'` if there is an error while logging the feedback.

15.4.5 Local Functions

N/A

16 MIS of Report Generation Module

16.1 Module

ReportGenerationModule

16.2 Uses

- Result Storage Module
- Media Processing Module
- Question Bank Module

16.3 Syntax

16.3.1 Exported Constants

N/A

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
generateReport	sessionID: str, metadata: JSON object	report: JSON	ReportGenerationException
getReport	reportID: str	report: JSON	NotFoundException

16.4 Semantics

16.4.1 State Variables

- reportStorage: Map(reportID, Report) - stores all generated reports with unique IDs.

16.4.2 Environment Variables

N/A

16.4.3 Assumptions

- Each report is assigned a unique reportID.
- Reports are generated using data fetched from the Result Storage Module and other sources like the Question Bank Module.
- Reports can be retrieved in JSON format then converted to PDF format

- The clinician requesting the report has access to the session data.

16.4.4 Access Routine Semantics

`generateReport()`:

- **transition:** Creates a new report using the ‘sessionID’ and ‘metadata’, and stores it in ‘reportStorage’.
- **output:** Returns the generated report in the specified format (JSON or PDF).
- **exception:** Throws ‘ReportGenerationException’ if there is an error during report generation.

`getReport()`:

- **transition:** None
- **output:** Returns the report associated with the ‘reportID’.
- **exception:** Throws ‘NotFoundException’ if the ‘reportID’ does not exist in ‘reportStorage’.

16.4.5 Local Functions

N/A

17 MIS of Video Processing Module

17.1 Module

VideoProcessingModule

17.2 Uses

N/A MediaProcessingModule

17.3 Syntax

17.3.1 Exported Constants

N/A

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
processVideo	videoFile: str, assessmentID: str	report: VideoAnalysisReport	VideoProcessingException

17.4 Semantics

17.4.1 State Variables

- processedVideos: Map(assessmentID, VideoAnalysisReport) - stores results of processed videos.

17.4.2 Environment Variables

N/A

17.4.3 Assumptions

- Video files are in a supported format (e.g., MP4, AVI).
- Video processing is done within a time threshold for real-time feedback.

17.4.4 Access Routine Semantics

processVideo():

- transition: Analyzes the video to identify any disturbances, bias, or cheating patterns.
- output: Returns a detailed VideoAnalysisReport containing flagged events and metrics.
- exception: Throws VideoProcessingException if the file cannot be processed or analyzed.

17.4.5 Local Functions

N/A

18 MIS of Audio Processing Module

18.1 Module

AudioProcessingModule

18.2 Uses

N/A MediaProcessingModule

18.3 Syntax

18.3.1 Exported Constants

N/A

18.3.2 Exported Access Programs

Name	In	Out	Exceptions
processAudio	audioFile: str, assessmentID: str	report: AudioAnalysisReport	AudioProcessingException

18.4 Semantics

18.4.1 State Variables

- processedAudio: Map(assessmentID, AudioAnalysisReport) - stores results of processed audio.

18.4.2 Environment Variables

N/A

18.4.3 Assumptions

- Audio files are in a supported format (e.g., WAV, MP3).
- Background noise levels are detectable and quantifiable.

18.4.4 Access Routine Semantics

processAudio():

- transition: Analyzes the audio for disturbances such as background noise or interruptions.
- output: Returns a detailed AudioAnalysisReport with flagged issues.
- exception: Throws AudioProcessingException if the file cannot be processed or analyzed.

18.4.5 Local Functions

N/A

19 MIS of English Question Bank Module

19.1 Module

EnglishQuestionBankModule

19.2 Uses

MatchingQuestionBankModule, RepetitionQuestionBankModule

19.3 Syntax

19.3.1 Exported Constants

N/A

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
getQuestion	questionID: str	question: JSON object	NotFoundException
addQuestion	question: JSON object	status: bool	StorageException

19.4 Semantics

19.4.1 State Variables

- englishQuestions: Map(questionID, JSON object) - stores English questions.

19.4.2 Environment Variables

N/A

19.4.3 Assumptions

- Questions are either matching or repetition type.
- Matching and Repetition modules are used to handle the respective types.

19.4.4 Access Routine Semantics

getQuestion():

- transition: None
- output: Returns the question corresponding to the questionID.
- exception: Throws `NotFoundException` if the questionID does not exist.

addQuestion():

- transition: Adds the input question to `englishQuestions`.
- output: Returns `true` if the question is successfully added.
- exception: Throws `StorageException` if there is an issue storing the question.

19.4.5 Local Functions

N/A

20 MIS of Mandarin Question Bank Module

20.1 Module

MandarinQuestionBankModule

20.2 Uses

MatchingQuestionBankModule, RepetitionQuestionBankModule

20.3 Syntax

20.3.1 Exported Constants

N/A

20.3.2 Exported Access Programs

Name	In	Out	Exceptions
getQuestion	questionID: str	question: JSON object	NotFoundException
addQuestion	question: JSON object	status: bool	StorageException

20.4 Semantics

20.4.1 State Variables

- `mandarinQuestions`: `Map(questionID, JSON object)` - stores Mandarin questions.

20.4.2 Environment Variables

N/A

20.4.3 Assumptions

- Questions are either matching or repetition type.
- Matching and Repetition modules are used to handle the respective types.

20.4.4 Access Routine Semantics

`getQuestion()`:

- `transition`: None
- `output`: Returns the question corresponding to the `questionID`.
- `exception`: Throws `NotFoundException` if the `questionID` does not exist.

`addQuestion()`:

- `transition`: Adds the input question to `englishQuestions`.
- `output`: Returns `true` if the question is successfully added.
- `exception`: Throws `StorageException` if there is an issue storing the question.

20.4.5 Local Functions

N/A

21 MIS of Matching Question Bank Module

21.1 Module

`MatchingQuestionBankModule`

21.2 Uses

N/A

21.3 Syntax

21.3.1 Exported Constants

N/A

21.3.2 Exported Access Programs

Name	In	Out	Exceptions
storeMatchingQuestion	question: JSON object	status: bool	StorageException
retrieveMatchingQuestion	questionID: str	question: JSON object	NotFoundException

21.4 Semantics

21.4.1 State Variables

- matchingQuestions: Map(questionID, JSON object) - stores matching questions.

21.4.2 Environment Variables

N/A

21.4.3 Assumptions

- Questions have a unique ID.
- Data is stored in a JSON format for flexibility.

21.4.4 Access Routine Semantics

storeMatchingQuestion():

- transition: Adds the question to `matchingQuestions`.
- output: Returns `true` if successfully stored.
- exception: Throws `StorageException` if there is a storage error.

retrieveMatchingQuestion():

- transition: None
- output: Returns the matching question corresponding to the questionID.
- exception: Throws `NotFoundException`

21.4.5 Local Functions

N/A

22 MIS of Repetition Question Bank Module

22.1 Module

RepetitionQuestionBankModule

22.2 Uses

N/A

22.3 Syntax

22.3.1 Exported Constants

N/A

22.3.2 Exported Access Programs

Name	In	Out	Exceptions
storeRepetitionQuestion	question: JSON object	status: bool	StorageException
retrieveRepetitionQuestion	questionID: str	question: JSON object	NotFoundException

22.4 Semantics

22.4.1 State Variables

- repetitionQuestions: Map(questionID, JSON object) - stores matching questions.

22.4.2 Environment Variables

N/A

22.4.3 Assumptions

- Questions have a unique ID.
- Data is stored in a JSON format.

22.4.4 Access Routine Semantics

storeRepetitionQuestion():

- transition: Adds the question to `repetitionQuestions`.
- output: Returns `true` if successfully stored.
- exception: Throws `StorageException` if there is a storage error.

retrieveRepetitionQuestion():

- transition: None
- output: Returns the matching question corresponding to the `questionID`.
- exception: Throws `NotFoundException`

22.4.5 Local Functions

N/A

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?
 - Promish: I think as a group we were very coordinated and had the important parts, like the module hierarchy diagram, completed before our TA meeting. This allowed us to ask Chris if our design looked good and if there was any feedback he could provide.
 - Jasmine: The team worked well to break down the deliverable into smaller section and distribute tasks so that the team could efficiently complete the deliverable. Designing the interface went fairly smoothly since there was a old incomplete version of the website to gather inspiration from, and meeting with our capstone supervisor to discuss and review our design was very useful since her research specializes in human computer interfaces.
 - Parisha: We defined who was doing what section of each MIS and MG document well, and all of our work aligned. We also met/discussed a lot to ensure we are on the same page and coordinated. We also asked questions to Chris which helped us with any confusion and continue working on track. The hierarchy diagram was made early so we could base everything off it and improve it as we went along.
 - Mitchell: The team worked well to split up the sections fairly, well ahead of the deadline to ensure that all schedules were accommodated, as we all had a very busy first week of the semester. Planning in advance allowed us to meet our deadline and complete our work to a high quality. I also think the separation of responsibilities for designing the user interfaces allowed us to learn more about the system and how the pages would flow between one another.
2. What pain points did you experience during this deliverable, and how did you resolve them?

- Promish: The hardest part of this deliverable is knowing that right after it, we have to build everything within three weeks. I found myself second-guessing between what would make a good design and what would make a feasible design. We overcame this by talking to our supervisor and setting up a good timeline.
 - Jasmine: Since the team chose a divide and conquer method for a significant portion of the deliverable, the team anticipated a pain point for ensuring consistency between members' work. To resolve this, the team made sure to communicate throughout the deliverable, discussing sections and sending examples to follow. Also, the team made sure to have a meeting with everyone's attendance to merge parts together and clean up/ make adjustments for a cohesive and consistent document.
 - Parisha: The main point was the amount of time it took to get to completing this deliverable. Although the time it took to fill in the deliverable did not take that long, we had to discuss multiples to ensure we were making the right hierarchy and module discussions. Talking to our supervisor and our clinician helped.
 - Mitchell: A pain point I experienced was learning how to properly write the MIS, as I have not seen an MIS in any of my previous courses. So the concepts of modules, and how to express them and ensure all integral information was present was a main pain point for me. I resolved this by looking at past groups' MIS' to learn more about the requirements for those documents and how to properly write 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?
- Promish: Most of the design came from prior knowledge of working in co-op as a backend and frontend developer. We also had our supervisor help with the frontend UI design, as there was a mock-up of what she wanted overall or what she didn't like.
 - Jasmine: Design decisions that stemmed from speaking to our client were interface usability decisions such as keeping the interface design simple and child-friendly, making sure navigation is as simple as possible so that it is intuitive for parents, and the assessment questions are displayed with large buttons and friendly images to be child friendly.
 - Parisha: We met with our supervisor/clinician to discuss their old design and what improvements and new features they would like to have. Much of the design decisions stemmed from these discussions and what our client wants to see. We will base off our UI design on the mock up and improve it to improve usability and design. Other design decisions like backend set up came from group members' experiences from previous co-ops.

- Mitchell: The design of the assessment pages came from speaking with Dr. Du, our supervisor's collaborator. Her team already had an early concept for the assessment interface, which allowed us to prototype a variety of interfaces to make improvements to the initial concept. The clinician dashboard did not stem from speaking with peers, as this was a brand new addition to the system, so I created a new interface and concept for how it would be interacted with to meet all requirements specified.
4. While creating the design doc, what parts of your other documents (e.g., requirements, hazard analysis, etc.), if any, needed to be changed, and why?
- Promish: Nothing needed to change as we had already discussed in-depth what we were building and had regular meetups with our supervisor. We referenced the SRS to ensure that our design met our goals and extended goals, but we didn't have to change any previous document because of our design.
 - Parisha: We referred to SRS and requirement docs to make the design documents. Not much has changed as we discussed in detail how the requirements should be set out early so minimal changes need to be made later unless absolutely necessary.
 - Mitchell: Since we frequently referenced our SRS document, not many details would need to be changed. As well, since our roles have remained consistent, our development plan should not need to change either.
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)
- Promish: I would say a big limitation is the time aspect of making the capstone. I feel like given more time, we would be able to build it like an agile team, so that features are constantly tested with users. We are also limited in terms of getting video metadata, so the accuracy of our video analysis might suffer because of that. Finally, we are also limited by our skills; there are some aspects of the design that we couldn't do because we didn't know how to and didn't have the time to figure it out.
 - Jasmine: The main limitation of our solution is development time. Other limitations include only having indirect access to user groups (bilingual children with speech difficulties), access to fluent mandarin speakers, etc.
 - Parisha: If given more time, we would be able to make a whole real time feedback system that flagged bias in realtime and would be able to train and make a model completely on our own that would be more specific and flag biases a lot more accurately. Given more time we could have planned and started work in a more agile way and be able to get a lot more done. We found with the current timeline, most is focused on documentation and design, making it difficult to plan to do more difficult features.

- Mitchell: One of the limitations of our solution is the development time, as since there was so much emphasis on the design, which took up a lot of the preparation time, the implementation time had been reduced. Given unlimited resources, I would look deeper into the machine learning models to get a more accurate bias detection, so that there would be less intervention required from the Clinicians to manually check on the bias detected.
6. Give a brief overview of other design solutions you considered. What are the benefits and trade-offs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO Explores)
- Promish: For our backend, we thought about a monolithic architecture style because it would be easier to implement, but the trade-off is that it's not as maintainable. Our supervisor stressed that maintainability is a big priority for her, so we ended up going with a microservice architecture style. We also considered the various interactions between how the clinician and the parent UI. We previously thought about the parent making an account and the clinician would just search for the parent to add them to their client list. However, we thought it was best if the clinician added the parent to their client list and then gave them a code. This way, we would reduce troll accounts and other security gaps.
 - Jasmine: Another design that was considered was having a singular question bank repository for all questions, this would lead to simpler architecture, but less organization and be inefficient to scale in the future. We went with modular question banks with dedicated submodules based on language + assessment type to balance flexibility and maintainability while also considering future expandability.
 - Parisha: Talking to our supervisors we came to the conclusion that maintainability and security were the main two focuses to improve on based off the old/mock off site. We based our design decisions on prioritizing these requirements.
 - Mitchell: Another design solution I considered was ordering the question banks first by question type, and then by language later. The benefit of this would be consistency in question types available, however, this would require that whenever a question type is selected, all languages would need to be available for that type. This would make scaling of the design somewhat difficult, as every time a new language is added, it would need questions of all types available. The design we went with allows us to add and swap languages using a strategy design pattern, so that when a user selects the language first, it will list the possible assessment types for that language, instead of selecting a question type, and realizing the language you wanted was unavailable.