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	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 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

[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 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	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 Mandarian 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

- AppController

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

- AppController

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

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

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

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

9.4.4 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]

9.4.5 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]

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.

10 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?

   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.

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.

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.

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.

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.

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.