Module Interface Specification for Software Engineering

Team #22, TeleHealth Insights
Mitchell Weingust
Parisha Nizam
Promish Kandel
Jasmine Sun-Hu

January 14, 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

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **1**

6 MIS of Question Bank Module **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 . . . 4

7 MIS of English Question Bank Module **4**
- 7.1 Module . . . 4
- 7.2 Uses . . . 4
- 7.3 Syntax . . . 4
  - 7.3.1 Exported Constants . . . 4
  - 7.3.2 Exported Access Programs . . . 4
- 7.4 Semantics . . . 4
  - 7.4.1 State Variables . . . 4
  - 7.4.2 Environment Variables . . . 5
  - 7.4.3 Assumptions . . . 5
  - 7.4.4 Access Routine Semantics . . . 5
  - 7.4.5 Local Functions . . . 5

8 MIS of Mandarin Question Bank Module **5**
- 8.1 Module . . . 5
- 8.2 Uses . . . 5
- 8.3 Syntax . . . 5
  - 8.3.1 Exported Constants . . . 5
  - 8.3.2 Exported Access Programs . . . 6

8.4 Semantics . . . 6
8.4.1 State Variables . . . 6
8.4.2 Environment Variables . . . 6
8.4.3 Assumptions . . . 6
8.4.4 Access Routine Semantics . . . 6
8.4.5 Local Functions . . . 6

9 MIS of Matching Question Bank Module **7**
9.1 Module . . . 7
9.2 Uses . . . 7
9.3 Syntax . . . 7
9.3.1 Exported Constants . . . 7
9.3.2 Exported Access Programs . . . 7
9.4 Semantics . . . 7
9.4.1 State Variables . . . 7
9.4.2 Environment Variables . . . 7
9.4.3 Assumptions . . . 7
9.4.4 Access Routine Semantics . . . 7
9.4.5 Local Functions . . . 8

10 MIS of Repetition Question Bank Module **8**
10.1 Module . . . 8
10.2 Uses . . . 8
10.3 Syntax . . . 8
10.3.1 Exported Constants . . . 8
10.3.2 Exported Access Programs . . . 8
10.4 Semantics . . . 8
10.4.1 State Variables . . . 8
10.4.2 Environment Variables . . . 9
10.4.3 Assumptions . . . 9
10.4.4 Access Routine Semantics . . . 9
10.4.5 Local Functions . . . 9

11 Appendix **11**

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 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	
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 Question Bank Module

6.1 Module

QuestionBankModule

6.2 Uses

EnglishQuestionBankModule, MandarinQuestionBankModule

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

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

6.4 Semantics

6.4.1 State Variables

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

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

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

6.4.4 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.

6.4.5 Local Functions

N/A

7 MIS of English Question Bank Module

7.1 Module

EnglishQuestionBankModule

7.2 Uses

MatchingQuestionBankModule, RepetitionQuestionBankModule

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

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

7.4 Semantics

7.4.1 State Variables

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

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

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

7.4.4 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.

7.4.5 Local Functions

N/A

8 MIS of Mandarin Question Bank Module

8.1 Module

MandarinQuestionBankModule

8.2 Uses

MatchingQuestionBankModule, RepetitionQuestionBankModule

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

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

8.4 Semantics

8.4.1 State Variables

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

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

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

8.4.4 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.

8.4.5 Local Functions

N/A

9 MIS of Matching Question Bank Module

9.1 Module

MatchingQuestionBankModule

9.2 Uses

N/A

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

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

9.4 Semantics

9.4.1 State Variables

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

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

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

9.4.4 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`

9.4.5 Local Functions

N/A

10 MIS of Repetition Question Bank Module

10.1 Module

RepetitionQuestionBankModule

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
storeRepetitionQuestion	question: JSON object	status: bool	StorageException
retrieveRepetitionQuestion	questionID: str	question: JSON object	NotFoundException

10.4 Semantics

10.4.1 State Variables

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

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

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

10.4.4 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`

10.4.5 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.

11 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)