Module Interface Specification for SyncMaster

Team 15, SyncMaster
Kyle D'Souza
Mitchell Hynes
Richard Fan
Akshit Gulia
Rafeed Iqbal

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 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 SyncMaster.

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 (-∞, ∞)
any	any	any data type

The specification of SyncMaster uses some derived data types: sequences, strings, tuples, map, enum, KeyCondition, AttributeCondition, S3File. 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, SyncMaster 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. Maps are a collection of key-value pairs, where thre does not necessarily need to be a restriction on the data types of the keys and values, but one can be placed. An enum is a set of values of which the data can be. AttributeConditions are conditionals placed on an attribute of a database entry. KeyConditions are a subset of AttributeConditions, and are condtionals placed on the key attributes of a database entry. An S3File, is a unique file that exists in AWS S3.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	N/A
Software Decision Modules	Audit and Compliance Module User Authentication Module Location Verification Module Logging Module Analytics and Reporting Module User Management Module Document Management Module Job Management Module
Behaviour-Hiding Modules	API Integration Module Database Interaction Module Blob Storage Interaction Module Request Routing Module Function Compute Module

Table 1: Module Hierarchy

6 MIS of [Module Name —SS]

[Use labels for cross-referencing —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 Database Interaction Module

7.1 Module

Database Interaction Module

7.2 Uses

boto3 (AWS SDK for Python), AWS DynamoDB (AWS Cloud Service for NoSQL Databases)

7.3 Syntax

7.3.1 Exported Constants

Name	Description
DBTable.Name	The name of the underlying DynamoDB table resource.
DBTable.Access	The level of access the DBTable object has on the DynamoDB table. Either "read" or "write".

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
DBTable	**TableName:** string **Access:** enum[“read”, “write”]	DBTable	-
DBTable.get	**Key:** map[string → any]	map[string → any]	**ItemNotFound:** Item with requested key does not exist in the database **ExternalServiceFailure:** An internal error from AWS
DBTable.put	**Item:** map[string → any] **Condition:** AttributeCondition	map[string → any]	**ConditionCheckFailed:** The given condition is not met **ExternalServiceFailure:** An internal error from AWS **PermissionException:** If the current access level is read-only
DBTable.delete	**Key:** map[string → any] **Condition:** AttributeCondition	map[string → any]	**ExternalServiceFailure:** An internal error from AWS **ConditionCheckFailed:** The given condition is not met **PermissionException:** If the current access level is read-only
DBTable.query	**KeyConditions:** sequence[KeyCondition] **AttributeConditions:** sequence[AttributeCondition]	sequence[map[string → any]]	**ExternalServiceFailure:** An internal error from AWS

7.4 Semantics

7.4.1 State Variables

Name	Description
Database	The underlying AWS DynamoDB table, can be represented as a set of items: $\{i_0, i_1, ..., i_n\}$, where $i_k : map[string \rightarrow any], k \in [0, n]$

7.4.2 Environment Variables

Name	Description
LAMBDA_EXECUTION_ROLE	When an AWS Lambda Function (The chosen AWS compute service for this project), it has an AWS IAM role attached to it, that it uses when running. This role needs permission for database access for this module to work

7.4.3 Assumptions

LAMBDA_EXECUTION_ROLE has the required permissions in AWS to access the database.

7.4.4 Access Routine Semantics

DBTable.put(Item: i_{new}, Condition: c):

- transition: $Database \rightarrow Database \cup \{i_{new}\}$, if $c == true \land DBTable.Access == \text{“}write\text{”}$
- output: i_{new}
- exception: ExternalServiceFailure, ConditionCheckFailed, PermissionException

DBTable.delete(Key: k, Condition: c):

- transition: $Database \rightarrow Database - \{i_{old}\}$, where $k == dbKey(i_{old}, DBTable.Name)$, if $c == true \land DBTable.Access == \text{“}write\text{”}$
- output: i_{old}
- exception: ExternalServiceFailure, ConditionCheckFailed, PermissionException

7.4.5 Local Functions

Name	In	Out	Description
dbKey	**Item:** map[string → any] **TableName:** string	map[string → any]	Returns the keys of the given db item, assuming it is from the given table

8 MIS of File Storage Interaction Module

8.1 Module

File Storage Interaction Module

8.2 Uses

boto3 (AWS SDK for Python), AWS S3 (AWS Cloud Service for storing files)

8.3 Syntax

8.3.1 Exported Constants

Name	Description
S3Bucket.Name	The name of the underlying S3 Bucket resource.
S3Bucket.Access	The level of access the S3Bucket object has on the S3 Bucket resource in AWS. Either "read" or "write".

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
S3Bucket	**BucketName:** string **Access:** enum["read", "write"]	S3Bucket	-
S3Bucket.createPresignedUrl	**Key:** string **VersionID:** string **ETag:** string **Method:** enum["get", "upload"] **ExpiresIn:** integer	string	**PermissionException:** If attempting to get an upload url, while only having read permissions **ExternalServiceFailure:** An internal error from AWS
S3Bucket.delete	**Key:** string **VersionID:** string **ETag:** string	-	**FileNotFound:** The given key and version do not match any file in the bucket **ETagMismatch:** The given ETag does not match the ETag of the file with the given key and version **ExternalServiceFailure:** An internal error from AWS **PermissionException:** If the current access level is read-only

8.4 Semantics

8.4.1 State Variables

Name	Description
Bucket	The underlying AWS S3 Bucket, can be represented as a set of files: $\{f_0, f_1, ..., f_n\}$, where $f_k : S3File \land k \in [0, n]$

8.4.2 Environment Variables

Name	Description
LAMBDA_EXECUTION_ROLE	When an AWS Lambda Function (The chosen AWS compute service for this project), it has an AWS IAM role attached to it, that it uses when running. This role needs permission for S3 bucket access for this module to work

8.4.3 Assumptions

LAMBDA_EXECUTION_ROLE has the required permissions in AWS to access the S3 bucket.

8.4.4 Access Routine Semantics

S3Bucket.delete(Key: k, VersionID: v, ETag: e):

- transition: $Bucket \rightarrow Bucket - \{f_{old}\}$, where $map\{Key : k, VersionID : v, ETag : e\} == S3Bucket.metadata(f_{old})$, if $S3Bucket.Access == \text{“}write''$
- exception: ExternalServiceFailure, ConditionCheckFailed, PermissionException

8.4.5 Local Functions

Name	In	Out	Description
S3Bucket.metadata	**File:** S3File	map[string → any]	Returns S3 key, versionID, and Etag of the given file, assuming it is from the given S3Bucket

9 MIS of Function Compute Module

9.1 Module

Function Compute Module

9.2 Uses

AWS Lambda (AWS Service that executes code in response to events and manages the compute resources needed to run the code)

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
Invoke	**FunctionName:** string **Event:** map[any → any]	map[any → any]	**ExternalServiceFailure:** An internal error from AWS **ExecutionError:** Any error that occurs while the function is running

9.4 Semantics

9.4.1 State Variables

N/A

9.4.2 Environment Variables

Name	Description
LAMBDA_EXECUTION_ROLE	When an AWS Lambda Function runs, it has an AWS IAM role attached to it, which it uses when running. This role gives the function the necessary permissions to execute without issue.

9.4.3 Assumptions

LAMBDA_EXECUTION_ROLE has the required permissions in AWS to execute the lambda's required tasks.

9.4.4 Access Routine Semantics

N/A

9.4.5 Local Functions

N/A

10 MIS of Routing Module

10.1 Module

Routing Module

10.2 Uses

AWS APIGateway (AWS Cloud Service for handling routing of API to an underlying serverless function)

10.3 Syntax

10.3.1 Exported Constants

Name	Description
BaseUrl	The base url of the REST API

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
SubmitRequest	**Request:** map[any → any] **Path:** String	map[any → any]	**ExternalServiceFailure:** An internal error from AWS **ExecutionError:** If the underlying compute resource that the request gets routed to encounters an error suring execution

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

N/A

10.4.5 Local Functions

N/A

11 MIS of User Management Module

11.1 Module

User Management Module

11.2 Uses

Database Interaction Module, Routing Module

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
createUser	**email:** string **password:** string	**userID:** string	**DuplicateUser:** Existing email address used
editUser	**userID:** string **edit:** map[string → any]	boolean	**UserNotFound:** userID not in database
deleteUser	**userID:** string	boolean	**UserNotFound:** userID not in database
getUser	**userID:** string	map[string → any]	**UserNotFound:** userID not in database

11.4 Semantics

11.4.1 State Variables

Name	Description
Database	Set of registered users, can be represented as set of items $\{i_0, i_1, ..., i_n\}$, where $i_k : map[string \rightarrow any], k \in [0, n]$

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

createUserUser(email: email, password: password):

- transition: $Database \rightarrow Database \cup \{User_{new}\}$, if $email \notin Database$
- output: $authToken$
- exception: InvalidCredentials

authenticateContractor(email: email, name: name, locationState: bool):

- transition: $issuedTokens \rightarrow issuedTokens \cup \{authToken\}$, if $(email \cap name) \in Database \wedge bool == true$, where $authToken$ is uniquely generated for the user.
- output: $authToken$
- exception: InvalidCredentials, LocationVerificationFailed

validateToken(authToken: token):

- transition: N/A
- output: true if $token \in issuedTokens$, else false.
- exception: TokenInvalid

11.4.5 Local Functions

Name	In	Out	Description
generateToken	**userID:** string	**authToken:** string	Generates unique secure token for authenticated user

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.

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)