Module Interface Specification for McMaster Engineering Society Custom Financial Expense Reporting Platform

Team #12, Reimbursement Rangers
Adam Podolak
Evan Sturmey
Christian Petricca
Austin Bennett
Jacob Kish

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

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **1**

6 MIS of **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 . . . 3

7 MIS of Account Management API **3**

7.1 Module . . . 3
7.2 Uses . . . 3
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 . . . 4
7.4.3 Assumptions . . . 4
7.4.4 Access Routine Semantics . . . 4
7.4.5 Local Functions . . . 5

8 MIS of Account Management Controller **6**

8.1 Module . . . 6
8.2 Uses . . . 6
8.3 Syntax . . . 6
8.3.1 Exported Constants . . . 6
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 7
8.4.5 Local Functions 7

9 Appendix **8**

3 Introduction

The following document details the Module Interface Specifications for the MES Finance Platform. The document specifies how each module interfaces with other parts of the program. Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/ausbennett/mes-finance-platform.

4 Notation

The structure of the MIS for modules comes from **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. 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 McMaster Engineering Society Custom Financial Expense Reporting Platform.

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 McMaster Engineering Society Custom Financial Expense Reporting Platform 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, McMaster Engineering Society Custom Financial Expense Reporting Platform 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	Account Management Module Requests Module Notification Module User Dashboard Module Authentication Module Email Module Account Management Controller Module Requests Controller Module
Software Decision	Clubs Database Users Database Requests Database Graphical User Interface

Table 1: Module Hierarchy

6 MIS of

6.1 Module

6.2 Uses

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
	-	-	-

6.4 Semantics

6.4.1 State Variables

6.4.2 Environment Variables

6.4.3 Assumptions

6.4.4 Access Routine Semantics

():

- transition:
- output:
- exception:

6.4.5 Local Functions

7 MIS of Account Management API

7.1 Module

Account Management API

7.2 Uses

Account Management Controller

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
auth	token: String	JSON	InvalidTokenException
loginUser	email: String	String	EmailNotFoundException
registerUser	userDetails: JSON	JSON	DatabaseException
getAllUsers	adminToken: String	JSON	AuthorizationException
getUser	userId: String	JSON	UserNotFoundException
editProfile	userId: String, updates: JSON	Boolean	DatabaseException
editClub	clubId: String, updates: JSON	JSON	AuthorizationException

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

MongoDB connection (via Mongoose)

7.4.3 Assumptions

- Valid and authenticated tokens are provided for admin and user-specific actions.
- All inputs are sanitized before being processed.

7.4.4 Access Routine Semantics

auth(token: String):

- transition: Validates the provided token and grants access.
- output: returns a JSON object with detailed information about the result.
- exception: InvalidTokenException if token is malformed or expired.

loginUser(email: String):

- transition: Sends a confirmation link to the provided email.
- output: returns a JSON object with detailed information about the result.

- exception: EmailNotFoundException if email does not exist in the system.

registerUser(userDetails: JSON):

- transition: Adds a new user record to the database.
- output: returns a JSON object with detailed information about the result.
- exception: DatabaseException if there is an issue saving to MongoDB.

getAllUsers(adminToken: String):

- input: admin auth token
- output: returns a JSON object with detailed information about the result and array (users).
- exception: DatabaseException if there is an issue communicating to MongoDB.

getUser(userID: String):

- input: userID of user
- output: returns a JSON object with detailed information about the result.
- exception: DatabaseException if there is an issue communicating to MongoDB.

editUser(userID: String, updates: JSON):

- input: userID, and a JSON object containing updates to user information.
- output: returns a JSON object with detailed information about the result.
- exception: DatabaseException if there is an issue communicating to MongoDB.

editClub(clubID: String, updates: JSON):

- input: clubID, and a JSON object containing updates to club information.
- output: returns a JSON object with detailed information about the result.
- exception: DatabaseException if there is an issue communicating to MongoDB.

7.4.5 Local Functions

None

8 MIS of Account Management Controller

8.1 Module

Account Management Controller

8.2 Uses

Mongoose Schema, MongoDB

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
createUser	userDetails: JSON	JSON	DatabaseException
findUser	userId: String	JSON	UserNotFoundException
updateUser	userId: String, updates: JSON	JSON	DatabaseException
deleteUser	userId: String	JSON	AuthorizationException

8.4 Semantics

8.4.1 State Variables

MongoDB User Schema (defines fields like email, password, roles, etc.)

8.4.2 Environment Variables

MongoDB connection via Mongoose (database connection client)

8.4.3 Assumptions

- Mongoose (database connection client) is properly configured and connected to MongoDB.
- User schema validations are performed automatically during operations.

8.4.4 Access Routine Semantics

createUser(userDetails: JSON):

- transition: Saves a new user record to MongoDB.
- output: Returns a JSON object with detailed information about the result.
- exception: DatabaseException if saving fails due to validation or connection issues.

findUser(userId: String):

- transition: Queries the MongoDB collection for the specified user.
- output: Returns user data in JSON format.
- exception: UserNotFoundException if the user ID does not exist.

updateUser(userId: String, updates: JSON):

- transition: Updates the MongoDB collection for the specified user information.
- output: Returns a JSON object with detailed information about the result.
- exception: UserNotFoundException if the user ID does not exist.

deleteUser(userId: String):

- transition: Removes the specified user from the MongoDB collection.
- output: Returns a JSON object with detailed information about the result.
- exception: UserNotFoundException if the user ID does not exist.

8.4.5 Local Functions

- Validation functions for email and password.

9 Appendix

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?
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)