

Module Guide for McMaster Engineering Society Custom Financial Expense Reporting Platform

Team #12, Reimbursement Rangers

Adam Podolak

Evan Sturmey

Christian Petricca

Austin Bennett

Jacob Kish

April 4, 2025

1 Revision History

Date	Version	Notes
1/17/2025	0	All sections
3/24/2025	1	Updated Authentication, see commit: 4cef88d
4/4/2025	1	Updated User Dashboard Module, see commit: a23d43b
4/4/2025	1	Improvements to Software decision module section, see commit: d28cc2c
4/4/2025	1	Improved traceability tables with descriptions for what each module does, see commit: 6ba09e4

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

Symbol	Description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
\progrname	Explanation of program name
UC	Unlikely Change

Contents

1	Revision History	i
2	Reference Material	ii
2.1	Abbreviations and Acronyms	ii
3	Introduction	1
4	Anticipated and Unlikely Changes	2
4.1	Anticipated Changes	2
4.2	Unlikely Changes	2
5	Module Hierarchy	3
6	Connection Between Requirements and Design	3
7	Module Decomposition	3
7.1	Hardware Hiding Modules (M1)	4
7.2	Behaviour-Hiding Module	4
7.2.1	Account Management Module (M2)	5
7.2.2	Requests Module (M3)	5
7.2.3	Notification Module (M4)	5
7.2.4	User Dashboard Module (M5)	6
7.2.5	Authentication Module (M6)	6
7.2.6	Emailer API (M7)	6
7.2.7	Account Management Controller (M8)	7
7.2.8	Requests Controller (M9)	7
7.3	Software Decision Module	7
7.3.1	Clubs Database (M10)	7
7.3.2	User Database (M11)	8
7.3.3	Requests Database (M12)	8
7.3.4	Graphical User Interface (M13)	8
8	Traceability Matrix	9
9	Use Hierarchy Between Modules	13
10	User Interfaces	14
10.1	Login/Signup Page	14
10.2	User Info Form (General Info)	15
10.3	User Info Form (Club Info)	16
10.4	User Info Form (Payment Info)	17
10.5	Student Dashboard	18

10.6 Admin Dashboard	19
10.7 Account Info Page	20
10.8 Edit Account Info Page	20
10.9 Request Page	21
10.10View Requests Page	22
10.11Edit Request Page (Student/Admins	23
10.12Review Request Page (Admins)	23
10.13Review Request Page (Students)	24
10.14Ledger Tracker	25
11 Design of Communication Protocols	26
12 Timeline	26

List of Tables

1 Module Hierarchy	4
2 Trace Between Requirements and Modules with Notes	9
3 Trace Between Additional Requirements and Modules with Notes	11
4 Trace Between Anticipated Changes and Modules	12

List of Figures

1 Use hierarchy among modules	13
---	----

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team. We advocate a decomposition based on the principle of information hiding. This principle supports design for change, because the “secrets” that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by, as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module’s data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed. The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers’ understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: Changes to the database schema to meet evolving data storage needs.

AC2: Expansion of the notification system to include push notifications and SMS.

AC3: Adjustments to the ledger views for new regulatory compliance requirements impacting data handling.

AC4: Enhancements to the user interface for improved accessibility and user experience.

AC5: Integration with additional third-party APIs for payment processing.

AC6: Security enhancements to the login process. (i.e Integrated 2FA McMaster Login)

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Major changes to the core system architecture, such as switching from cloud to on-premises infrastructure.

UC2: Transitioning from the existing programming language stack to a completely different stack.

UC3: Removal of essential core modules such as authentication or role-based access control.

UC4: Fundamental changes in the communication protocols, such as moving from REST to an entirely different protocol.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Account Management Module

M3: Requests Module

M4: Notification Module

M5: User Dashboard Module

M6: ~~Authentication Module~~

M7: Email Module

M8: Account Management Controller Module

M9: Requests Controller Module

M10: Clubs Database

M11: Users Database

M12: Requests Database

M13: Graphical User Interface

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table ??.

7 Module Decomposition

Modules are decomposed according to the principle of “information hiding” proposed by ?. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the

Level 1	Level 2
Hardware-Hiding Module	
	Account Management Module
	Requests Module
	Notification Module
Behaviour-Hiding Module	User Dashboard Module
	Authentication Module
	Email Module
	Account Management Controller Module
	Requests Controller Module
Software Decision Module	Clubs Database
	Users Database
	Requests Database
	Graphical User Interface

Table 1: Module Hierarchy

Implemented By title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *McMaster Engineering Society Custom Financial Expense Reporting Platform* means the module will be implemented by the McMaster Engineering Society Custom Financial Expense Reporting Platform software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module

serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Account Management Module (M2)

Secrets: User account details, including personal information, access levels, and account statuses.

Services: Provides functionality for creating, updating, and deleting user accounts. It manages user roles and permissions, ensuring that appropriate access is granted to each user based on their role within the system. This module ensures secure storage of account information and integrates with the authentication module for login and identity verification purposes.

Implemented By: Account Management Controller

Type of Module: Abstract Data Type

7.2.2 Requests Module (M3)

Secrets: Internal data structure of requests, including status, history, and associated user transaction information.

Services: Manages the creation, tracking, and approval of reimbursement requests. This module handles the submission of new requests by users, updates the status of existing requests, and reconciles transactions with external plaid API for auditing purposes. It communicates with the notification module to inform users about the progress of their requests.

Implemented By: Requests Controller

Type of Module: Abstract Data Type

7.2.3 Notification Module (M4)

Secrets: Notification content, delivery methods, and scheduling information.

Services: Sends notifications to users regarding updates to their reimbursement requests, account changes, and other system events. It supports email. The module ensures that notifications are timely.

Implemented By: Emailer API

Type of Module: Abstract Data Type

7.2.4 User Dashboard Module (M5)

Secrets: Defines the organization and structure of displayed data, including summaries and real-time updates.

Services: Delivers a user-friendly interface showing the current status of reimbursement requests, notifications, and account information. It updates dynamically based on user interactions and backend data changes. Real-time updates are handled through periodic polling, ensuring timely reflection of state changes while maintaining performance efficiency. The module integrates with the requests and account management modules to provide a comprehensive, up-to-date user experience.

Implemented By: MRM

Type of Module: Abstract Data Type

7.2.5 Authentication Module (M6)

This was deemed out of scope. Authentication will be done through integrating with MES' existing authentication process.

Secrets: ~~Validation processes, and session management details.~~

Services: ~~Handles user login, logout, and session management. It ensures secure authentication using encrypted credentials. The module integrates with the account management module to verify user identities and enforce access control.~~

Implemented By: ~~MRM~~

Type of Module: ~~Abstract Data Type~~

7.2.6 Emailer API (M7)

Secrets: Email templates, SMTP server credentials.

Services: Provides an interface for sending emails to users. It handles notification delivery, and verification communications, and ensures successful delivery through integration with external SMTP services. The module is used by the notification module to send email-based alerts to users.

Implemented By: TBD

Type of Module: Abstract Data Type

7.2.7 Account Management Controller (M8)

Secrets: The flow of control for account operations and error handling mechanisms.

Services: Acts as a controller for the account management module. It receives requests from the user dashboard and other modules, processes them, and invokes appropriate functions within the account management module. It ensures consistency and error handling in account-related operations.

Implemented By: MRM

Type of Module: Abstract Data Type

7.2.8 Requests Controller (M9)

Secrets: The flow of control for request operations and error handling mechanisms.

Services: Acts as a controller for the requests module. It handles user inputs for creating and managing requests, processes these inputs, and invokes the necessary functions within the requests module. The controller ensures that all requests are valid and consistent with the system's requirements.

Implemented By: MRM

Type of Module: Abstract Data Type

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: This module encompasses the system's data storage components that do not provide direct interaction with the user but support core functionalities. The databases are responsible for reliable transaction handling and efficient data querying. They expose methods to relevant controllers and modules to facilitate logic. These modules are designed with scalability in mind, including support for integration with third-party APIs (e.g., payment processors) and extendable schema models for anticipated data format changes.

Implemented By:

7.3.1 Clubs Database (M10)

Secrets: The structure and format of the database storing information about McMaster University clubs, including club names, member rosters, and financial data.

Services: Provides methods to store, retrieve, and update club-related data, ensuring data integrity and quick access. It supports queries for club membership, financial histories, and active/inactive statuses, which are used by other modules to process reimbursement requests.

Implemented By: MongoDB

Type of Module: Library

7.3.2 User Database (M11)

Secrets: The structure and format of the database storing user account details, including personal information, roles, and activity logs.

Services: Stores user registration data, access roles, and status fields. Interfaces with the Authentication Module to verify user identity and with the Account Management Module to fetch/update user details.
purposes.

Implemented By: MongoDB

Type of Module: Library

7.3.3 Requests Database (M12)

Secrets: The structure and format of the database containing reimbursement request records, including statuses, timestamps, and related documents.

Services: Manages creation and updates of reimbursement/payment requests from the Requests Module. Supports historical tracking via logs and timestamps. Ensures transaction integrity when updates are made through the Requests Controller. Interacts with third-party APIs (e.g., Plaid or payment systems) to log transaction results.

Implemented By: MongoDB

Type of Module: Library

7.3.4 Graphical User Interface (M13)

Secrets: The structure and layout of user interaction elements, including buttons, forms, and visual themes. It also contains methods to style and manage user inputs efficiently.

Services: Builds a visual interface for users to interact with the platform. Handles user input events (e.g., button clicks, form submissions) and sends commands to the appropriate modules. Provides error handling for invalid inputs and visual feedback.

Implemented By: React

Type of Module: Library

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes. Note: hyperlinks to requirements in the SRS will be added in a later version of this document.

Table 2: Trace Between Requirements and Modules with Notes

Req.	Modules	Notes
FR1.1	M3 , M5 , M9 , M10 , M11 , M12 , M13	Request handling, UI display, data persistence
FR1.2	M3 , M5 , M9 , M10 , M11 , M12 , M13	Edit flow across controller, DBs, and UI
FR1.3	M3 , M5 , M9 , M10 , M11 , M12 , M13	Full request lifecycle coverage
FR1.4	M3 , M5 , M9 , M10 , M11 , M12 , M13	Same modules involved in view/edit flow
FR1.5	M4 , M7	Email delivery and template management
FR1.6	M12	Request status tracking
FR1.7	M6	Handles authentication
FR1.8	M5 , M3 , M9 , M13	UI (M13), request DB (M12), controller (M9), and logic (M3)
FR1.9	M12	Tracks updates in request database
FR3.1	M10 , M11	User/club info persistence
FR3.2	M6	Authentication via existing MES system
FR3.3	M2 , M8	Account management and controller logic
APR1	M13	Responsive frontend design
APR2	M13	UI enhancement handling
STYR1	M13	GUI styling for themes
STYR2	M13	Accessibility theming

Table 2 – continued from previous page

Req.	Modules	Notes
EUR1	M13	UI structure
EUR2	M13	Form design layout
EUR3	M13	Interactive elements
PIR1	M2, M8	Account roles and permissions
PIR2	M2, M8	Role editing logic
UAPR1	M13	Visual layout engine
ACSR1	M13	GUI component arrangement
ACSR2	M13	Frontend component styling
SPLR1	M3, M9, M10, M11, M12	Club/user/request logic coordination
SPLR2	M3, M9, M10, M11, M12	Multi-module reconciliation
SPLR3	M3, M9, M10, M11, M12	Data integrity during request flow
SFCR1	M11, M12	DB integrity and cross-link validation
SFCR2	M6	Authentication flow security
POAR1	M9	Request form validation control

Table 3: Trace Between Additional Requirements and Modules with Notes

Req.	Modules	Notes
CPR1	M3, M9	Club/request form logic
CPR2	M3, M9	Same as CPR1 with extended role validation
CPR3	M10, M11, M12	Core DBs for persistence of forms
OR1	M1	Hardware abstraction
IAR1	M1 – M13	Full system integration from hardware to frontend
PRD1	M10, M11, M12	Data model for third-party integration
PRD2	M6	Auth for secured external access
PRD3	M13	UI layer interacting with APIs
PRD4	M1 – M13	Full stack extensibility
PRD5	M1 – M13	Reusable across integrations
PRD6	M1 – M13	Modular design for third-party support
PRD7	M10, M11, M12	Schema control for partner data
RLR1	M1 – M13	System-wide logging and recovery
MR1	M10, M11, M12	DB scalability and replication
MR2	M1 – M13	Whole-system modularity
MR3	M1 – M13	Flexible module interconnectivity
SR1	M1 – M13	Runtime security architecture
AR1	M1 – M13	Architecture for extensibility
SCR1	M6	Login security

Table 3 – continued from previous page

Req.	Modules	Notes
SCR2	M6, M2, M8	Auth, account, and controller validation
SCR3	M6, M4, M7	Email and auth security features
SCR4	M2, M8, M6	Controller role security
SCR5	M2, M8, M6	Consistent role-based logic
SCR6	M2, M8, M6	Secure module boundaries
INR1	M3, M9	Request form controller logic
PVR1	M2, M3, M4, M6, M7, M8, M9, M10, M11, M12	System-wide data privacy controls
PVR2	M2 – M13	Full system privacy enforcement
ADR1	M2, M3, M4, M6, M7, M8, M9, M10, M11, M12	All data modules support change
ADR2	M12	Tracks and stores audit logs
IMM1	M10, M11, M12	Core DB schema updatable
CLTR1	M13	UI display of ledger totals
LR1	M1 – M13	Logging through system layers
LR2	M1 – M13	Recovery logic system-wide

AC	Modules
AC1	M10, M11, M12
AC2	M4, M7
AC3	M9, M5, M3
AC4	M13
AC5	M2, M8
AC6	M6, M1

Table 4: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. ? said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

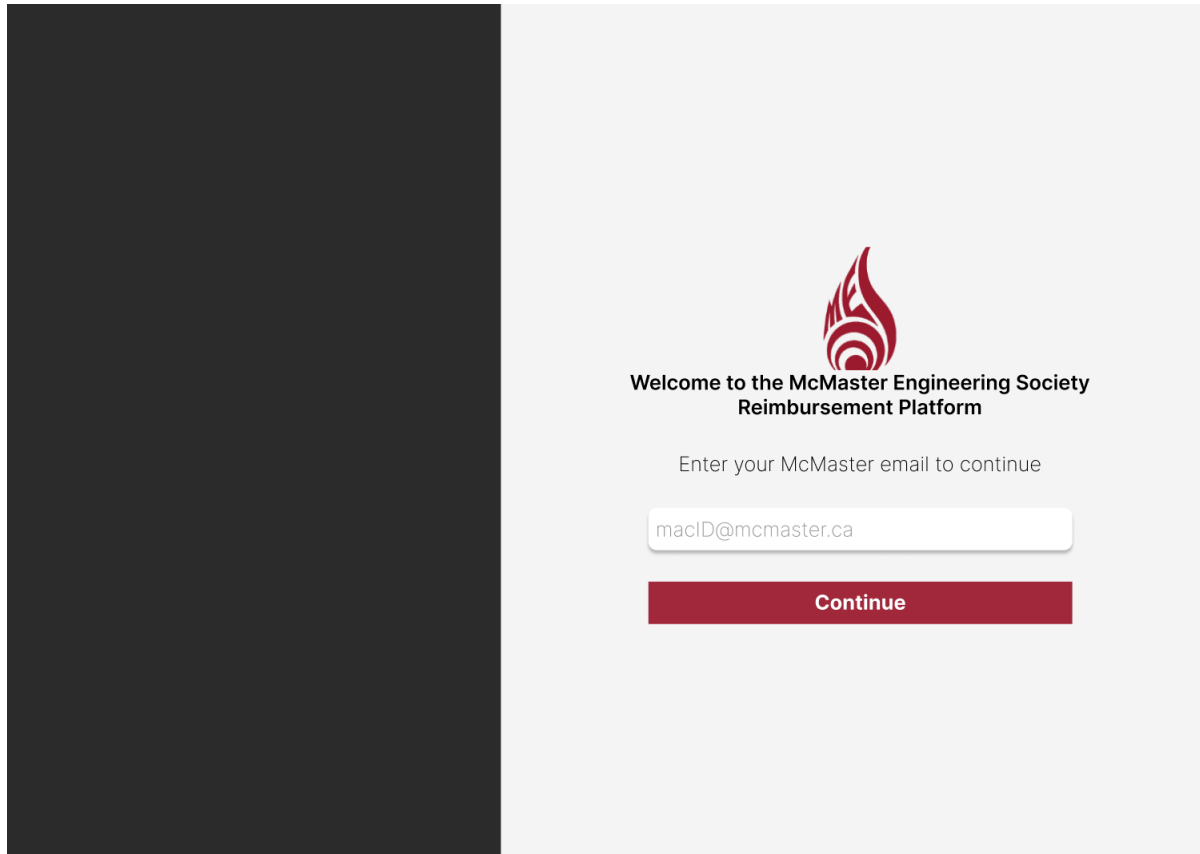


Figure 1: Use hierarchy among modules

10 User Interfaces

Design of the user interface screens are located on [Figma](#). Image references are also provided below:

10.1 Login/Signup Page



10.2 User Info Form (General Info)



Let's get you started

- 1** User Info **2** Club Info **3** Payment Info

First Name

Last Name

Phone Number

Continue

10.3 User Info Form (Club Info)



Let's get you started

1 User Info

2 Club Info

3 Payment Info

Who are you?

Club, Team, or Program Society ▼

Club

Baja Racing ▼

Role

Design Lead ▼

Add Another Position

Continue

10.4 User Info Form (Payment Info)



Let's get you started

- 1 User Info 2 Club Info 3 Payment Info

Payment Method

Interac E-Transfer

E-Transfer Email

username@email.com

E-transfer Phone Number

905-123-4567

Add Another Payment Method

Continue

10.5 Student Dashboard



Hello User, Welcome to the MES Reimbursement & Payment Platform!



Submit a new reimbursement or payment request

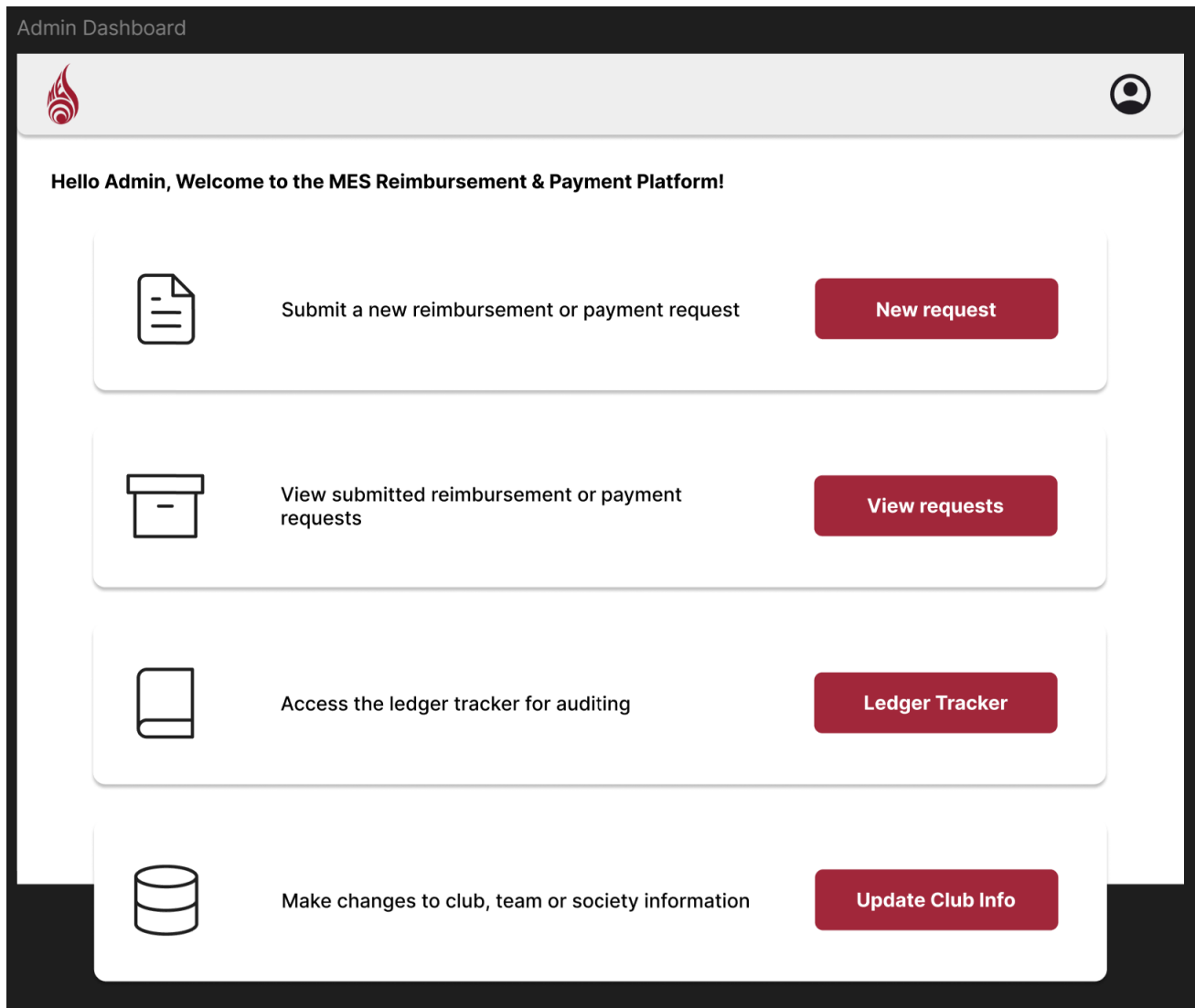
New request





View your submitted reimbursement and payment requests

View requests


10.6 Admin Dashboard



10.7 Account Info Page

Dashboard

Your Account

Hello, User

First Name

John

Last Name

Smith

Email

macID@mcmaster.ca



Phone Number

905-123-4567

Club Information

Club Information

10.8 Edit Account Info Page

Dashboard

Edit Account

Hello, User

First Name

John

Last Name

Smith

Email

macID@mcmaster.ca

Phone Number

905-123-4567

Club Information

Club Information

Save

10.9 Request Page



Dashboard



Select a request to submit

☒ Reimbursement Request

You have already paid for something,
and would like the MES to pay you back

☐ Payment Request

You are requesting the MES pay
someone on your behalf

Budget Line Item

Select ▼

Reimbursement Request

Recipient

Search

Payment Date

1/23/2025

Description

description of payment

Subtotal

\$12.34



HST

\$12.34

Add Recipient

Submit

10.10 View Requests Page



Dashboard

John Smith's Submitted Requests

× Sort by: Most Recent ▾

John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request
John Smith Baja Racing - Team Lead January 1st, 2025	Request: Reimbursement Amount: \$12.34 Status: Pending	View request

10.11 Edit Request Page (Student/Admins)

 Dashboard 

Select a request to submit

☒ Reimbursement Request
You have already paid for something, and would like the MES to pay you back

☐ Payment Request
You are requesting the MES pay someone on your behalf

Budget Line Item
Select ▼

Reimbursement Request

Recipient
John Smith

Payment Date
1/23/2025



Description
description of payment

Subtotal
\$12.34

HST
\$12.34

Save

10.12 Review Request Page (Admins)

 Dashboard 

Select a request to submit

☒ Reimbursement Request
You have already paid for something, and would like the MES to pay you back

☐ Payment Request
You are requesting the MES pay someone on your behalf

Budget Line Item
Select ▼

Reimbursement Request

Set Status
Status: approved ▼

Recipient
John Smith

Payment Date
1/23/2025


Description
description of payment


Subtotal
\$12.34

HST
\$12.34

Edit Request

10.13 Review Request Page (Students)



Dashboard 

Select a request to submit

☒ Reimbursement Request
You have already paid for something,
and would like the MES to pay you back

☐ Payment Request
You are requesting the MES pay
someone on your behalf

Budget Line Item

Select ▼

Reimbursement Request

Recipient

John Smith

Payment Date

1/23/2025

Description

description of payment

Subtotal



\$12.34

HST

\$12.34

Edit Request

10.14 Ledger Tracker

Dashboard

Ledger Tracker

×

Sort by: Most Recent ▾

Reimbursement Request: John Smith - \$12.34 View Request	^
• 1/12/2025 - Action: Status Changed to "Disbursed"	User: Admin - Jane Smith
• 1/11/2025 - Action: Status Changed to "Approved"	User: Admin - Jane Smith
• 1/9/2025 - Action: Status Changed to "Pending Review"	User: Admin - Jane Smith
• 1/8/2025 - Action: Request Edit	User: Student - John Smith
• 1/4/2025 - Action: Request Submitted	User: Student - John Smith
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾
Reimbursement Request: John Smith - \$12.34 View Request	▾

11 Design of Communication Protocols

Not applicable for this project.

12 Timeline

Timeline will be found on GitHub Projects [here!](#)