Module Guide for MES-ERP

Team #26, Ethical Pals
Sufyan Motala
Rachid Khneisser
Housam Alamour
Omar Muhammad
Taaha Atif

April 4, 2025

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

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
MES-ERP	Explanation of program name
UC	Unlikely Change
HH	Hardware-Hiding
BH	Behaviour-Hiding
SD	Software Decision
DB	Database
GUI	Graphical User Interface
CI/CD	Continuous Integration / Continuous Deployment
[etc. —SS]	[... —SS]

Contents

List of Tables

List of Figures

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 (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). 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 principles laid out by Parnas et al. (1984), as follows:

- System details that are likely to change independently are the secrets of separate modules.
- Each data structure is implemented in only one module, ensuring clarity and responsibility for managing that structure.
- Any program requiring information from a module's data structures must access it via well-defined interfaces.

The MES-ERP financial platform's design focuses on:

- Modularity: Each functional requirement is encapsulated in a distinct module to isolate changes and enhance maintainability.
- Scalability: Modules are designed to accommodate future enhancements, such as support for new input formats or integration with third-party systems.
- Robustness: Validation and error-handling mechanisms ensure data integrity across all modules.

After completing the Software Requirements Specification (SRS), the Module Guide (MG) was developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify and work on the system's components.

The potential readers of this document include:

- New Project Members: Provides an overview of the system's structure, allowing new members to quickly identify relevant modules.
- Maintainers: Ensures maintainers can understand and update the system while minimizing the impact on unrelated modules.
- Designers: Allows designers to verify consistency, feasibility, and flexibility of the system's architecture.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system, categorized as either anticipated or unlikely. These changes reflect both user-driven requirements and system-level constraints.

4.1 Anticipated Changes

Anticipated changes are those expected to occur due to user feedback, evolving requirements, or regular system updates. These changes are addressed by hiding information in the appropriate modules.

AC1: Input Data Format

- Secret: The specific format (e.g., CSV, JSON, potentially OCR from images) of receipt/expense data.
- Impacted Modules: ~~Input Format Module, Data Validation Module~~ Expense Submission Module, Data Validation Module
- Examples: Adding support for new formats (e.g., images, PDFs) or OCR for scanned receipts.

AC2: Notification Methods

- Secret: The specific method (Email, SMS, In-App) and external service (SendGrid, Twilio) used for notifications.
- Impacted Modules: Notification System Module
- Examples: Adding SMS or push notification support.

AC3: Reimbursement Workflow Rules

- Secret: The specific rules defining the steps, conditions, and approvers for processing requests.
- Impacted Modules: ~~Reimbursement Submission, Approval Workflow~~ Approval Workflow Module
- Examples: Modifying approval rules (e.g., adding budget holder sign-off), changing thresholds, or automating ~~incomplete submission rejections~~ rejection for incomplete submissions.

AC4: Financial Reporting Formats

- Secret: The specific format (PDF, CSV, Excel) and content structure of generated financial reports.
- Impacted Modules: Reporting Module
- Examples: Adding new report types (e.g., budget variance), changing export formats.

4.2 Unlikely Changes

Unlikely changes are less probable due to their disruptive nature or misalignment with project goals.

UC1: Core Technology Stack

- Impacted Modules: All
- Examples: Switching from TypeScript/Next.js to another framework (e.g., Python/Django, Java/Spring).

UC2: Database Provider

- Impacted Modules: Database Interaction Layer Module
- Examples: Migrating from Supabase/PostgreSQL to MySQL, MongoDB, or a self-hosted solution.

5 Module Hierarchy

Modules are organized to encapsulate secrets and ensure clear responsibilities. The hierarchy reflects relationships between modules and their dependencies, as shown in Table 1. This decomposition follows the standard Hardware-Hiding (HH), Behaviour-Hiding (BH), and Software Decision (SD) layers.

M1: Hardware-Hiding Modules: Abstracts ~~database and file system~~ interactions with external hardware or services, primarily the database.

M2: Behaviour-Hiding Modules: Implements ~~workflows for submissions, reviews, and dashboards~~ the externally visible functionality specified in the SRS.

M3: Software Decision Modules: Encapsulates ~~shared logic for validation, testing, and CI/CD~~ internal design choices, algorithms, or data structures not directly visible to the user but necessary for implementation.

Level 1	Level 2
Hardware-Hiding (HH)	Database Interaction Layer (M4)
Behaviour-Hiding (BH)	User Authentication & Profile Management (M5) Expense Submission & Tracking (M6) Budget and Funding Management (M7) Approval Workflow and Review (M8) Notifications & Communication (M9) Policy & Compliance Management (M10) Reporting and Analytics (M11) Administrator and Configuration Panel (M12)
Software Decision (SD)	Data Validation Module (M13) GUI Module (M14)

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the MES-ERP system has been structured to ensure that the requirements outlined in the Software Requirements Specification (SRS) are met comprehensively. Table 2 illustrates the connection between the requirements and the modules designed to fulfill them.

6.1 Design Decisions for Each Requirement

- **R1: Secure Authentication**
  - **Module**: User Authentication & Profile Management Module (M5)
  - **Design Decision**: Implements secure login via Supabase Auth, ~~with~~ session timeouts, role-based access control (RBAC), and ~~account lockout mechanisms~~ profile management.
- **R2: Expense Submission & Tracking**
  - **Module**: Expense Submission & Tracking Module (M6)
  - **Design Decision**: Provides forms for submission, receipt uploads (incl. OCR attempt), and tracking expense status via the database.

Table 2: Connection Between Requirements and Modules

Requirement (R)	**Modules**
R1: Secure Authentication	User Authentication & Profile Management Module (M5)
R2: Expense Submission & Tracking	Expense Submission & Tracking Module (M6)
R3: Budget Validation	Budget and Funding Management Module (M7)
R4: Approval Workflow	Approval Workflow and Review Module (M8)
R5: Notifications	Notifications & Communication Module (M9)
R6: Compliance	Policy & Compliance Management Module (M10)
R7: Reporting	Reporting and Analytics Module (M11)
R8: Administrative Tools	Administrator and Configuration Panel Module (M12)

- **R3: Budget Validation**
  - **Module**: Budget and Funding Management Module (M7)
  - **Design Decision**: Ensures expense requests ~~do not exceed departmental budgets~~ are checked against allocated group budgets and integrates with ~~the financial system~~ the operating budget data stored in the database.
- **R4: Approval Workflow**
  - **Module**: Approval Workflow and Review Module (M8)
  - **Design Decision**: Implements dynamic routing for approvals (based on role/group) and ~~notifications for pending actions~~ triggers status updates.
- **R5: Notifications**
  - **Module**: Notifications & Communication Module (M9)
  - **Design Decision**: Sends alerts via email~~, SMS,~~ and ~~dashboard notifications~~ (using SendGrid) for ~~system events~~ request status changes. (SMS is a potential future extension).
- **R6: Compliance**
  - **Module**: Policy & Compliance Management Module (M10)

  - **Design Decision**: Validates expense requests against predefined policies (e.g., spending limits, required documentation) ~~to ensure compliance with regulations~~. Includes audit logging of key actions.

- **R7: Reporting**
  - **Module**: Reporting and Analytics Module (M11)
  - **Design Decision**: Generates ~~detailed reports in PDF/CSV format~~ on-screen analytics visualizations (charts/tables) for expense tracking and system usage. (Export functionality (PDF/CSV) is an anticipated change AC4).

- **R8: Administrative Tools**
  - **Module**: Administrator and Configuration Panel Module (M12)
  - **Design Decision**: Enables administrators to manage user roles, ~~configure approval workflows,~~ groups, permissions, and access ~~system logs~~ the operating budget interface.

7 Module Decomposition

The MES-ERP system is designed following the principle of *information hiding*, which ensures that each module encapsulates decisions likely to change independently. The modules are organized into a hierarchy, with higher-level modules relying on lower-level ones for functionality. This hierarchy forms a directed acyclic graph (DAG). ~~Figure ?? illustrates the module decomposition.~~ Figure 1 (in Section 9) illustrates the use relation between modules. The primary decomposition follows the HH, BH, and SD layers. Supporting processes like CI/CD and test automation, while crucial for development, are not considered core architectural modules in this decomposition as they do not hide specific design decisions related to the system's runtime behavior.

7.1 Hardware-Hiding (HH) Modules

M4 Database Interaction Layer:

- **Secrets:** The specific database technology (Supabase/PostgreSQL), schema details, query language (SQL), and connection management.
- **Services:** Provides abstract functions for creating, reading, updating, and deleting (CRUD) data related to users, groups, roles, requests, budgets, etc. Hides the underlying implementation details of data persistence.
- **Implemented By:** Backend API using Supabase client library.

7.2 Behaviour-Hiding (BH) Modules

M5 User Authentication & Profile Management:

- **Secrets:** The mechanism for verifying user identity (delegated to Supabase Auth), session management details, and the structure of user profile data (excluding role/group assignments handled elsewhere).
- **Services:** Provides functions for user login, registration, logout, retrieving profile information (~~name, email, etc.~~), and updating profile details. Corresponds to SRS R1.
- **Implemented By:** Backend API (using Supabase Auth), Frontend UI (Login/Register/Account Pages).

M6 Expense Submission & Tracking:

- **Secrets:** The specific fields required for a request, receipt data format (AC1) including OCR processing logic, and how request statuses are represented internally before approval.
- **Services:** Allows users to create new payment/reimbursement requests, upload receipts, view the status of their own submitted requests. Corresponds to SRS R2.
- **Implemented By:** Frontend UI (Forms Page), Backend API (Request Creation Endpoint).

M7 Budget and Funding Management:

- **Secrets:** How budget allocations are stored and retrieved for groups, the rules for checking available funds against requests.
- **Services:** Provides functions to check if a request amount is within a group's allocated budget, potentially update remaining budget upon approval/reimbursement. Allows admins to view/manage operating budget lines. Corresponds to SRS R3.
- **Implemented By:** Backend API (Budget Checking Logic), Frontend UI (Operating Budget Page).

M8 Approval Workflow and Review:

- **Secrets:** The specific sequence of approval steps, the conditions for routing requests (AC3), and the internal representation of approval states.
- **Services:** Enables authorized users (admins) to view pending requests, change their status (approve/reject), and potentially add comments. Triggers notifications upon status change. Corresponds to SRS R4.
- **Implemented By:** Backend API (Status Update Logic), Frontend UI (Requests Page for Admins).

M9 Notifications & Communication:

- **Secrets:** The specific notification channels (Email, ~~SMS~~) and the external services used (AC2). The content templates for different notification types.

- **Services:** Provides a function to send notifications to users based on system events (e.g., request status change). Corresponds to SRS R5.
- **Implemented By:** Backend API (Notification Endpoints using SendGrid).

M10 Policy & Compliance Management:

- **Secrets:** The specific compliance rules checked during submission or approval (e.g., maximum request amounts, required fields based on type). The structure and storage of audit logs.
- **Services:** Validates requests against defined policies. Records significant actions (creation, update, deletion of requests/users/roles) in an audit log. Corresponds to SRS R6.
- **Implemented By:** Backend API (Validation Logic, Audit Logging Hooks).

M11 Reporting and Analytics:

- **Secrets:** The specific data aggregation methods and calculations used to generate analytics insights. The format of generated reports (AC4).
- **Services:** Provides aggregated data for display in analytics charts/tables (e.g., spending trends, status summaries). Corresponds to SRS R7.
- **Implemented By:** Backend API (Analytics Data Endpoint), Frontend UI (Analytics Page with Charts).

M12 Administrator and Configuration Panel:

- **Secrets:** The specific administrative functions available (user management, role management, group management, budget configuration) and how configurations are stored.
- **Services:** Provides interfaces for administrators to manage core system entities like users, roles, groups, and the operating budget structure. Corresponds to SRS R8.
- **Implemented By:** Frontend UI (Users, Roles, Groups, Operating Budget pages), Backend API (Endpoints for managing these entities).

7.3 Software Decision (SD) Modules

M13 Data Validation Module:

- **Secrets:** The specific validation rules (e.g., regex for emails, numeric checks for amounts, file type restrictions) and libraries used for input validation.
- **Services:** Provides functions to validate data submitted through forms or API requests, ensuring data consistency and integrity before it reaches the database or core logic modules.
- **Implemented By:** Shared utility functions/library used by Backend API and potentially Frontend.

M14 GUI Module:
- **Secrets:** The specific UI library (Shadcn/ui), CSS framework (Tailwind), and rendering engine (React/Next.js) used to build the user interface. The component structure and state management approach within the frontend.
- **Services:** Renders the web interface, handles user input events (clicks, form submissions), displays data fetched from the backend API, and manages frontend application state.
- **Implemented By:** The entire Next.js frontend application code (`src/app`, `src/components`).

8 Traceability Matrix

This section outlines the anticipated changes (from Section 4.1) to the MES-ERP system and identifies the modules (from Section 7) that would be impacted by each change. The traceability matrix maps these anticipated changes to the affected modules, ensuring alignment with the principle of *information hiding*.

8.1 Anticipated Changes to Modules

Table 3 provides a traceability matrix that lists the anticipated changes and their associated modules.

Table 3: Traceability Matrix: Anticipated Changes to Modules

Anticipated Change (AC)	**Affected Modules**
AC1: Input Data Format	M6, M13
AC2: Notification Methods	M9
AC3: Reimbursement Workflow Rules	M8
AC4: Financial Reporting Formats	M11

8.2 Impact of Anticipated Changes

- **AC1: Input Data Format**
  - **Affected Modules**: M6, M13
  - **Impact**: M6 would need updated logic for parsing new formats (e.g., adding OCR library call, handling XML). M13 would require new rules to check the validity of the new formats.
  - **Mitigation**: Encapsulation of parsing logic within M6 and validation rules within M13 isolates the impact.

- **AC2: Notification Methods**
  - **Affected Module**: M9
  - **Impact**: Adding SMS would require integrating a new service (e.g., Twilio) within M9 and potentially adding phone numbers to user profiles (M5).
  - **Mitigation**: The abstract ‘sendNotification‘ service provided by M9 hides the specific channel implementation (Email vs. SMS) from calling modules like M8.
- **AC3: Reimbursement Workflow Rules**
  - **Affected Module**: M8
  - **Impact**: Modifying approval steps (e.g., adding a second approver for amounts ¿ $1000) requires changing the internal state machine or rule engine within M8.
  - **Mitigation**: The workflow logic is contained within M8, preventing changes from affecting submission (M6) or notification (M9) modules directly.
- **AC4: Financial Reporting Formats**
  - **Affected Module:** M11
  - **Impact:** Adding CSV export would require new data formatting logic and potentially a new function within M11. Changing PDF layout involves modifying the PDF generation template/code used by this module.
  - **Mitigation:** Report generation is isolated in M11. Other modules only provide data to it, they don’t depend on the output format.

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) 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.

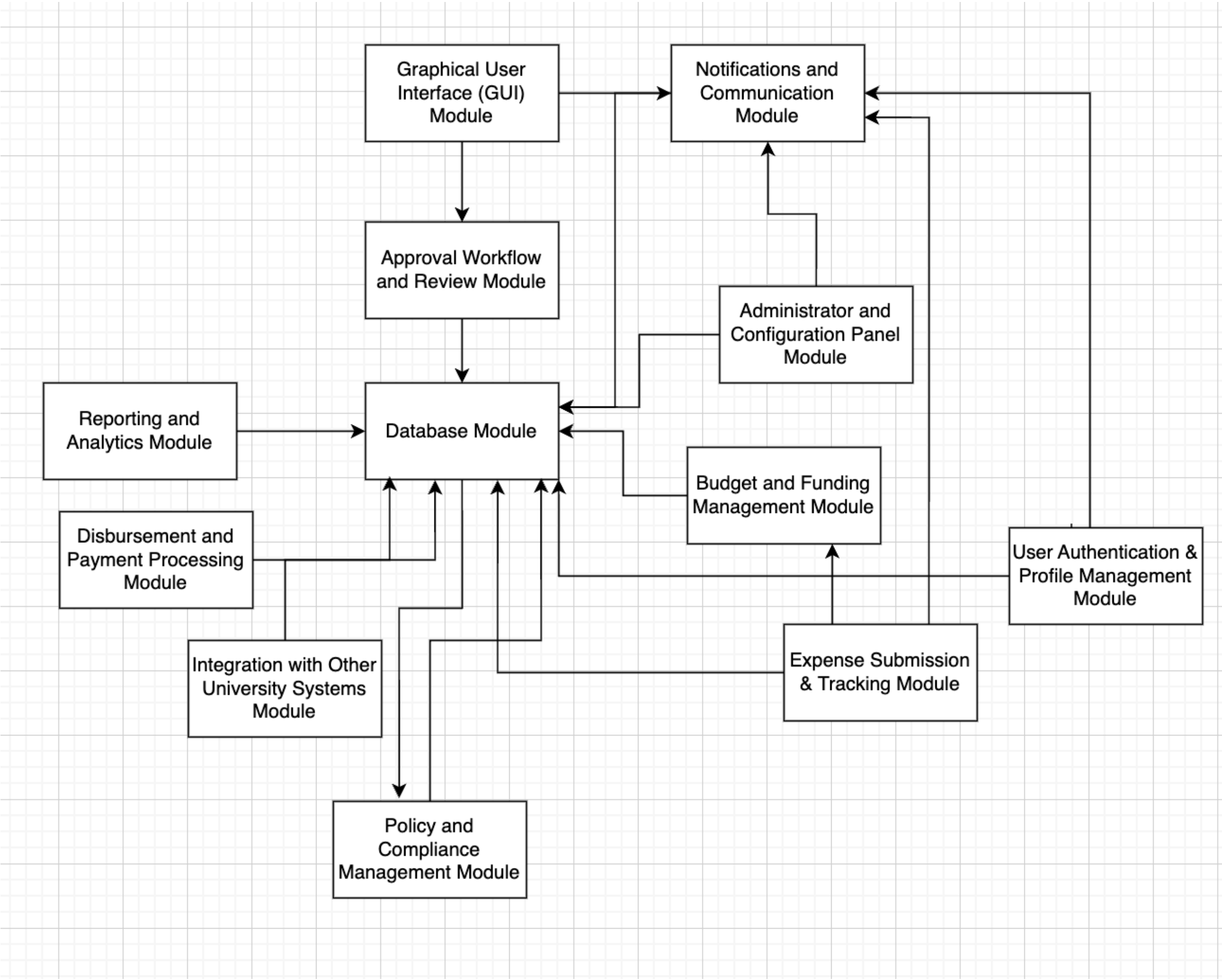

Graphical User Interface (GUI) Module
Notifications and Communication Module
Approval Workflow and Review Module
Administrator and Configuration Panel Module
Reporting and Analytics Module
Database Module
Budget and Funding Management Module
Disbursement and Payment Processing Module
User Authentication & Profile Management Module
Integration with Other University Systems Module
Expense Submission & Tracking Module
Policy and Compliance Management Module

Figure 1: Module Use Hierarchy (Illustrative - Requires Update based on Section 7 modules)

The dependencies shown in Figure 1 (and implied by the decomposition) include:

- Most Behaviour-Hiding (BH) and Software Decision (SD) modules **use** the **Database Interaction Layer (M4)** to persist and retrieve data.
- The **GUI Module (M14) uses** many BH modules to display data and trigger actions, for example:
  - Uses **User Authentication (M5)** for login/registration UI.
  - Uses **Expense Submission (M6)** to render forms and display request status.
  - Uses **Budget Management (M7)** to show budget dashboards.
  - Uses **Approval Workflow (M8)** to display pending requests for admins.
  - Uses **Reporting (M11)** to display analytics charts.

  - Uses **Admin Panel (M12)** to render user/role/group management interfaces.

- The **Approval Workflow Module (M8) uses**:
  - **Budget Management (M7)** to check funds before approval.
  - **Notifications (M9)** to send alerts after status changes.
  - **Policy & Compliance (M10)** to log the approval action.

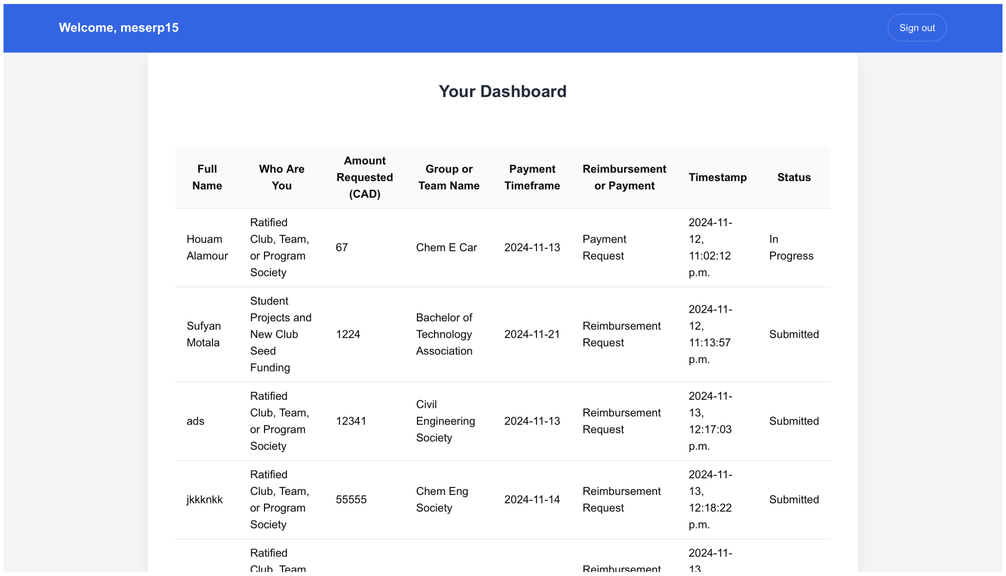
- The **Expense Submission Module (M6) uses**:
  - **Data Validation (M13)** to check inputs before saving.
  - **Policy & Compliance (M10)** to log the submission action.

- The **Admin Panel (M12) uses** modules like **User Authentication (M5)** (to manage user details), **Budget Management (M7)** (to configure budgets), etc.

This structure aims for low coupling, where higher-level modules depend on the services provided by lower-level ones without knowing their internal implementation details.

10 User Interfaces

10.1 Dashboard (GUI Module - M14)



Welcome, meserp15

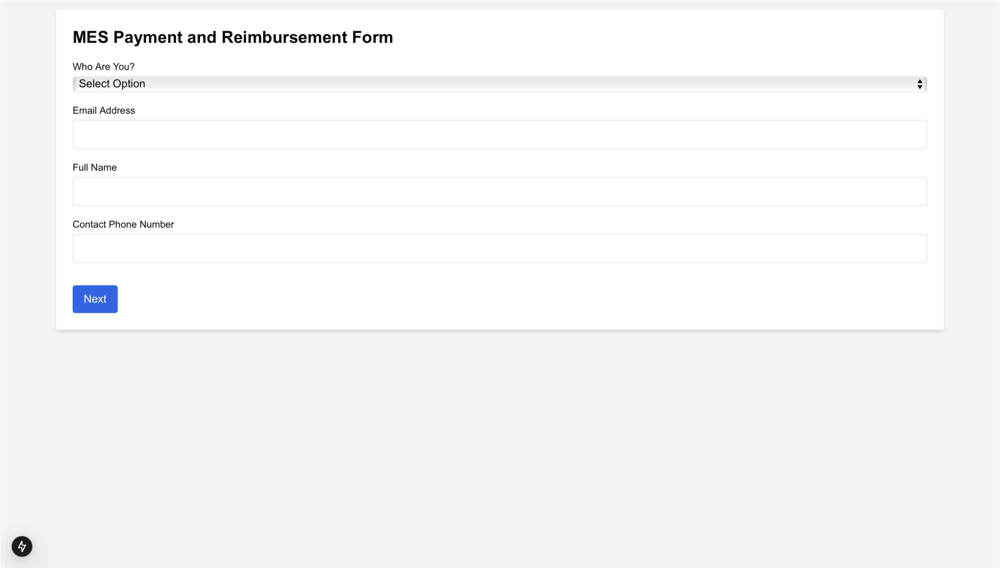
Sign out

Your Dashboard

Full Name	Who Are You	Amount Requested (CAD)	Group or Team Name	Payment Timeframe	Reimbursement or Payment	Timestamp	Status
Houam Alamour	Ratified Club, Team, or Program Society	67	Chem E Car	2024-11-13	Payment Request	2024-11-12, 11:02:12 p.m.	In Progress
Sufyan Motala	Student Projects and New Club Seed Funding	1224	Bachelor of Technology Association	2024-11-21	Reimbursement Request	2024-11-12, 11:13:57 p.m.	Submitted
ads	Ratified Club, Team, or Program Society	12341	Civil Engineering Society	2024-11-13	Reimbursement Request	2024-11-13, 12:17:03 p.m.	Submitted
jkkknkk	Ratified Club, Team, or Program Society	55555	Chem Eng Society	2024-11-14	Reimbursement Request	2024-11-13, 12:18:22 p.m.	Submitted
	Ratified Club, Team				Reimbursement	2024-11-13	

Figure 2: Dashboard View (Presents data from M11, M6)

10.2 Form (GUI Module - M14, uses M6)



MES Payment and Reimbursement Form

Who Are You?

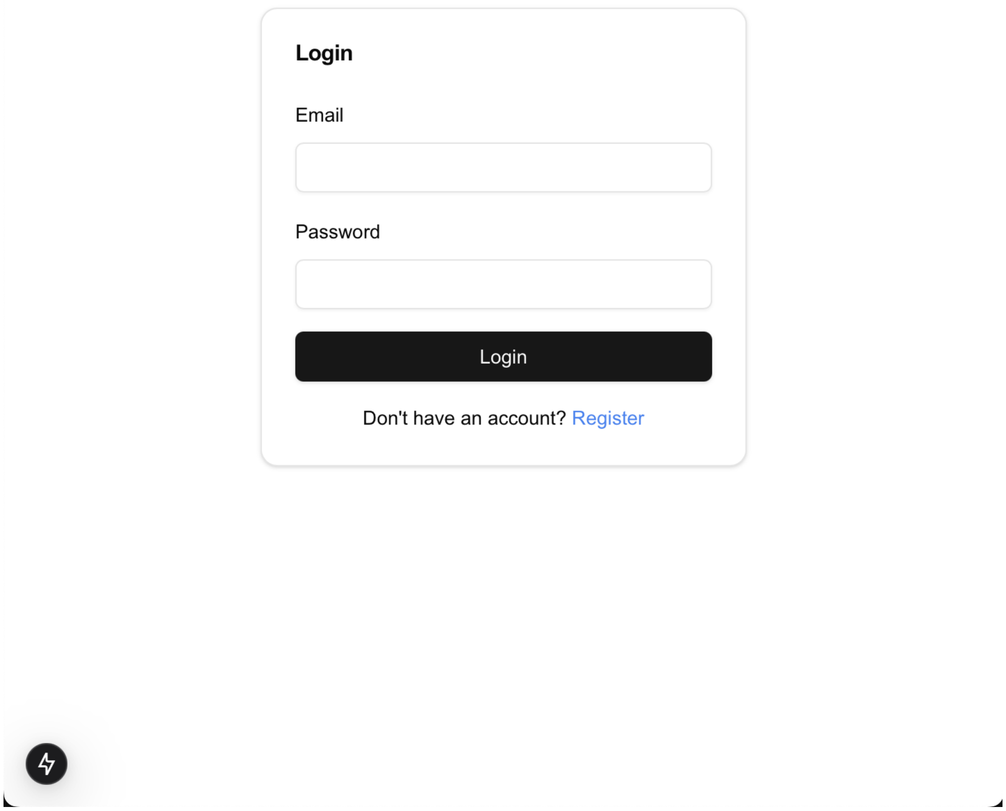
Email Address

Full Name

Contact Phone Number

Figure 3: Form View (Interface for M6)

10.3 Login (GUI Module - M14, uses M5)



Don't have an account? Register

Figure 4: Login View (Interface for M5)

10.4 Receipt Input (GUI Module - M14, part of M6)

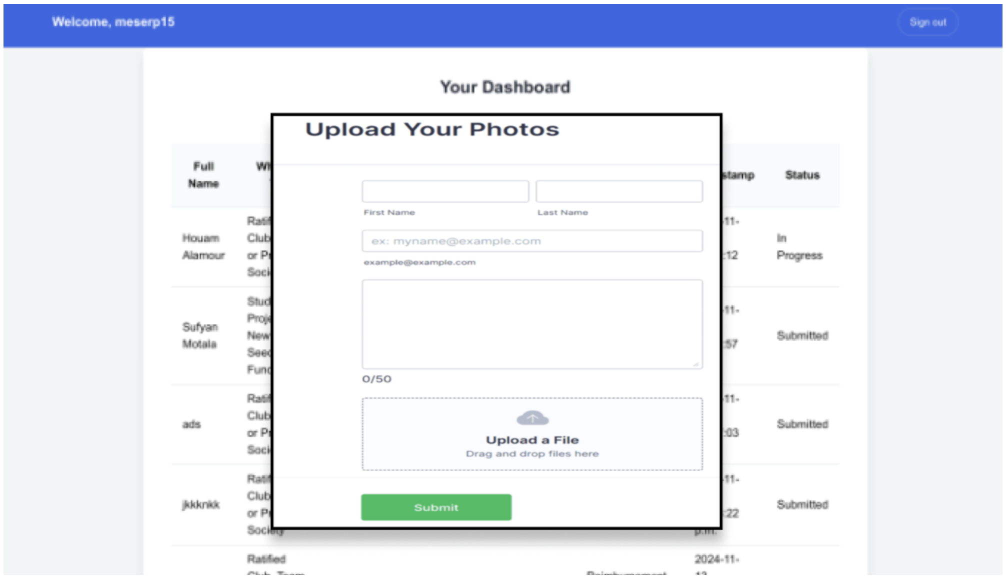


Figure 5: Receipt Scanning View (Input for M6)

10.5 Website Tutorial (GUI Module - M14)

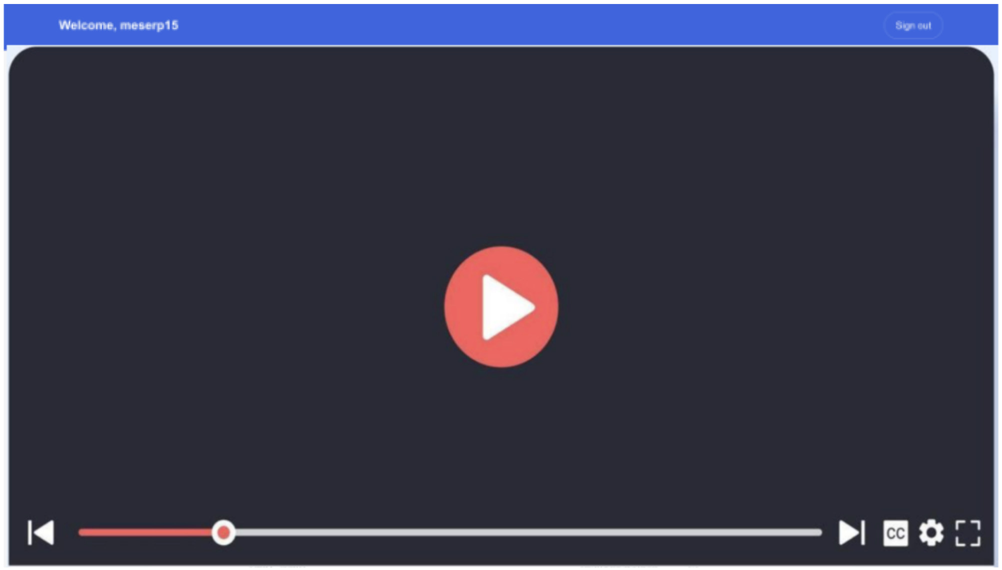


Figure 6: Tutorial Page

11 Design of Communication Protocols

- **APIs:** The **GUI Module (M14)** uses APIs (implemented as Next.js API Routes) to communicate between the backend and the frontend of the application.
- **Email Registration:** When creating an account, the **User Authentication Module (M5)** (via Supabase Auth) ~~there~~ will ~~be email authentication~~ trigger email verification to ensure valid users.
- **Email communication:** When a reimbursement request is ~~made or~~ edited or its status changes, the **Notifications Module (M9)** ~~, the correct groups~~ will ~~be notified via email~~ notify the relevant user via email.

12 Timeline

- **Team Formed, Project Selected:** September 16
  - **All Team Members (Omar, Taaha, Rachid, Sufyan, Housam):** Establish initial roles, select project scope, and identify stakeholder needs.
- **Problem Statement, POC Plan, Development Plan:** September 23
  - **All Team Members:** Finalize the problem statement for the MES-ERP system. Outline Proof-of-Concept (POC) objectives and develop a high-level development plan.
  - **Sufyan & Omar:** Draft initial architecture sketches for key modules (HH: M4; BH: e.g., M6; SD: e.g., M13).
- **Requirements Document Revision 0:** October 9
  - **Housam & Rachid:** Gather and refine user stories for Expense Submission (M6), Approval Workflow (M8), Budget Management (M7), and Notifications (M9).
  - **Taaha:** Integrate feedback on Data Validation (M13) and Database Interaction (M4) requirements.
  - **All Team Members:** Review and finalize the Requirements Specification for Rev 0.
- **Hazard Analysis 0:** October 23
  - **Omar & Sufyan:** Identify potential failure points for the Data Validation Module (M13) (e.g., corrupted or missing receipts) and the Database Module (M4) (e.g., concurrency issues).

  - **Rachid:** Document mitigations for critical hazards in the Expense Submission (M6) workflow (e.g., incorrect approvals).

- **V&V Plan Revision 0:** November 1

  - **Housam & Taaha:** Draft initial testing approach for each module:
    - **HH Module (M4):** Verify database connections and mocks.
    - **BH Modules (e.g., M6, M9, M7):** Unit tests for core logic.
    - **SD Modules (e.g., M13, M14):** Automated tests for validation rules, basic UI component rendering.
  - **All Team Members:** Incorporate plan into the project schedule.

- **Proof of Concept Demonstration:** November 11–22

  - **Omar, Taaha, Housam:** Implement a simplified Expense Submission (M6) + Data Validation (M13) flow (happy path) for the demonstration.
  - **Rachid & Sufyan:** Set up the Database Module (M4) with mock data; create a minimal GUI (M14) prototype for user interaction.
  - **All Team Members:** Conduct internal testing before the demonstration.

- **Design Document Revision 0:** January 17

  - **Omar & Housam:** Document final architecture decisions for each module:
    - **HH Module (M4):** Database schema and environment specifics.
    - **BH Modules (e.g., M6, M9):** Detailed class/function descriptions or sequence diagrams.
    - **SD Modules (M13, M14):** Data Validation logic details, GUI component structure.
  - **Rachid & Sufyan:** Cross-check design artifacts (MG, MIS) with SRS and POC feedback.
  - **Taaha:** Integrate all design sections into a cohesive Revision 0 document.

- **Revision 0 Demonstration:** February 3–February 14

  - **GUI Module (M14) (Rachid & Sufyan):** Build an interactive front-end for Expense Submission (M6), including field-level validation feedback.
  - **Data Validation Module (M13) (Taaha):** Finalize core validation checks (e.g., correct file formats, mandatory receipt uploads).
  - **Approval Workflow Module (M8) (Omar):** Implement the basic approval workflow with role-based permissions.
  - **Budget Management (M7) (Housam):** Provide partial functionality (e.g., budget overview) and real-time data fetch from the Database Module (M4).

  - **All Team Members:** Present Rev 0 to stakeholders, focusing on end-to-end demonstration of a reimbursement request.

- **V&V Report Revision 0:** March 7
  - **Taaha & Omar:** Compile test results from module-level tests, including pass/fail statistics and coverage metrics.
  - **Sufyan, Housam, Rachid:** Validate the correctness of reported bugs and retest major fixes in Expense Submission (M6) and Budget Management (M7).

- **Final Demonstration (Revision 1):** March 24–March 30
  - **All Team Members:** Incorporate feedback from Revision 0 and finalize all modules.
  - **GUI Enhancements (M14) (Sufyan & Rachid):** Polish front-end UX and incorporate advanced form validations for user-friendly error handling.
  - **Notifications Module (M9) (Housam):** Integrate notifications and advanced triggers (e.g., overdue approvals).

- **Add Receipt Scanning/Image Processing:** March 24–March 30
  - **Omar, Rachid, Housam:** Incorporate an OCR (Optical Character Recognition) service in the Expense Submission Module (M6).
  - **Taaha:** Update the Data Validation Module (M13) to handle scanned data edge cases (e.g., partial text scans).

- **Add User Manual to Application:** March 24–March 30
  - **Sufyan:** Create comprehensive end-user documentation and how-to guides, focusing on reimbursements and approvals.
  - **All Team Members:** Review for clarity and completeness before final release.

- **Refine the UI, Functions, and Backend Connectivity:** March 24–March 30
  - **Taaha:** Implement advanced error handling across all modules and polish backend endpoints for reliability.
  - **All Team Members:** Conduct integration testing for the updated features.

- **Reach Out to MES Rep (Weekly):** March 24–March 30
  - **Omar, Taaha, Rachid, Sufyan, Housam:** Schedule status meetings, collect user feedback, and prioritize final tweaks for Rev 1.

- **EXPO Demonstration:** April (TBD)

  - **All Team Members:** Showcase the fully functional application, highlighting compliance auditing, budgeting, and user flows.

- **Final Documentation (Revision 1):** April 2
  - **All Team Members:** Update the Design Document (MG, MIS), V&V Report, and User Manual with final revisions and new features.
  - **Omar & Taaha:** Proofread overall consistency and finalize submission to the department.

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.