Module Guide for MES-ERP

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

January 17, 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
[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 rules layed out by Parnas et al. (1984), 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 (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 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 **??** specifies the connections between the software requirements and the modules. Section **??** gives a detailed description of the modules. Section **??** 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. The changes are categorized as either anticipated or unlikely, based on their probability and potential impact. Anticipated changes represent areas of the system expected to evolve due to regular updates or new requirements. Unlikely changes are those that would require significant rework and are less probable given the current context.

4.1 Anticipated Changes

Anticipated changes are the source of information hidden within the modules. These changes are expected to occur as part of system updates or user feedback and are designed to be handled with minimal impact on the overall structure.

AC1: The specific hardware on which the software is hosted.

- Examples include MES-provided laptops with Windows 10 and above-average specifications or the potential transition to cloud-hosted environments like Azure or AWS.
- Impact: Hardware dependencies are encapsulated in the hardware-hiding module to isolate changes and avoid propagation to higher-level modules.

AC2: The format of input data, such as receipts or invoices uploaded by student groups.

- Examples: Adding support for new file formats (e.g., PDFs, images, CSVs) or incorporating OCR (Optical Character Recognition) for scanned documents.
- Impact: Changes to input formats affect only the Input Format and Data Validation Modules, which transform raw input into system-compatible data structures.

AC3: Notification methods, such as expanding delivery channels.

- Examples: Adding SMS notifications, push alerts via a mobile app, or integration with messaging platforms like Slack or Teams.
- Impact: The Notifications System Module handles all updates to notification methods, ensuring other modules remain unaffected.

AC4: Changes to the workflow for reimbursement request approvals.

- Examples: Modifying multi-level approval processes, incorporating dynamic routing based on the requester's department, or enabling automated rejections for incomplete submissions.
- Impact: The Approval Workflow Module encapsulates these decisions, minimizing disruptions to related modules.

AC5: Requirements for audit log duration and format.

- Examples: Updating log retention policies from 6 months to 2 years or adding additional metadata to logs for enhanced traceability.
- Impact: Changes are confined to the Audit and Compliance Module, ensuring audit functionality evolves independently.

4.2 Unlikely Changes

Unlikely changes are less probable due to their disruptive nature or because they contradict established constraints and design principles. While the system design is modular, accommodating these changes would require significant rework across multiple modules.

UC1: Supported devices.

- Examples: Expanding compatibility to entirely new platforms like gaming consoles or wearables.
- Impact: Extensive updates to the Graphical User Interface (GUI) and hardware-hiding modules would be required.

UC2: The technology stack used.

- Examples: Replacing TypeScript and Next.js with entirely different frameworks or programming languages.
- Impact: This change would affect almost all modules, requiring a complete system overhaul and significant developer retraining.

UC3: Legal or compliance policies.

- Examples: Abandoning compliance with AODA (Accessibility for Ontarians with Disabilities Act) or PIPEDA (Personal Information Protection and Electronic Documents Act).
- Impact: Changes would cascade through the Policy and Compliance Management Module, potentially invalidating existing processes and documentation.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1.

M1: Hardware-Hiding Module

M2: Behaviour-Hiding Module

M3: Software Decision Module

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Reimbursement Submission
	Reimbursement Review and Approval
	Audit and Compliance Module
	Budget Dashboard
	Notification System
	User Management
	Financial Reporting
Software Decision Module	CI/CD Integration
	Test Automation Framework
	Data Validation Module

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.

Table 2: Connection Between Requirements and Modules

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

6.1 Design Decisions for Each Requirement

- **R1: Secure Authentication**
  - **Module**: User Authentication & Profile Management Module
  - **Design Decision**: Implements secure login with session timeouts, role-based access control, and account lockout mechanisms.
- **R2: Expense Submission & Tracking**
  - **Module**: Expense Submission & Tracking Module
  - **Design Decision**: Provides forms for submission, receipt uploads, and tracking expense status.
- **R3: Budget Validation**
  - **Module**: Budget and Funding Management Module
  - **Design Decision**: Ensures expense requests do not exceed departmental budgets and integrates with the financial system.
- **R4: Approval Workflow**
  - **Module**: Approval Workflow and Review Module
  - **Design Decision**: Implements dynamic routing for approvals and notifications for pending actions.
- **R5: Notifications**
  - **Module**: Notifications & Communication Module
  - **Design Decision**: Sends alerts via email, SMS, and dashboard notifications for system events.
- **R6: Compliance**
  - **Module**: Policy & Compliance Management Module
  - **Design Decision**: Validates expense requests against predefined policies to ensure compliance with regulations.
- **R7: Reporting**
  - **Module**: Reporting and Analytics Module
  - **Design Decision**: Generates detailed reports in PDF/CSV format for expense tracking and system usage.
- **R8: Administrative Tools**

  - **Module**: Administrator and Configuration Panel Module
  - **Design Decision**: Enables administrators to manage user roles, configure approval workflows, and access system logs.

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.

7.1 Module Levels

- **Hardware-Hiding Modules**
  - **Database Module**: Provides a foundational interface for storing and retrieving data. It serves as the core data storage system, used by nearly all other modules.
- **Behavior-Hiding Modules**
  - **User Authentication & Profile Management Module**: Manages secure login, user sessions, and profiles.
  - **Expense Submission & Tracking Module**: Handles expense submission, receipt uploads, and tracking.
  - **Policy & Compliance Management Module**: Ensures submitted expenses comply with organizational policies.
- **Software Decision Modules**
  - **Approval Workflow and Review Module**: Manages dynamic workflows for expense approvals.
  - **Budget and Funding Management Module**: Validates budgets and tracks funding.
  - **Reporting and Analytics Module**: Provides reporting tools and analytics for system data.
- **Presentation Layer Modules**
  - **Graphical User Interface (GUI) Module**: Interacts with users by displaying data and capturing inputs.
  - **Notifications & Communication Module**: Sends alerts and notifications to users via email, SMS, or the dashboard.

7.2 Use Relations Between Modules

- **Database Module**
  - **Used By**: User Authentication & Profile Management, Expense Submission & Tracking, Budget and Funding Management, Approval Workflow and Review, Notifications & Communication, Reporting and Analytics, Policy & Compliance Management.
  - **Purpose**: Acts as the central data storage and retrieval system.
- **User Authentication & Profile Management Module**
  - **Used By**: Approval Workflow and Review, Notifications & Communication, GUI Module.
  - **Purpose**: Provides secure access to the system and maintains user roles and profiles.
- **Expense Submission & Tracking Module**
  - **Used By**: Approval Workflow and Review, Budget and Funding Management, Notifications & Communication, Reporting and Analytics.
  - **Purpose**: Handles expense submissions, categorization, and tracking.
- **Budget and Funding Management Module**
  - **Used By**: Expense Submission & Tracking, Approval Workflow and Review, Notifications & Communication.
  - **Purpose**: Validates and updates budgets associated with submitted expenses.
- **Approval Workflow and Review Module**
  - **Used By**: Notifications & Communication, GUI Module.
  - **Purpose**: Implements dynamic routing and rules for approving expenses.
- **Notifications & Communication Module**
  - **Used By**: GUI Module, all other modules requiring alerts or updates.
  - **Purpose**: Sends notifications and alerts via email, SMS, or dashboard.
- **Reporting and Analytics Module**
  - **Used By**: GUI Module, Administrator and Configuration Panel.
  - **Purpose**: Provides analytics and reporting tools for system data.
- **Graphical User Interface (GUI) Module**

- **Used By**: End users interacting with the system.
- **Relies On**: Backend modules to fetch and display data.
- **Purpose**: Displays system data and handles user interactions.

7.3 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

7.4 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: –

7.4.1 Input Format Module (M??)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the input parameters module.

Implemented By: [Your Program Name Here]

Type of Module: [Record, Library, Abstract Object, or Abstract Data Type] [Information to include for leaf modules in the decomposition by secrets tree.]

7.4.2 Etc.

7.5 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: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

7.5.1 Etc.

8 Traceability Matrix

This section outlines the anticipated changes to the MES-ERP system and identifies the modules 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: Changes in Authentication Protocols	User Authentication & Profile Management Module
AC2: New Expense Categories	Expense Submission & Tracking Module
AC3: Budget Limits Adjustment	Budget and Funding Management Module
AC4: Workflow Modifications	Approval Workflow and Review Module
AC5: New Notification Methods	Notifications & Communication Module
AC6: Policy Updates	Policy & Compliance Management Module
AC7: Report Formats or Types	Reporting and Analytics Module
AC8: Additional Admin Features	Administrator and Configuration Panel Module

8.2 Impact of Anticipated Changes

- **AC1: Changes in Authentication Protocols**
  - **Affected Module**: User Authentication & Profile Management Module
  - **Impact**: Updates to the `authenticate()` function to support new protocols (e.g., multi-factor authentication) and modifications to constants like session timeouts.

  - **Mitigation**: Encapsulation of authentication logic minimizes impact on other modules.

- **AC2: New Expense Categories**
  - **Affected Module**: Expense Submission & Tracking Module
  - **Impact**: Updates to the `EXPENSE_CATEGORIES` constant and adjustments to the `categorizeExpense()` function.
  - **Mitigation**: Modular handling of categories allows isolated changes without affecting other modules.

- **AC3: Budget Limits Adjustment**
  - **Affected Module**: Budget and Funding Management Module
  - **Impact**: Modifications to the `MAX_BUDGET` constant and related validation logic.
  - **Mitigation**: Centralized budget handling simplifies updates.

- **AC4: Workflow Modifications**
  - **Affected Module**: Approval Workflow and Review Module
  - **Impact**: Updates to `routeRequest()` and `updateStatus()` functions for new workflow rules.
  - **Mitigation**: Dynamic routing logic supports flexible workflow updates.

- **AC5: New Notification Methods**
  - **Affected Module**: Notifications & Communication Module
  - **Impact**: Addition of new constants (e.g., `NOTIFICATION_TYPES`) and updates to notification functions.
  - **Mitigation**: Modular notification handling allows easy integration of new methods.

- **AC6: Policy Updates**
  - **Affected Module**: Policy & Compliance Management Module
  - **Impact**: Updates to `validateRequest()` to handle new policy rules and thresholds.
  - **Mitigation**: Encapsulation of policy logic ensures minimal impact on unrelated modules.

- **AC7: Report Formats or Types**
  - **Affected Module**: Reporting and Analytics Module

  - **Impact**: Modifications to `generateReport()` to support new formats or metrics.
  - **Mitigation**: Flexible report generation logic minimizes system-wide impact.

- **AC8: Additional Admin Features**
  - **Affected Module**: Administrator and Configuration Panel Module
  - **Impact**: New access programs (e.g., `addFeature()`, `updateRolePermissions()`).
  - **Mitigation**: Well-defined admin tools enable straightforward updates.

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 **??** 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.

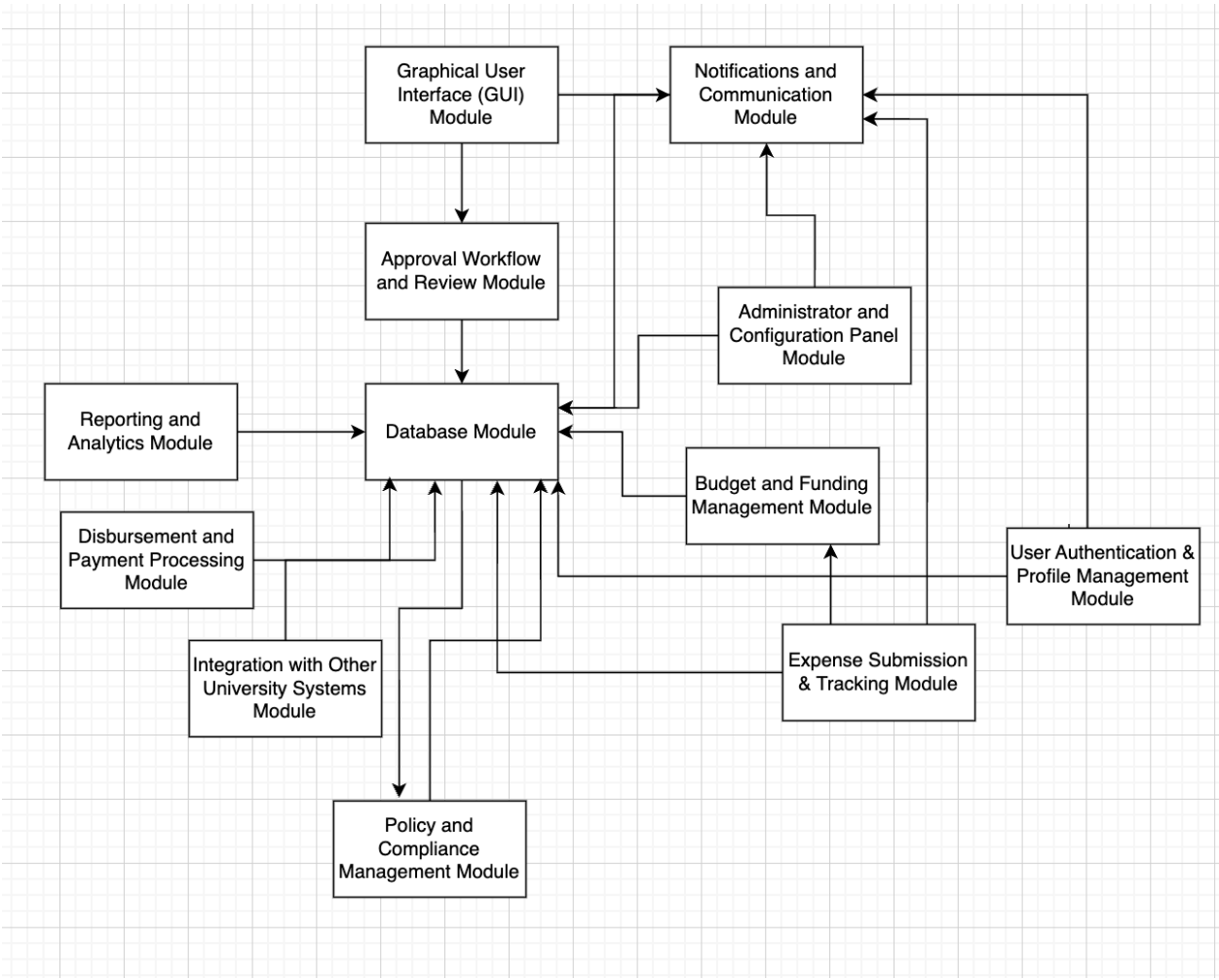
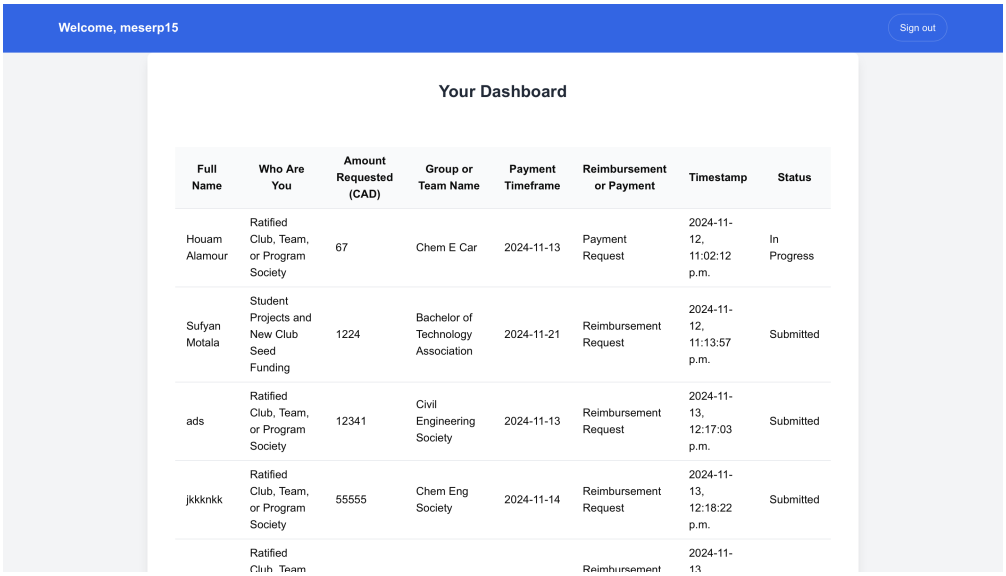


Figure 1: Module Hierarchy

10 User Interfaces

10.1 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

10.2 Form

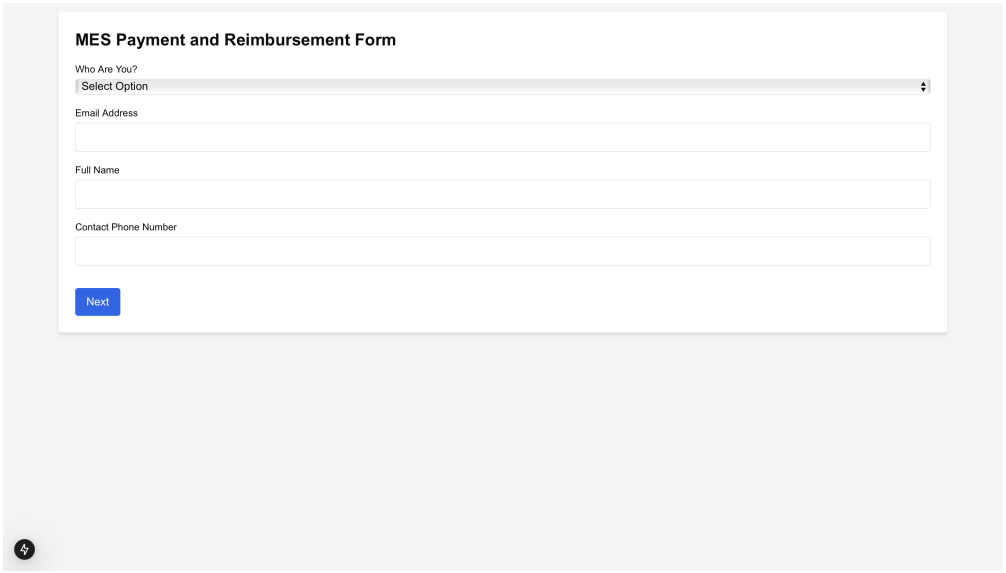


Figure 3: Form View

10.3 Login

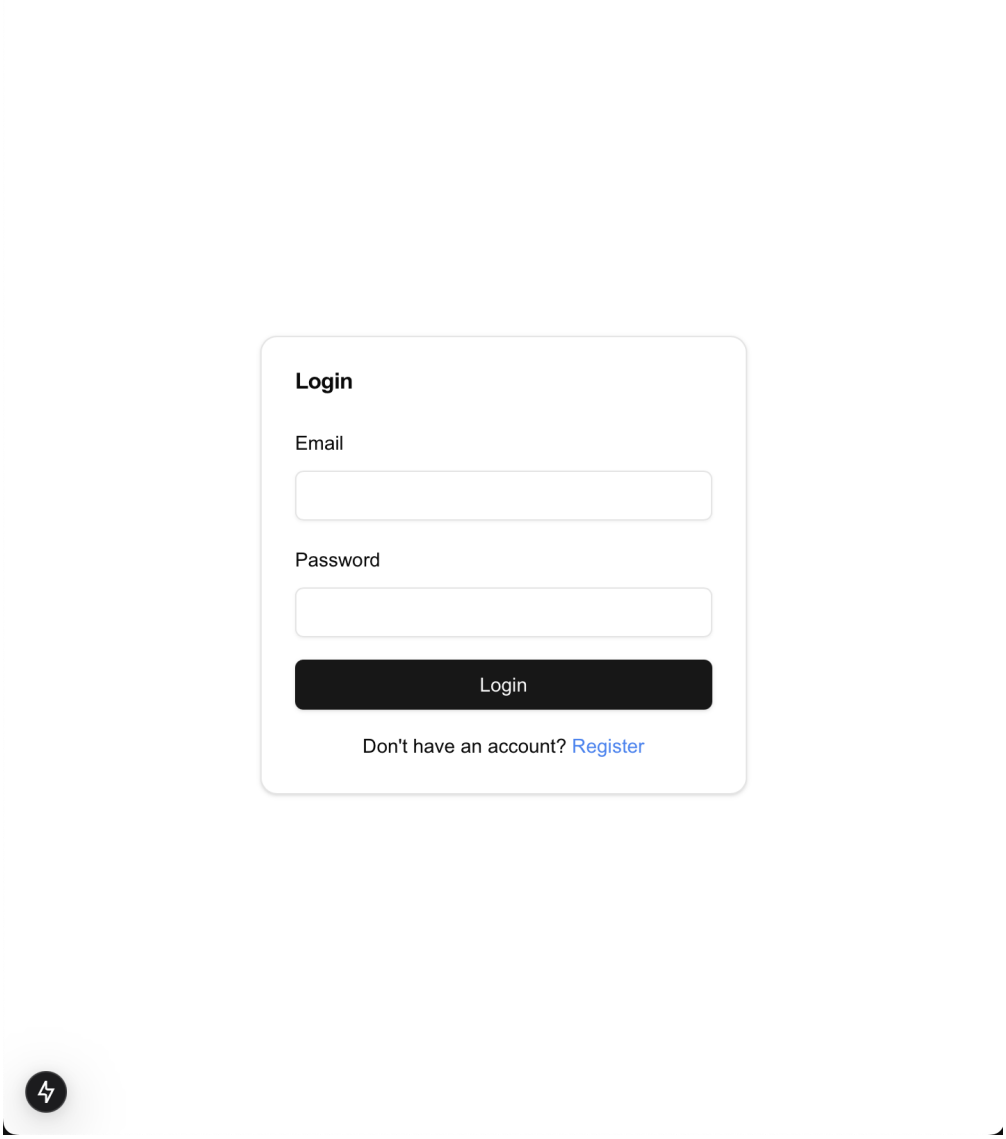


Figure 4: Login View

10.4 Receipt Input

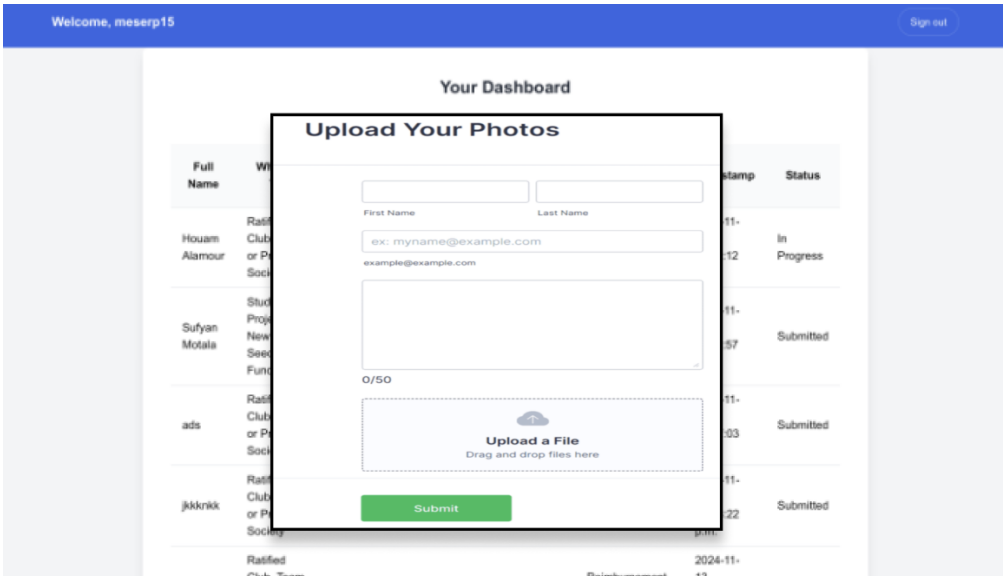


Figure 5: Receipt Scanning View

10.5 Wesbsite Tutorial

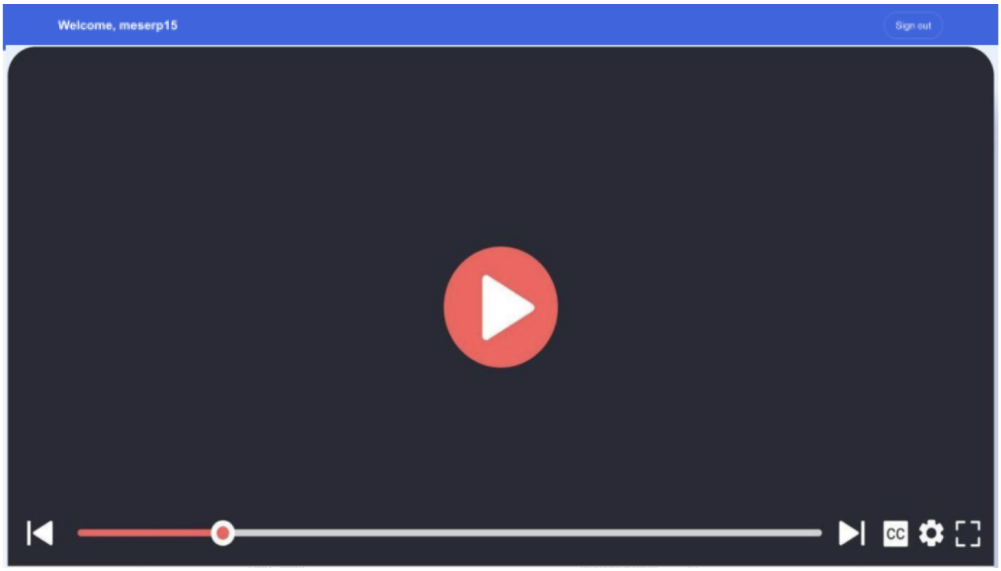


Figure 6: Tutorial Page

11 Design of Communication Protocols

- **APIs:** APIs will be used to communicate between the backend and the frontend of the application
- **Email Registration:** When creating an account there will be email authentication to ensure valid users
- **Email communication:** When a reimbursement request is made or edited, the correct groups will be notified via email

12 Timeline

- **Team Formed, Project Selected:** September 16 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Problem Statement, POC Plan, Development Plan:** September 23 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Requirements Document Revision 0:** October 9 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Hazard Analysis 0:** October 23 [Omar, Taaha, Rachid, Sufyan, Housam]
- **V&V Plan Revision 0:** November 1 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Proof of Concept Demonstration:** November 11–22 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Design Document Revision 0:** January 15 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Revision 0 Demonstration:** February 3–February 14 [Omar, Taaha, Rachid, Sufyan, Housam]
- **V&V Report Revision 0:** March 7 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Final Demonstration (Revision 1):** March 24–March 30 [Omar, Taaha, Rachid, Sufyan, Housam]
- **Add Receipt Scanning/Image Processing:** March 24–March 30 [Omar, Rachid, Housam]
- **Add User Manual to Application:** March 24–March 30 [Sufyan]
- **Refine the UI, Functions, and Backend Connectivity:** March 24–March 30 [Taaha]

- **Reach Out to MES Rep (Weekly):** March 24–March 30 [Omar, Taaha, Rachid, Sufyan, Housam]
- **EXPO Demonstration:** April (TBD) [Omar, Taaha, Rachid, Sufyan, Housam]
- **Final Documentation (Revision 1):** April 2 [Omar, Taaha, Rachid, Sufyan, Housam]

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.