

# 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
<a href="#">[etc. —SS]</a>	<a href="#">[... —SS]</a>

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Reference Material</b>	<b>ii</b>
2.1	Abbreviations and Acronyms . . . . .	ii
<b>3</b>	<b>Introduction</b>	<b>1</b>
<b>4</b>	<b>Anticipated and Unlikely Changes</b>	<b>2</b>
4.1	Anticipated Changes . . . . .	2
4.2	Unlikely Changes . . . . .	2
<b>5</b>	<b>Module Hierarchy</b>	<b>3</b>
<b>6</b>	<b>Connection Between Requirements and Design</b>	<b>3</b>
<b>7</b>	<b>Module Decomposition</b>	<b>4</b>
7.1	Hardware Hiding Modules (M1) . . . . .	5
7.2	Behaviour-Hiding Module . . . . .	5
7.2.1	Input Format Module (M??) . . . . .	5
7.2.2	Etc. . . . .	6
7.3	Software Decision Module . . . . .	6
7.3.1	Etc. . . . .	6
<b>8</b>	<b>Traceability Matrix</b>	<b>6</b>
<b>9</b>	<b>Use Hierarchy Between Modules</b>	<b>7</b>
<b>10</b>	<b>User Interfaces</b>	<b>8</b>
10.1	Dashboard . . . . .	8
10.2	Form . . . . .	9
10.3	Login . . . . .	10
10.4	Receipt Input . . . . .	11
<b>11</b>	<b>Design of Communication Protocols</b>	<b>11</b>
<b>12</b>	<b>Timeline</b>	<b>11</b>

## List of Tables

1	Module Hierarchy . . . . .	4
2	Trace Between Requirements and Modules . . . . .	6
3	Trace Between Anticipated Changes and Modules . . . . .	7

## List of Figures

1	Use hierarchy among modules . . . . .	8
2	Dashboard View . . . . .	8
3	Form View . . . . .	9
4	Login View . . . . .	10
5	Receipt Scanning View . . . . .	11

### 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:** The specific hardware on which the software is running.

**AC2:** The format of the initial input data.

...

[Anticipated changes relate to changes that would be made in requirements, design or implementation choices. They are not related to changes that are made at run-time, like the values of parameters. —SS]

### 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:** Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

...

## 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

...

## 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 2.

[The intention of this section is to document decisions that are made “between” the requirements and the design. To satisfy some requirements, design decisions need to be made. Rather than make these decisions implicit, they are explicitly recorded here. For instance, if a program has security requirements, a specific design decision may be made to satisfy those requirements with a password. —SS]



Level 1	Level 2
Hardware-Hiding Module	
70.3Behaviour-Hiding Module	?
	?
	?
	?
	?
	?
	?
	?
30.3Software Decision Module	?
	?
	?

Table 1: Module Hierarchy

## 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 *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. *MES-ERP* means the module will be implemented by the MES-ERP 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 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.]

### 7.2.2 Etc.

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

**Implemented By:** –

### 7.3.1 Etc.

## 8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
R1	M1, M??, M??, M??
R2	M??, M??
R3	M??
R4	M??, M??
R5	M??, M??, M??, M??, M??, M??
R6	M??, M??, M??, M??, M??, M??
R7	M??, M??, M??, M??, M??
R8	M??, M??, M??, M??, M??
R9	M??
R10	M??, M??, M??
R11	M??, M??, M??, M??

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??
AC??	M??

Table 3: 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.

[The uses relation is not a data flow diagram. In the code there will often be an import statement in module A when it directly uses module B. Module B provides the services that module A needs. The code for module A needs to be able to see these services (hence the import statement). Since the uses relation is transitive, there is a use relation without an import, but the arrows in the diagram typically correspond to the presence of import statement. —SS]

[If module A uses module B, the arrow is directed from A to B. —SS]

Figure 1: Use hierarchy among modules

# 10 User Interfaces

[Design of user interface for software and hardware. Attach an appendix if needed. Drawings, Sketches, Figma —SS]

## 10.1 Dashboard

Welcome, meserp15

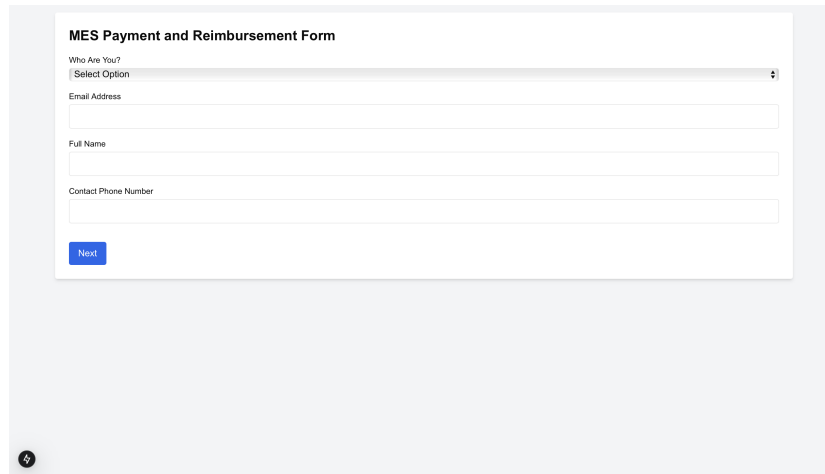
Sign out

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
jkknkk	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



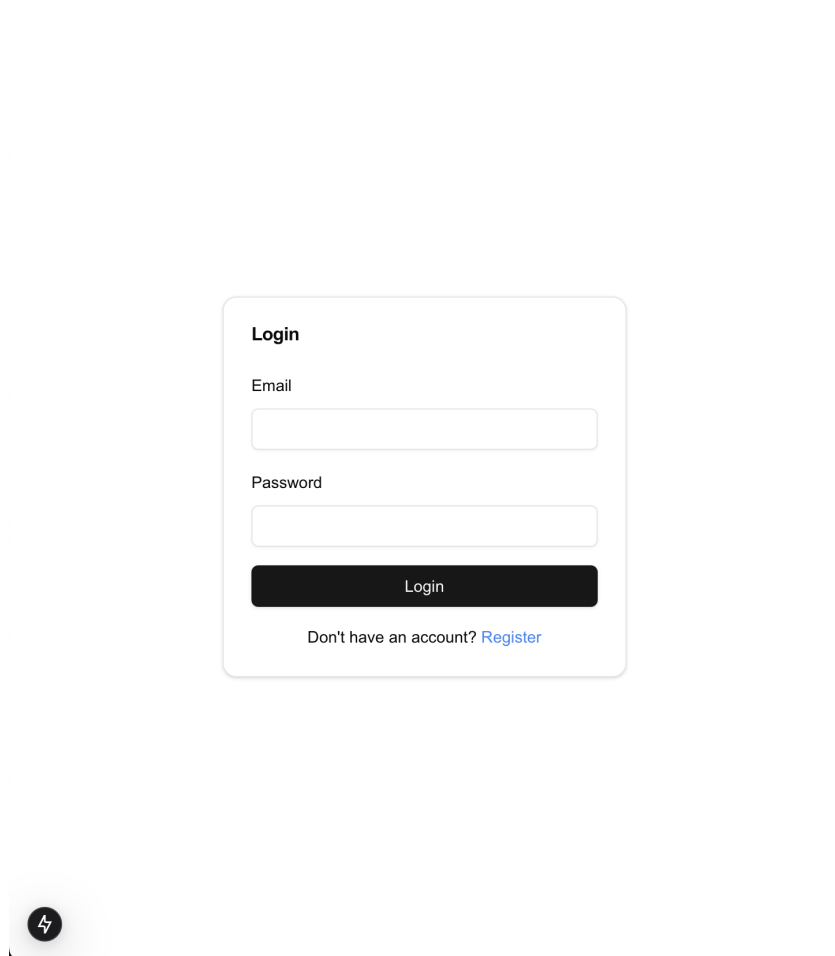
The image shows a web form titled "MES Payment and Reimbursement Form". The form is contained within a white box with a thin grey border, set against a light grey background. The form fields are as follows:

- Who Are You?**: A dropdown menu with "Select Option" as the placeholder text.
- Email Address**: A text input field.
- Full Name**: A text input field.
- Contact Phone Number**: A text input field.

At the bottom left of the form box is a blue button labeled "Next". In the bottom left corner of the overall page, there is a small black circle containing the number "4".

Figure 3: Form View

## 10.3 Login

A login form centered on a light gray background. The form is a white rounded rectangle with a thin gray border. It contains the title "Login" in bold, followed by "Email" and "Password" labels, each with a corresponding text input field. Below the inputs is a dark gray "Login" button. At the bottom, it says "Don't have an account?" followed by a blue "Register" link. A small circular icon with a lightning bolt is in the bottom-left corner of the background.

**Login**

Email

Password

Login

Don't have an account? [Register](#)

Figure 4: Login View

## 10.4 Receipt Input

The screenshot displays a web application interface. At the top, a blue header bar contains the text 'Welcome, meserp15' on the left and a 'Sign out' link on the right. Below the header, the main content area is titled 'Your Dashboard'. A modal window titled 'Upload Your Photos' is centered on the screen. The modal contains the following elements: two input fields for 'First Name' and 'Last Name'; an email input field with placeholder text 'ex: myname@example.com' and 'example@example.com'; a large text area for a description; a file upload section with a cloud icon, the text 'Upload a File', and 'Drag and drop files here'; and a green 'Submit' button at the bottom. The background of the dashboard shows a table with columns 'Full Name', 'Role', and 'Status'. The table lists several users, including Housam, Sufyan, and others, with their respective roles and status (e.g., 'In Progress', 'Submitted').

Figure 5: Receipt Scanning View

## 11 Design of Communication Protocols

[If appropriate —SS]

## 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

[Schedule of tasks and who is responsible —SS]

[You can point to GitHub if this information is included there —SS]