

# Module Guide for SFWRENG 4G06 - Capstone Design Process

**Team 17, DomainX**

Awurama Nyarko  
Haniye Hamidizadeh  
Fei Xie  
Ghena Hatoum

November 12, 2025

# 1 Revision History

Table 1: Revision History

| Date        | Developer(s) | Change |
|-------------|--------------|--------|
| Nov 3, 2025 | Fei          | Rev -1 |

## 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   |
| SFWRENG 4G06 - Capstone Design Process | Software Engineering Capstone Project |
| UC                                     | Unlikely Change                       |
| AHP                                    | Analytical Hierarchy Process          |
| BWM                                    | Best-Worst Method                     |
| SSB                                    | Skew-Symmetric Bilinear               |
| Domain                                 | Research Software Domain              |
| Packages                               | Software Packages                     |
| API                                    | Application Programming Interface     |
| ADT                                    | Abstract Data Type                    |
| POC                                    | Proof of Concept                      |

# Contents

|   |    |
|---|----|
| <b>1 Revision History</b>                           | i  |
| <b>2 Reference Material</b>                         | ii |
| 2.1 Abbreviations and Acronyms . . . . .            | ii |
| <b>3 Introduction</b>                               | 1  |
| <b>4 Anticipated and Unlikely Changes</b>           | 2  |
| 4.1 Anticipated Changes . . . . .                   | 2  |
| 4.2 Unlikely Changes . . . . .                      | 2  |
| <b>5 Module Hierarchy</b>                           | 3  |
| <b>6 Connection Between Requirements and Design</b> | 4  |
| <b>7 Module Decomposition</b>                       | 4  |
| 7.1 Hardware Hiding Modules (M1) . . . . .          | 5  |
| 7.1.1 Browser Module (M2) . . . . .                 | 5  |
| 7.2 Behaviour-Hiding Module . . . . .               | 5  |
| 7.2.1 Application UI Module (M3) . . . . .          | 5  |
| 7.2.2 Data Edit Module (M4) . . . . .               | 6  |
| 7.2.3 User Authentication Module (M5) . . . . .     | 6  |
| 7.2.4 User Role Access Module (M6) . . . . .        | 6  |
| 7.2.5 User Page Module (M7) . . . . .               | 6  |
| 7.2.6 Automated Metrics Module (M8) . . . . .       | 7  |
| 7.2.7 Domains Page Module (M9) . . . . .            | 7  |
| 7.2.8 Comparison Module (M16) . . . . .             | 7  |
| 7.2.9 Configuration Module (M19) . . . . .          | 7  |
| 7.3 Software Decision Module . . . . .              | 8  |
| 7.3.1 System API Gateway Module (M10) . . . . .     | 8  |
| 7.3.2 Ranking Algorithm Module (M11) . . . . .      | 8  |
| 7.3.3 Graphing Module (M12) . . . . .               | 8  |
| 7.3.4 File Import Module (M13) . . . . .            | 9  |
| 7.3.5 File Export Module (M14) . . . . .            | 9  |
| 7.3.6 Repository API Module (M15) . . . . .         | 9  |
| 7.3.7 Database Persistence Module (M17) . . . . .   | 9  |
| 7.3.8 Logging Module (M18) . . . . .                | 10 |
| <b>8 Traceability Matrix</b>                        | 10 |
| <b>9 Use Hierarchy Between Modules</b>              | 14 |

|   |           |
|---|-----------|
| <b>10 User Interfaces</b>                   | <b>16</b> |
| <b>11 Design of Communication Protocols</b> | <b>19</b> |
| <b>12 Timeline</b>                          | <b>19</b> |

## List of Tables

|   |    |
|---|----|
| 1 Revision History . . . . .                                      | i  |
| 2 Module Hierarchy . . . . .                                      | 4  |
| 3 Trace Between Functional Requirements and Modules . . . . .     | 10 |
| 4 Trace Between Non-Functional Requirements and Modules . . . . . | 13 |
| 5 Trace Between Anticipated Changes and Modules . . . . .         | 14 |

## List of Figures

|  |    |
|--|----|
| 1 Use hierarchy among modules . . . . .              | 15 |
| 2 Main domain summary view page . . . . .            | 16 |
| 3 Main domain data view page . . . . .               | 16 |
| 4 Domain editing . . . . .                           | 17 |
| 5 Domain management (collaborator role) . . . . .    | 17 |
| 6 User management (super admin role) . . . . .       | 18 |
| 7 Main user information page (viewer role) . . . . . | 18 |
| 8 Domain X Database Schema . . . . .                 | 21 |

### 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 laid 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 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 metrics used to assess a domain's state of practice may be updated (e.g. Addition of new metrics, such as code coverage percentage)

**AC2:** The system should support extension to user authentication (e.g. Using two-factor authentication on top of username and password)

**AC3:** The ranking algorithm used within a domain for packages may be changed (e.g. Alternatives to AHP, such as BWM, SSB, etc)

**AC4:** The comparison algorithms used between domains may be changed

**AC5:** The user access roles available might be expanded (e.g. Admin, User, Contributor)

**AC6:** The APIs used to extract repository metrics

**AC7:** The visualization libraries used to display data

**AC8:** Additional export and import file formatting may be added (e.g. CSV, Excel)

### 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:** The schema structure of our domain and metrics systems are designed for long-term use and allows for the addition of new metric types

**UC2:** The platform of the tool will remain as a web application

**UC3:** The development stack of our web application will remain the same ([React](#), [Django](#))

**UC4:** Input types of existing metrics, as outlined in the methodology paper ([Smith et al. \(October 2021\)](#))

**UC5:** Tool owner, Dr. Spencer Smith, is not expected to change for the duration of the tool's lifespan

## 5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

**M1:** Hardware Hiding Module

**M2:** Browser Module

**M3:** Application UI Module Module

**M4:** Data Edit Module

**M5:** User Authentication Module

**M6:** User Role Access Module

**M7:** User Page Module

**M8:** Automated Metrics

**M9:** Domains Page Module

**M10:** System API Gateway Module

**M11:** Ranking Algorithm Module

**M12:** Graphing Module

**M13:** File Import Module

**M14:** File Export Module

**M15:** Repository API Module

**M16:** Comparison Module

**M17:** Database Persistence Module

**M18:** Logging Module

**M19:** Configuration Module

| <b>Level 1</b>           | <b>Level 2</b>   |
|--------------------------|--|
| Hardware-Hiding Module   | Browser Module   |
| Behaviour-Hiding Module  | Domains Page Module<br>Application UI Module<br>Data Edit Module<br>User Authentication Module<br>User Role Access Module<br>User Page Module<br>Automated Metrics Module<br>Comparison Module<br>Configuration Module |
| Software Decision Module | System API Gateway Module<br>Ranking Algorithm Module<br>Graphing Module<br>File Import Module<br>File Export Module<br>Repository API Module<br>Database Persistence Module<br>Logging Module                         |

Table 2: Module Hierarchy

## 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 4.

## 7 Module Decomposition

Modules are decomposed according to the principle of “information hiding” proposed by Parnas et al. (1984). 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. *SFWRENG 4G06 - Capstone Design Process* means the module will be implemented by the

SFWRENG 4G06 - Capstone Design Process 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

### 7.1.1 Browser Module (M2)

**Secrets:** The data structure and algorithm used to implement the browser, which is outside of the scope of the project.

**Services:** The browser allows all users of the product to view and retrieve the project through the internet, and displays the contents for use.

**Implemented By:** Web browser

## 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 Application UI Module (M3)

**Secrets:** The data structures and algorithms to display all visuals.

**Services:** Displays and renders the interactive user interface elements.

**Implemented By:** [React](#)

**Type of Module:** ADT

### **7.2.2 Data Edit Module (M4)**

**Secrets:** The data structures and algorithms to edit domain data.

**Services:** Handles and validates user inputs of domain/package/metric data.

**Implemented By:** DomainX

**Type of Module:** ADT

### **7.2.3 User Authentication Module (M5)**

**Secrets:** The data structures and algorithms used to securely store, validate and manage user credentials.

**Services:** Provides user registration, login, and session management services.

**Implemented By:** DomainX

**Type of Module:** Library

### **7.2.4 User Role Access Module (M6)**

**Secrets:** The data structures and algorithms used to store and validate users access level within the system.

**Services:** Provides user role and capabilities related to the role.

**Implemented By:** DomainX

**Type of Module:** ADT

### **7.2.5 User Page Module (M7)**

**Secrets:** The algorithms and components used to display the interactive user settings page.

**Services:** Displays user settings where user can update their settings and view information related to their account profile.

**Implemented By:** DomainX

**Type of Module:** Abstract Object

### **7.2.6 Automated Metrics Module (M8)**

**Secrets:** The data structures and algorithms used for adding automatable metrics.

**Services:** Handles the automated data entry into system using the Repository Api Module M15.

**Implemented By:** DomainX

**Type of Module:** Library

### **7.2.7 Domains Page Module (M9)**

**Secrets:** The algorithms and components used to display domain management data.

**Services:** Displays available domains and the domain contents, including it's corresponding packages, metrics, description.

**Implemented By:** DomainX

**Type of Module:** ADT

### **7.2.8 Comparison Module (M16)**

**Secrets:** The data structures and algorithm that store the package comparison methods.

**Services:** Defines available comparison methods based on user request.

**Implemented By:** DomainX

**Type of Module:** Abstract Object

### **7.2.9 Configuration Module (M19)**

**Secrets:** The data structures and algorithm that stores each individual user.

**Services:** Using the user authentication and provides interface to retrieve and update user information.

**Implemented By:** DomainX

**Type of Module:** Abstract Object

### 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 System API Gateway Module (M10)

**Secrets:** The backend apis that services the flow of business logic.

**Services:** Manages application state and serves as the central communication hub for the system.

**Implemented By:** [Django](#)

**Type of Module:** Abstract Object

#### 7.3.2 Ranking Algorithm Module (M11)

**Secrets:** AHP ranking algorithm used for package comparison.

**Services:** Computes comparative rankings using configurable methods and outputs the result.

**Implemented By:** [AHPy](#)

**Type of Module:** Abstract Object

#### 7.3.3 Graphing Module (M12)

**Secrets:** The data structures and algorithms used for graphing data.

**Services:** Takes metric data as input and outputs requested graphs.

**Implemented By:** [Matplotlib](#)

**Type of Module:** Abstract Object

#### 7.3.4 File Import Module (M13)

**Secrets:** The data structures and algorithms used to import data of varying formats (e.g. Excel, CSV).

**Services:** Facilitates the process of processing a inputted file into data for use by the rest of the system.

**Implemented By:** [pandas](#)

**Type of Module:** Abstract Object

#### 7.3.5 File Export Module (M14)

**Secrets:** The data structures and algorithms used to export data in varying formats (e.g. Excel, CSV).

**Services:** Facilitates the process of transforming system data into an exportable format (e.g. Excel, CSV).

**Implemented By:** [pandas](#)

**Type of Module:** Abstract Object

#### 7.3.6 Repository API Module (M15)

**Secrets:** API endpoints, tokens, and rate limit strategies.

**Services:** Fetches metrics and metadata from external repositories (e.g., GitHub, GitLab).

**Implemented By:** [Github API](#)

**Type of Module:** Library

#### 7.3.7 Database Persistence Module (M17)

**Secrets:** The algorithms used for interacting with the database.

**Services:** Provides database methods related to updating and querying the stored data and the connection to the database itself. The expected database design is shown in Figure 8

**Implemented By:** [MySQL](#)

**Type of Module:** Library

### 7.3.8 Logging Module (M18)

**Secrets:** The algorithms and parameters used for logging all interactions that happens with the system.

**Services:** Provides logging capabilities and logs user actions and system actions, provides a way to retrieve logged details.

**Implemented By:** [Logging \(Python\)](#)

**Type of Module:** Abstract Object

## 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      |
|------|--------------|
| FR1  | M7           |
| FR2  | M5, M7       |
| FR3  | M5           |
| FR4  | M19          |
| FR5  | M9, M6       |
| FR6  | M9           |
| FR7  | M4, M17, M14 |
| FR8  | M4           |
| FR9  | M13          |
| FR10 | M8, M15      |
| FR11 | M9           |
| FR12 | M11, M16     |
| FR13 | M12          |
| FR14 | M14          |
| FR15 | M12, M14     |

Table 3: Trace Between Functional Requirements and Modules

| Req.    | Modules      |
|---------|--------------|
| LF-AR1  | M3           |
| LF-AR2  | M2, M10      |
| LF-AR3  | M3, M12, M10 |
| LF-AR4  | M3, M9       |
| LF-AR5  | M3           |
| LF-AR6  | M3           |
| LF-SR1  | M3           |
| LF-SR2  | M12          |
| LF-SR3  | M3,          |
| UH-EU1  | M3           |
| UH-EU2  | M3           |
| UH-EU3  | M3           |
| UH-LR1  | M3           |
| UH-LR2  | M3, M9       |
| UH-UP1  | M3, M9       |
| UH-UP2  | M3, M9       |
| UH-UP3  | M3           |
| UH-AR1  | M3           |
| UH-AR2  | M3           |
| PR-SL1  | M16          |
| PR-SC1  | M5           |
| PR-SC2  | M5, M19      |
| PR-SC3  | M6, M4       |
| PR-PA1  | M8           |
| PR-RFT1 | M10          |
| PR-RFT2 | M17          |
| PR-RFT3 | M17          |
| PR-CR1  | M10          |
| PR-CR2  | M17          |
| PR-CR3  | M10, M17     |
| PR-SE1  | M3, M10, M17 |
| PR-SE2  | M5, M6       |
| PR-LR1  | M10, M17     |

| Req.    | Modules (continued) |
|---------|---------------------|
| PR-LR2  | M17                 |
| OE-EPE1 | M10                 |
| OE-EPE2 | M1, M2              |
| OE-EPE3 | M1                  |
| OE-WE1  | M2, M10, M15        |
| OE-WE2  | M1                  |
| OE-WE3  | M2                  |
| OE-IA1  | M15                 |
| OE-IA2  | M17                 |
| OE-IA3  | M12                 |
| OE-IA4  | M10, M15            |
| OE-PR1  | Not Relevant        |
| OE-PR2  | Not Relevant        |
| OE-PR3  | Not Relevant        |
| OE-PR4  | M14, M12            |
| OE-RR1  | Not Relevant        |
| OE-RR2  | Not Relevant        |
| OE-RR3  | Not Relevant        |
| OE-RR4  | Not Relevant        |
| MS-MR1  | Not Relevant        |
| MS-MR2  | Not Relevant        |
| MS-MR3  | Not Relevant        |
| MS-MR4  | Not Relevant        |
| MS-MR5  | Not Relevant        |
| MS-MR6  | Not Relevant        |
| MS-MR7  | M18, M4             |
| MS-MR8  | M3, M10, M15, M17   |
| MS-SR1  | Not Relevant        |
| MS-SR2  | Not Relevant        |
| MS-SR3  | M18                 |
| MS-SR4  | Not Relevant        |
| MS-AR1  | M10                 |
| MS-AR2  | M17                 |
| MS-AR3  | M15                 |

| Req.    | Modules (continued) |
|---------|---------------------|
| MS-AR4  | M1, M2, M10         |
| MS-AR5  | M10, M3, M17        |
| SR-AC1  | M6, M9              |
| SR-AC2  | M6                  |
| SR-AC3  | M19, M6             |
| SR-AC4  | M6                  |
| SR-AC5  | M3                  |
| SR-AC6  | M15, M17            |
| SR-INT1 | M4                  |
| SR-INT2 | M17                 |
| SR-INT3 | M4, M17             |
| SR-INT4 | M3, M4, M8, M17     |
| SR-INT5 | M17, M18            |
| SR-INT6 | M4                  |
| SR-INT7 | M14, M10, M3        |
| SR-P1   | M17, M5, M19        |
| SR-P2   | M17, M5, M19        |
| SR-AU1  | M17, M18            |
| SR-AU2  | M18                 |
| SR-AU3  | M18, M6             |
| SR-IM1  | M17                 |
| SR-IM2  | M3                  |
| SR-IM3  | M15                 |
| SR-IM3  | Not Relevant        |
| CU-CR1  | M3                  |
| CU-CR2  | M3                  |
| CU-CR3  | M3                  |
| CU-CR4  | Not Relevant        |
| CR-LR1  | Not Relevant        |
| CR-LR2  | Not Relevant        |
| CR-LR3  | Not Relevant        |

Table 4: Trace Between Non-Functional Requirements and Modules

| <b>AC</b> | <b>Modules</b> |
|-----------|----------------|
| AC1       | M4, M17        |
| AC2       | M5             |
| AC3       | M11, M16       |
| AC4       | M16            |
| AC5       | M6             |
| AC6       | M15            |
| AC7       | M12            |
| AC8       | M14, M13       |

Table 5: Trace Between Anticipated Changes and Modules

## 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.

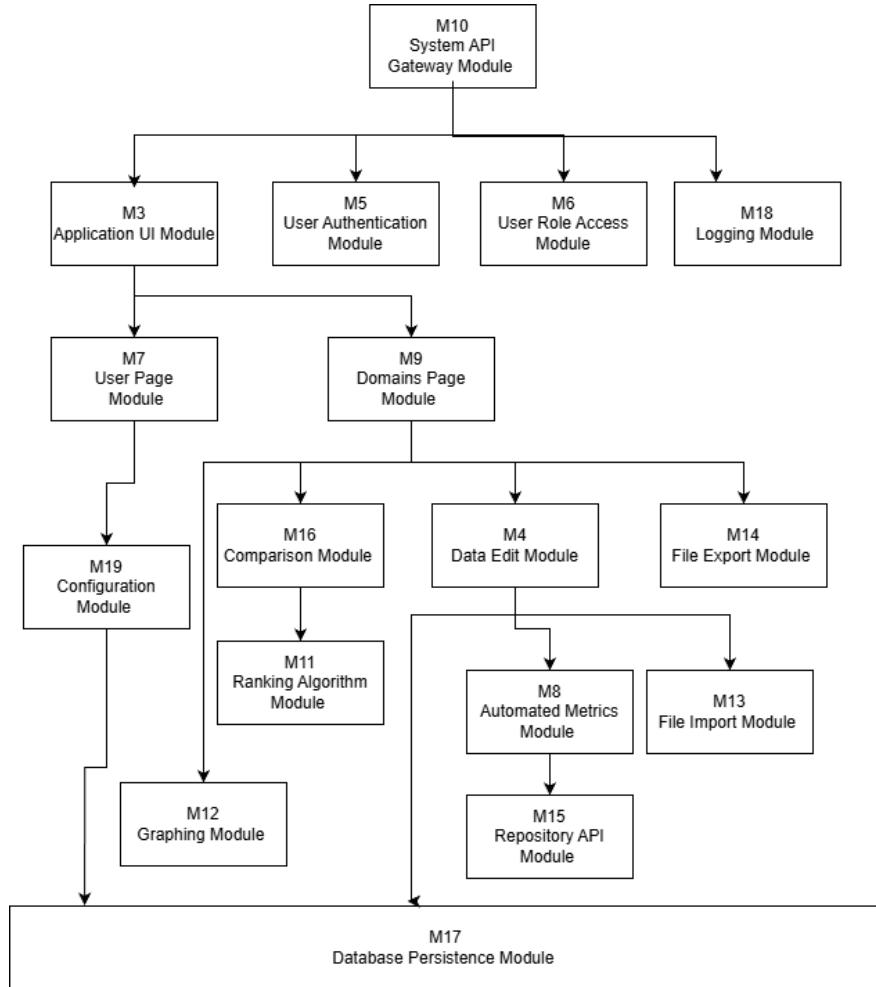


Figure 1: Use hierarchy among modules

## 10 User Interfaces

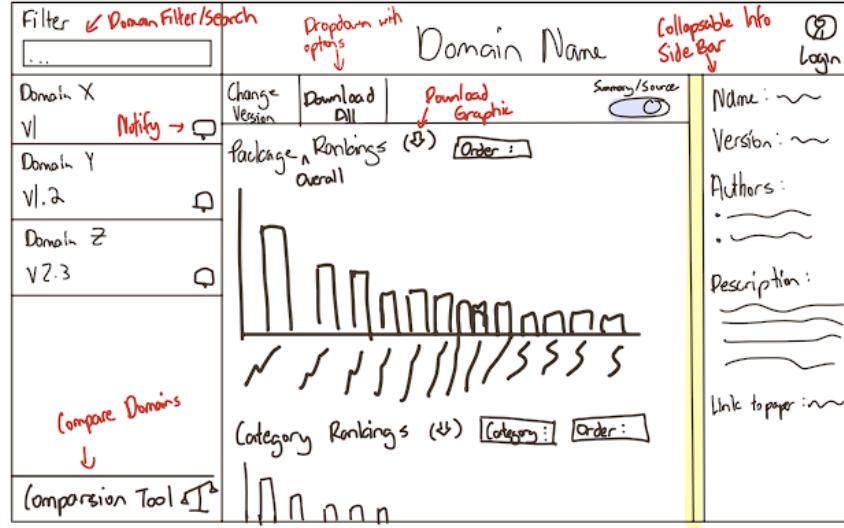


Figure 2: Main domain summary view page

Figure 2 shows the main page users of any role can see. This page allows the user to filter published domains, compare domains, and view domain details. The summary view of the domain allows users to quickly view the graphs associated with the data.

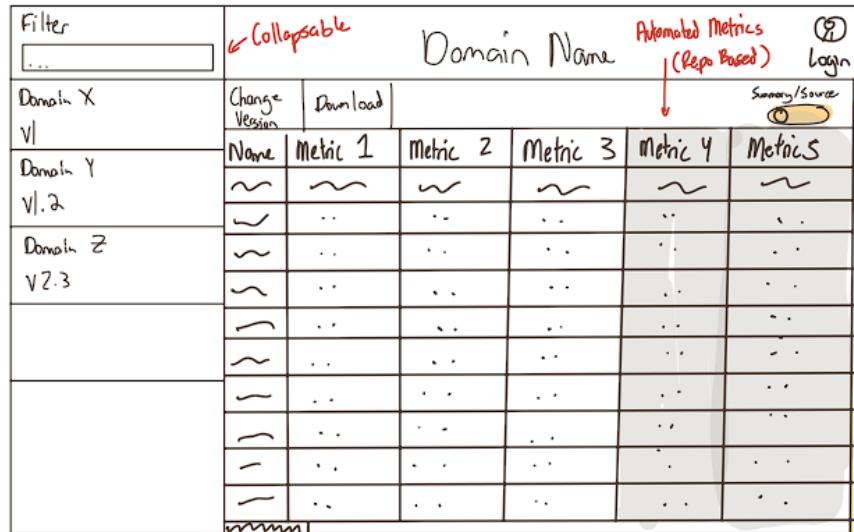


Figure 3: Main domain data view page

Figure 3 shows the secondary source view of a domain. This allows users to see the underlying metric data used to display the graphs from the summary view.

Currently Editing

| Exit                                     | Save | Domain |     |          |          |          |          | Unsaved | Saved |
|--|------|--------|-----|----------|----------|----------|----------|---------|-------|
|  |      | Name   | Vrl | Metric 1 | Metric 2 | Metric 3 | Metric 4 |         |       |
| Automation Tool                          |      | ~      | ~~  |          |          |          |          |         |       |
| Selected Packages                        |      | ~      | ~~  |          |          |          |          |         |       |
| • P1                                     |      | ~      | ~~  |          |          |          |          |         |       |
| • P2                                     |      | ~      | ~~  |          |          |          |          |         |       |
| • P3                                     |      | ~      | ~~  |          |          |          |          |         |       |
| Metric                                   |      |        |     |          |          |          |          |         |       |
| • M3                                     |      |        |     |          |          |          |          |         |       |
| • M4                                     |      |        |     |          |          |          |          |         |       |
| Notification                             |      |        |     |          |          |          |          |         |       |
| <input type="checkbox"/> Email: ~        |      |        |     |          |          |          |          |         |       |
| <input type="checkbox"/> Phone: xxx-1111 |      |        |     |          |          |          |          |         |       |
| <input type="button" value="Start"/>     |      |        |     |          |          |          |          |         |       |

Figure 4: Domain editing

Figure 4 shows the domain editing view accessible for collaborator roles and above (admin, super admin). This page allows user to enter metric data manually and triggering the automation tool to input data automatically for the selected packages and metrics. Automatable metrics and data are indicated with a different colour.

| Back  | Domain X |         | Description                          |                |
|---|----------|---------|--------------------------------------|----------------|
| Information                                 | Version  | : 1.0   | Domain Expert: Sam Max xxx@gmail.com |                |
| Preference                                  | Process  |         |                                      | Initial List   |
| Your Domains                                | Edit     | Publish | • Domain Expert meeting              | • Initial List |
| Notifications                               |          |         | • ~                                  |                |
| <input type="button" value="Request Role"/> |          |         |                                      |                |

Figure 5: Domain management (collaborator role)

Figure 5 shows the domain management page that a collaborator can see. This page allows them to quickly glance information about the domains they are working on, and the current process of the domains. As well as the option to publish domains.

| <input type="button" value="Back"/> | Name                                | Role | Domain      | Edit                 | Privileges | Email         |
|-------------------------------------|-------------------------------------|------|-------------|----------------------|------------|---------------|
| Information                         | <input type="button" value="Edit"/> | ~~~  | Super Admin | *                    |            | xxx@gmail.com |
| Preference                          | <input type="button" value="Edit"/> | ~~   | Contributor | Domain X<br>Domain Y |            | xxx@gmail.com |
| Your Domains                        | <input type="button" value="Edit"/> | ~~   | Viewer      |                      |            | xxx@gmail.com |
| Notifications                       | <input type="button" value="Edit"/> | ~~~  | Admin       | *                    |            | xxx@gmail.com |
| Admin                               |                                     |      |             |                      |            |               |
| API Keys                            |                                     |      |             |                      |            |               |
|                                     |                                     |      |             |                      |            | .             |

Figure 6: User management (super admin role)

Figure 6 shows the user management page, this page is visible for admin and super admins. Allows admins to edit the role access level of current users, user information, and sending invites to directly invite users.

| <input type="button" value="Back"/> |   |
|-------------------------------------|---|
| Information                         |  |
| Preference                          | Jane Doe  |
| Your Domains                        | Role : Contributor  |
| Notifications                       | Change  |
|                                     | Language : English  |
|                                     | Email : xxx@email.com   |
|                                     | Password : * * * *  |
|                                     | <input type="button" value="Edit Information"/>                                     |
| Request Role                        |   |

Figure 7: Main user information page (viewer role)

Figure 7 shows the main user information page that any users can see. This apge allows users to set their basic information. Additional side columns are added based on the role access level.

## 11 Design of Communication Protocols

N/A

## 12 Timeline

The following is the timeline for development of the modules, building up from the POC. Issues related to the timeline will be created and tracked through [Github Issues](#) and the associated [project board](#).

| Week | Dates           | Assigned To | Deliverables (Module Focus)   |
|------|-----------------|-------------|---|
| 1    | Jan 5 – Jan 11  | Haniye      | Implement <b>User Authentication Module (M5)</b> for secure login, registration, token management, and password encryption. |
|      |                 | Ghena       | Implement <b>User Role Access Module (M6)</b> with Admin/Analyst/Viewer roles and middleware for route protection.          |
|      |                 | Fei         | Setup Logging Module (M18)  |
|      |                 | Fei         | Setup testing framework   |
|      |                 | Awurama     | Build User Page (M7)  |
| 2    | Jan 12 – Jan 18 | Haniye      | Extend backend to support <b>Database Persistence (M17)</b> : ORM models, migrations, CRUD operations.                      |
|      |                 | Ghena       | Expand <b>System API Gateway (M10)</b> to integrate Auth & Repo APIs, include error-handling layer.                         |
|      |                 | Awurama     | Expand user page with <b>configurable functionalities (M19)</b> based on user role  |
| 3    | Jan 19 – Jan 25 | Awurama     | Build <b>Application UI Module (M3)</b> with global navigation, sidebar, responsive design, and state handling.             |
|      |                 | Fei         | Implement <b>Domains Page Module (M9)</b> to display domains and metrics with integrated backend data.                      |
|      |                 | Ghena       | Finalize <b>Repository API Module (M15)</b> : error handling, caching, and rate-limit retries.                              |
|      |                 | Fei         | Build <b>File Export Module (M14)</b> to export domain data in CSV/XLSX formats.  |
| 4    | Jan 26 – Feb 1  | Awurama     | Develop <b>File Import Module (M13)</b> to upload, validate, and parse CSV/XLSX files.                                      |

| Week | Dates          | Assigned To    | Deliverables (Module Focus)  |
|------|----------------|----------------|--|
|      |                | Haniye         | Implement <b>Automated Metrics Module (M8)</b> with background job scheduler and automatic metric retrieval. |
| 5    | Feb 2 – Feb 8  | Awurama + Fei  | Implement <b>Data Edit Module (M4)</b> with React validation, inline feedback, and API integration.          |
|      |                | Ghena          | Implement <b>Ranking Algorithm Module (M11)</b> using AHP/BWM/SSB methods with test coverage.                |
|      |                | Haniye         | Develop <b>Graphing Module (M12)</b> for data visualizations using <a href="#">Matplotlib</a> .              |
| 6    | Feb 9 – Feb 15 | Ghena + Haniye | Build <b>Comparison Module (M16)</b> for cross-domain and in-domain analysis, visualization, and reporting.  |
|      |                | Awurama + Fei  | Conduct accessibility and usability testing.   |
| 9-11 | Mar 2 – Mar 23 | Awurama + Fei  | Conduct full end-to-end testing; finalize documentation and update traceability links.                       |
|      |                | All Developers | Code freeze and integration testing; bug fixes, performance optimization, and final release demo.            |

## 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.
- Spencer Smith, Jacques Carette, Peter Michalski, Ao Dong, and Olu Owojaiye. Methodology for assessing the state of the practice for domain x. *arXiv preprint arXiv:2110.11575v1*, abs/2110.11575:1–15, October 2021. URL <https://arxiv.org/abs/2110.11575>.

# Appendix A: Database Schema

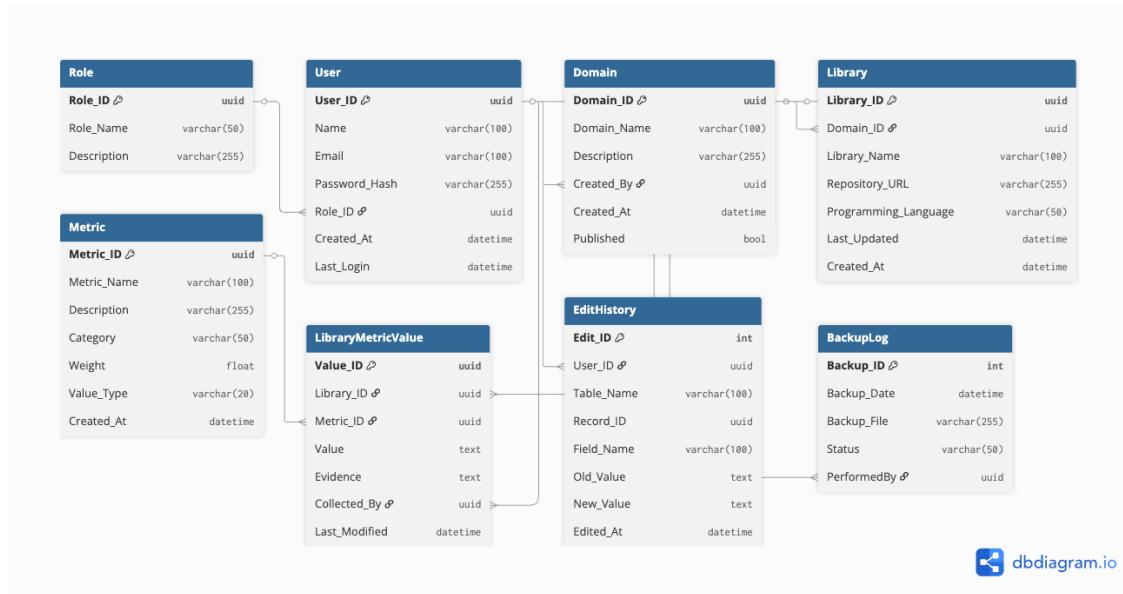


Figure 8: Domain X Database Schema