****

**SOFTWARE DESIGN DOCUMENT FOR**

**LOST AND FOUND MANAGEMENT SYSTEM**

**GROUP-B**

**Date: 18TH/4/2025**

|  |  |  |
| --- | --- | --- |
| **Name** | **Registration Number** | **Student Number** |
| Ssenoga Herman | 23/U/17648/EVE | 2300717648 |
| Nyonyozi Maria Lisa Loyce | 23/U/16424/EVE | 2300716424 |
| Nabirye Anita | 23/U/12826/EVE | 2300712826 |
| Mugoya Yusuf | 23/U/11615/EVE | 2300711615 |
| Walusimbi Ashraf | 23/U/18466/EVE | 2300718466 |

TABLE OF CONTENTS

[1. INTRODUCTION 4](#_Toc196342319)

[1.1 Purpose 4](#_Toc196342320)

[1.2 Scope 4](#_Toc196342321)

[1.3 Document Overview 5](#_Toc196342322)

[1.3 Reference Material 6](#_Toc196342323)

[1.4. Definitions and Acronyms 7](#_Toc196342324)

[2. SYSTEM OVERVIEW 8](#_Toc196342325)

[2.1 System Description 8](#_Toc196342326)

[2.2. Software requirements 9](#_Toc196342327)

[2.3 Hardware Requirements 9](#_Toc196342328)

[3. SYSTEM ARCHITECTURE 11](#_Toc196342329)

[3.1 Architectural Design 11](#_Toc196342330)

[3.2 Decomposition Description 14](#_Toc196342331)

[3.3 Design Rationale 18](#_Toc196342332)

[4. DATA DESIGN 20](#_Toc196342333)

[4.1 Data description 20](#_Toc196342334)

[4.2 Data dictionary 23](#_Toc196342335)

[5. COMPONENT DESIGN 24](#_Toc196342336)

[5.1 Item Management Component 24](#_Toc196342337)

[Post lost or found item 24](#_Toc196342338)

[Submit claim for an item 25](#_Toc196342339)

[Approve Claim 25](#_Toc196342340)

[6. HUMAN INTERFACE DESIGN. 26](#_Toc196342341)

[6.1 Overview of User Interface 26](#_Toc196342342)

[6.1.1 Administrator Interface 26](#_Toc196342343)

[6.1.2 Student Interface 26](#_Toc196342344)

[6.2 Screen Images 27](#_Toc196342345)

**LIST OF TABLES**

[Table 1:Acronyms 8](#_Toc196342296)

[Table 2: Definitions 8](#_Toc196342297)

[Table 3: Data Description of the System 21](#_Toc196342298)

[Table 4: User data 24](#_Toc196342299)

[Table 5: Post data 24](#_Toc196342300)

**LIST OF FIGURES**

[Figure 3.1 1: System Architectural Design 13](#_Toc196305632)

[Figure 3.2 1: Use Case Diagram of the System 16](#_Toc196305619)

[Figure 3.2 2: Sequence Diagrams of the System 17](#_Toc196305620)

[Figure 3.2 3: Activity Diagram of Administrator 18](#_Toc196305621)

[Figure 3.2 5: Activity diagram of interaction between student and the System 19](#_Toc196305622)

[Figure 4.1 1: Class Diagram of the System 23](#_Toc196305611)

[Figure 4.1 2: ERD for the System 24](#_Toc196305612)

[Figure 6.2 2: Student dashboard Interface 28](#_Toc196342279)

# 1. INTRODUCTION

## 1.1 Purpose

This software design document describes the necessary information required to effectively describe the architecture and system design in order to give the system developers and testers guidance on the architecture of the Lost and Found Management System. The system is to enable Makerere University students to report any lost or found items by enabling them post information on the website of the system and also enable students to recover their lost items. The goal of this document is to give the development team a better understanding of how the system is to be developed.

## 1.2 Scope

The scope of the system design is as follows:

This SDD details the design for:

* A web interface developed using PHP and the Laravel framework, intended for users to upload information about lost or found items and also be able to directly message other users.
* A database system that stores information about users and lost or found items.
* A web interface developed using PHP and the Laravel framework, intended for administrators to manage the lost or found items, approve item claims, and generate reports.

The system aims to provide a seamless experience for all users, from registration to the normal use of the system, and through to the generation of analytics and reports.

It will as well generate reports including items that have been lost in certain period of time and items found in a given amount of time, the users of the system, and the users that post any lost or found items. It will enable administrators approve items being claimed by individuals.

**Goal**

The goal of the Lost and Found Management System is to provide a digital platform for university students to report, search, and claim lost or found items in a secure and organized manner, minimizing loss and enhancing item recovery.

**Benefits**

* Reduces the time and effort required to locate lost items.
* Prevents fraudulent claims through admin-based approval workflows.
* Encourages accountability and trust within the university community.
* Offers 24/7 access to item reports from any device.
* Eliminates the need for traditional bulletin boards or manual tracking systems.

**Objectives**

* Enable students to post lost or found items using a user-friendly interface.
* Allow users to upload images and item descriptions.
* Provide a system for administrators to verify and approve claims of ownership.
* Send notifications to users when a match or update is made on an item.
* Maintain a searchable record of all lost and found items.
* Ensure system access is restricted based on user roles (students, admins).

## 1.3 Document Overview

This document consists of 6 sections.

Section 1 is the introduction that describes a brief overview of the document and its significance to the project. It also consists of subsections like purpose section which explains the goals, objectives and benefits of the system, overview section which explains all the different parts of the document, reference material section which lists all the citations and references, and the definitions and acronyms section which explains terms used in this document.

Section 2 of this document is system overview which gives a general description of the functionality, context and design of the project and provide any background information if necessary.

Section 3 is the system architecture which contains the architectural design, decomposition description and the design rationale. This section breaks down the project into smaller modules illustrating them in terms of interface specifications, hierarchical diagrams and more.

Section 4 is the data design that contains the data description which explains how the information domain of the Lost and found management System is transformed into data structures and the data dictionary which contains the major data along with their types and descriptions.

Section 5 is the component design which contains a closer look at what each component does in a more systematic way.

Section 6 is the human interface design which contains the overview of the user interface, screen images, and screen objects and actions. The overview of the user interface describes the functionality of the system from the users’ perspective. The screen images display screenshots showing the interface from the users’ perspective. The screen objects and actions details a discussion of screen objects and actions associated with those objects.

## 1.3 Reference Material

L. Bass, P. Clements, and R. Kazman. Software Architecture in

Practice 2nd ed. Addison Wesley, 2003.

J. Burge, J. Carroll, R. McCall, and I. Mistrík. Rationale-Based

Software Engineering. Springer-Verlag, 2008.

J. S. van der Ven, A. Jansen, J. Nijhuis, and J. Bosch. Design

Decisions: The Bridge between Rationale and Architecture

R. S. Pressman, Software Engineering A Practitioner’s Approach 5th ed. 2001.

## 1.4. Definitions and Acronyms

**Acronyms**

Table 1:Acronyms

|  |  |
| --- | --- |
| Term | Details |
| DB | Database |
| UI | User Interface |
| SQL | Structured Query Language |
| SDD | Software Design Document |

**Definitions**

Table 2: Definitions

|  |  |
| --- | --- |
| Definition | Meaning |
| Sequence Diagram: | Shows the sequence of messages passed between objects of the system. |
| Use case Diagram: | This visually represents the functional interactions between actors and a system |
| Class Diagram: | A static structure diagram that describes that shows classes, their attributes, methods and relationships among objects. |
| Activity Diagram: | This visually represents the flow of activities or actions in a system, showing how processes and decisions lead to various outcomes. |

# 2. SYSTEM OVERVIEW

## 2.1 System Description

The Lost and Found Management System is a web-based application developed using the Laravel framework, designed to streamline the process of reporting, managing, and recovering lost or found items within a university environment. Inspired by the social media model, the system provides an intuitive interface where students can post items they have lost or found, accompanied by images and relevant descriptions.

Users interact with the system by submitting posts tagged as either “lost” or “found,” which are then publicly visible to the university community. The system allows other students to view these posts and contact the original poster if there is a potential match.

To ensure integrity and prevent misuse, item claims must be reviewed and approved by system administrators before final handover. Administrators are responsible for validating claim details and verifying ownership before approving or rejecting claims.

The system also includes features such as user role management, search and filter options for locating specific items, and notification mechanisms to alert users of updates related to their posts or claims.

## 2.2. Software requirements

* Laravel: Laravel is a popular PHP framework used for web application development. It offers a syntax that is clean and easy to understand, which makes it an excellent choice to write expressive and readable code for the system.
* Laravel provides a flexible routing system and supports middleware, which can be used to filter HTTP requests entering your application.
* A Database management system: DBMS software that enables the creation, management, and manipulation of databases. DBMSs will provide efficient and organized data management, reducing redundancy and inconsistency.
* Apache Web server: A Web server will efficiently handle HTTP requests and responses, enabling web applications to communicate with users. It is highly customizable, supports a wide range of features through extensions and modules.

## 2.3 Hardware Requirements

* Servers: as hardware components they play a pivotal role in modern computing and offer numerous advantages that drive the efficiency, reliability, and scalability of IT infrastructure.
* Servers are designed for high reliability and uptime, often featuring redundant components like power supplies, network interfaces, and storage. This redundancy ensures that the server remains operational even if one component fails, minimizing downtime.
* **Processor (CPU):** The CPU will handle all the computations and processing tasks. A powerful, multi-core CPU enhances the system's ability to handle multiple tasks simultaneously, improving overall speed and responsiveness.
* **Power Supply Unit (PSU):** The PSU will provide power to all server components. A reliable PSU with sufficient wattage and redundancy will ensure stable operation, minimizing downtime and hardware failures.
* **Solid-State Drives (SSDs):** SSDs will provide much faster read/write speeds leading to quicker data retrieval and improved system responsiveness.
* **High-Speed Network Interface Card (NIC):** this will ensure high-speed and low-latency network connectivity, which is essential for fast access to the system by users.

# 3. SYSTEM ARCHITECTURE

This section discusses the architectural structure for the system. It describes the architectural design, decomposition description for each system feature and the justification for the type of architecture chosen.

## 3.1 Architectural Design

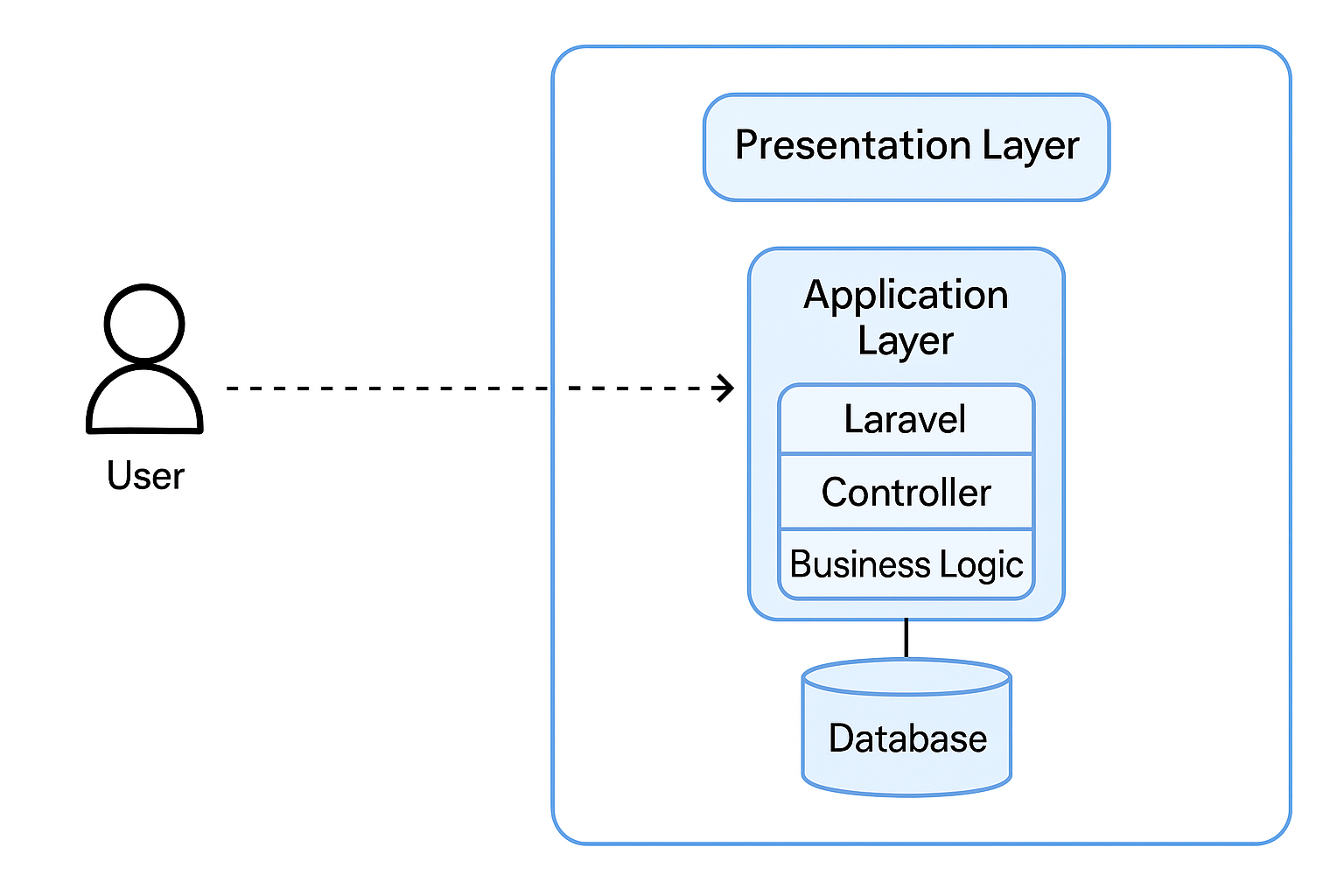


Figure 3.1 1: System Architectural Design

**Architecture Description**

The Lost and Found Management System follows a three-tier architecture, consisting of:

1. Presentation Layer (Frontend)
2. Application Layer (Backend/Business Logic)
3. Data Layer (Database)

Each layer is responsible for a specific part of the system’s operation:

1. Presentation Layer (Frontend)

* Built using Laravel Blade templates and optionally JavaScript for interactivity.
* This layer handles user interaction: submitting lost/found posts, searching, claiming items, and admin actions.
* Responsive design ensures usability across different desktop screens.

2. Application Layer (Business Logic)

* Implemented in Laravel using MVC (Model-View-Controller) structure.
* Controllers process user requests, interact with models, and return appropriate views or responses.
* Middleware handles authentication, authorization, and route protection based on user roles.

3. Data Layer (Database)

* MySQL is used to store persistent data: users, posts, claims, notifications, etc.
* Eloquent ORM is used for database interactions.
* Data is validated, sanitized, and secured to ensure integrity.

Component Interaction Overview

Here's a typical flow:

* A student logs in and creates a "lost item" post (Presentation → Controller → Model → DB).
* Other users view this post and submit a claim).
* Admin reviews claim requests and approves/rejects them (Admin Controller → Claim Model → DB).
* System sends notifications using Laravel’s notification system (email or in-app).

Tech Stack Summary

| Layer | Technologies Used |
| --- | --- |
| Presentation | HTML, CSS, Laravel Blade, JavaScript |
| Application | Laravel (PHP), MVC Pattern |
| Data | MySQL, Eloquent ORM |
| Others | Laravel Auth, Laravel Notifications |

## 3.2 Decomposition Description

Use Case Diagram of the system.



Figure 3.2 1: Use Case Diagram of the System

This figure describes the various actors (administrators and students) and how they interact with the system.

Sequence Diagrams for the System.





Figure 3.2 2: Sequence Diagrams of the System

The figure above shows a sequence diagram showing message passing between different system objects.

Activity diagrams of the system

The figures below show activity diagrams representing the interaction between the student, administrator, and the system.

1. An activity diagram showing the interaction between the administrator and the system.



Figure 3.2 3: Activity Diagram of Administrator

1. An activity diagram showing the interaction between the student and the system.



Figure 3.2 5: Activity diagram of interaction between student and the System

## 3.3 Design Rationale

* **Architecture Chosen: MVC (Model-View-Controller) Architecture**
  + Rationale: The MVC architecture is widely used in Laravel applications and provides a clear separation between business logic (Model), user interface (View), and input control (Controller). This architecture simplifies maintenance and allows for scalable, modular design, which is crucial for a system that might grow as the number of students and items increases.
  + It helps ensure that the backend logic (like user authentication, posting and claiming items) is well-separated from the presentation layer (the user interface) and user input handling.
* **Backend Framework: Laravel**
  + Rationale: Laravel is a robust PHP framework that provides built-in tools like authentication, routing, and database migrations. It supports rapid development, which is beneficial for a project with limited time constraints (as in a university environment). Laravel’s ORM (Eloquent) simplifies database interactions, making it easier to handle relationships between items, users, and claims.
  + Laravel’s built-in functionalities like authentication, form validation, and database migration were key to developing a secure and maintainable application without needing to reinvent common web application features. Additionally, Laravel’s extensive documentation and large developer community make it easier to troubleshoot and get support.
* **Database: MySQL**
  + Rationale: Relational databases like MySQL are well-suited for the structured nature of the data (users, items, claims, etc.). The relationships between entities (e.g., a user can post multiple items, and an item can have one claim) are easily represented using foreign keys and relational tables in MySQL.
  + MySQL is widely used, well-documented, and provides performance benefits when handling structured data. It also integrates seamlessly with Laravel.
* **Frontend Framework: Laravel Blade**
  + Rationale: Laravel Blade is a simple templating engine that helps render dynamic views. For the UI, using Blade ensures tight integration with the backend and allows for easy rendering of data from the controllers.
  + Blade is easy to use with Laravel and does not require additional setup or complexity.
* **User Authentication: Laravel’s built-in authentication system**
  + Rationale: Laravel provides a robust authentication system that includes features like password hashing, user registration, and session management.
  + It saves time, provides secure authentication out of the box, and ensures that students and administrators can log in securely. Additionally, Laravel includes built-in CSRF protection to prevent cross-site request forgery attacks.

# 4. DATA DESIGN

## 4.1 Data description

Table 3: Data Description of the System

|  |  |
| --- | --- |
| Entity | Attribute |
| User | id (PK)  name  email  username  image\_name  password  is\_admin  status  is\_delete |
| Post | id (PK)  user\_id (FK)  description  location  lost  found |
| message | id (PK)  conversation\_id  sender\_id FK)  receiver\_id (FK)  body |
| Administrator | email(PK)  password  name |



Figure 4.1 1: Class Diagram of the System



Figure 4.1 2: ERD for the System

## 4.2 Data dictionary

**User**

Table 4: User data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Data Type | Size | Key | Description |
| name | VARCHAR | 40 |  | Name of the user |
| id | INT |  | PK | Unique identifier of user |
| email | VARCHAR | 30 |  | Email of the user |
| username | VARCHAR | 30 |  | Name that is visible to other users |
| password | VARCHAR | 100 |  | Password used to login |
| Image\_name | VARCHAR | 30 |  | Path to user image |
| Is\_Admin | BOOLEAN |  |  | Specifies if a user is an admin or a student |

**Post**

Table 5: Post data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Data Type | Size | Key | Description |
| id | INT |  | PK | Unique identifier for a post |
| User\_id | INT |  | FK | Id of the user that made the post |
| description | VARCHAR | 100 |  | Description of the lost or found item |
| location | VARCHAR | 30 |  | Place where the item was found or expected to be lost |
| type | VARCHAR | 6 |  | Classifies the item as either lost or found |

# 5. COMPONENT DESIGN

This section describes the system designs for the Lost and found management System using the procedural description language which elaborates the algorithms that can be used to implement the major system features.

## 5.1 Item Management Component

### Post lost or found item

|  |
| --- |
| FUNCTION postItem(userId, title, description, image, category, type, location)  IF user is authenticated THEN  CREATE new Item  SET item.title = title  SET item.description = description  SET item.image = image  SET item.category = category  SET item.type = type // 'lost' or 'found'  SET item.location = location  SET item.status = 'posted'  SET item.postedBy = userId  SAVE item to database  RETURN success message  ELSE  RETURN error: "User not authenticated"  END FUNCTION |

### Submit claim for an item

|  |
| --- |
| FUNCTION submitClaim(userId, itemId, reason)  IF item.status IS NOT 'claimed' THEN  CREATE new Claim  SET claim.itemId = itemId  SET claim.userId = userId  SET claim.reason = reason  SET claim.status = 'pending'  SET claim.date = current date  SAVE claim to database  RETURN success message  ELSE  RETURN error: "Item already claimed"  END FUNCTION |

### Approve Claim

|  |
| --- |
| FUNCTION approveClaim(adminId, claimId)  FETCH claim FROM database WHERE claim.id == claimId  IF claim.status == 'pending' THEN  SET claim.status = 'approved'  SET claim.approvedBy = adminId  SET item.status = 'claimed'  SAVE changes to claim and item  RETURN success message  ELSE  RETURN error: "Claim already processed"  END FUNCTION |

# 6. HUMAN INTERFACE DESIGN.

## 6.1 Overview of User Interface

This overview outlines the user interfaces required for the lost and found management system. The system will have distinct interfaces for administrators (web-based), and Students, each designed to facilitate their specific roles. The administrator interface will manage displaying analytics about posts, reviewing item claims as well as communication among administrators, while student interfaces will support uploading posts, messaging, claiming items as well as viewing profiles of other users.

Here’s a detailed overview:

### 6.1.1 Administrator Interface

**Login Interface:** This field displays Username and Password. It secures login for administrators

**Dashboard:**

The following fields are displayed;

* Claims
* Analytics for lost and found items
* Users of the system

### 6.1.2 Student Interface

**Login Interface:** This field displays Username and Password. Its function is to secure login for students.

Dashboard:

The dashboard is made of different UI components that include:

* Notification area.
* Users followed.
* Post section.
* Profile area
* Recently posted items

## 6.2 Screen Images

The system shall provide friendly user interfaces to provide the best interactive sessions with its users. Users shall be able to interface with the system through a browser provided they have internet connection.

Below are some of the user mockups for the System.

**Home page Interfaces**

Students who have logged into the system shall have access to the page below after login.

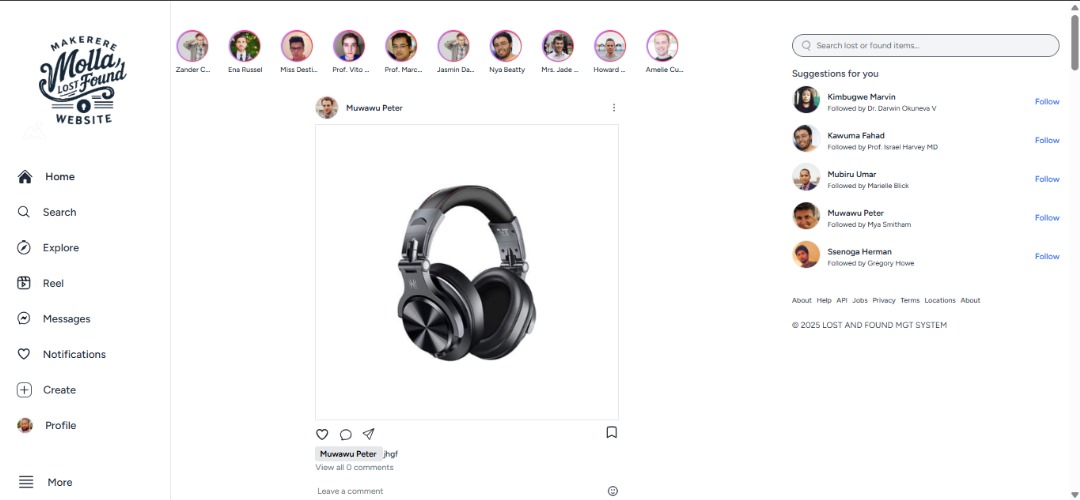
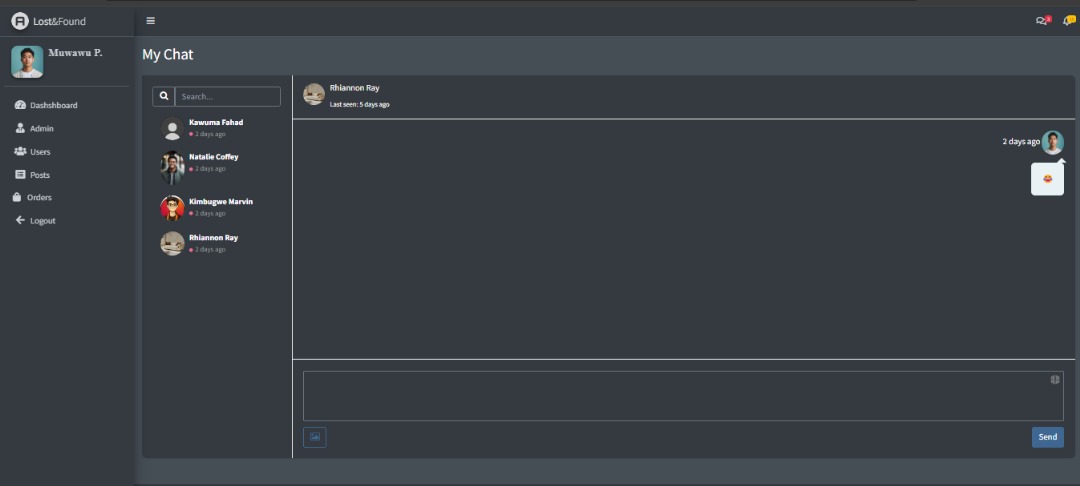


Figure 6.2 2: Student dashboard Interface

**Administrator dashboard**



**Administrator chat interface**

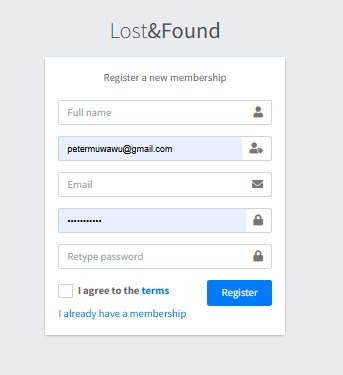


**Administrator log in page**

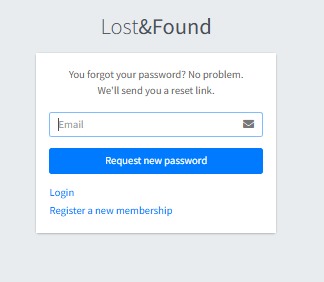
A screenshot of a login form

AI-generated content may be incorrect.

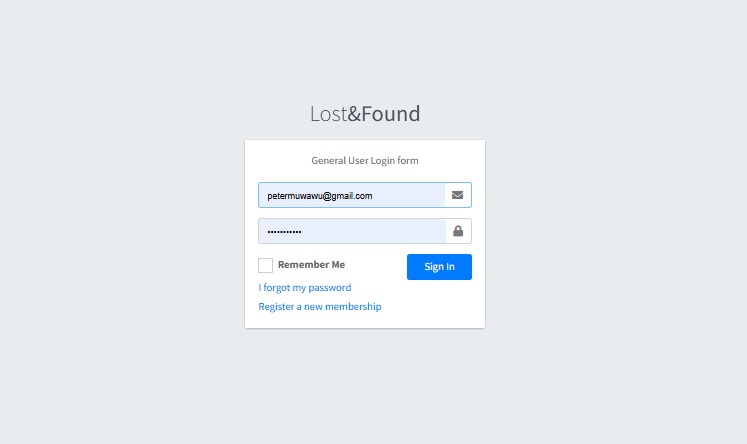
**Student registration page**



**Password reset page**



**Student log in page**



**System administrators page**

