# SOFTWARE DESIGN DOCUMENT


**Campus Connect**

**Submitted to:** Sir Shahzad Khan

**Submitted by:** Tech Bugs

| **Name** | **Roll No** |
|---|---|
| Farzeen Batool | F21BB111 |
| Muhammad Hassan | F21BB126 |
| Ardeen Zaib | F21BB142 |
| Rabeet Arif | F21BB143 |

## Table of Contents

# Introduction

- ## Purpose of the Document

This document provides a comprehensive software design for the Student Resource Portal developed as part of the Final Year Project. It outlines the architectural and detailed component-level design decisions made to implement the functionalities defined in the Software Requirements Specification (SRS). The document serves as a guide for developers, testers, and stakeholders to understand the structural and technical aspects of the system.

- ## Scope

As detailed in the SRS, the Student Resource Portal is a web-based system designed to centralize and simplify student access to academic records, attendance tracking, personal information, and shared resources. The system supports multiple academic programs within an institute and offers role-based access for students, faculty, and administrators. This design document elaborates on how the specified requirements will be translated into a functional and scalable software product.

- ## Intended Audience

This document is intended for:

- **Project Supervisors and Evaluators** – to assess the technical design of the portal.
- **Developers** – to use as a blueprint for building the system.
- **Testers** – to understand the system's structure and logic for test planning.

# Architectural Design

- ## System Architecture Overview

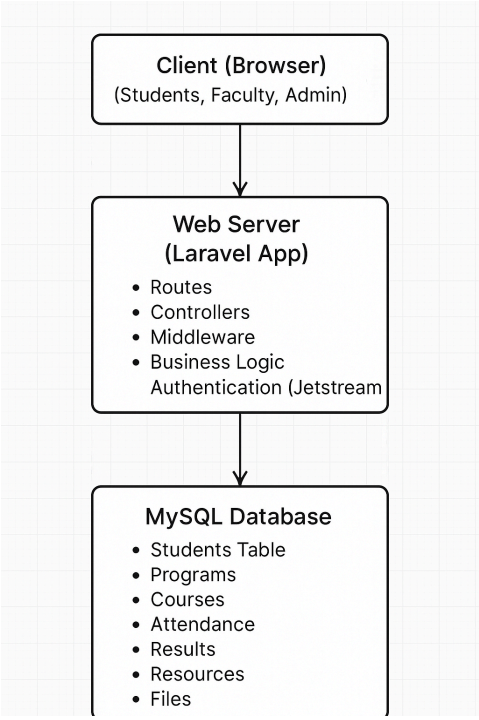
The Student Resource Portal follows a **Client-Server** architecture. The client interacts with the server through a RESTful API interface, while the server handles business logic, database interactions, and authentication.

The system consists of the following layers:

- **Presentation Layer**: Front-end developed using Tailwind CSS and Jetstream.
- **Application Layer**: Backend powered by Laravel, handling routing, logic, and API endpoints.
- **Data Layer**: MySQL database storing user data, academic records, attendance, and resources.

This architecture ensures simplicity and scalability for academic environments, where centralized control and straightforward deployment are priorities.

- ## High-Level Architecture Diagram



- **Technology Stack**

| Layer | Technology |
|---|---|
| **Frontend** | Tailwind CSS, Blade (Laravel UI), Jetstream |
| **Backend** | PHP (Laravel Framework) |
| **Database** | MySQL |
| **Authentication** | Laravel Jetstream with Sanctum |
| **Server Environment** | Apache / Nginx (LAMP/LEMP Stack) |
| **Operating System** | Linux-based (Ubuntu preferred) |
| **Version Control** | Git (GitHub) |

## ● Module Design

This section outlines the design of each major module within the Student Resource Portal. Each module is structured to encapsulate its own responsibilities while interacting seamlessly with others through well-defined interfaces and shared layout components.

### 1. Profile Module

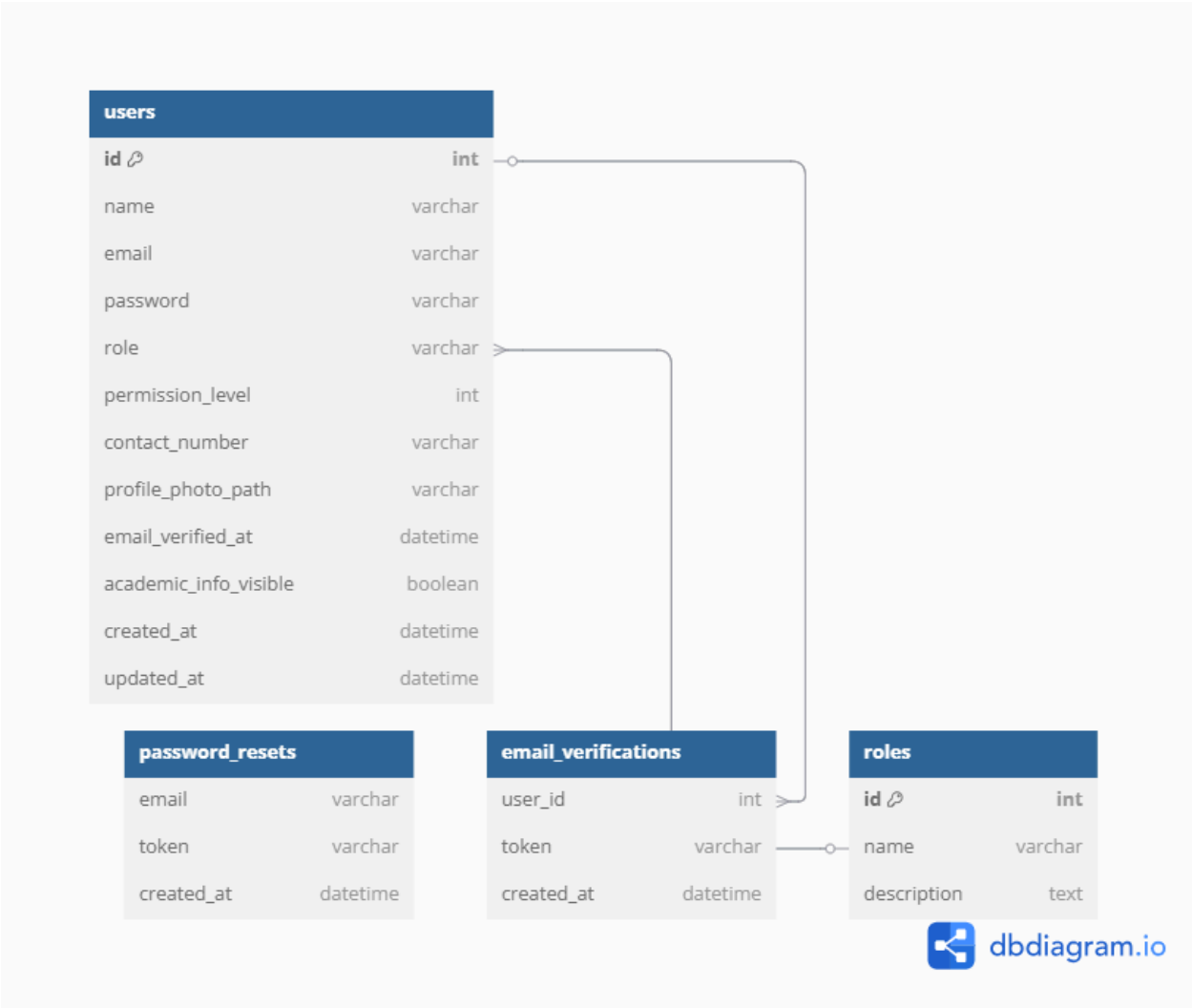
**Responsibilities**: Allows users to view and update their profile information.
**Features**:

- Name, contact, profile photo
- Password change
- Academic info visibility

**Key Components**

- `ProfileController`
- Blade views under `resources/views/profile/`

**Entity Relationship Diagram:**




## 2. Admin Module

**Responsibilities**: Admin-exclusive functions for managing users, posts, resources, and portal-wide configurations.
**Features**:

- User management dashboard
- Post/resource moderation
- Academic data entry/edit

**Key Components**:

- `AdminController`
- Admin-only views (Blade templates with access control)

## 3. Main Layout Module

**Responsibilities**: Shared layout used across all pages, includes navigation, headers, footers, and notifications.
**Features**:
Responsive navbar with role-aware links

- Notification popups or dropdowns
- Dark/light mode toggle (optional)

**Key Components**:

- Blade templates in `layouts/app.blade.php`
- Includes for `nav`, `footer`, `notifications`

## 4. Users Module

**Responsibilities**: Manages user data and access control.
**Features**:

- Registration/login (Jetstream)
- Role assignment (e.g., student, admin, faculty)
- Email verification & password reset

**Key Components**:

- `User` Model
- `UserController` (for admin control)
- Middleware for role-based access

**Entity Relationship Diagram:**


## 5. Posts Module

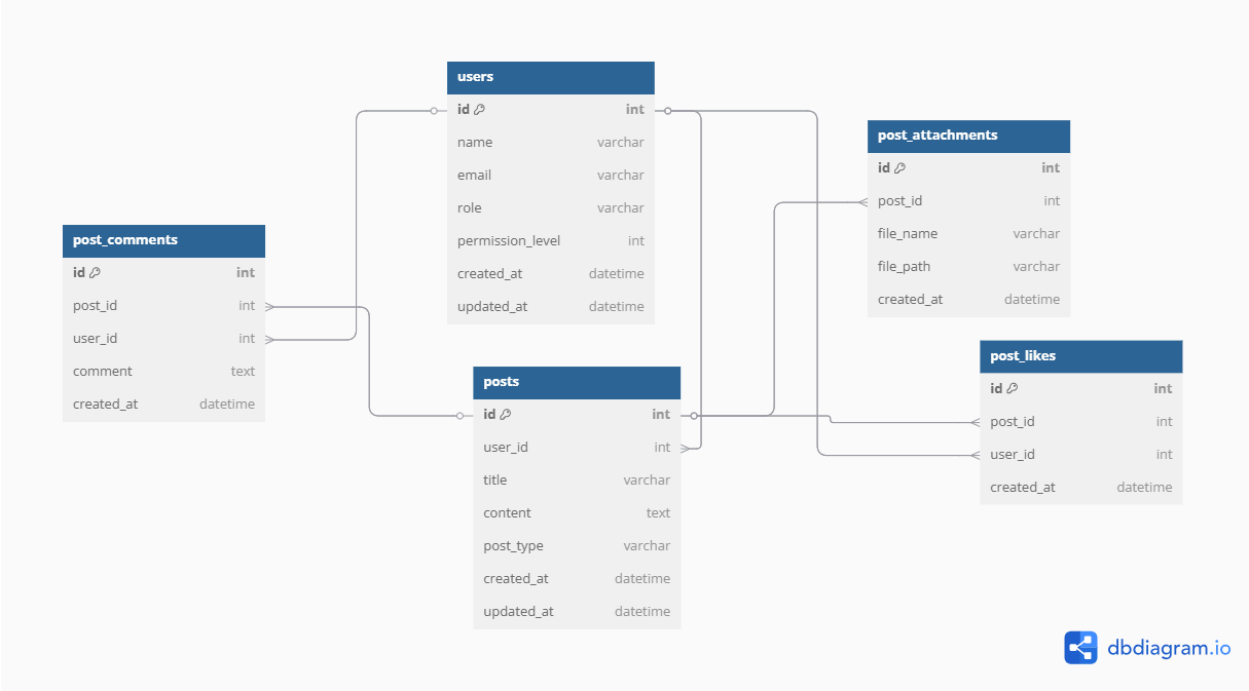
**Responsibilities**: Allows users to create, view, edit, and delete posts (text or media-based). Used to share announcements, updates, or discussion threads
**Features**:

- Post creation with text/media upload
- Like/comment functionality (optional)
- Sorting/filtering

**Key Components**:

- `PostController`
- `Post` Model and Migration
- Blade views for feed, post details

**Entity Relationship Diagram:**




## 6. Chat Module

**Responsibilities**: Enables real-time or near-real-time communication between users.

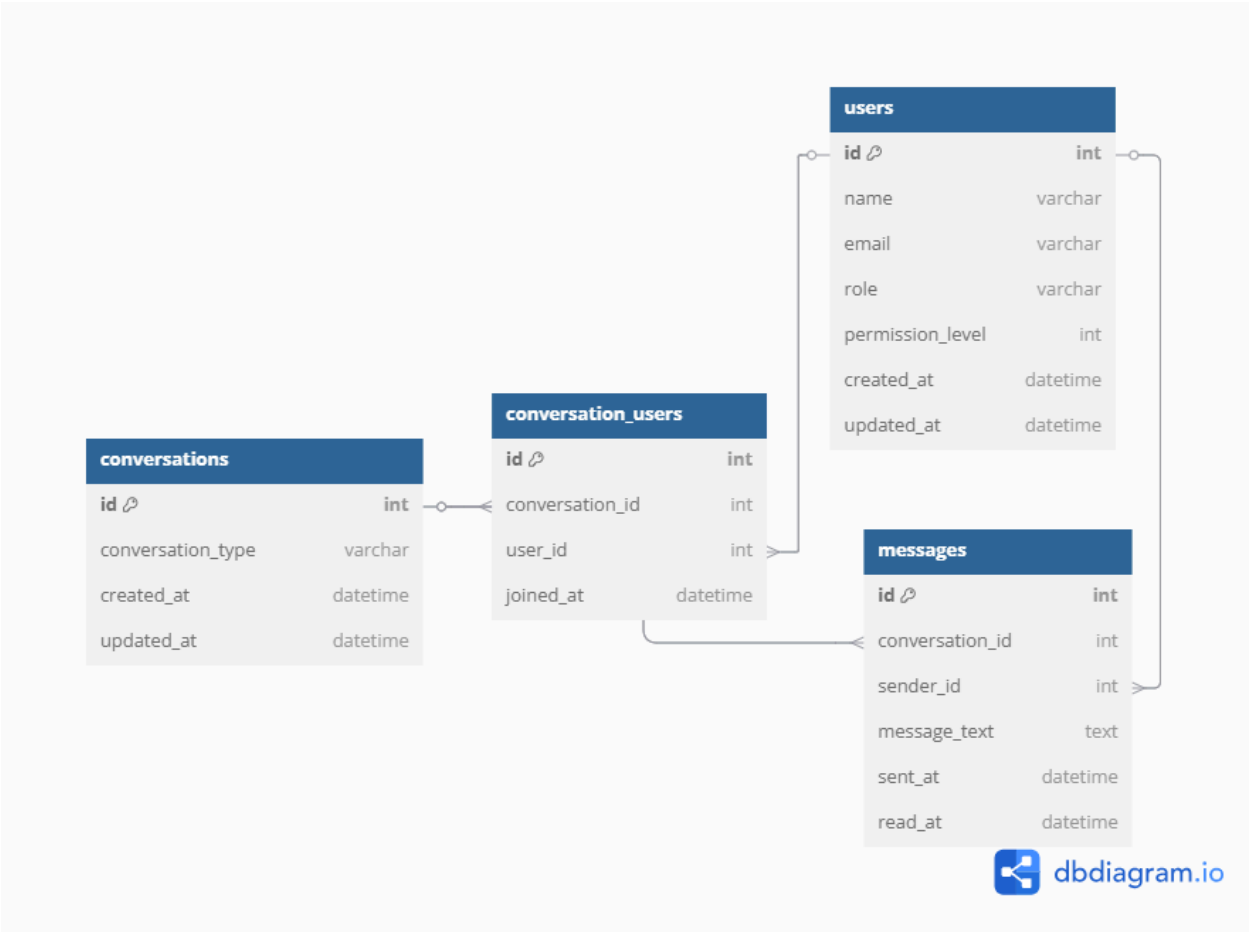
**Features**:

- One-on-one chat
- Chat history
- Notifications for new messages

**Key Components**:

- `ChatController`
- `Message` Model
- WebSockets (optional for live chat)
- Blade/JS chat interface

**Entity Relationship Diagram:**


users
id int
name varchar
email varchar
role varchar
permission_level int
created_at datetime
updated_at datetime
conversation_users
id int
conversation_id int
user_id int
joined_at datetime
conversations
id int
conversation_type varchar
created_at datetime
updated_at datetime
messages
id int
conversation_id int
sender_id int
message_text text
sent_at datetime
read_at datetime
dbdiagram.io

- **Data Design**

## Entity RelationShip Diagram


## Schema Design (Tables and Fields)

### 1. `programs`

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Program ID |
| name | varchar(255) | Program name |
| total_semesters | int | Number of semesters |

**2. `courses`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Course ID |
| program_id | int (FK) | Linked to `programs` |
| semester_number | int | Semester of the course |
| course_code | varchar(255) | Course code |
| course_name | varchar(255) | Course title |

**3. `students`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Student ID |
| name | varchar(255) | Full name |
| program_id | int (FK) | Enrolled program |
| current_semester | int | Ongoing semester |
| status | varchar(255) | Active, frozen, graduated, etc. |
| role | varchar(255) | For access control |
| permission_level | int | Defines scope of access |

**4. `student_course_enrollments`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Record ID |
| student_id | int (FK) | Linked to `students` |
| course_id | int (FK) | Linked to `courses` |
| taken_in_semester | int | When enrolled |
| is_repeat | boolean | Indicates a repeated course |
| created_at | datetime | Timestamp |

**5. `results`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Result ID |
| student_id | int (FK) | Linked to `students` |
| course_id | int (FK) | Linked to `courses` |
| taken_in_semester | int | Semester of attempt |
| marks | float | Numerical marks |
| grade | varchar(255) | Letter grade |

**6. `attendance`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Attendance ID |
| student_id | int (FK) | Linked to `students` |
| course_id | int (FK) | Linked to `courses` |
| taken_in_semester | int | Semester when recorded |

| | | |
|---|---|---|
| date | date | Date of class |
| present | boolean | Attendance status |

**7. `semester_freeze_logs`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Record ID |
| student_id | int (FK) | Linked to `students` |
| semester_number | int | Frozen semester |
| reason | text | Freeze reason |
| from_date | date | Start date |
| to_date | date | End date |

**8. `users`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | User ID |
| name | varchar(255) | Full name |
| email | varchar(255) | Login credential |
| role | varchar(255) | Role (admin, student, etc.) |
| permission_level | int | Authorization scope |
| created_at | datetime | Created |
| updated_at | datetime | Last updated |

**9. `posts`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Post ID |

| | | |
|---|---|---|
| user_id | int (FK) | Creator |
| title | varchar(255) | Title |
| content | text | Main content |
| post_type | varchar(255) | Announcement, discussion, etc. |
| created_at | datetime | Timestamp |
| updated_at | datetime | Last updated |

**10. `post_likes`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Like ID |
| post_id | int (FK) | Liked post |
| user_id | int (FK) | Liking user |
| created_at | datetime | Timestamp |

**11. `post_comments`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Comment ID |
| post_id | int (FK) | Related post |
| user_id | int (FK) | Commenting user |
| comment | text | Text content |
| created_at | datetime | Timestamp |

**12. `post_attachments`**

| Field | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| id | int (PK) | Attachment ID |
| post_id | int (FK) | Related post |
| file_name | varchar(255) | Name of file |
| file_path | varchar(255) | File storage path |
| created_at | datetime | Uploaded on |

**13. `conversations`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Conversation ID |
| conversation_type | varchar(255) | One-on-one / group |
| created_at | datetime | Start time |
| updated_at | datetime | Last updated |

**14. `conversation_users`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Record ID |
| conversation_id | int (FK) | Linked conversation |
| user_id | int (FK) | Linked user |
| joined_at | datetime | Participation time |

**15. `messages`**

| Field | Type | Description |
|---|---|---|
| id | int (PK) | Message ID |
| conversation_id | int (FK) | Related conversation |

| | | |
|---|---|---|
| sender_id | int (FK) | Sender |
| message_text | text | Content |
| sent_at | datetime | Sent time |
| read_at | datetime | When seen (nullable) |

## Relationships

| Table 1 | Table 2 | Relationship |
|---|---|---|
| `programs` | `students`, `courses` | One-to-Many |
| `students` | `student_course_enrollments` | One-to-Many |
| `courses` | `student_course_enrollments` | One-to-Many |
| `student_course_enrollments` | Composite of `students` + `courses` | Many-to-Many |
| `students` | `results`, `attendance`, `semester_freeze_logs` | One-to-Many |
| `users` | `posts`, `post_likes`, `post_comments`, `conversation_users`, `messages` | One-to-Many |
| `posts` | `post_likes`, `post_comments`, `post_attachments` | One-to-Many |
| `conversations` | `conversation_users`, `messages` | One-to-Many |

# Interface Design

## User Interface (UI) Mockups / Wireframes

Home

There is no email program associated to perform the requested action. Please install an email program or, if one is already installed, create an association in the Default Programs control panel.

PR Posted by @parthrana27

test

for testing

Ahmedabad/GJ 13 hours ago

1 0 Comments

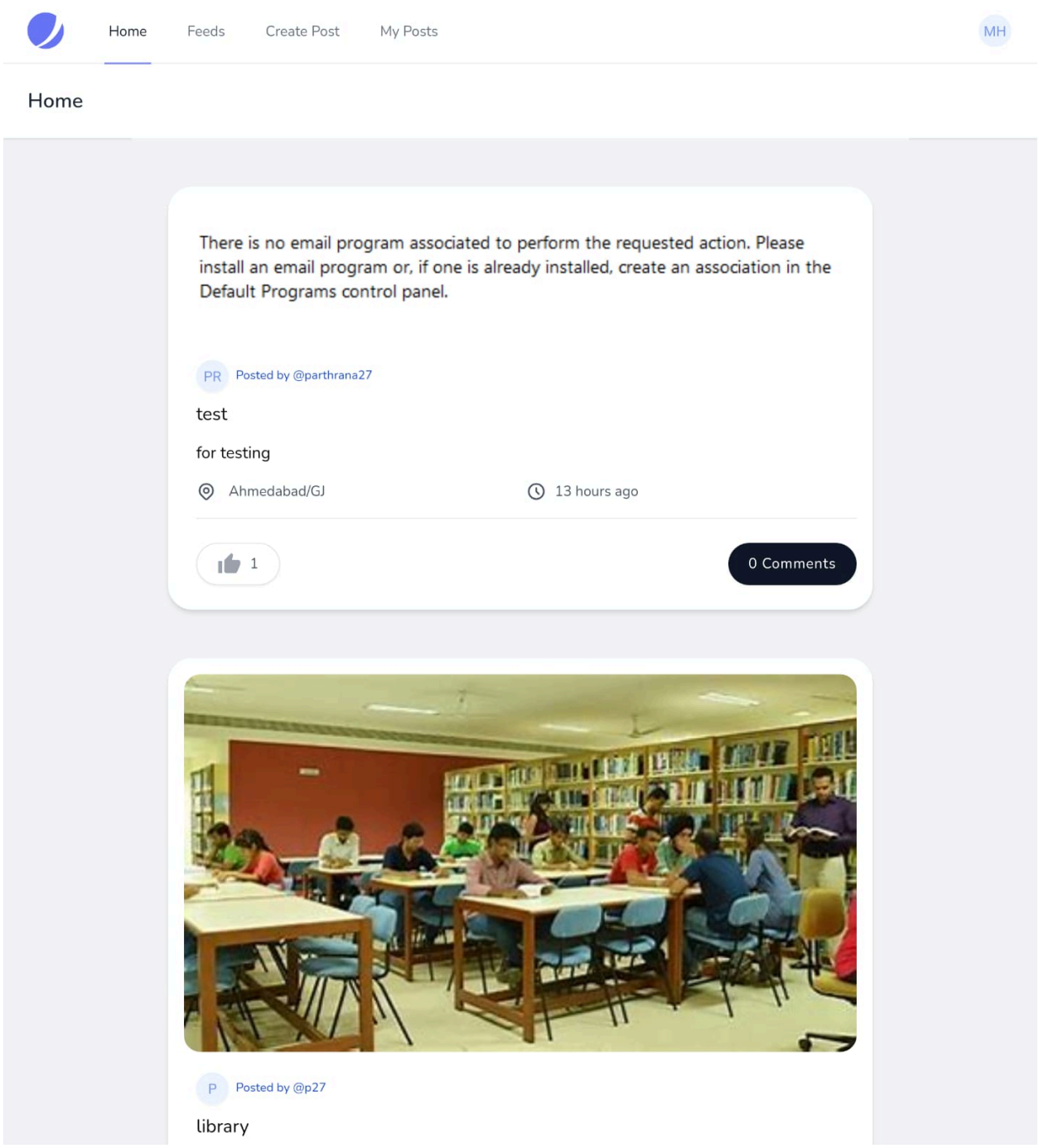
P Posted by @p27

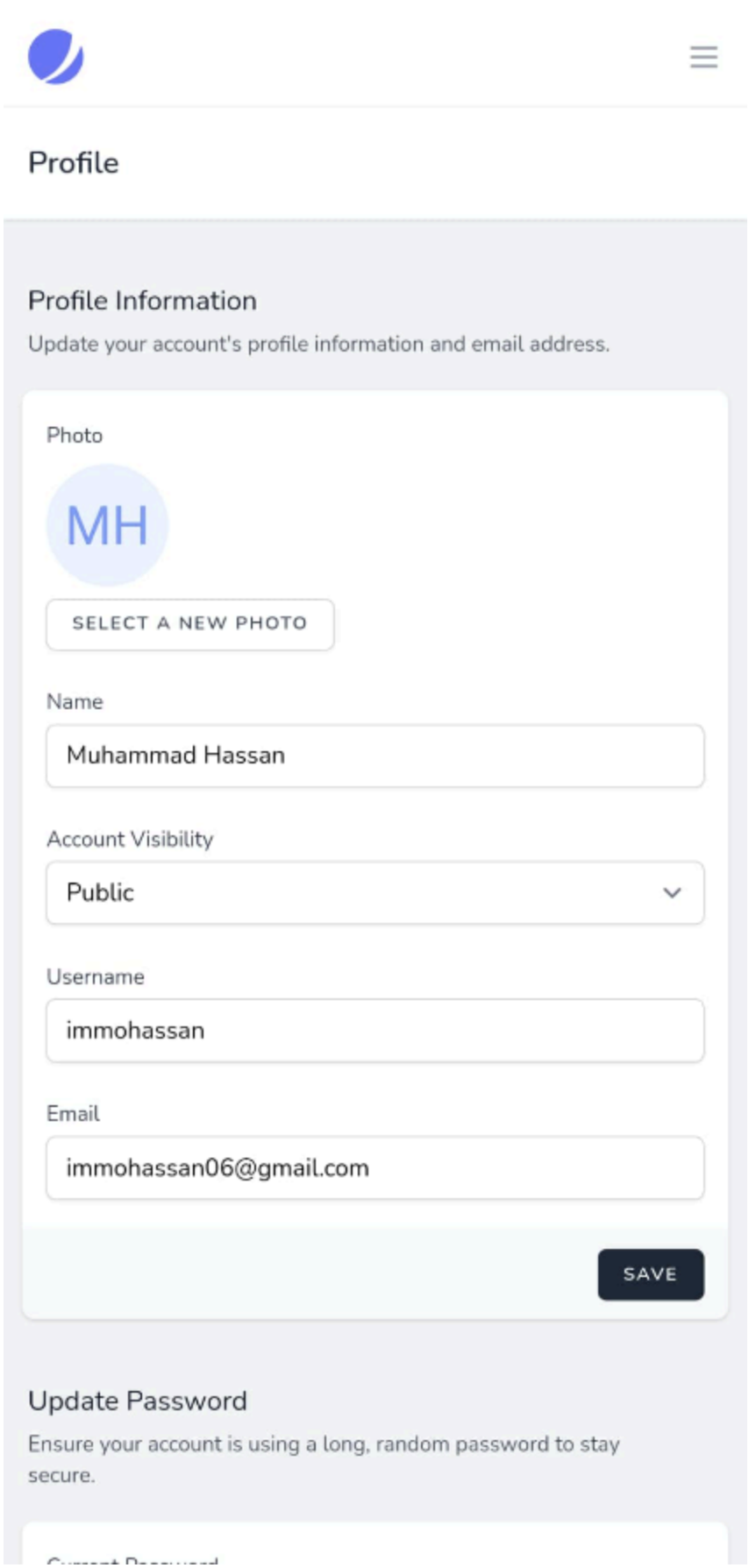
library

library time updated till 10

Ahmedabad/GJ 13 hours ago

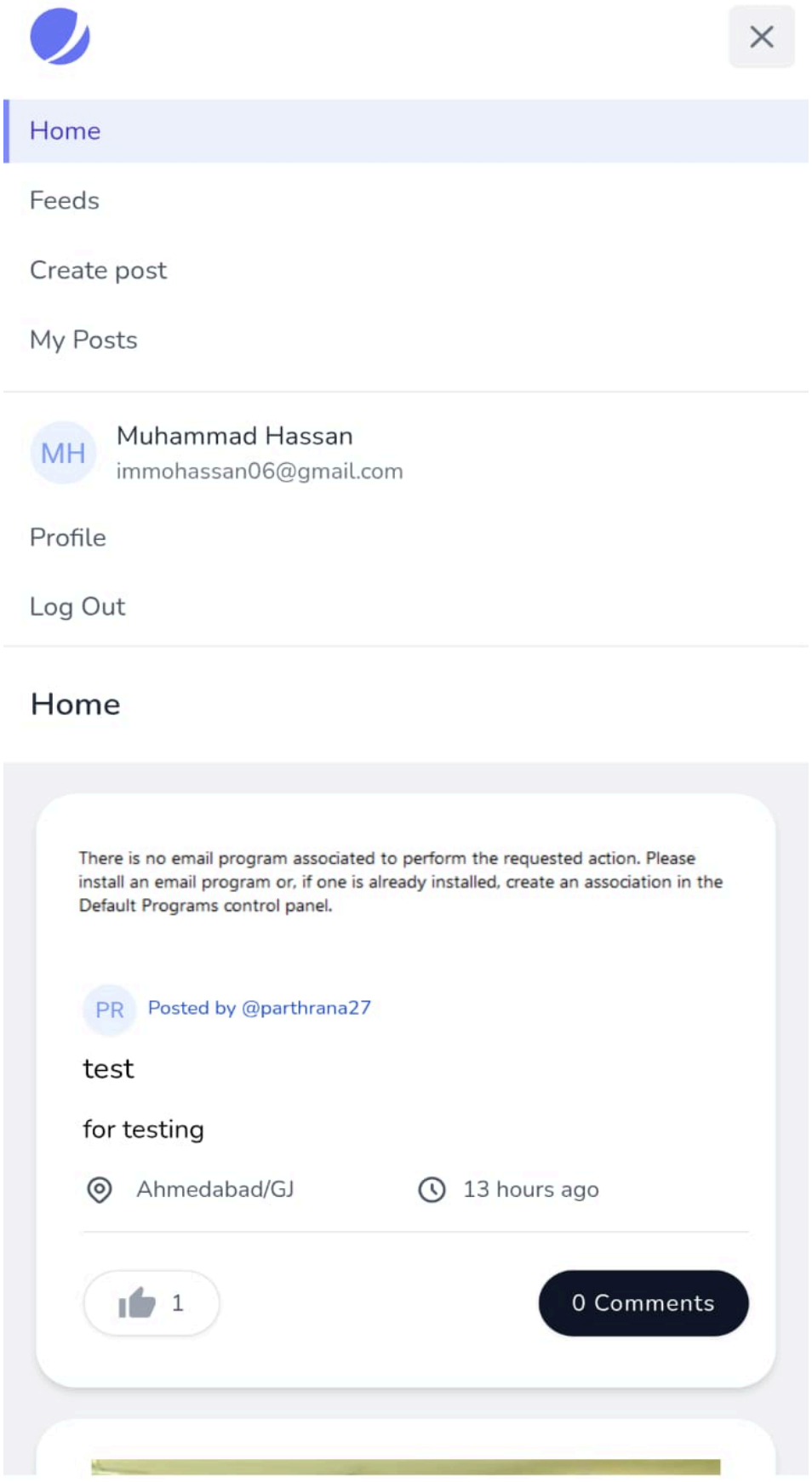
1 0 Comments


Home
Feeds
Create Post
My Posts
MH
Home
There is no email program associated to perform the requested action. Please install an email program or, if one is already installed, create an association in the Default Programs control panel.
PR
Posted by @parthrana27
test
for testing
Ahmedabad/GJ
13 hours ago
1
0 Comments
P
Posted by @p27
library

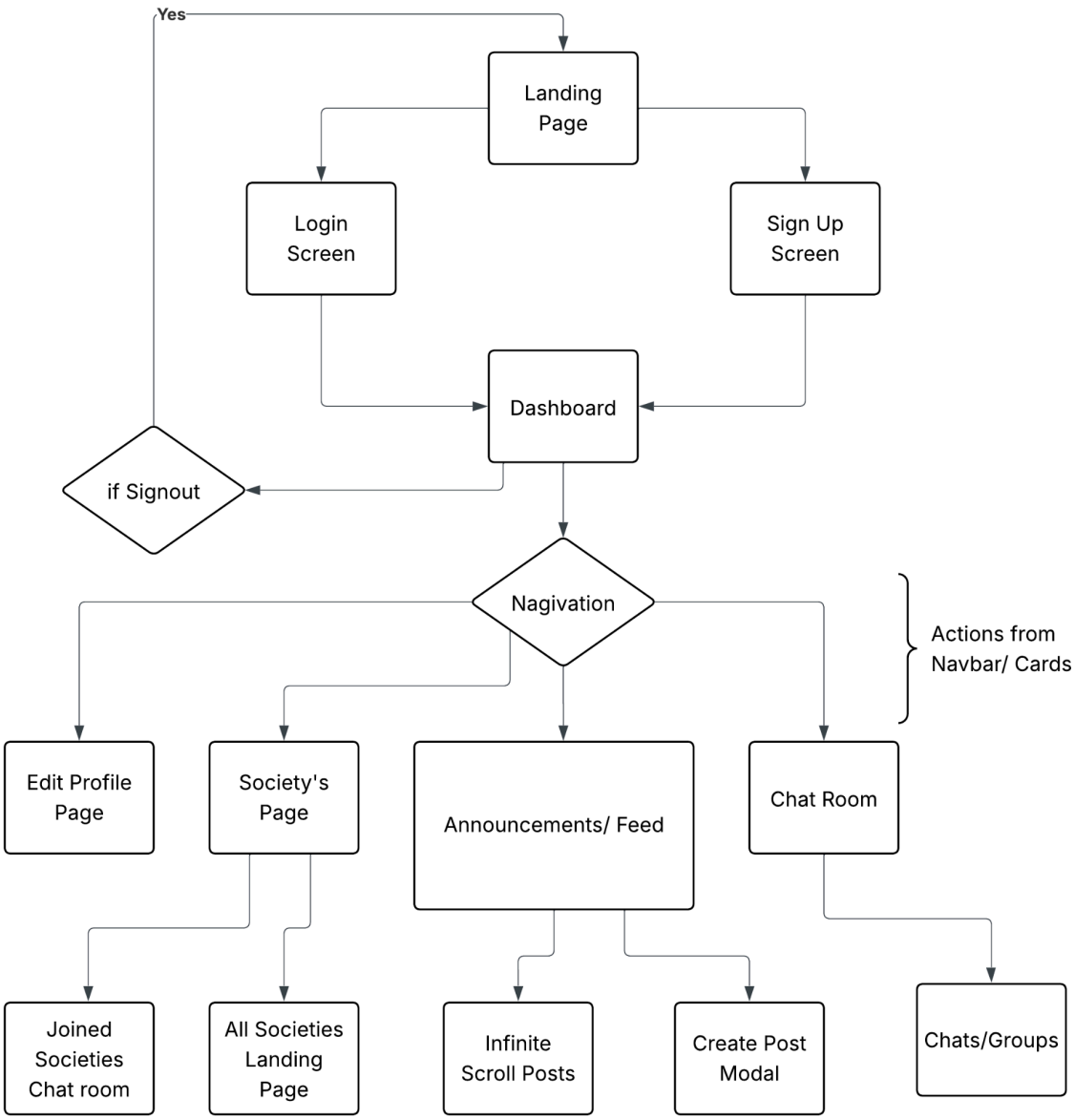

Profile
Profile Information
Update your account's profile information and email address.
Photo
MH
SELECT A NEW PHOTO
Name
Muhammad Hassan
Account Visibility
Public
Username
immohassan
Email
immohassan06@gmail.com
SAVE
Update Password
Ensure your account is using a long, random password to stay secure.

My Posts

No Posts found..!!!


Home
Feeds
Create post
My Posts
MH
Muhammad Hassan
immohassan06@gmail.com
Profile
Log Out
Home
There is no email program associated to perform the requested action. Please install an email program or, if one is already installed, create an association in the Default Programs control panel.
PR
Posted by @parthrana27
test
for testing
Ahmedabad/GJ
13 hours ago
1
0 Comments

## Wire-Frame/Flow Diagram



## References


1. **Software Requirements Specification (SRS)** – detailing the functional and non-functional requirements of the Student Resource Portal.
2. **Laravel Documentation** – for backend and API development (https://laravel.com/docs)
3. **Tailwind CSS Documentation** – for front-end UI styling (https://tailwindcss.com/docs)
4. **Jetstream Documentation** – for authentication scaffolding and UI presets (https://jetstream.laravel.com)
5. **MySQL Documentation** – for relational database design (https://dev.mysql.com/doc)
6. **UML Diagrams and Modeling Standards** – for system modeling and diagram consistency