A

Project Report

On

‘Notice Management System’

Presented by

Sarvesh Sudhir Kulkarni
&
Chaitanya Rajesh Koyalwar

TY[CSE-A]

Computer Science and Engineering

2025-2026

Guided By

Ms. Mukta G. Shelke

(Department of Computer Science and Engineering)

Submitted to

MGM’S COLLEGE OF ENGINEERING, NANDED

Under

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

Certificate

This is to certify that the report entitled

'Notice Management System'

Submitted By

Sarvesh Sudhir Kulkarni
&
Chaitanya Rajesh Koyalwar

in satisfactory manner as a partial fulfillment of

TY [CSE-A] in Dept. Of Computer Science and Engineering

To

MGM'S COLLEGE OF ENGINEERING, NANDED

Under

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

has been carried out under my guidance,

Ms. Mukta G. Shelke

Project Guide

Dr. Archana M. Rajurkar

HOD.

Computer Science and Engineering

Dr. Geeta S. Lathkar

Director

MGM's College of Engineering,Nanded

ACKNOWLEDGEMENT

We are greatly indebted to our project guide Ms. Mukta G. Shelke for her able guidance throughout this work. It has been an altogether different experience to work with her and we would like to thank her for her help, suggestions and numerous discussions.

We gladly take this opportunity to thank Dr. Rajurkar A.M (Head of CSE DEPT, MGM's College of Engineering, Nanded).

We are heartily thankful to Dr. Lathkar G. S (Director, MGM's College of Engineering, Nanded) for providing facility during the progress of the project; also for her kindly help, guidance and inspiration.

Last but not least we are also thankful to all those who help directly or indirectly to develop this project and complete it successfully.

With Deep Reverence,

Sarvesh Sudhir Kulkarni

Chaitanya Rajesh Koyalwar

ABSTRACT

CampusChrono is a comprehensive web-based notice management system designed specifically for educational institutions. The system facilitates efficient communication between administration, faculty, and students through a centralized digital platform. It eliminates the traditional paper-based notice distribution system and provides real-time notifications via email.

The system implements role-based access control with three distinct user roles: Admin, Staff, and Students. Each role has specific permissions and functionalities tailored to their needs. The platform features secure user registration with OTP verification, admin approval workflow, targeted notice distribution, comment system, file attachments, and comprehensive user management capabilities.

Built using modern web technologies including HTML5, CSS3, JavaScript, PHP, and MySQL, CampusChrono ensures scalability, security, and ease of use. The system integrates PHPMailer for reliable email delivery and implements industry-standard security practices including password hashing, SQL injection prevention, and XSS protection.

Keywords: Notice Management, Educational Institution, Role-Based Access Control, OTP Verification, Email Notification, Web Application.

Sarvesh Sudhir Kulkarni.

Chaitanya Rajesh Koyalwar

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION & BACKGROUND

In most educational institutions, communication between administration, faculty, and students plays a critical role in maintaining academic flow. Traditionally, institutions rely on physical notice boards placed in hallways or central areas. Although effective decades ago, this method now creates several limitations due to increasing student count, reduced physical presence on campus, and the need for faster updates. The shift toward digital learning environments further highlights the need for an efficient online notice distribution mechanism. This chapter introduces the background, problems, and goals of creating CampusChrono as a modern digital solution.

1.1 Background :

Notices are an integral part of daily operations in schools, colleges, and universities. They contain essential information such as exam schedules, event updates, administrative announcements, deadlines, and academic instructions. The physical notice board method is slow, unreliable, and dependent on students being physically present. Notices may go unnoticed, and there is no record-keeping mechanism. Digital notice systems solve these issues by providing easy access, reducing manual effort, and delivering information instantly.

1.2 Problem Statement:

Educational institutions face several challenges with traditional notice distribution: Students must physically check the board to stay updated.Notices may be missed due to crowding or late arrival.There is no centralized database to store older notices. Manual posting consumes time and effort.Printing notices increases paper usage and cost.No system exists to send notices to a specific department, year, or class. To address these challenges, there is a need for a centralized, automated, and accessible system.

1.3 Objectives :

The primary objective of CampusChrono is to digitalize notice delivery within institutions. The system aims to:

1. Provide a centralized platform for all announcements.

2. Reduce manual errors and paper usage by eliminating printed notices.
3. Deliver notices instantly to all concerned users.
4. Facilitate targeted notice distribution (specific department or class).
5. Maintain a history of notices for future reference.
6. Enhance accessibility by allowing users to view notices 24/7 from any device.
7. Improve security using OTP verification, admin approval, and password hashing.

1.4 Scope of the Project:

The system focuses primarily on notice distribution and role-based communication. It includes:

- Admin login and notice posting
- Student and staff login
- Viewing and searching notices
- Commenting on notices
- Notification through email

The project does not include features like push notifications, SMS services, chat modules, or mobile applications. These can be future developments.

1.5 System Overview :

CampusChrono is designed as a web-based platform where the Admin can upload notices in text or file format. They can target notices to specific classes or departments. Students and staff can log in and view notices based on their role and category. The system ensures secure login, maintains user data, and stores notice history for future reference.

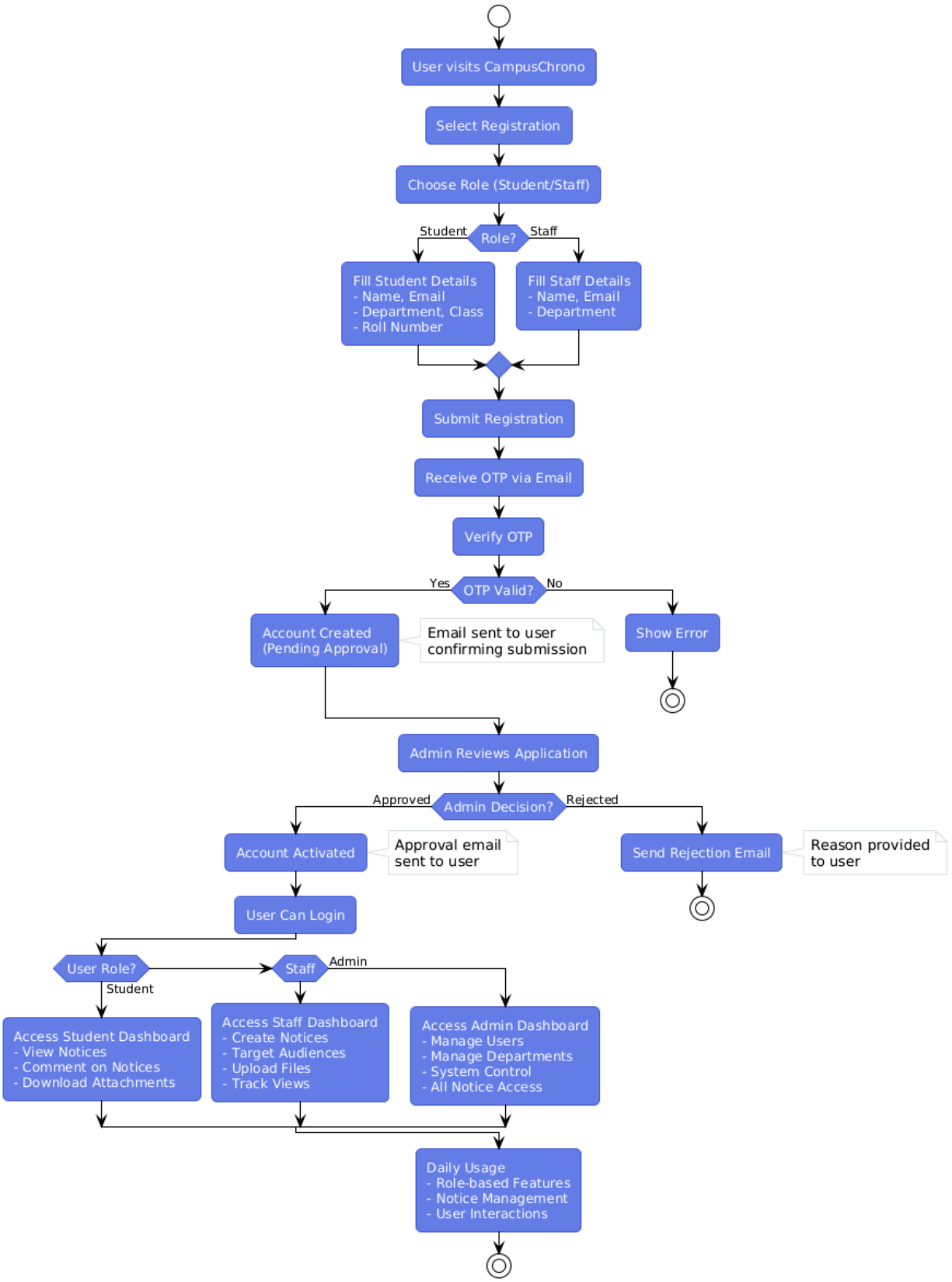

Fig. 1.5 : System Workflow Diagram

Chapter 2

SYSTEM ANALYSIS

Before developing CampusChrono, a detailed study of existing systems and operational challenges was performed. This chapter discusses traditional systems, their drawbacks, and the benefits of adopting a digital alternative. It also includes a feasibility study covering technical, economic, and operational aspects.

2.1 Existing System Analysis :

The traditional system requires notices to be printed and pasted physically on notice boards. This approach has several disadvantages:

- Notices reach students late.
- Information cannot be accessed remotely.
- Physical damage or removal of notices is possible.
- No digital archive exists for old notices.
- Staff must manually update boards daily.

Such limitations create the need for a centralized, automated solution.

2.2 Proposed System :

The proposed digital system overcomes all the shortcomings of the traditional approach. It provides real-time access, digital storage, secure logins, and targeted notice sharing. Users are notified instantly through email, and notices remain accessible at any time.

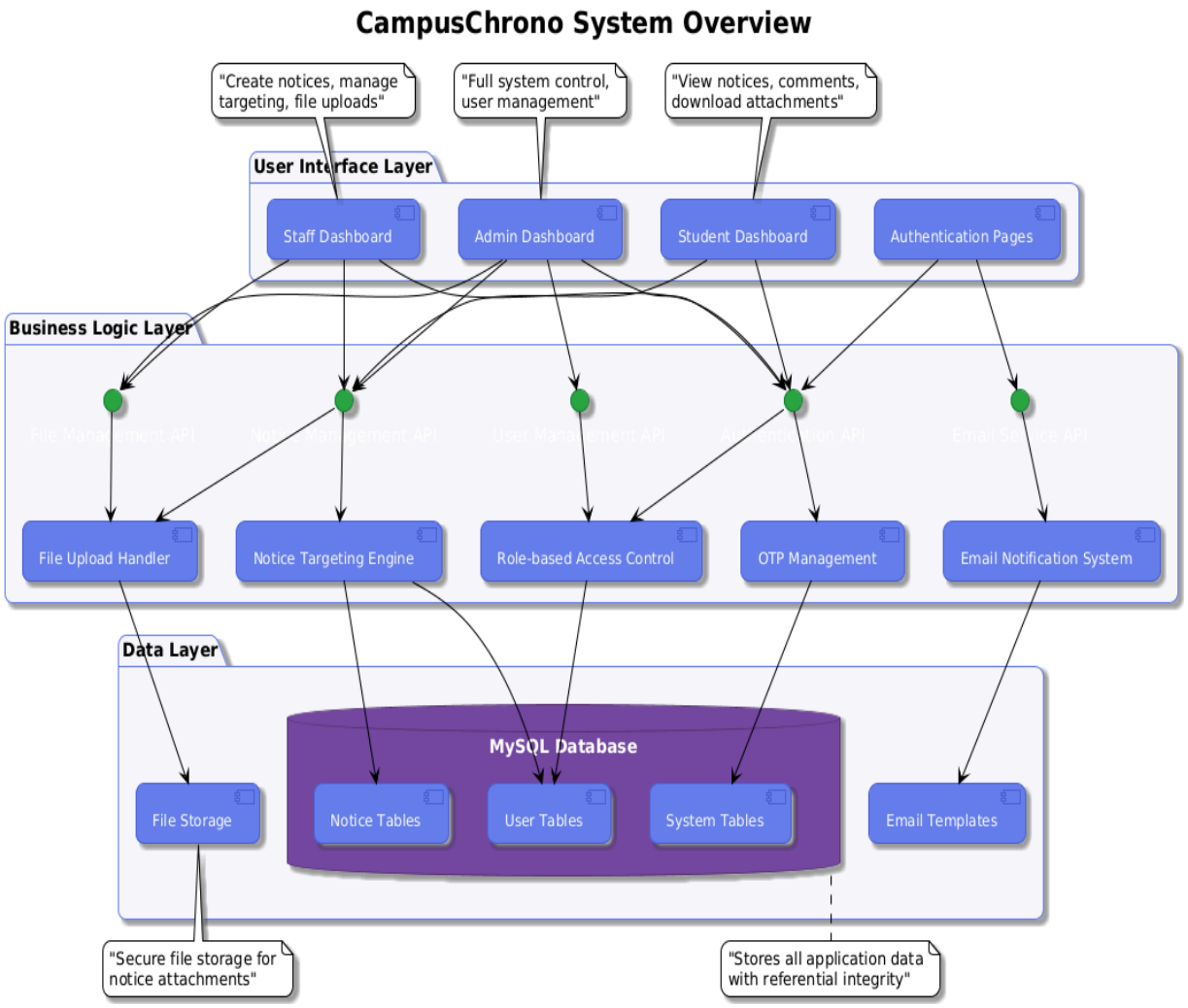

Fig. 2.2 : System Overview Diagram

2.3 Feasibility Study :

Technical Feasibility : CampusChrono is built using widely available technologies like PHP, MySQL, HTML, CSS, and JavaScript, making it easy to deploy on any standard server.

Economic Feasibility : Development cost is minimal, and maintenance requires no specialized infrastructure. Only a basic hosting plan and domain are needed.

Operational Feasibility : The interface is user-friendly. Students and faculty require minimal training to operate the system.

Chapter 3

SYSTEM DESIGN

System design defines the structural, architectural, and database foundations of CampusChrono. This chapter explains how different layers of the system interact, how data flows inside the application, and how the relational database schema supports all functionalities. A well-structured design ensures performance, scalability, and maintainability of the overall system.

3.1 System Architecture Overview:

CampusChrono is developed using a Three-Tier Architecture, which separates the system into independent layers for clarity, security, and modularity. Each layer performs a dedicated function while interacting through well-defined protocols.

1. **Presentation Layer :** This includes the user interface components developed using HTML, CSS, and JavaScript. It handles user interactions, displays pages, and sends user requests to the server.
2. **Application Layer :** This layer contains the server-side logic implemented in PHP. It processes user requests, validates input, performs authorization checks, executes business logic, and communicates with the database.
3. **Database Layer :** All system data is stored and managed in MySQL. It includes tables for users, notices, departments, classes, comments, OTP tokens, notice attachments, and more. Foreign key relationships ensure data integrity.

This layered structure ensures that each component can be developed, tested, and maintained independently, making the system scalable and secure.

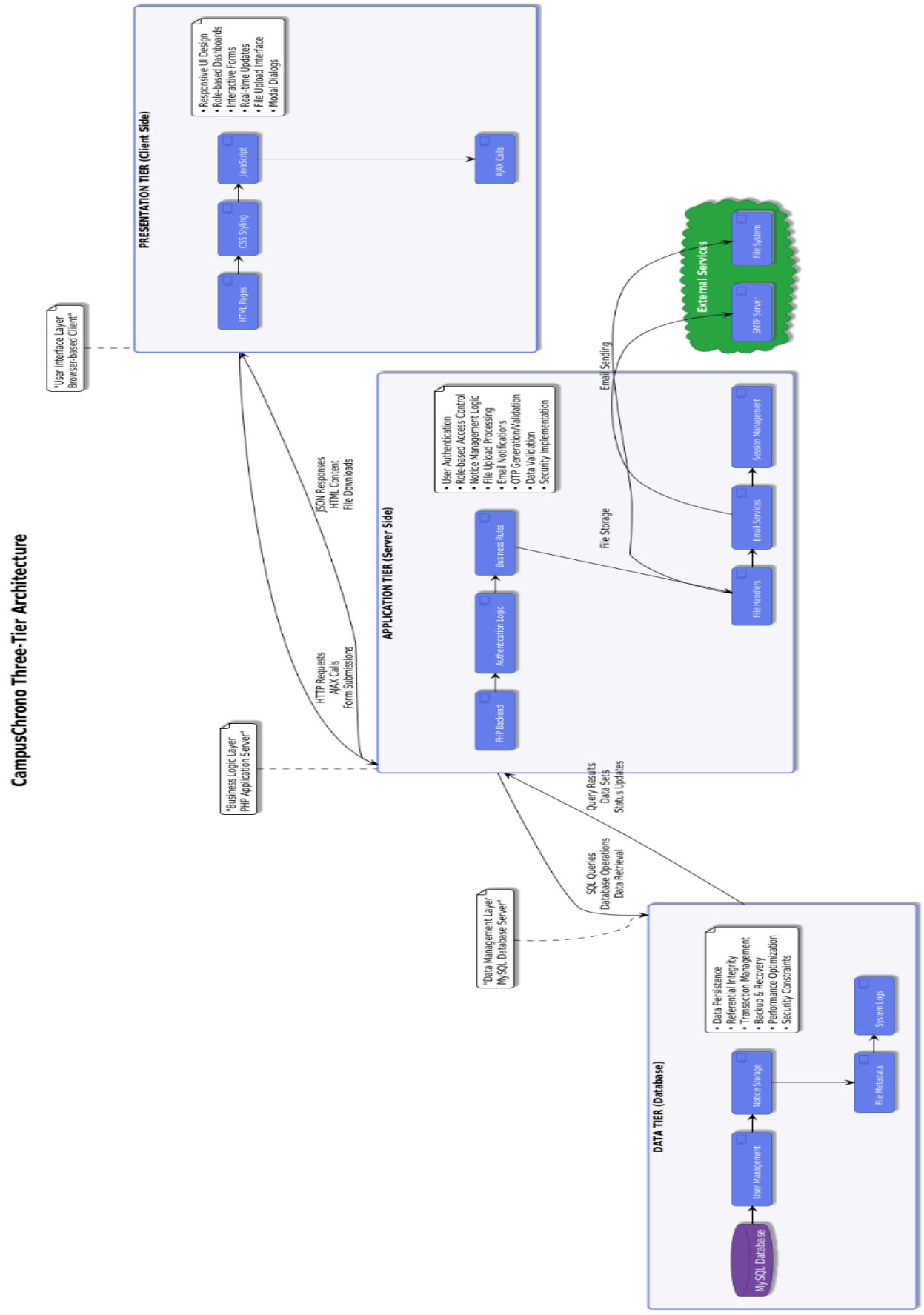

Fig. 3.1: Three-Tier Architecture Diagram

3.2 ER Diagram :

The Entity Relationship (ER) Diagram visually represents the logical structure of the CampusChrono database. It shows how major entities—such as Users, Roles, Departments, Classes, Notices, Notice Targets, Comments, and Notice Attachments—are connected.

The system ensures:

- Proper relational mapping
- Data consistency using foreign keys
- Fast data retrieval using indexes
- Support for notice tracking and role-based access

The ER diagram forms the blueprint for the SQL schema implemented in the system.

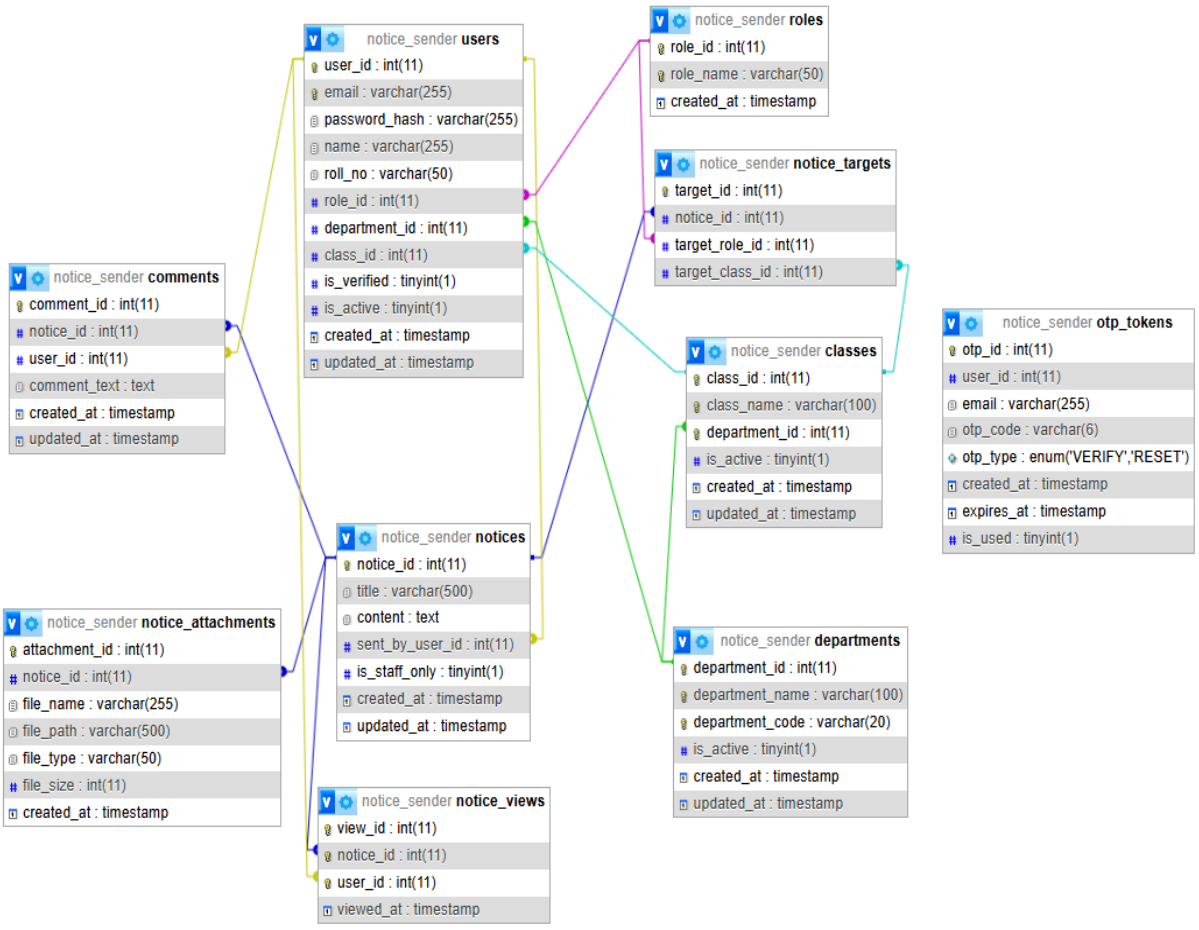

Fig. 3.2: ER Diagram of CampusChrono

3.3 Database Schema :

The complete SQL schema for CampusChrono is provided below. It includes table creation scripts, relationship definitions, sample inserts, and constraints. This schema ensures secure, normalized, and efficient data storage.

```
-- Notice Sender Database Schema
-- Drop database if exists and create fresh
DROP DATABASE IF EXISTS notice_sender;
CREATE DATABASE notice_sender CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
USE notice_sender;

-- Table: roles
CREATE TABLE roles (
    role_id INT PRIMARY KEY AUTO_INCREMENT,
    role_name VARCHAR(50) NOT NULL UNIQUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB;

-- Table: departments
CREATE TABLE departments (
    department_id INT PRIMARY KEY AUTO_INCREMENT,
    department_name VARCHAR(100) NOT NULL UNIQUE,
    department_code VARCHAR(20) NOT NULL UNIQUE,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    INDEX idx_active (is_active)
) ENGINE=InnoDB;

-- Table: classes
CREATE TABLE classes (
```

```
    class_id INT PRIMARY KEY AUTO_INCREMENT,
    class_name VARCHAR(100) NOT NULL,
    department_id INT NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (department_id) REFERENCES departments(department_id) ON
DELETE RESTRICT,
    UNIQUE KEY unique_class_dept (class_name, department_id),
    INDEX idx_department (department_id),
    INDEX idx_active (is_active)
) ENGINE=InnoDB;

-- Table: users
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(255) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    name VARCHAR(255) NOT NULL,
    roll_no VARCHAR(50) DEFAULT NULL,
    role_id INT NOT NULL,
    department_id INT DEFAULT NULL,
    class_id INT DEFAULT NULL,
    is_verified BOOLEAN DEFAULT FALSE,
    is_active BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES roles(role_id) ON DELETE RESTRICT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id) ON
DELETE SET NULL,
    FOREIGN KEY (class_id) REFERENCES classes(class_id) ON DELETE SET
NULL,
```

```
    INDEX idx_email (email),
    INDEX idx_role (role_id),
    INDEX idx_department (department_id),
    INDEX idx_class (class_id)
) ENGINE=InnoDB;

-- Table: notices
CREATE TABLE notices (
    notice_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(500) NOT NULL,
    content TEXT NOT NULL,
    sent_by_user_id INT NOT NULL,
    is_staff_only BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (sent_by_user_id) REFERENCES users(user_id) ON DELETE
CASCADE,
    INDEX idx_sender (sent_by_user_id),
    INDEX idx_created (created_at)
) ENGINE=InnoDB;

-- Table: notice_targets
CREATE TABLE notice_targets (
    target_id INT PRIMARY KEY AUTO_INCREMENT,
    notice_id INT NOT NULL,
    target_role_id INT DEFAULT NULL,
    target_class_id INT DEFAULT NULL,
    FOREIGN KEY (notice_id) REFERENCES notices(notice_id) ON DELETE
CASCADE,
    FOREIGN KEY (target_role_id) REFERENCES roles(role_id) ON DELETE
CASCADE,
    FOREIGN KEY (target_class_id) REFERENCES classes(class_id) ON DELETE
CASCADE,
```

```
    INDEX idx_notice (notice_id),
    INDEX idx_role (target_role_id),
    INDEX idx_class (target_class_id)
) ENGINE=InnoDB;

-- Table: otp_tokens
CREATE TABLE otp_tokens (
    otp_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT DEFAULT NULL,
    email VARCHAR(255) NOT NULL,
    otp_code VARCHAR(6) NOT NULL,
    otp_type ENUM('VERIFY', 'RESET') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    expires_at TIMESTAMP NOT NULL,
    is_used BOOLEAN DEFAULT FALSE,
    INDEX idx_email (email),
    INDEX idx_otp (otp_code),
    INDEX idx_expires (expires_at)
) ENGINE=InnoDB;

-- Table: comments
CREATE TABLE comments (
    comment_id INT PRIMARY KEY AUTO_INCREMENT,
    notice_id INT NOT NULL,
    user_id INT NOT NULL,
    comment_text TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (notice_id) REFERENCES notices(notice_id) ON DELETE
CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE
CASCADE,
    INDEX idx_notice (notice_id),
```

```
    INDEX idx_user (user_id),
    INDEX idx_created (created_at)
) ENGINE=InnoDB;

-- Table: notice_attachments
CREATE TABLE notice_attachments (
    attachment_id INT PRIMARY KEY AUTO_INCREMENT,
    notice_id INT NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_path VARCHAR(500) NOT NULL,
    file_type VARCHAR(50) NOT NULL,
    file_size INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (notice_id) REFERENCES notices(notice_id) ON DELETE
CASCADE,
    INDEX idx_notice (notice_id)
) ENGINE=InnoDB;

-- Table: notice_views
CREATE TABLE notice_views (
    view_id INT PRIMARY KEY AUTO_INCREMENT,
    notice_id INT NOT NULL,
    user_id INT NOT NULL,
    viewed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (notice_id) REFERENCES notices(notice_id) ON DELETE
CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE
CASCADE,
    UNIQUE KEY unique_view (notice_id, user_id),
    INDEX idx_notice (notice_id),
    INDEX idx_user (user_id)
) ENGINE=InnoDB;

-- Insert default roles
```

```
INSERT INTO roles (role_name) VALUES
('Admin'),
('Staff'),
('Student');

-- Insert sample departments
INSERT INTO departments (department_name, department_code) VALUES
('Computer Science and Engineering', 'CSE'),
('Information Technology', 'IT'),
('Electronics and Telecommunication', 'EXTC'),
('Mechanical Engineering', 'MECH'),
('Civil Engineering', 'CIVIL'),
('Automation and Robotics', 'AR'),
('Artificial Intelligence and Machine Learning', 'AIML');

-- Insert sample classes (linked to departments)
INSERT INTO classes (class_name, department_id) VALUES
('F.E.A', 1),
('S.Y.A', 1),
('T.Y.A', 1),
('B.E.A', 1),
('F.E.B', 1),
('S.Y.B', 1),
('T.Y.B', 1),
('B.E.B', 1),
('F.E.A', 2),
('S.Y.A', 2),
('T.Y.A', 2),
('B.E.A', 2),
('F.E.A', 3),
('S.Y.A', 3),
('T.Y.A', 3),
('B.E.A', 3);
```

```
-- Insert default admin user
INSERT INTO users (email, password_hash, name, role_id, is_verified, is_active)
VALUES ('admin@noticeboard.com',
'$2y$10$e0MYzXyjpJS7Pd0RVvHwHe.N9Aq5cJZxJxJxJxJxJxJxJxJxJxJu',
'System Admin', 1, TRUE, TRUE);
```

3.4 Module Description:

Admin Module: Admin can create, edit, delete, and manage notices. They approve user registrations and handle departments and classes.

Staff Module: Staff can post notices (if permitted) and view student/staff notices.

Student Module: Students can view class-wise notices and comment.

Notice Module: Handles notice creation, file uploads, and distribution.

Chapter 4

SYSTEM IMPLEMENTATION

This chapter describes the practical implementation of the CampusChrono system, focusing on the technologies used, integration of major modules, and a demonstration of the user interface through screenshots. The goal of implementation is to translate the design into a working and efficient system that fulfills all specified functional and non-functional requirements. Each component—frontend, backend, database, and email service—works together to provide a seamless experience for users.

4.1 Technology Used:

CampusChrono has been developed using a combination of modern web technologies chosen for efficiency, reliability, and ease of deployment.

Frontend Technologies: The frontend is responsible for the user interface, ensuring ease of use and responsive design.
Technologies used:

- HTML5 – Structure and content of web pages
- CSS3 – Styling, layout, responsiveness
- JavaScript – Form validation, dynamic content, UI interactions

Backend Technology: The backend handles system logic, authentication, authorization, and communication with the database.
PHP (server-side scripting)
PHP ensures secure request handling, session management, and integration of email services.

Database Technology:

- MySQL
    - MySQL is used to store all data securely, including users, departments, classes, notices, comments, attachments, OTP tokens, etc.
- Email Service
    - PHPMailer: PHPMailer is integrated to send OTPs and email alerts for verification and notifications.

These technologies collectively ensure that the system is secure, scalable, and maintainable.

4.2 Screenshots:

This section presents the major user interface components of CampusChrono. Each screenshot highlights how different modules of the system function from the perspectives of Admin, Staff, and Students.

1. Login Page:

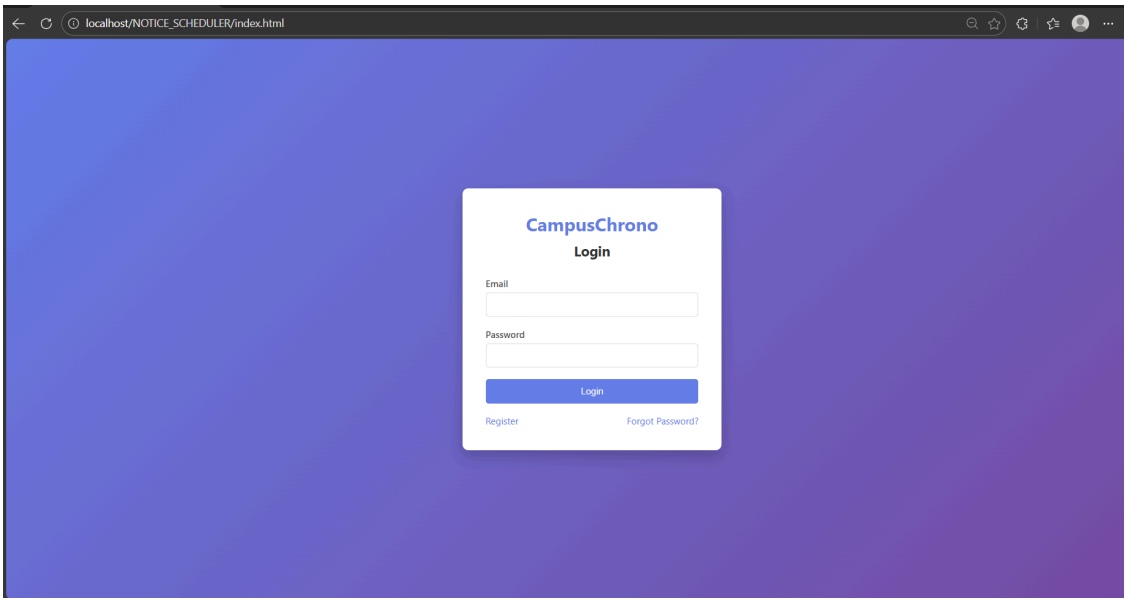

Fig. 4.2.1 Login page

The Login Page serves as the secure entry point to the CampusChrono system. Users must enter their registered email address and password to authenticate themselves. The page includes built-in validation to prevent incorrect or empty submissions, ensuring that only legitimate users can access the platform. A “Forgot Password” option is provided to help users recover access through an OTP-based reset mechanism. During registration, OTP verification ensures the authenticity of email addresses and prevents unauthorized account creation. The interface is designed to be clean, responsive, and easily accessible across devices, providing a seamless user experience while maintaining strong security protocols.

2. Admin Dashboard:

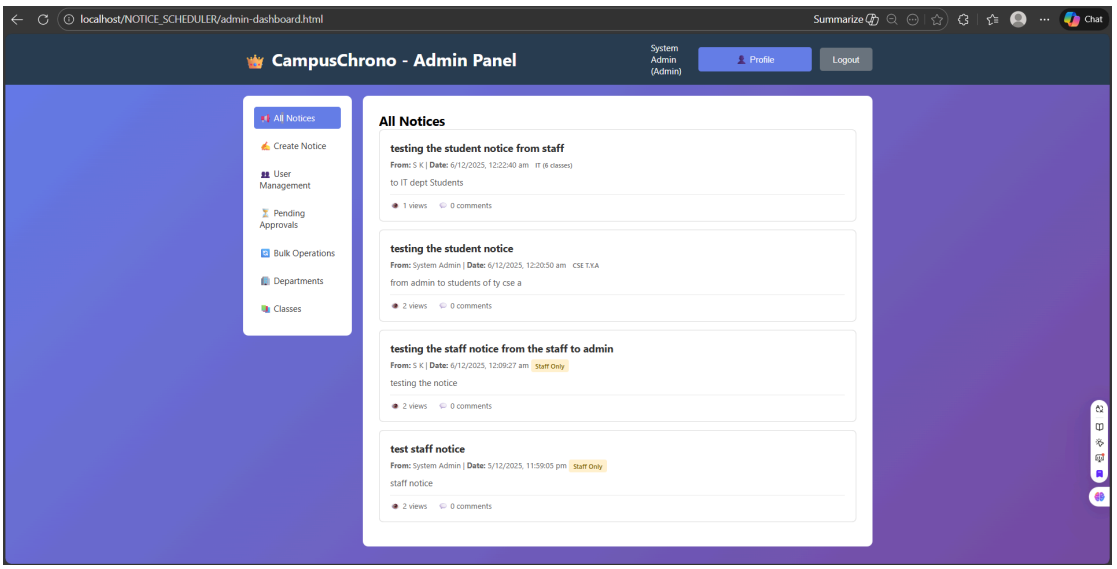

Fig. 4.2.2 Admin Dashboard

The Admin Dashboard provides a centralized panel where the administrator can manage all system operations. Admins can approve or reject new user registrations, add departments and classes, view total users, create notices, and monitor system activities. The dashboard displays important statistics and quick-access cards for easy navigation, ensuring efficient administration.

3. Notice Creation Page:

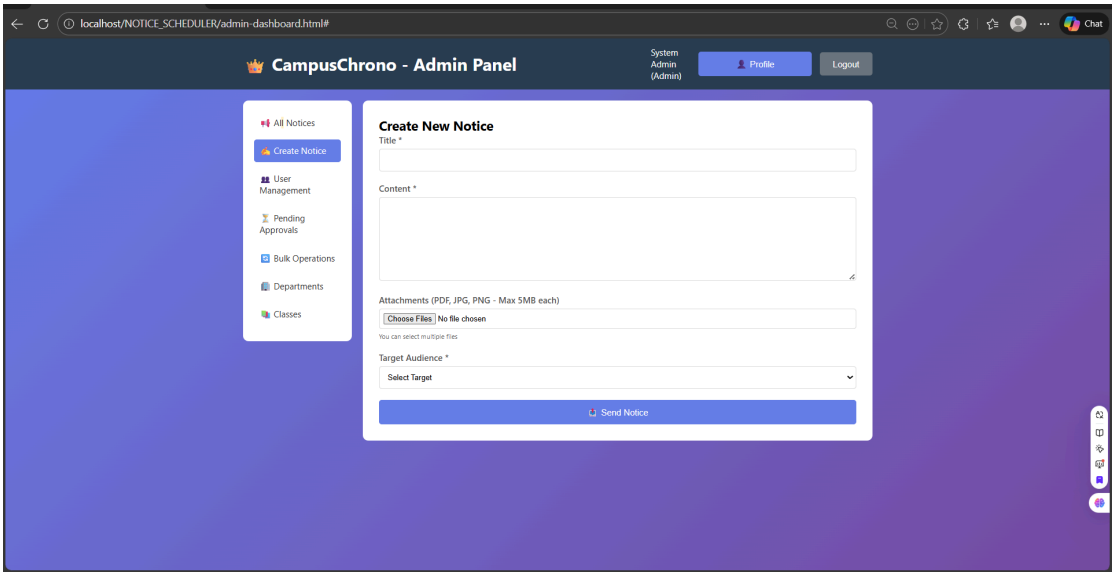

Fig. 4.2.3 Notice Creation Page

This page allows Admin or authorized Staff users to create and publish new notices. The form includes fields for title, description, file attachments, and target selection (role-wise or class-wise). The notice creation module ensures structured communication by enabling targeted distribution. Validation ensures correct input before submitting the notice to the database.

4. Staff Dashboard:

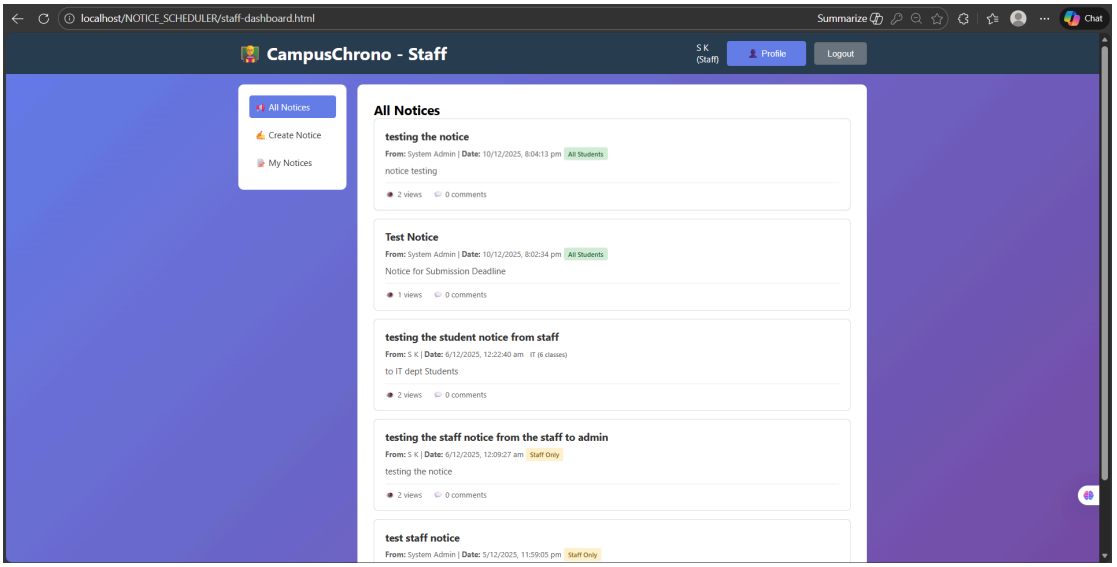

Fig. 4.2.4 Staff Dashboard

The Staff Dashboard provides an interface tailored specifically for faculty members. Staff users can view notices relevant to their department and class, create notices (if permission is granted by the administrator), and manage their previously posted notices. The dashboard displays important quick-access options such as posting a new notice, viewing all institutional notices, and responding to student comments. The layout ensures that staff members can efficiently communicate updates, academic schedules, assignments, and announcements with minimal effort. The interface remains clean and intuitive, allowing educators to focus on effective information-sharing without dealing with technical complexities.

5. Student Dashboard:

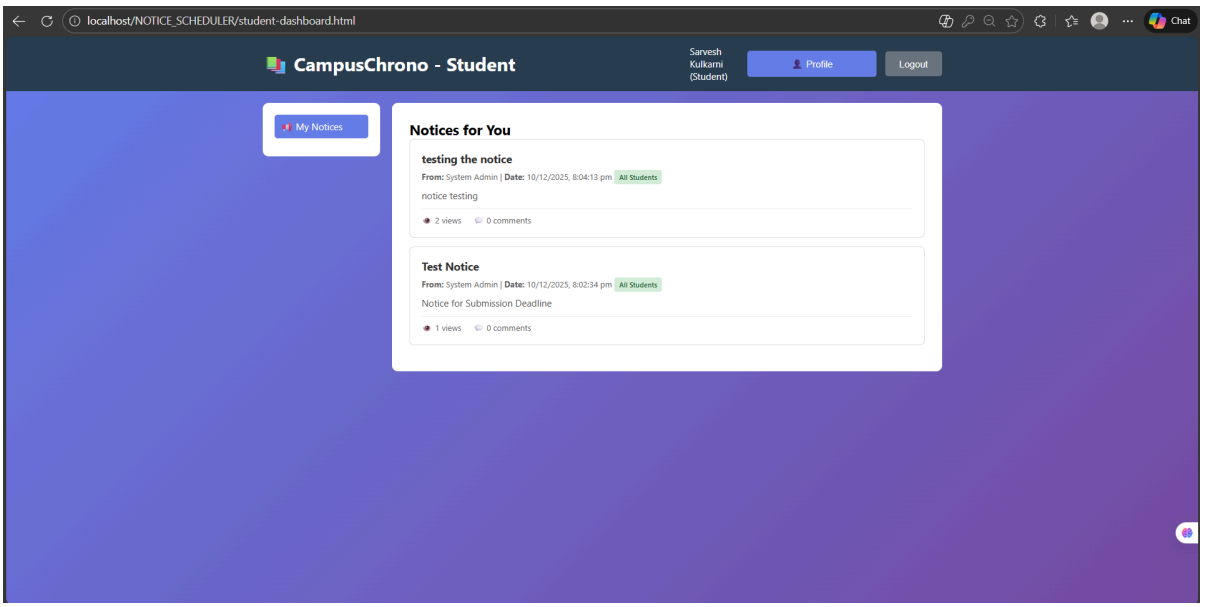

Fig. 4.2.5 Student Dashboard

The Student Dashboard displays all notices relevant to the logged-in student based on their department and class. Notices are listed chronologically with clear indicators for new or unread notices. Students can open notices, download attachments, add comments, and track previously viewed notices. The dashboard ensures an easy and clutter-free experience tailored for students.

6. Notice View Page:

Fig. 4.2.6 Notice-Viewpage

The Notice View Page shows the complete details of a selected notice, including the title, content, date, sender information, and downloadable attachments. A comments section enables interactive communication, allowing students to clarify doubts or share feedback. This page also records “notice views” to track user engagement, helping administrators assess reach and visibility.

Chapter 5

TESTING

Testing is an essential phase in system development to ensure the reliability, security, and proper functioning of the system. Various testing techniques were applied to validate that CampusChrono works as intended under different conditions.

5.1 Unit Testing:

Unit testing was conducted individually on all functional modules such as:

- User registration
- OTP verification
- Login/Logout
- Notice creation and posting
- Comments module
- File attachment module
- User approval by admin

Each module was tested with valid and invalid data to confirm expected behavior and handle errors gracefully.

5.2 Integration Testing:

Integration testing ensured that multiple modules work together seamlessly.
Examples include:

- Login system interacting correctly with the user database
- Notice posting module interacting with notice_targets
- File attachments linking correctly with notices
- Comments linked properly to users and notices

Integration testing confirmed that combined functions performed accurately without conflicts.

5.3 Security Testing:

Security testing was performed to ensure that the system is safe against common vulnerabilities.

Key tests included:

- SQL Injection Prevention – Prepared statements were used to prevent injection attacks.
- XSS Protection – Input sanitization was implemented to avoid cross-site scripting.
- Password Security – All passwords were hashed securely using password_hash() before storage.
- Access Control Testing – Role-based access was verified for Admin, Staff, and Students.

The system successfully passed all essential security checks for deployment.

5.4 User Acceptance Testing (UAT):

User Acceptance Testing was conducted with:

- Admins
- Staff members
- Students

Each user group tested the system for usability, accuracy, interface quality, and performance.

Results showed:

- Users found the UI simple and intuitive
- Notices were delivered accurately
- OTP verification worked efficiently
- System speed and reliability were satisfactory

Based on the feedback, minor UI refinements were made before final deployment.

CONCLUSION

The CampusChrono Notice Management System successfully modernizes communication within educational institutions by replacing outdated physical notice boards with a centralized, efficient, and fully digital platform. The system significantly enhances accessibility and transparency by allowing students and staff to receive timely updates from any location. Features such as role-based access control, secure login mechanisms, OTP verification, real-time email notifications, file attachments, commenting functionality, and a well-structured database ensure reliable and streamlined information sharing across the institution. Through careful design and implementation, CampusChrono meets all functional requirements while reducing manual workload and maintaining a complete historical record of notices. Moreover, the system is highly scalable and future-ready, with potential for enhancements such as mobile applications, push notifications, cloud deployment, and AI-assisted notice prioritization. Overall, the project demonstrates an effective, secure, and user-friendly solution that significantly improves communication workflow in academic environments.

REFERENCES

[1] Sanchez L. (2023). *Web Programming with HTML, CSS, Bootstrap, JavaScript, React.JS, PHP, and MySQL*, (Fourth Edition). IngramSpark. ISBN-13: 978-1088239872.

[2] Kogent Learning Solutions Inc. (2009). *Web Technologies: HTML, JAVASCRIPT, PHP, JAVA, JSP, ASP.NET, XML and Ajax, Black Book*. Dreamtech Press. ISBN-13: 978-8177229974.

[3] PHPMailer GitHub Repository. (2024). *PHPMailer – A full-featured email creation and transfer class for PHP*. Available at: https://github.com/PHPMailer/PHPMailer

[4] W3Schools. (2024). *HTML, CSS, JavaScript and PHP Tutorials*. Available at: https://www.w3schools.com/