Orbit Campus

Orbit Campus: A smart campus management platform designed to streamline student services, enhance communication, and simplify academic operations.

Empowering Students and Administration with Smart Technology:

(Orbit campus)

A Seamlessly Integrated Web System Crafted with Frontend and Backend Technologies

Submitted By

D. Chiranjeevi **23072-CM-016**

Report Submitted to

DR. B. R. AMBEDKAR GOVERNMENT MODEL RESIDENTIAL POLYTECHNIC

Under the Guidance of

Kum. N.SANDYA RANI (L/CME, M. TECH)

DEPARTMENT OF COMPUTER ENGINEERING

Dr. B.R. Ambedkar Government Model Residential Polytechnic

RAJAMAHENDRAVARAM-533124 2025-2026

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that this project work entitled "ORBIT CAMPUS "is the Bonafide record work done by Mr./Ms.

_____ Bearing the Pin Number. _____of the final year, along with batch mates submitted on the partial fulfilment of the Requirement for the award of DIPLOMA IN COMPUTER ENGINEERING during the academic year 2023-2026.

Signature of project guide
Kum. N.SANDYA RANI L/CME, M. TECH

Head of the Department
P.V. LAKSHMINARAYANA, M. Tech

Signature of the External Examiner

ACKNOWLEDGEMENT

I have made every effort to complete this project with the support and valuable suggestions from my classmates, ensuring it functions efficiently and effectively.

I would like to express my sincere gratitude to my project guide, Kum. N. Sandya Rani, for her guidance, cooperation, and valuable suggestions, which motivated me to successfully complete this project.

I also extend my thanks to our Senior Lecturer, P.V.LAKSHMINARAYANA, and Lecturers Smt. G.SriDevi, Smt.CH. Kalyani Bindu, Smt. N. Pavani for their valuable suggestions throughout the course of the project.

I am thankful to our Head of the Computer Engineering Department and project coordinator, Sri P.V.LAKSHMINARAYANA, for his cooperation and unwavering dedication,which greatly inspired me to complete this project.

I also grateful to our respected Principal, Sri A. Murali, for his kind support in the successful completion of this project.

Lastly, I extend my gratitude to all the teaching and non-teaching staff of the Computer Engineering department, who have assisted me, either directly or indirectly, in the completion of this project.

PROJECT ASSOCIATES

Name	Pin Number
D. Chiranjeevi	23072-CM-016
K. Manikanta Karthik	23072-CM-029
P. Nanda Kishore	23072-CM-038
B. Siddhardha	23072-CM-004
P. Jaswanth Laxman	23072-CM-047
B. Bala Manohar	23072-CM-058

TABLE OF CONTENT

S.no	CONTENTS	PG.no
1	ABSTRACT	
2	SOFTWARE REQUIREMENTS	
3	INTRODUCTION	
4	SYSTEM ANALYSIS AND FEASIBILITY STUDY	
5	SRS & DESIGN	
6	SOFTWARE INSTALLATION	
7	SOURCE CODE AND TESTING	
8	USER MANUAL	
9	OUTPUT SCREENS AND BIOGRAPHY	

ABSTRACT

Campus Sphere is a web-based campus management system engineered to centralize academic, administrative, and communication workflows within a unified digital platform. The system employs a modular architecture integrating role-based access control (RBAC), secure authentication mechanisms, and a scalable backend to streamline operations for students, faculty, and administrators. Core functionalities include automated attendance processing, digital notice dissemination, event and timetable management, academic content delivery, and two-way communication modules.

The platform is built using modern web technologies to support efficient data handling, real-time updates, and responsive user interactions. A structured relational database ensures reliable storage, retrieval, and synchronization of campus data, while optimized backend APIs facilitate smooth transactions across modules. By digitizing manual procedures and enabling seamless information flow, Campus Sphere enhances operational efficiency, reduces redundancy, and improves overall campus connectivity. This project demonstrates a technically robust approach to campus digitization through secure architecture, modular design, and user-centric functionality.

Existing System:

- Most educational institutions still rely on traditional and semi-digital methods to manage campus activities, communication, and student services. Many processes—such as announcements, attendance, timetable management, event updates, and academic resource sharing—are handled manually or distributed across multiple unconnected platforms. This leads to inefficiencies, delays, and a lack of centralized control.
- Current campus management systems face several limitations:
- Fragmented Communication – Notices and updates are shared through physical notice boards, SMS groups, or separate apps, causing inconsistent information flow.
- Manual Administrative Processes – Attendance, event registration, and academic documentation are often handled manually, making them time-consuming and error-prone.
- Lack of Centralized Access – Students and faculty must use multiple channels to access timetables, study materials, announcements, and services.
- Limited Transparency – Tracking activities such as academic progress, campus events, or service requests becomes difficult without a unified system.
- Accessibility Issues – There is no single platform for accessing real-time campus information anytime, anywhere.
- These limitations highlight the need for an integrated, automated, and efficient campus management solution.

Proposed System:

Smart Campus Sphere introduces a unified, web-based campus management platform designed to streamline academic, administrative, and communication processes within a single system. The platform provides role-based access for students, faculty, and

administrators, enabling seamless interaction and real-time information sharing across the campus ecosystem.

Key features and advantages include:

- Centralized Communication Hub – Instantly shares notices, announcements, and updates to all users through a unified dashboard.
- Automated Administrative Tasks – Digital attendance, event management, and academic record handling reduce manual work and improve accuracy.
- Unified Resource Access – Provides students and staff with organized timetables, study materials, and campus services from one platform.
- Real-Time Updates – Ensures immediate access to academic schedules, upcoming events, and urgent notifications.
- Secure and Scalable Architecture – Implements authentication, role-based access control (RBAC), and structured database management, ensuring data privacy and future scalability.
- Smart Campus Sphere significantly enhances campus efficiency, transparency, and connectivity by combining automation, centralized access, and real-time information delivery into a single intelligent platform.

SOFTWARE REQUIREMENTS

SOFTWARE REQUIREMENTS :

1. Frontend Requirements

- Your frontend is built using React + Vite, so the following software is required:

- Development Tools
- Node.js (v16+)
- npm (for installing dependencies)
- Vite (already used inside your project)

Code Editor:

- VS Code (recommended)

Frontend Libraries:

- React JS
- React Router
- Axios (for API communication)
- Tailwind / CSS (if included)
- Node Modules (installed using npm install)

Browser Requirements

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
  (Any browser supporting ES6+ JS and React)

2. Backend Requirements

- Your backend uses Core PHP + REST APIs, so:
- Server Software
- Apache Server (included in XAMPP)
- PHP 7.4 or above
- MySQL
- phpMyAdmin (for database management)
- PHP Extensions Required
- mysqli
- pdo
- json
- fileinfo
- curl (optional for external APIs)
- Backend Structure Found
- /api folder with endpoints
- config/cors.php — for CORS handling

- config/database.php — DB connection
- models/ — PHP model classes
- database.sql — to create DB tables
- File upload scripts (profile photo, documents)

3. Operating System Compatibility

- Windows 10 / 11
- Linux (Ubuntu recommended)
- macOS

HARDWARE REQUIREMENTS

1. Minimum Hardware Requirements

- Processor: Dual-core 2.0 GHz
- RAM: 4 GB
- Storage: 2–4 GB free space (Node modules + PHP + Database)
- Display: 720p or higher
- Input: Keyboard & Mouse

2. Recommended Hardware Requirements

- Running React + PHP + MySQL + Node.js together requires more power:
- Processor: Intel i5 / Ryzen 5 or higher
- RAM: 8 GB or above
- Storage: 10–20 GB free (node_modules are large)
- Display: Full HD (1080p)

3. Server-Side Requirements

- If your project will run on an internal server or cloud:
- Server Hardware
- Processor: Quad-core CPU
- RAM: 8–16 GB

- Storage:
- 100+ GB SSD for documents, logs, DB backups
- Network:
- High-speed LAN or WiFi
- Stable internet (for cloud apps)
- Power Backup
- UPS or generator (for physical campus server)

4. Client Device Requirements

- Students/Faculty can access your system using:
- Smartphone (Android/iOS)
- Laptop/Desktop
- Tablet
- Minimum:
- RAM: 2 GB
- Browser: Any modern browser
- OS: Android/iOS/Windows/Linux

WEB BROWSER:

- CHROME.

INTRODUCTION

smart Campus Sphere is a web-based campus management system designed to digitalize and streamline essential academic and administrative activities within an educational institution. The platform provides a unified interface for students, faculty, and administrators to manage daily campus operations efficiently. Built with a **React + Vite frontend** and a **PHP–MySQL backend**, the system ensures fast performance, secure data handling, and smooth communication between all campus stakeholders.

The objective of Smart Campus Sphere is to eliminate manual paperwork, reduce administrative delays, and create a centralized digital ecosystem where users can access services such as academic records, document requests, fee information, notifications, and profile management.

The backend follows a structured REST API architecture, enabling seamless data exchange with the modern, interactive frontend.

The system uses a **role-based access model**, ensuring that students, faculty, and administrators only access features relevant to their roles. With modules like user authentication, document uploads, request handling, dashboards, and communication features, Smart Campus Sphere enhances transparency, reduces human errors, and improves overall campus workflow efficiency.

Objectives

- **Centralize Campus Services** – Provide a single digital platform for students, faculty, and admin workflows.
- **Improve Information Accessibility** – Enable users to quickly access academic and administrative data from anywhere.
- **Enhance Operational Efficiency** – Reduce paperwork and manual processing through automated modules.
- **Streamline Communication** – Deliver real-time updates, notices, and notifications to users.
- **Ensure Security & Reliability** – Implement secure authentication and database-driven data management.
- **Core Functionalities Supported**
- **User Authentication** – Secure login for students, faculty, and administrators.
- **Dashboard System** – Role-based dashboards for personalized access to campus services.
- **Document Management** – Uploading, requesting, and tracking academic documents.
- **Profile Handling** – Updating user profiles and uploading profile photos.
- **Fee & Academic Records** – Viewing fee details, academic information, and history.
- **Notifications & Announcements** – Real-time updates from the administration.

Smart Campus Sphere is designed to be **scalable, secure, and user-friendly**, leveraging modern web technologies to provide a responsive interface and efficient backend processing. This document outlines the system's technical design, functional modules, and requirements necessary for the successful development and deployment of the platform.

1. HTML (HyperText Markup Language)

Purpose in the Project:

- Forms the **structural foundation** of all web pages.
- Defines layout elements such as login forms, dashboards, tables, cards, and content sections.
- Works together with React components to display user interfaces dynamically.

2. CSS (Cascading Style Sheets)

Purpose in the Project:

- Responsible for the **styling and visual design** of the interface.
- Provides layout formatting, color schemes, animations, spacing, and responsive design.
- Ensures a clean and user-friendly UI for students, faculty, and administrators.

3. JavaScript

Purpose in the Project:

- Powers the **interactive and dynamic behavior** of the frontend.
- Used extensively inside **React components** to manage state, events, routing, and API calls.
- Controls dashboard elements, form validations, data rendering, and navigation.
- Communicates with backend APIs using **Axios** or Fetch requests.

4. React JS

(Framework built using JavaScript)

Purpose in the Project:

- Used to build the **entire frontend application** with reusable components.
- Handles:
    - Routing between pages
    - Dashboard rendering
    - Data fetching from backend
    - State management
    - Real-time UI updates
- Offers a highly responsive, fast, and modern user interface for the Smart Campus Sphere.

5. PHP (Hypertext Preprocessor)

Purpose in the Project:

- Serves as the **backend language** responsible for:
    - Handling user authentication

    - Verifying login credentials
    - Managing requests for documents, fees, notices, and user data
    - Running business logic for admin and student modules
- Implements **REST API endpoints** to communicate with the React frontend.
- Ensures secure data processing using sessions, validation, and database queries.

6. MySQL (Structured Query Language)

Purpose in the Project:

- Acts as the **database system** for storing and managing:
    - User details
    - Login credentials
    - Student records
    - Document uploads
    - Fees, notices, and academic information
- Provides efficient data retrieval for dashboards and API responses.
- Ensures secure and structured storage through relational database design.

7. JSON (JavaScript Object Notation)

Purpose in the Project:

- Used for **data exchange** between frontend and backend.
- API responses from PHP backend to React frontend are formatted in JSON.
- Makes communication lightweight, fast, and structured.

---

8. Vite Build Tool

Purpose in the Project:

- Used for compiling and bundling the React frontend.
- Provides ultra-fast development server and optimized production build.

- Ensures smooth and fast UI loading for end users.

SYSTEM ANALYSIS AND FEASIBILITY STUDY

1. System Analysis

Smart Campus Sphere is designed as a full-stack web-based platform that simplifies and digitizes campus-related operations such as student management, document handling, profile updates, notifications, and administrative tasks. The uploaded ZIP files show a React + Vite frontend communicating with a PHP + MySQL backend, indicating a modern and scalable system architecture.

1.1 Existing Problems Identified

- Before the system was designed, the following limitations were observed in traditional campus workflows:
- Manual handling of student records leads to delays and errors.
- No centralized platform for accessing academic, fee, and personal information.
- Communication gaps between students, faculty, and administration.
- Document requests and verifications handled offline, causing delays.
- No automated system to track user activities, updates, or notifications.
- Lack of transparency and real-time updates for academic/administrative processes.

1.2 System Requirements Identified

- From the project structure, the following requirements were derived:
- A secure authentication system (present in the PHP backend).
- A role-based dashboard for different users (students/admin).
- A user-friendly UI built with React components.
- Database connectivity for storing profile, documents, and fee details.
- API-level communication using JSON responses.
- Support for uploading files (profile photos, documents).
- A fast frontend build system (Vite).

1.3 System Workflow Overview

- User logs into the system (React → PHP API → MySQL).
- Backend validates credentials and returns role-based data.
- React renders dynamic dashboard using fetched JSON responses.

- Users can update profiles, request documents, or check records.
- Admins can add notices, process requests, and manage students.
- All updates are stored in MySQL and retrieved on demand.

2. Feasibility Study

A feasibility study was conducted to analyze the practicality of developing and deploying Smart Campus Sphere. Based on the source code, the system is technically and operationally feasible for real-world use.

2.1 Technical Feasibility

- Smart Campus Sphere uses widely supported and reliable technologies:
- Frontend Technology Feasibility
- React JS ensures a fast, component-driven UI.
- Vite provides high-speed bundling and development experience.
- Fully compatible across devices (mobile, desktop).
- Backend Technology Feasibility
- Core PHP is lightweight, easy to deploy, and compatible with shared hosting.
- REST APIs make frontend-backend communication structured and efficient.
- MySQL ensures stable and secure data storage.
- The technologies used are easy to maintain, scalable, and have a strong community support base, making the system technically feasible.

2.2 Operational Feasibility

- The system is easy to use and reduces workload for both students and administrators.
- Operational Benefits
- Clean React UI ensures high user usability.
- Centralized database simplifies management and retrieval.
- Automated workflows reduce manual errors and delays.
- Role-based dashboards improve clarity and operations.
- Admin panel supports quick updates and management.
- Users (students/admins) require very minimal training, proving the system’s operational feasibility.

2.3 Economic Feasibility

- Smart Campus Sphere is cost-effective because:
- Development Cost
- Uses open-source technologies (React, PHP, MySQL).

- No need for expensive servers — works on XAMPP or shared hosting.
- Maintenance Cost
- Easy to maintain due to modular frontend and API-based backend.
- Requires only basic hosting and domain expenses.
- Deployment Cost
- Can run on low-cost servers or educational institution infrastructure.
- Overall, the system is economically feasible with minimal setup and maintenance costs.

2.4 Schedule Feasibility

- Based on the folder structure and development progress:
- Frontend and backend modules are already well-structured.
- API endpoints and React components show modular development.
- Can be completed within a typical academic project timeline.
- Thus, the system meets realistic schedule feasibility.

SOFTWARE REQUIREMENTS & SPECIFICATION (SRS)

1. Introduction

The Smart Campus Sphere system is a web-based campus management platform designed to streamline student services, administrative workflows, and data management. The system uses a React + Vite frontend and PHP + MySQL backend, communicating through REST APIs to provide a fast, secure, and user-friendly experience.
This SRS document describes functional and non-functional requirements essential for the system's development, deployment, and maintenance.

2. Overall Description

- Smart Campus Sphere provides:
- A centralized dashboard for students and administrators
- Profile management and document upload features
- Real-time notifications
- Academic and personal data handling
- Admin controls for user management and updates
- It operates through a client–server architecture where the browser (React UI) communicates with PHP APIs connected to a MySQL database.

3. Software Requirements

Below are the tools, technologies, and libraries required to build and run the system.

3.1 Frontend Software Requirements

Technologies Used

Component	Details
React JS	For building UI components and dynamic pages
Vite	For bundling, fast development server, hot reloading
JavaScript (ES6+)	Main programming language for frontend logic
HTML5 & CSS3	Structure and styling of UI
Axios / Fetch API	For making API calls to backend
Node.js + NPM	For installing and managing frontend dependencies

- Purpose in Project
- Rendering responsive dashboards
- Handling user interactions (login, forms, updates)
- Displaying data returned from PHP API
- Providing smooth UI/UX with reusable components
- Managing state and session data on client side

3.2 Backend Software Requirements

Component	Details
PHP 7+	Server-side scripting, API handling, authentication
MySQL / MariaDB	Database for storing user details, records, documents
Apache Server	Required for running PHP APIs (XAMPP/WAMP/LAMP)
JSON	Data format for frontend–backend communication
phpMyAdmin	DB management tool

- Purpose in Project
- Authenticate users and manage roles (student/admin)
- Store and retrieve profile, academic, and admin data

- Handle document uploads (photos, PDFs, etc.)
- Return JSON responses to the React frontend
- Perform CRUD operations on database records

3.3 Additional Software Requirements

- Browser Support
- Chrome
- Edge
- Firefox
- Safari
- Operating System Compatibility
- Windows 10/11
- Linux (Ubuntu/CentOS)
- macOS

4. Functional Requirements

4.1 User Authentication

- User login with email and password
- Role-based access (student/admin)

4.2 Student Dashboard

- View personal information
- Update profile details
- Upload required documents
- Check academic/administrative notifications

4.3 Admin Dashboard

- Manage students (add/update/delete)
- Approve/verify student documents
- Post campus announcements
- Monitor system usage

4.4 Database Management

- Store student profiles
- Maintain academic and personal records

- Store uploaded documents securely

5. Non-Functional Requirements

5.1 Performance Requirements

- Fast load times ensured by Vite + React
- API responses optimized for low latency
- Database queries must execute under 2 seconds

5.2 Security Requirements

- Passwords stored using hashing
- Prevention of SQL injection
- Secure API endpoints with validation
- Document access restricted by user roles

5.3 Reliability

- System must handle multiple concurrent users
- Database backup and restore options must be available

5.4 Usability

- Simple and intuitive UI for students and admin
- Responsive design for mobile and desktop

5.5 Scalability

- Can integrate future modules: attendance, fees, hostel, exams
- React and PHP architecture supports horizontal scaling

6. Interface Requirements

- User Interface
- React-based dashboards for both student and admin
- Forms, tables, modals designed using HTML/CSS/JS
- Backend API Interface
- REST API endpoints returning JSON
- Supports GET, POST, PUT, DELETE operations

7. Database Requirements

- MySQL database with structured tables:
- students
- admins
- documents
- notifications
- activity_logs
- Proper indexing for fast queries
- Foreign keys for relational integrity

SYSTEM DESIGN

System Design

System design for Campus Sphere involves structuring the architecture, components, and workflow to ensure an efficient, scalable, and reliable college management platform.

Architecture Overview

1. Campus Sphere follows a classic three-tier architecture:
2. **Presentation Layer (Frontend):** The user interface built with React and TypeScript. This is what users (students, teachers, admins, etc.) interact with to access information, submit data, and view dashboards.
3. **Application Layer (Backend):** A PHP-based server that handles all business logic. It processes API requests from the frontend, manages user authentication, interacts with the database, and executes core functionalities.
4. **Data Layer (Database):** A MySQL database that stores all persistent data for the application. This includes user records, academic information, financial details, and service requests, all defined by the campus_sphere.sql schema.

System Components

- User Interface (UI): Built using React, TypeScript, and styled with Tailwind CSS. It consists of interactive components for each user role.
- PHP Backend: Handles user authentication, serves API endpoints for all CRUD (Create, Read, Update, Delete) operations, and manages application logic.

- MySQL Database: Stores all relational data,
  including users, courses, grades, fee_records, and service_requests tables.
- Authentication & State Management: A context-based authentication system (AuthContext.tsx) manages user sessions and roles on the frontend.

System Workflow

1. **User Authentication:** Secure login for all roles (Student, Teacher, Admin, Parent, HOD). The backend verifies credentials against the users table in the database.
2. **Data Submission:** Users submit data via forms on the frontend (e.g., a student requesting a document via NewDocumentRequest.tsx).
3. **API Request:** The frontend sends the data to the appropriate PHP API endpoint (e.g., create_document_request.php).
4. **Backend Processing:** The PHP script sanitizes the input, applies business logic, and executes a SQL query to interact with the MySQL database.
5. **Data Storage & Retrieval:** The query result is stored in or retrieved from the relevant database table (e.g., a new entry in service_requests). The result is sent back to the frontend.
6. **User Access:** The frontend updates dynamically, allowing users to view the results of their actions (e.g., the student sees their new request in the "My Documents" list).

System Overview

Campus Sphere is a comprehensive, web-based college management platform that streamlines administrative and academic processes. The system provides role-based dashboards and modules for students, teachers, parents, administrators, and Heads of Department (HODs), facilitating seamless interaction and data management across the institution.

Key Components

- **Role-Based Dashboards:** A central interface for each user role (AdminDashboard.tsx, StudentDashboard.tsx, etc.) to access relevant modules and at-a-glance statistics.
- **Core Modules:**
  - **User Management** – (Admin) Create, update, and manage all user accounts.
  - **Document Request System** – (Student/Admin) Allows students to request official documents and admins to process them.
  - **Fee Management** – (Student/Admin/Parent) Manage, view, and process fee payments.

    - **Academic & Grade Management** – (Teacher/Student) Enables teachers to enter grades and students to view their academic records.
    - **Parent-Student Linking** – (Admin) A system for linking parent accounts to student accounts to view their progress.
- **Profile & Settings:** A common area for all users to manage their personal information and security settings.

Technology Stack

- **Backend:** PHP – Manages all API requests, server-side logic, and database interactions.
- **Frontend:** React, TypeScript, Vite, Tailwind CSS – Builds a modern, fast, and responsive user interface.
- **Database:** MySQL – Stores all application data in a relational format.
- **IDE:** Visual Studio Code – A flexible development environment.

User Management

- Secure registration and login system with credential verification against the database.
- Role-based access control (RBAC) implemented throughout the frontend and backend to ensure users can only access features appropriate for their role.

Accessibility & Scalability

- As a web-based system, it is accessible from any device with a browser, centralizing data and reducing manual administrative work.
- The component-based frontend and modular backend architecture allow for easy expansion with new features and scaling to handle more users.

Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a standardized visual language used for designing and modeling software systems. It helps developers, designers, and stakeholders visualize a system's structure, behavior, and interactions using diagrams. UML consists of structural and behavioral diagrams that define system components and their interactions.
A Use Case Diagram visually represents how different users (actors) interact with a system.
It includes:

- **Actors:** External entities (users or systems) that perform actions.
- **Use Cases:** Functionalities represented as ovals.
- **Relationships:** Lines connecting actors to use cases, showing interactions.

Class Diagram

A Class Diagram shows the system's structure by defining its classes, attributes, and relationships.

- **Classes:** Represent system components with attributes and methods.
- **Relationships:**
    - **Association:** One class interacts with another.
    - **Dependency:** One class depends on another.

Data Flow Diagram (DFD)-Level 0

A **DFD Level 0** represents the entire system as a single process interacting with external entities.

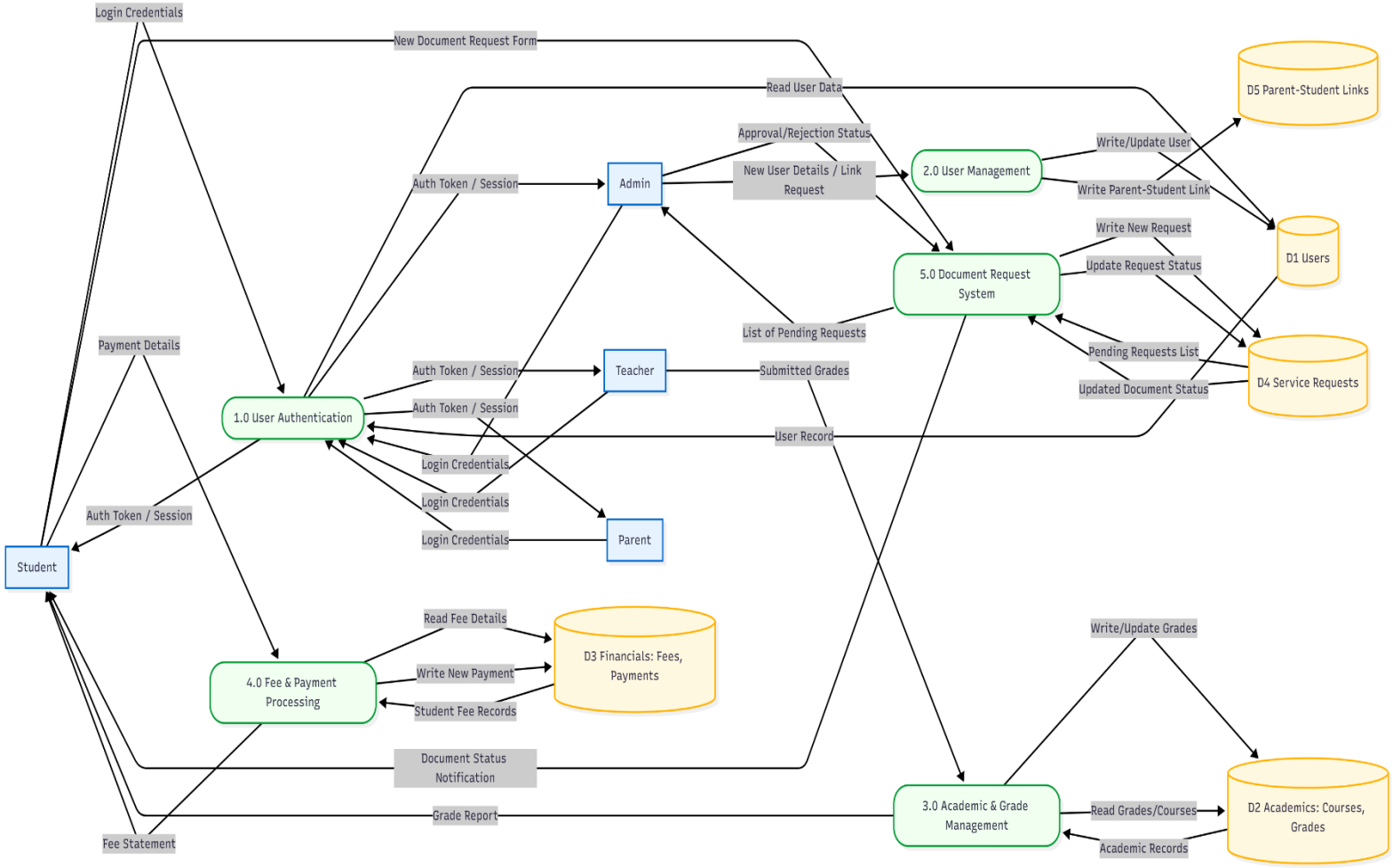
- **External Entities:** Users or external systems that exchange data.
- **Processes:** The central system handling input and output.
- **Data Flow:** Arrows representing data movement between entities and the system.

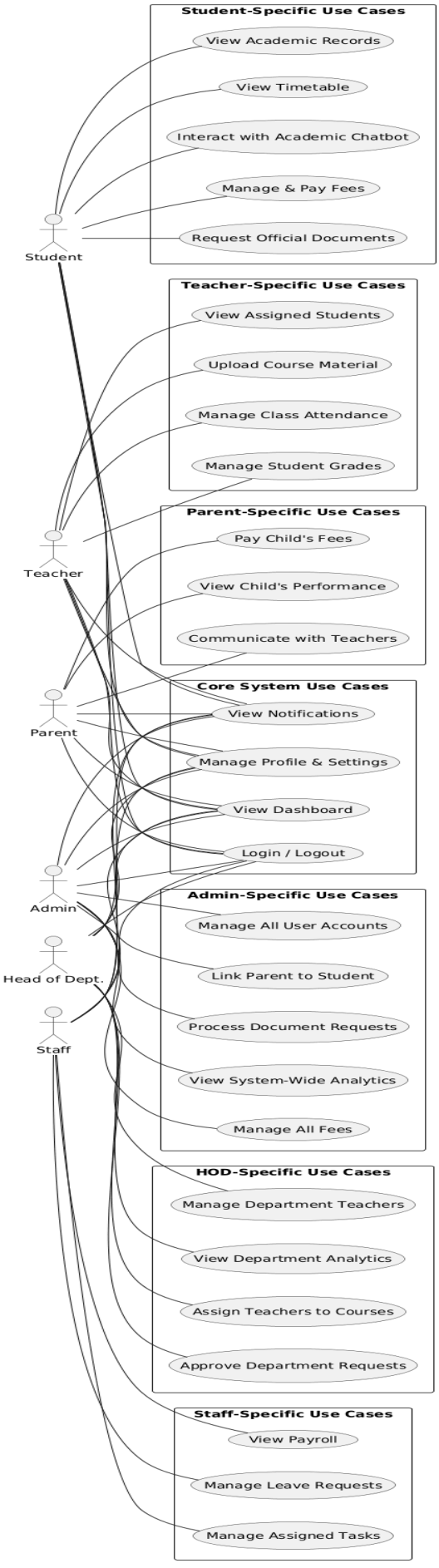
Data Flow Diagram (DFD)-Level 1

A DFD Level 1 breaks down the system into multiple sub-processes.

- **Sub-processes:** Individual tasks like data input, processing, and storage.
- **Data Stores:** Represent where information is saved (e.g., database tables).
- **Data Flow:** Shows how data moves between sub-processes, entities, and data stores

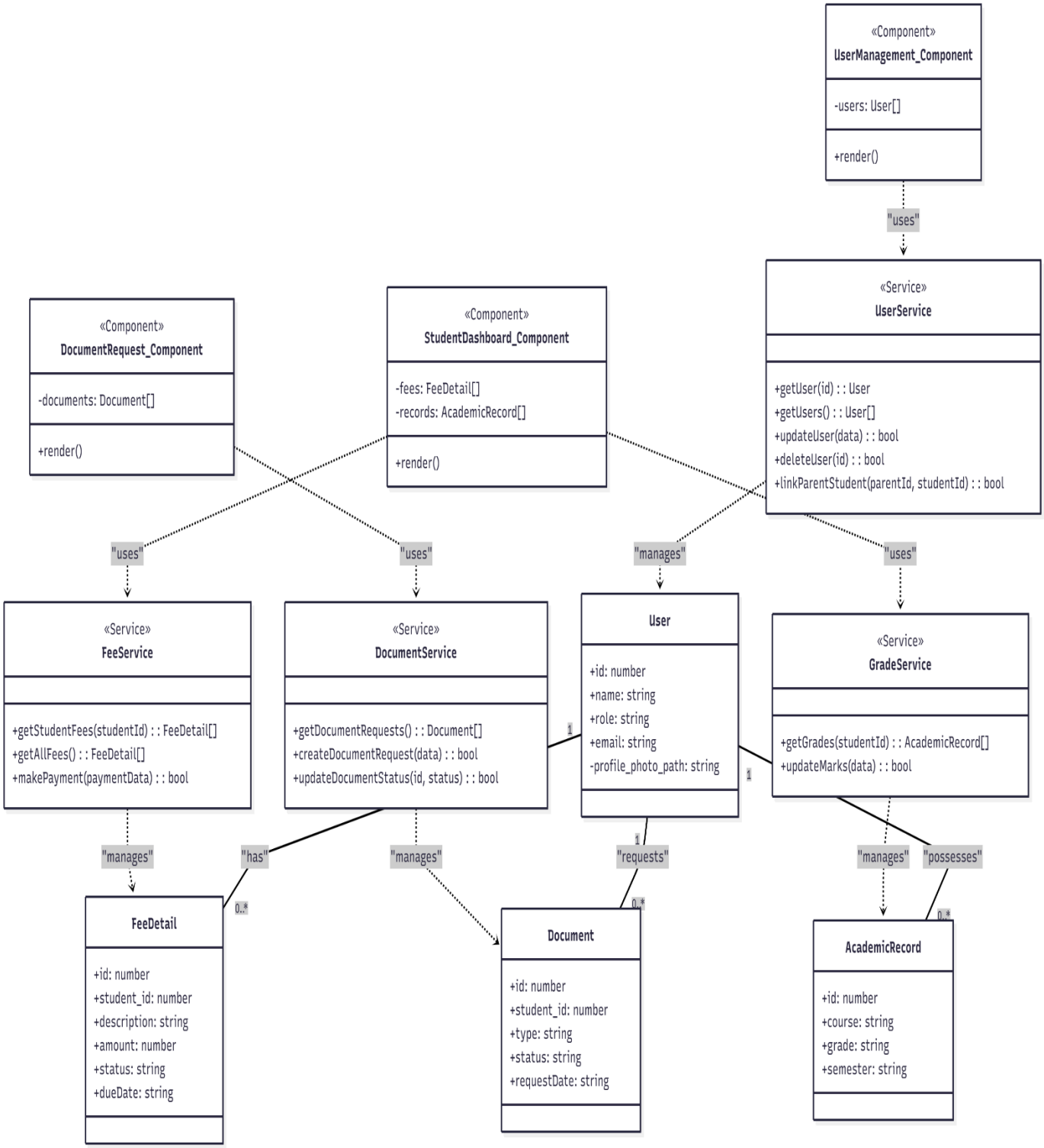
DATA FLOW DIAGRAM



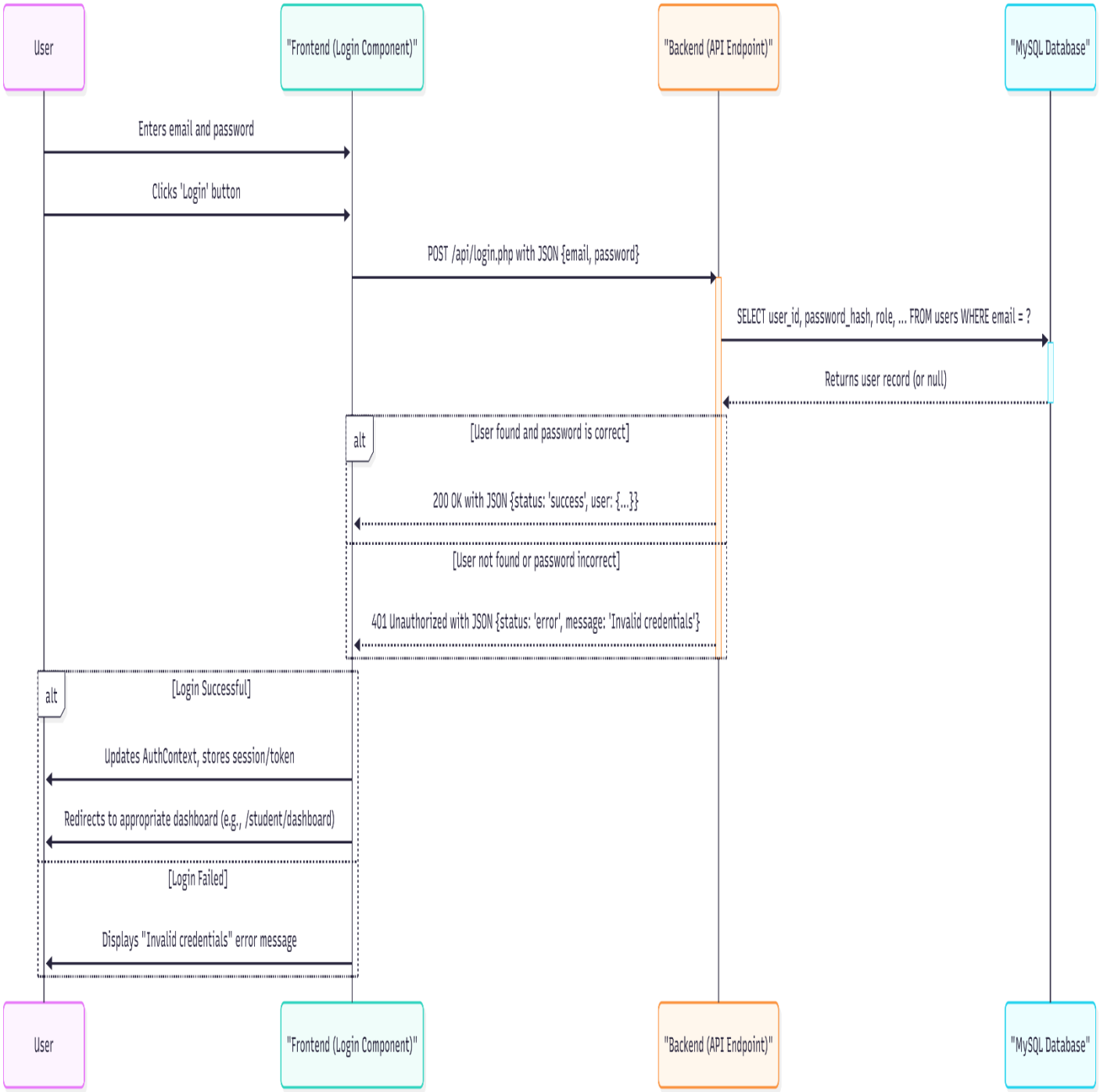


USE CASE DIAGRAM

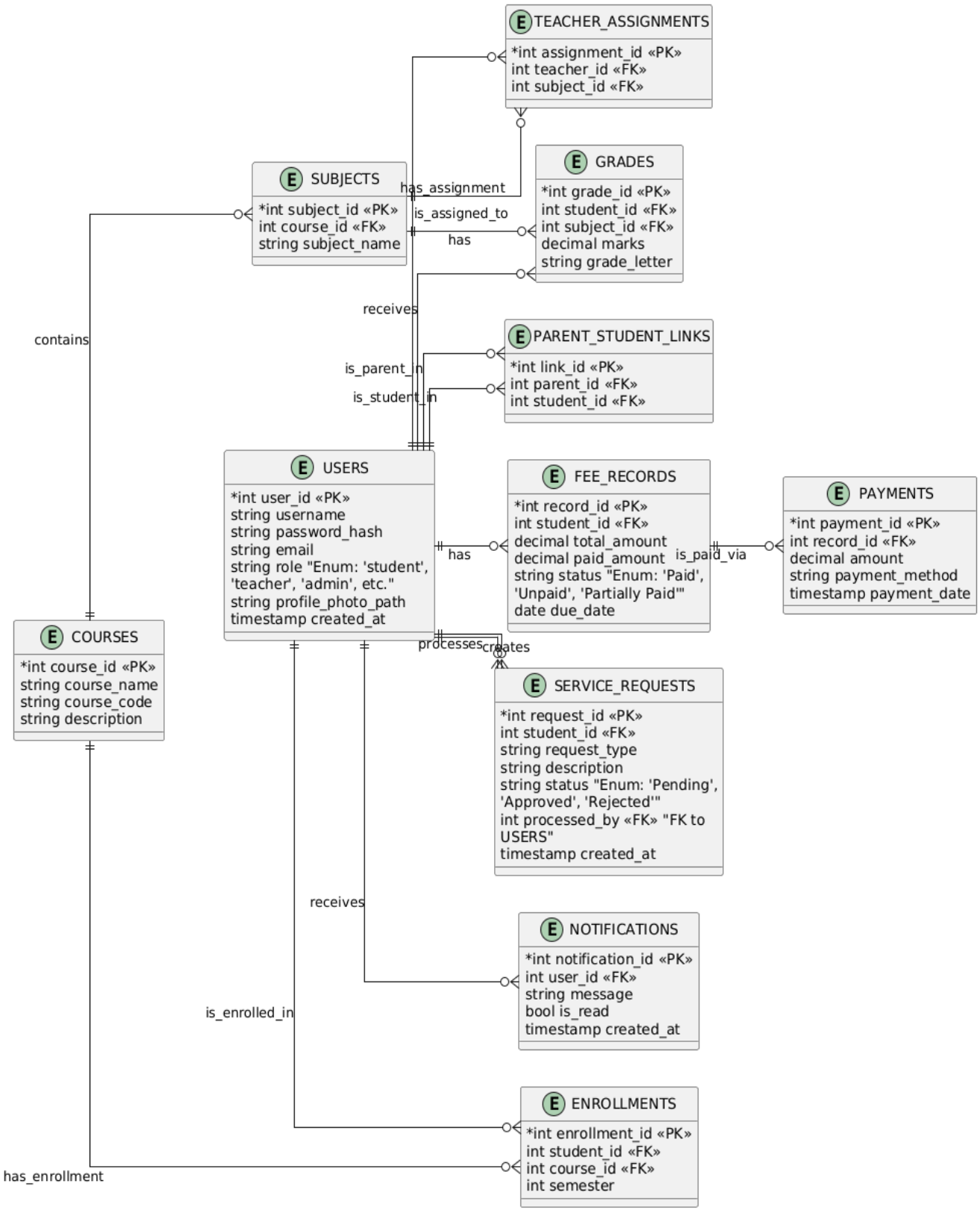
CLASS DIAGRAM



SEQUENCE DIAGRAM



ER DIAGRAM


TEACHER_ASSIGNMENTS
*int assignment_id «PK»
int teacher_id «FK»
int subject_id «FK»
SUBJECTS
*int subject_id «PK»
int course_id «FK»
string subject_name
GRADES
*int grade_id «PK»
int student_id «FK»
int subject_id «FK»
decimal marks
string grade_letter
has_assignment
is_assigned_to
has
receives
contains
PARENT_STUDENT_LINKS
*int link_id «PK»
int parent_id «FK»
int student_id «FK»
is_parent_in
is_student_in
USERS
*int user_id «PK»
string username
string password_hash
string email
string role "Enum: 'student', 'teacher', 'admin', etc."
string profile_photo_path
timestamp created_at
FEE_RECORDS
*int record_id «PK»
int student_id «FK»
decimal total_amount
decimal paid_amount
string status "Enum: 'Paid', 'Unpaid', 'Partially Paid'"
date due_date
has
PAYMENTS
*int payment_id «PK»
int record_id «FK»
decimal amount
string payment_method
timestamp payment_date
is_paid_via
COURSES
*int course_id «PK»
string course_name
string course_code
string description
processes
creates
SERVICE_REQUESTS
*int request_id «PK»
int student_id «FK»
string request_type
string description
string status "Enum: 'Pending', 'Approved', 'Rejected'"
int processed_by «FK» "FK to USERS"
timestamp created_at
receives
NOTIFICATIONS
*int notification_id «PK»
int user_id «FK»
string message
bool is_read
timestamp created_at
is_enrolled_in
ENROLLMENTS
*int enrollment_id «PK»
int student_id «FK»
int course_id «FK»
int semester
has_enrollment

SOFTWARE INSTALLATION

Software Installation:

The following software tools and dependencies are required for the development and deployment of the Smart Campus Sphere web-based campus management system. The platform uses a React + Vite frontend and a PHP–MySQL backend powered by XAMPP.

1. Frontend Technologies:

- Node.js (18+)
- Required to run the React development server.
- Provides npm (Node Package Manager) to install frontend dependencies.
- React (18.x)
- Framework used for building dynamic user interfaces.
- Handles routing, real-time UI updates, and component rendering.
- Vite
- A fast frontend build tool providing a lightning-fast dev server.
- Used for bundling the React application for production.
- NPM Packages Installed
- react, react-dom – Core React library.
- react-router-dom – Routing and navigation.
- axios or fetch – API integration with PHP backend.
- tailwindcss – Fast CSS utility framework.
- lucide-react – Icons for the UI.
- @types/react, @types/node – TypeScript definitions (if TS enabled).
- These packages ensure smooth rendering, styling, and data communication.

2. Backend Technologies:

- PHP (7+ recommended)
- Main backend scripting language.
- Handles authentication, validation, file uploads, and role-based operations.
- Generates JSON responses for the React frontend.
- XAMPP
- Provides:
- Apache Server to host PHP backend APIs.
- MySQL Server to store campus records.
- phpMyAdmin for database management.

- MySQL Database
- Stores:
- User accounts
- Student information
- Faculty & parent details
- Academic records
- Fee information
- Document requests
- Notifications & announcements
- Required PHP Extensions
- These come pre-configured in XAMPP:
- mysqli
- openssl
- json
- fileinfo
- session

3. Additional Tools:

- For testing backend APIs.
- VS Code
- Used for writing React and PHP code with syntax support.
- Git
- Manages project version control and deployments.

Software Installation & Environment Setup

The following steps explain how to set up the complete development environment for Smart Campus Sphere.

Step 1: Install Node.js:

- Check if Node is installed:
- node -v
- If not installed, download from: **https://nodejs.org**

Step 2: Install Frontend Dependencies:

- Navigate to the project folder:
- cd smart-campus-sphere
- Install all required packages:

- npm install
- This installs React, Vite, Tailwind CSS, and all other dependencies.

Step 3: Configure Vite Development Server:

- Start the frontend development server:
- npm run dev
- Vite will run the app on:
- http://localhost:5173/

Step 4: Install XAMPP:

- Download XAMPP from: **https://www.apachefriends.org/**
- Install and start:
- Apache (for PHP backend)
- MySQL (for database)

Step 5: Configure PHP Backend:

- Place your backend folder inside:
- C:\xampp\htdocs\campus-backend\
- Start Apache and MySQL from XAMPP.
- Test backend:
- http://localhost/campus-backend/api/test.php

Step 6: Create MySQL Database:

- Open phpMyAdmin:
- http://localhost/phpmyadmin/
- Create a new database:
- campus_sphere
- Import your SQL schema:
- Go to Import
- Select the SQL file from your backend
- Click Go
- This creates all required tables:
- users
- academic_records
- fee_details
- notifications
- documents
- parent_records

- staff_records
- tasks

Step 7: Configure Backend Environment Variables:

Create a config.php file:

```
<?php

$host = "localhost";

$user = "root";

$pass = "";

$db   = "campus_sphere";

$conn = new mysqli($host, $user, $pass, $db);

if($conn->connect_error){

die("Connection failed: " . $conn->connect_error);

}

?>
```

Step 8: Connect React Frontend with PHP Backend:

- Set API base URL in your React configuration (usually inside services/api.js):
- export const BASE_URL = "http://localhost/campus-backend/";

Step 9: Test Full System:

- Open frontend:
  http://localhost:5173
- Login as student/teacher/admin.
- Ensure:
- Dashboard loads correctly
- API calls reach the backend
- Data is fetched from MySQL
- Profile uploads work
- Document requests are stored

Step 10: Build Frontend for Deployment:

- For production build:

- npm run build
- Vite generates optimized files inside:
- dist/
- These can be hosted on:
- Apache Server
- Nginx
- Any static hosting platform

Source Code And Testing

Coding:

In the **OrbitCampus** project, coding forms the core foundation that enables seamless interaction between the frontend user interface, backend logic, and database management systems. The system is built using a modern tech stack consisting of React + Vite, PHP, and MySQL, ensuring high performance, data security, and a smooth user experience.
Below are the primary areas where coding is applied:

1. Frontend Development (React JS, HTML, CSS, JavaScript):

a. React (Main Frontend Framework):

React is used to build the entire user interface using component-based architecture.
Key Implementations:

- Page routing using react-router-dom.
- Dynamic dashboards for students, faculty, and administrators.
- State management using React Hooks (useState, useEffect, useContext).
- Fetching backend data through Axios or Fetch API.
- UI updates without reloading the page (SPA architecture).

b. HTML (Structure Layer):

Used within React components (JSX) to define:

- Login forms
- Dashboard layouts

- Tables, cards, modals, and content sections
- Document upload interfaces

c. CSS (Styling & UI Design):

CSS is used to create:

- Responsive layouts
- Color themes
- Component spacing and alignment
- Modern UI elements (buttons, cards, grids)
- Smooth animations and transitions

d. JavaScript (Logic & Interaction):

JavaScript powers:

- Form validations
- API requests to the PHP backend
- Displaying real-time data
- User interactions (clicks, navigation, menu toggles)
- Rendering dynamic elements inside React components

2. Backend Development (PHP):

PHP functions as the server-side engine of OrbitCampus.
Key Backend Coding Responsibilities:

a. REST API Development:

PHP scripts are written to create API endpoints for:

- Login authentication
- Fetching student/faculty/admin dashboards
- Document requests and uploads
- Profile updates

- Notification retrieval

b. User Authentication:

- Secure login using hashed passwords
- Session-based authentication
- Role-based access (student, admin, faculty)

c. Business Logic Implementation:

Backend code handles:

- Validating form data
- Approving or rejecting document requests
- Updating fee records
- Managing announcements and notifications

d. Error Handling & Security:

PHP includes:

- SQL injection prevention (prepared statements)
- Input sanitization
- Session handling
- File upload restrictions

3. Database Layer (MySQL):

MySQL is used to store all essential campus data.

Key Coding Activities:

- Writing SQL queries for CRUD operations
- Designing relational tables for users, documents, notices, fees, and profiles
- Managing foreign key relationships
- Optimizing query performance
- Integrating MySQL with PHP using mysqli or PDO

Examples of data stored:

- Login credentials
- Student & faculty details
- Uploaded documents
- Fee structures
- Announcements

4. Data Communication Layer (JSON):

JSON format is used for exchanging data between:
React Frontend ↔ PHP Backend

Examples:

- Sending login credentials from React to PHP
- Returning dashboard data in structured JSON arrays
- Sending request statuses and form responses

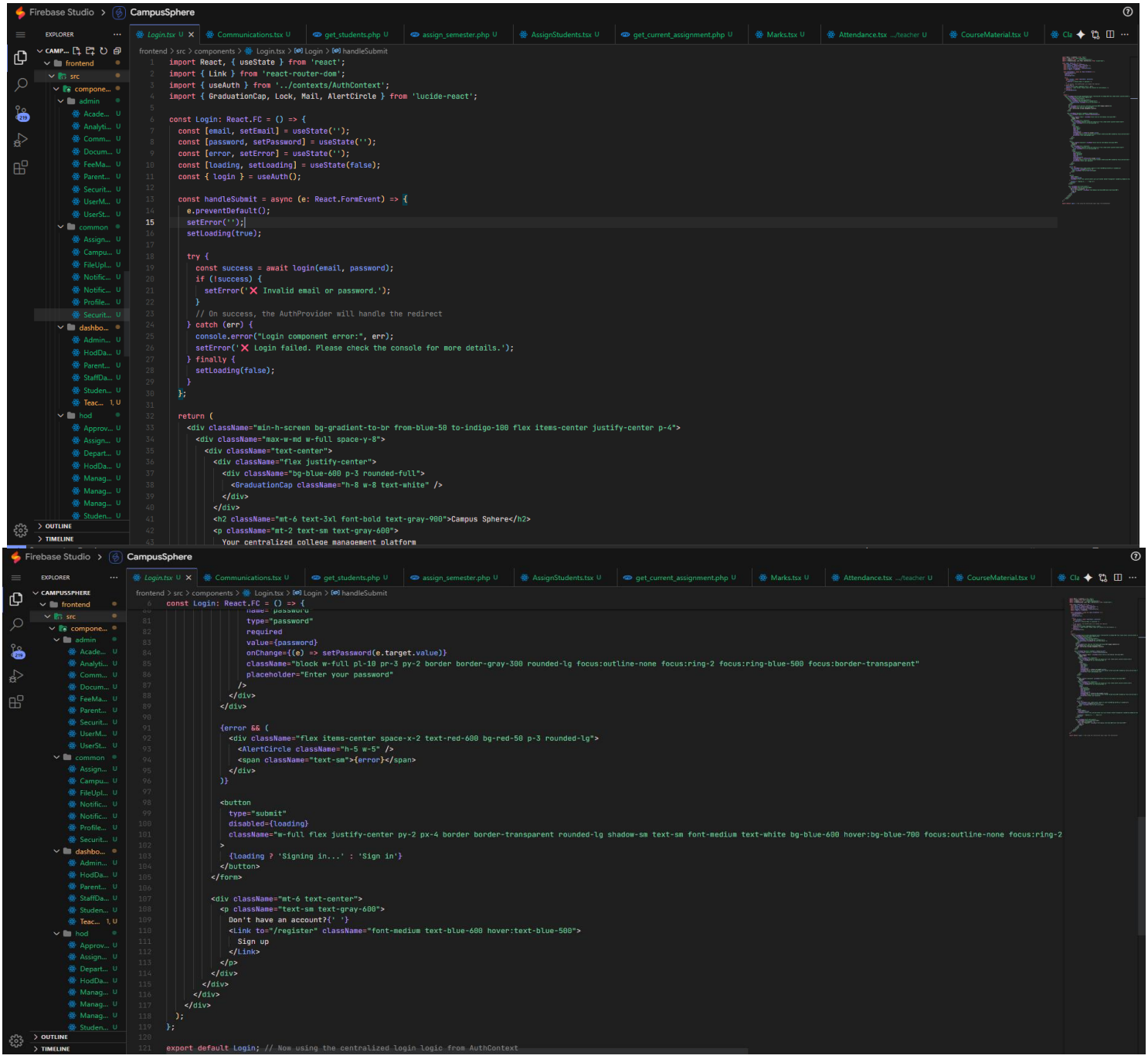
JSON ensures lightweight, secure, and fast data transfer.

5. Development & Build Tooling (Vite):

Role of Vite in Coding

- Provides ultra-fast development environment
- Hot Module Reloading (HMR) for instant UI preview
- Optimizes React code for production builds
- Bundles CSS, JS, and assets efficiently

LOGIN PAGE

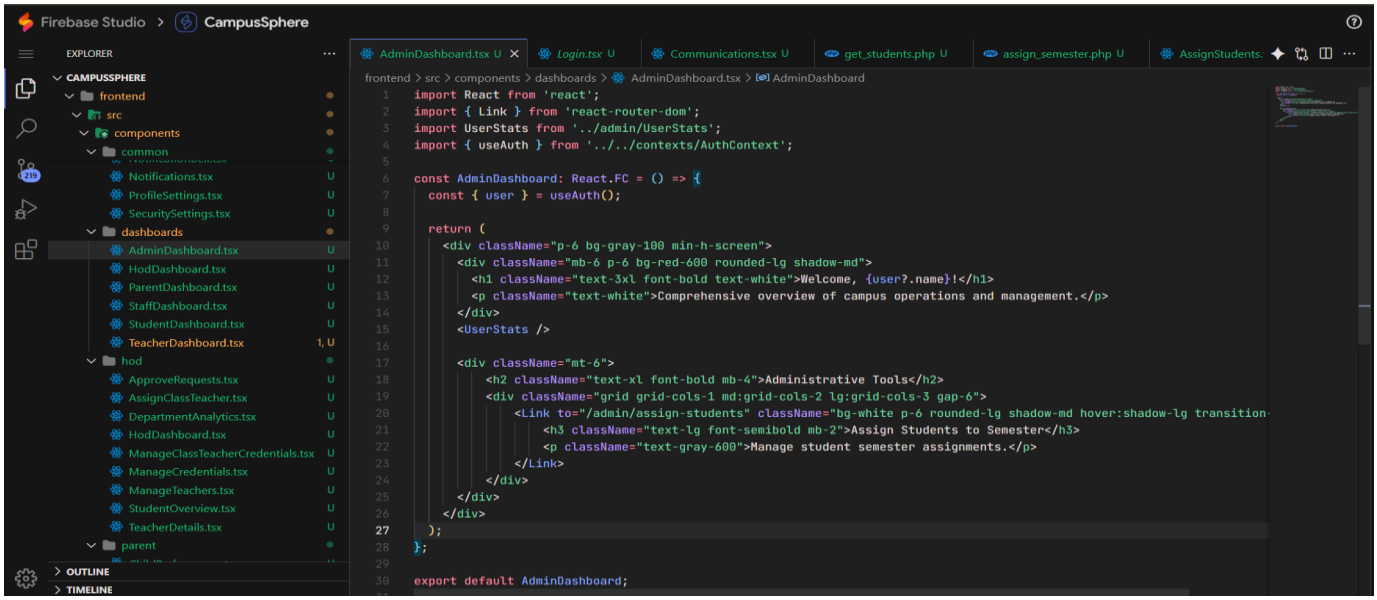

Firebase Studio
CampusSphere
Login.tsx
Communications.tsx
get_students.php
assign_semester.php
AssignStudents.tsx
get_current_assignment.php
Marks.tsx
Attendance.tsx
CourseMaterial.tsx
frontend > src > components > Login.tsx > Login > handleSubmit
import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import { useAuth } from '../contexts/AuthContext';
import { GraduationCap, Lock, Mail, AlertCircle } from 'lucide-react';
const Login: React.FC = () => {
const [email, setEmail] = useState('');
const [password, setPassword] = useState('');
const [error, setError] = useState('');
const [loading, setLoading] = useState(false);
const { login } = useAuth();
const handleSubmit = async (e: React.FormEvent) => {
e.preventDefault();
setError('');
setLoading(true);
try {
const success = await login(email, password);
if (!success) {
setError('❌ Invalid email or password.');
}
// On success, the AuthProvider will handle the redirect
} catch (err) {
console.error("Login component error:", err);
setError('❌ Login failed. Please check the console for more details.');
} finally {
setLoading(false);
}
};
return (

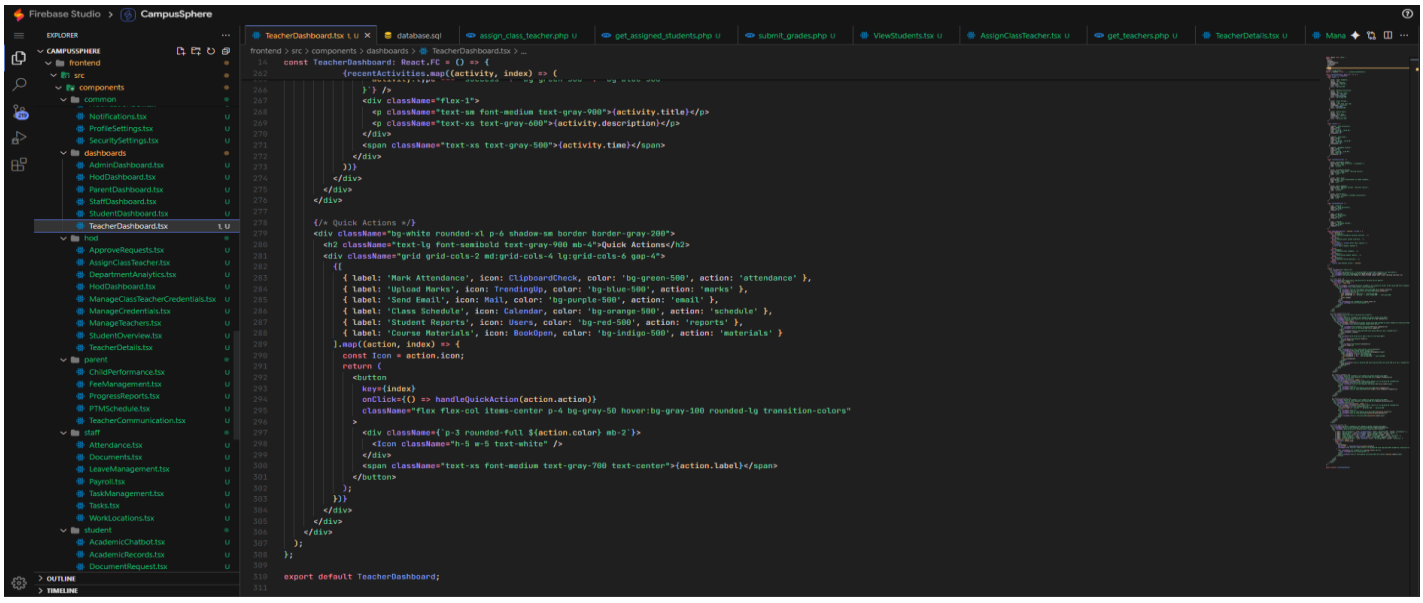
<GraduationCap className="h-8 w-8 text-white" />

<h2 className="mt-6 text-3xl font-bold text-gray-900">Campus Sphere</h2>
<p className="mt-2 text-sm text-gray-600">
Your centralized college management platform

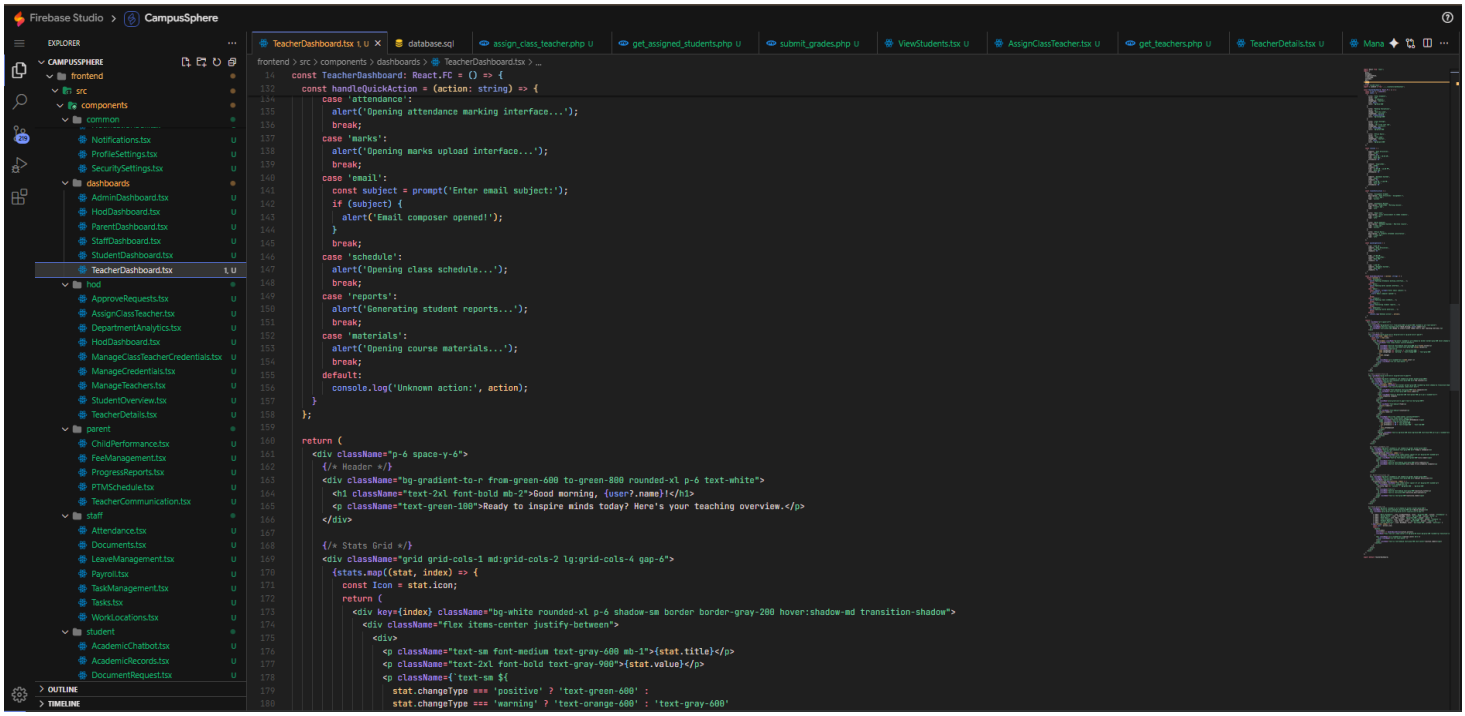
Firebase Studio
CampusSphere
Login.tsx
frontend > src > components > Login.tsx > Login > handleSubmit
const Login: React.FC = () => {
type="password"
required
value={password}
onChange={(e) => setPassword(e.target.value)}
className="block w-full pl-10 pr-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500 focus:border-transparent"
placeholder="Enter your password"
{error && (
<AlertCircle className="h-5 w-5" />
{error}
)}
<button
type="submit"
disabled={loading}
className="w-full flex justify-center py-2 px-4 border border-transparent rounded-lg shadow-sm text-sm font-medium text-white bg-blue-600 hover:bg-blue-700 focus:outline-none focus:ring-2
{loading ? 'Signing in...' : 'Sign in'}
Don't have an account?{' '}
<Link to="/register" className="font-medium text-blue-600 hover:text-blue-500">
Sign up
export default Login; // Now using the centralized login logic from AuthContext

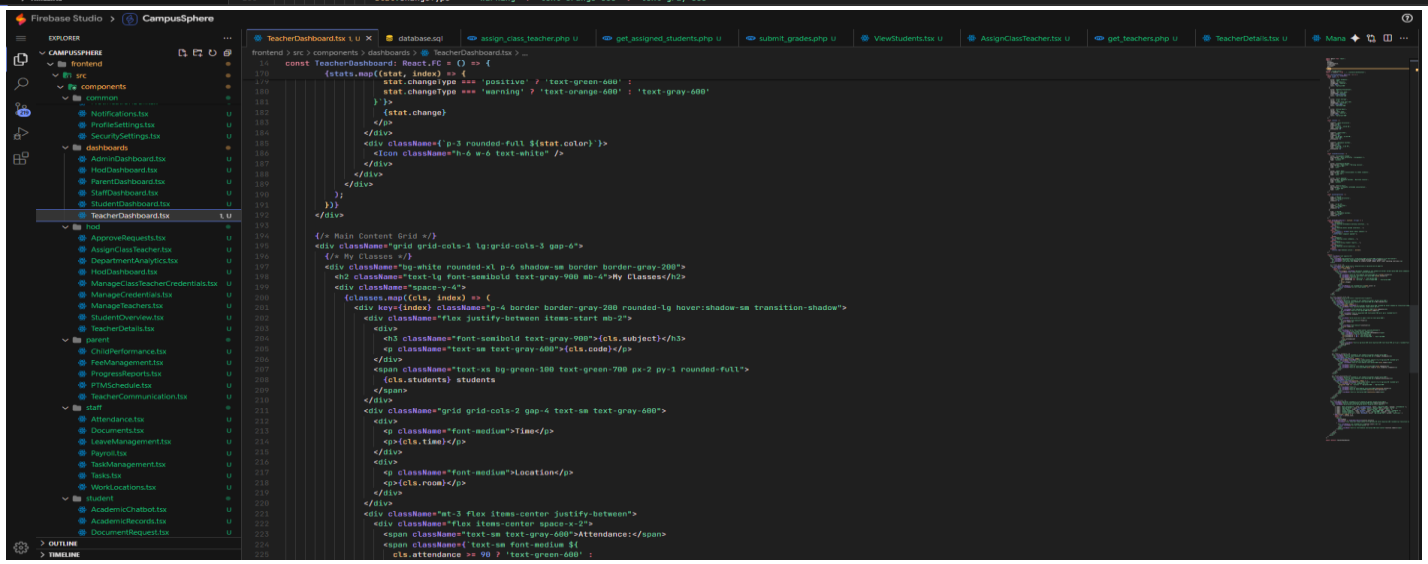
ADMIN DASHBOARD



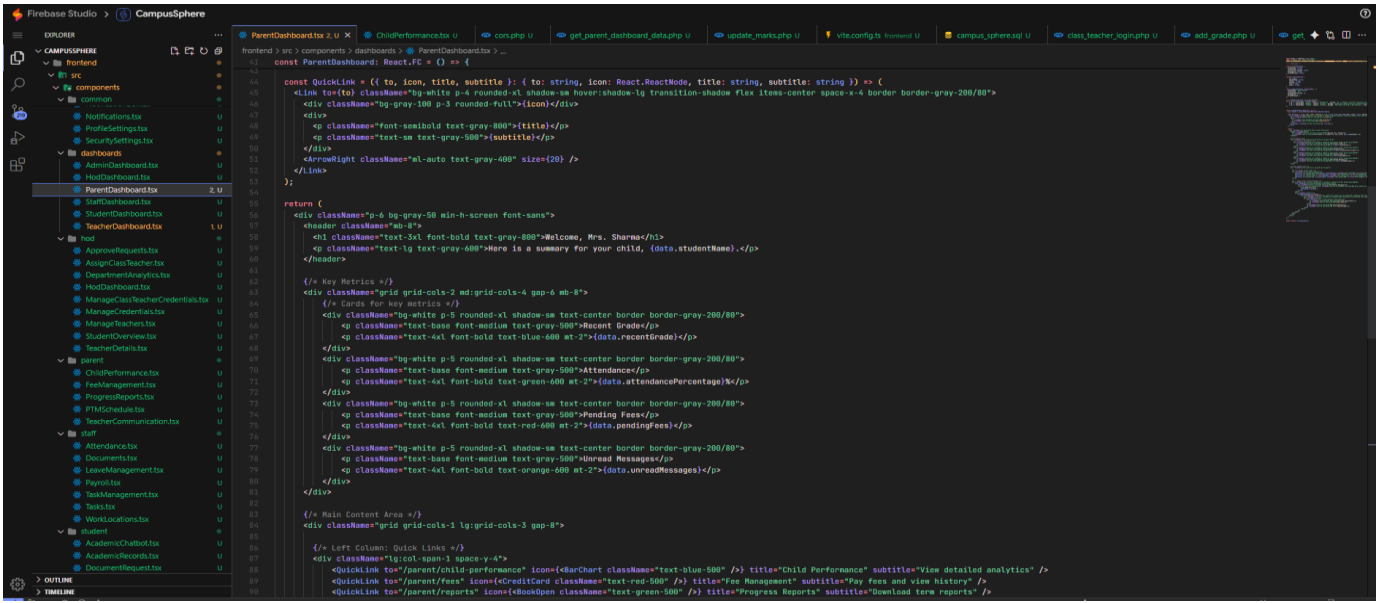


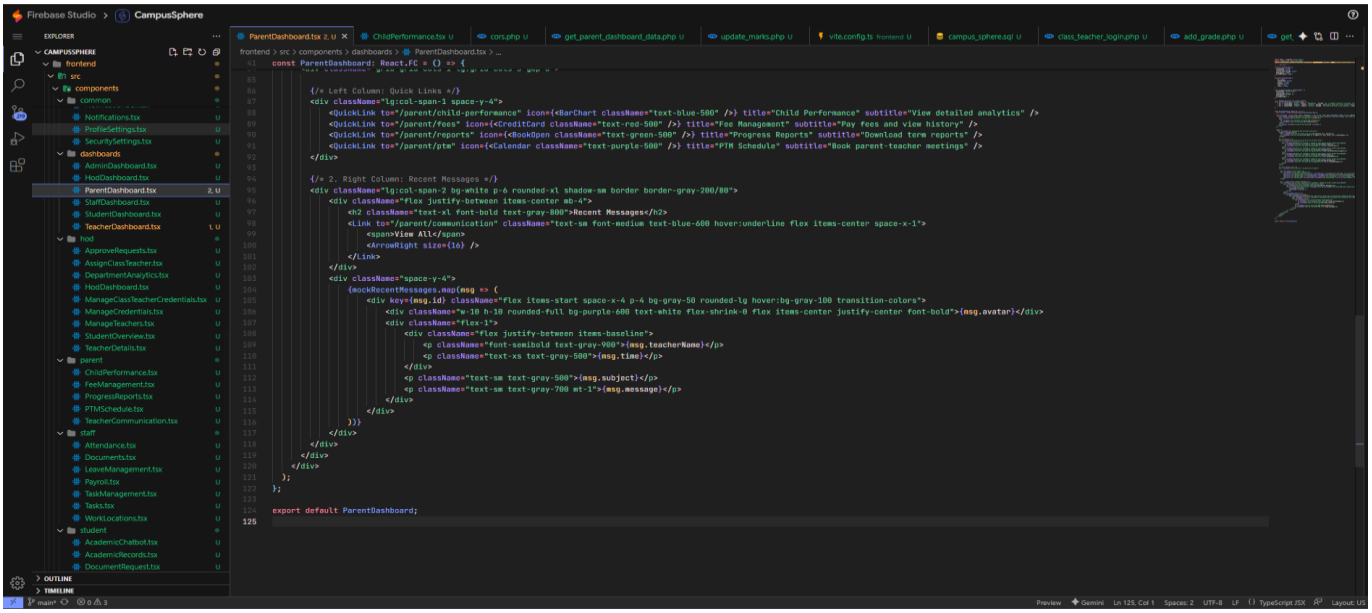
TEACHER DASHBOARD



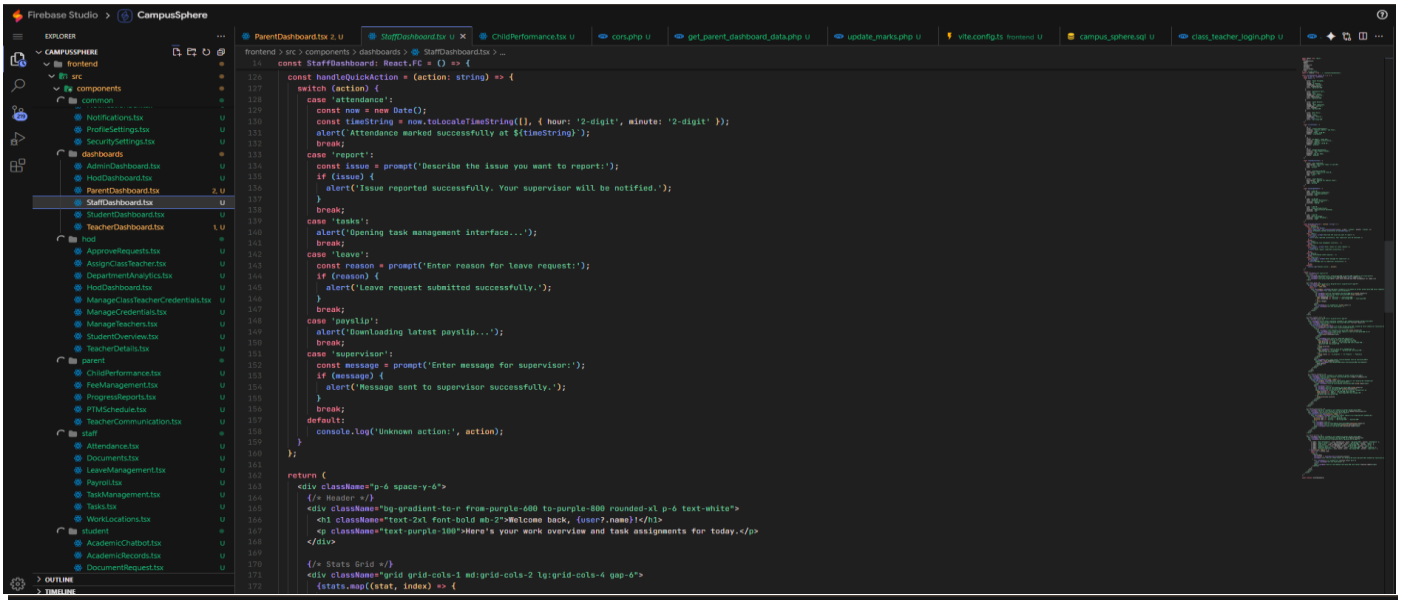


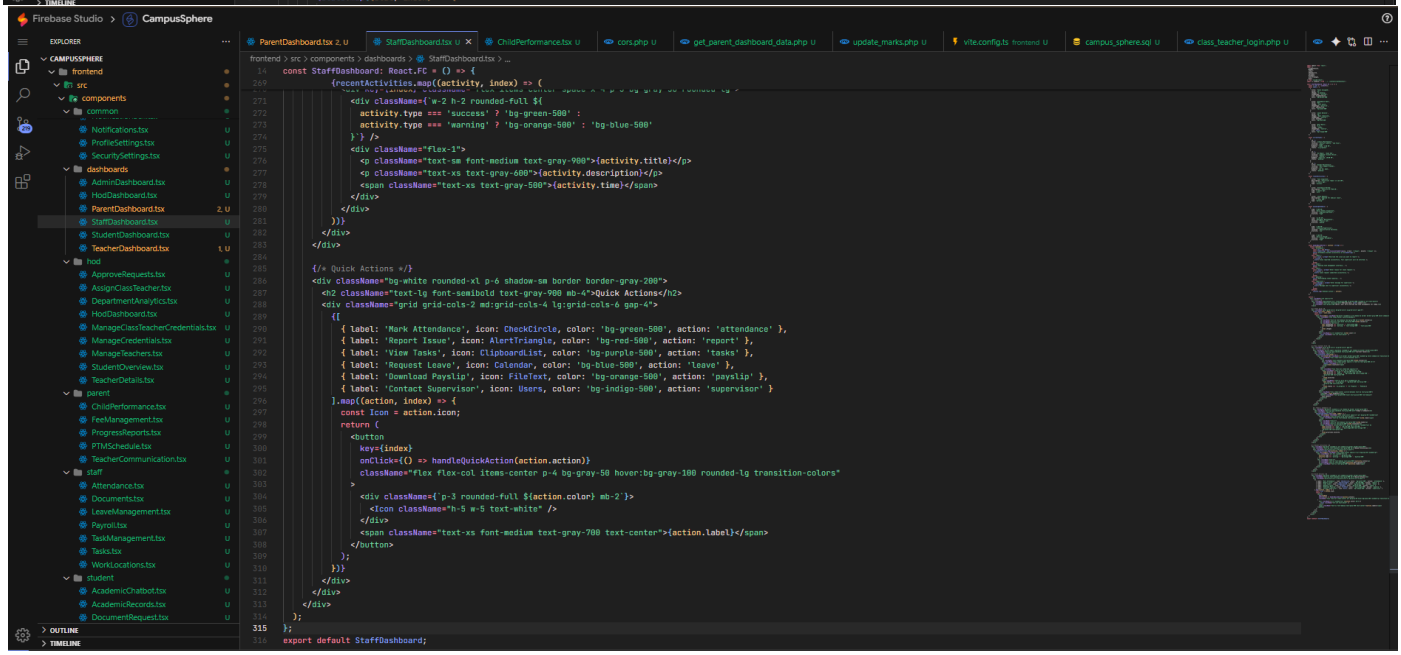
PARENT DASHBOARD



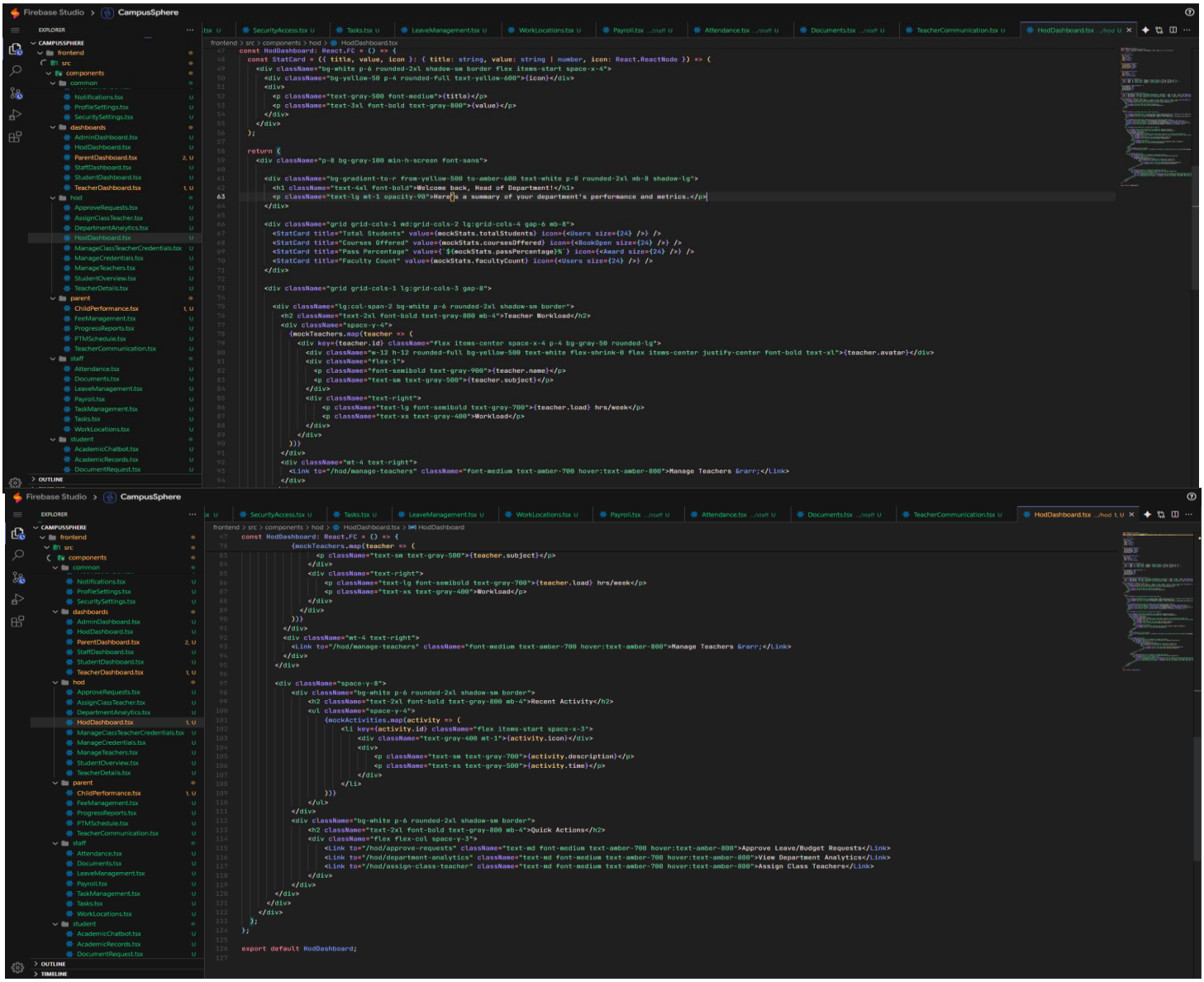


STAFF DASHBOARD



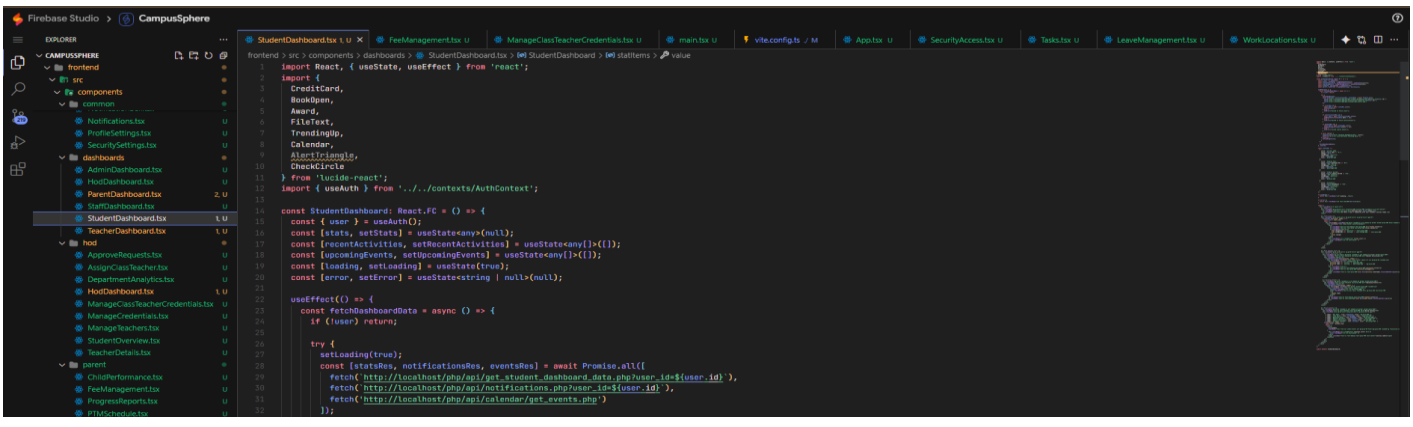


HOD DASHBOARD

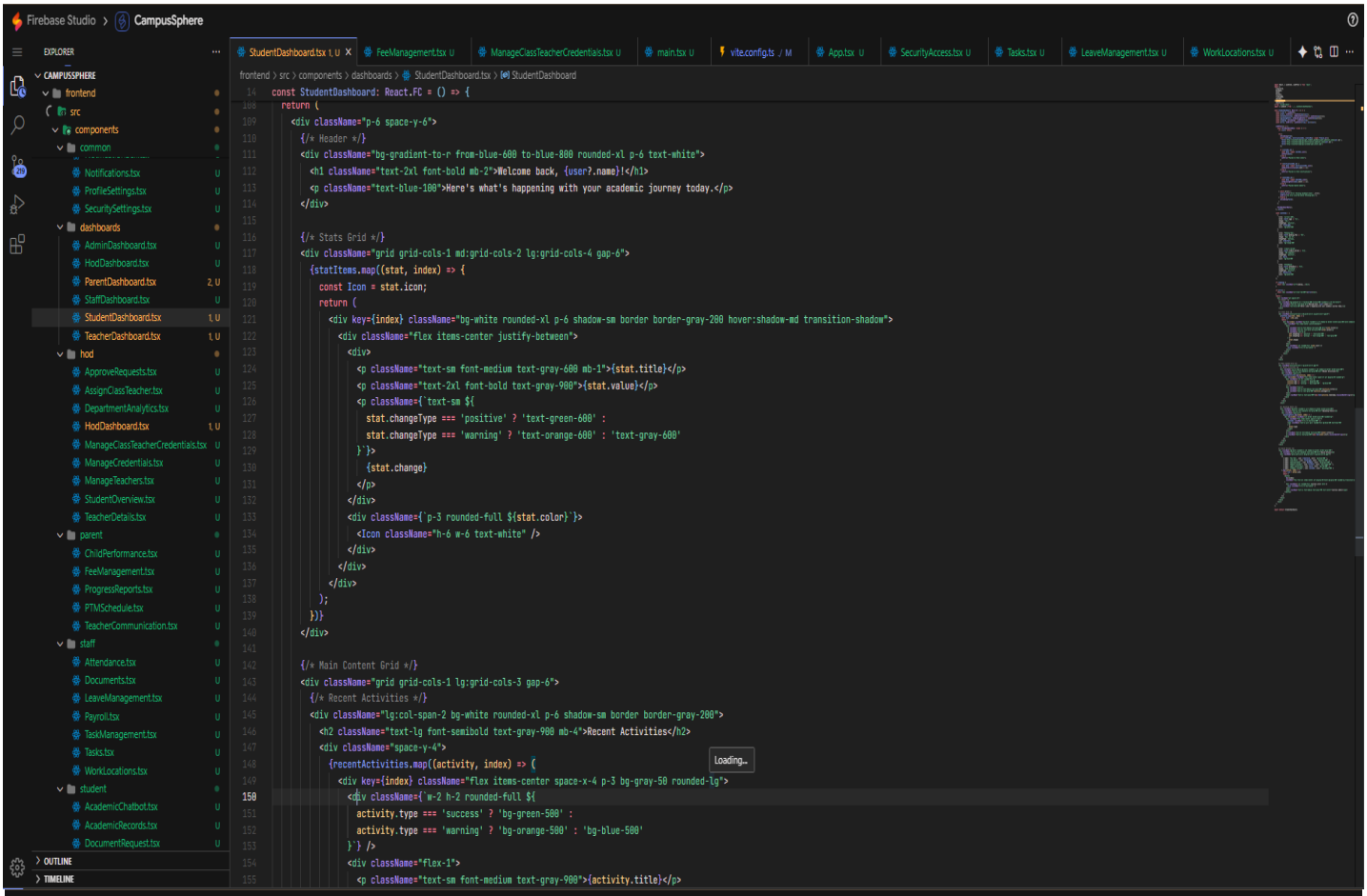


```tsx
const HodDashboard: React.FC = () => {
  const StatCard = ({ title, value, icon }: { title: string, value: string | number, icon: React.ReactNode }) => (
    <div className="bg-white p-6 rounded-2xl shadow-sm border flex items-start space-x-4">
      <div className="bg-yellow-50 p-4 rounded-full text-yellow-600">{icon}</div>
      <div>
        <p className="text-gray-500 font-medium">{title}</p>
        <p className="text-3xl font-bold text-gray-800">{value}</p>
      </div>
    </div>
  );

  return (
    <div className="p-8 bg-gray-100 min-h-screen font-sans">

      <div className="bg-gradient-to-r from-yellow-500 to-amber-600 text-white p-8 rounded-2xl mb-8 shadow-lg">
        <h1 className="text-4xl font-bold">Welcome back, Head of Department!</h1>
        <p className="text-lg mt-1 opacity-90">Here's a summary of your department's performance and metrics.</p>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-8">
        <StatCard title="Total Students" value={mockStats.totalStudents} icon={<Users size={24} />} />
        <StatCard title="Courses Offered" value={mockStats.coursesOffered} icon={<BookOpen size={24} />} />
        <StatCard title="Pass Percentage" value={`${mockStats.passPercentage}%`} icon={<Award size={24} />} />
        <StatCard title="Faculty Count" value={mockStats.facultyCount} icon={<Users size={24} />} />
      </div>

      <div className="grid grid-cols-1 lg:grid-cols-3 gap-8">

        <div className="lg:col-span-2 bg-white p-6 rounded-2xl shadow-sm border">
          <h2 className="text-2xl font-bold text-gray-800 mb-4">Teacher Workload</h2>
          <div className="space-y-4">
            {mockTeachers.map(teacher => (
              <div key={teacher.id} className="flex items-center space-x-4 p-4 bg-gray-50 rounded-lg">
                <div className="w-12 h-12 rounded-full bg-yellow-500 text-white flex-shrink-0 flex items-center justify-center font-bold text-xl">{teacher.avatar}</div>
                <div className="flex-1">
                  <p className="font-semibold text-gray-900">{teacher.name}</p>
                  <p className="text-sm text-gray-500">{teacher.subject}</p>
                </div>
                <div className="text-right">
                    <p className="text-lg font-semibold text-gray-700">{teacher.load} hrs/week</p>
                    <p className="text-xs text-gray-400">Workload</p>
                </div>
              </div>
            ))}
          </div>
          <div className="mt-4 text-right">
            <Link to="/hod/manage-teachers" className="font-medium text-amber-700 hover:text-amber-800">Manage Teachers &rarr;</Link>
          </div>
        </div>

        <div className="space-y-8">
            <div className="bg-white p-6 rounded-2xl shadow-sm border">
                <h2 className="text-2xl font-bold text-gray-800 mb-4">Recent Activity</h2>
                <ul className="space-y-4">
                    {mockActivities.map(activity => (
                        <li key={activity.id} className="flex items-start space-x-3">
                            <div className="text-gray-400 mt-1">{activity.icon}</div>
                            <div>
                                <p className="text-sm text-gray-700">{activity.description}</p>
                                <p className="text-xs text-gray-500">{activity.time}</p>
                            </div>
                        </li>
                    ))}
                </ul>
            </div>
            <div className="bg-white p-6 rounded-2xl shadow-sm border">
                <h2 className="text-2xl font-bold text-gray-800 mb-4">Quick Actions</h2>
                <div className="flex flex-col space-y-3">
                    <Link to="/hod/approve-requests" className="text-md font-medium text-amber-700 hover:text-amber-800">Approve Leave/Budget Requests</Link>
                    <Link to="/hod/department-analytics" className="text-md font-medium text-amber-700 hover:text-amber-800">View Department Analytics</Link>
                    <Link to="/hod/assign-class-teacher" className="text-md font-medium text-amber-700 hover:text-amber-800">Assign Class Teachers</Link>
                </div>
            </div>
        </div>
      </div>
    </div>
  );
};

export default HodDashboard;
```

STUDENT DASHBOARD



```tsx
import React, { useState, useEffect } from 'react';
import {
  CreditCard,
  BookOpen,
  Award,
  FileText,
  TrendingUp,
  Calendar,
  AlertTriangle,
  CheckCircle
} from 'lucide-react';
import { useAuth } from '../../contexts/AuthContext';

const StudentDashboard: React.FC = () => {
  const { user } = useAuth();
  const [stats, setStats] = useState<any>(null);
  const [recentActivities, setRecentActivities] = useState<any[]>([]);
  const [upcomingEvents, setUpcomingEvents] = useState<any[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const fetchDashboardData = async () => {
      if (!user) return;

      try {
        setLoading(true);
        const [statsRes, notificationsRes, eventsRes] = await Promise.all([
          fetch(`http://localhost/php/api/get_student_dashboard_data.php?user_id=${user.id}`),
          fetch(`http://localhost/php/api/notifications.php?user_id=${user.id}`),
          fetch('http://localhost/php/api/calendar/get_events.php')
        ]);
```


Firebase Studio
CampusSphere
EXPLORER
CAMPUSSPHERE
frontend
src
components
common
Notifications.tsx
ProfileSettings.tsx
SecuritySettings.tsx
dashboards
AdminDashboard.tsx
HodDashboard.tsx
ParentDashboard.tsx
StaffDashboard.tsx
StudentDashboard.tsx
TeacherDashboard.tsx
hod
ApproveRequests.tsx
AssignClassTeacher.tsx
DepartmentAnalytics.tsx
HodDashboard.tsx
ManageClassTeacherCredentials.tsx
ManageCredentials.tsx
ManageTeachers.tsx
StudentOverview.tsx
TeacherDetails.tsx
parent
ChildPerformance.tsx
FeeManagement.tsx
ProgressReports.tsx
PTMSchedule.tsx
TeacherCommunication.tsx
staff
Attendance.tsx
Documents.tsx
LeaveManagement.tsx
Payroll.tsx
TaskManagement.tsx
Tasks.tsx
WorkLocations.tsx
student
AcademicChatbot.tsx
AcademicRecords.tsx
DocumentRequest.tsx
OUTLINE
TIMELINE
StudentDashboard.tsx 1, U
FeeManagement.tsx U
ManageClassTeacherCredentials.tsx U
main.tsx U
vite.config.ts / M
App.tsx U
SecurityAccess.tsx U
Tasks.tsx U
LeaveManagement.tsx U
WorkLocations.tsx U
frontend > src > components > dashboards > StudentDashboard.tsx > StudentDashboard
const StudentDashboard: React.FC = () => {
return (

{/* Header */}

<h1 className="text-2xl font-bold mb-2">Welcome back, {user?.name}!</h1>
Here's what's happening with your academic journey today.

{/* Stats Grid */}

{statItems.map((stat, index) => {
const Icon = stat.icon;
return (

{stat.title}
{stat.value}
<p className={`text-sm ${
stat.changeType === 'positive' ? 'text-green-600' :
stat.changeType === 'warning' ? 'text-orange-600' : 'text-gray-600'
}`}>
{stat.change}

<div className={`p-3 rounded-full ${stat.color}`}>
<Icon className="h-6 w-6 text-white" />

);
})}

{/* Main Content Grid */}

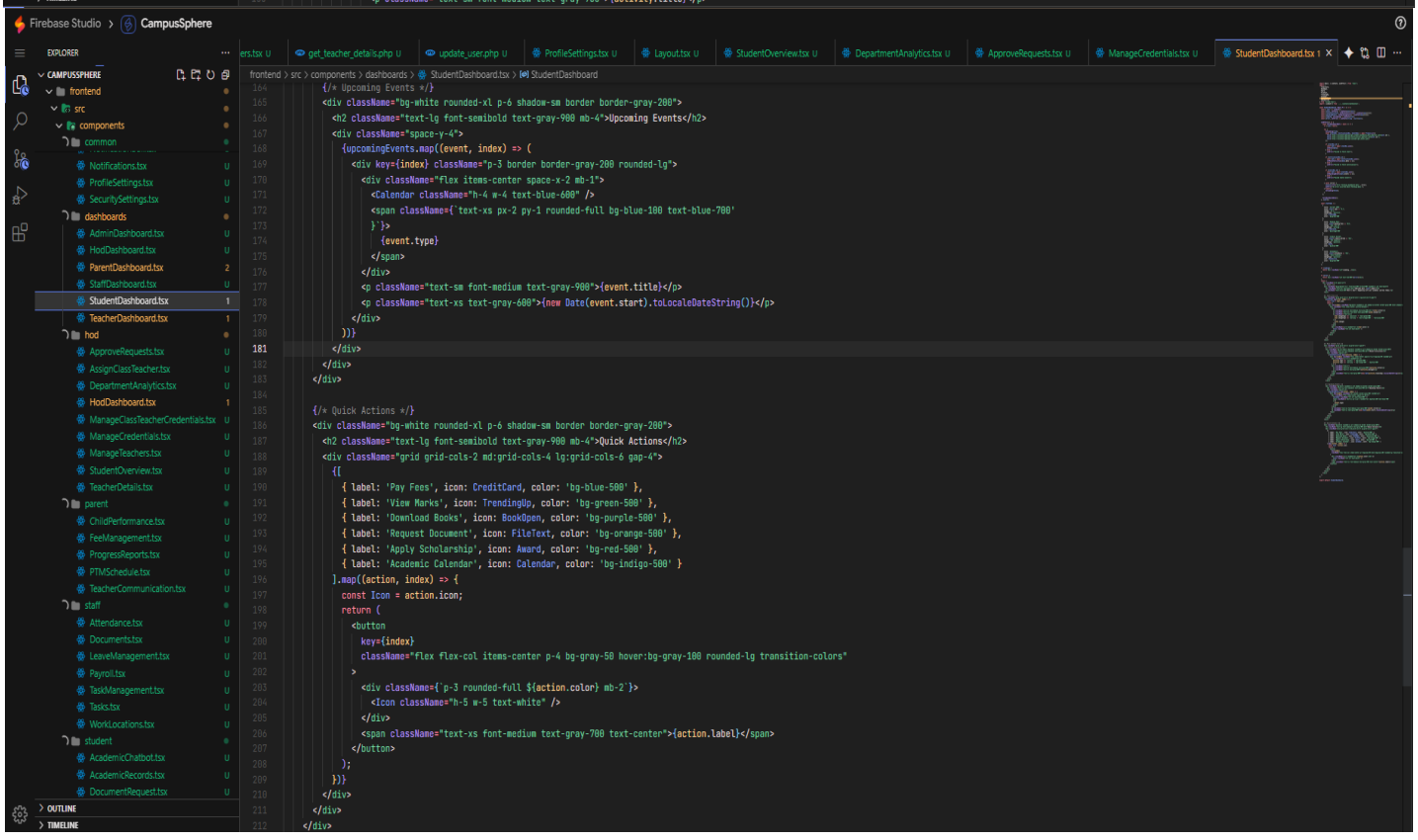
{/* Recent Activities */}

<h2 className="text-lg font-semibold text-gray-900 mb-4">Recent Activities</h2>

{recentActivities.map((activity, index) => (

<div className={`w-2 h-2 rounded-full ${
activity.type === 'success' ? 'bg-green-500' :
activity.type === 'warning' ? 'bg-orange-500' : 'bg-blue-500'
}`} />

{activity.title}
Loading...


Firebase Studio
CampusSphere
EXPLORER
CAMPUSSPHERE
frontend
src
components
common
Notifications.tsx
ProfileSettings.tsx
SecuritySettings.tsx
dashboards
AdminDashboard.tsx
HodDashboard.tsx
ParentDashboard.tsx
StaffDashboard.tsx
StudentDashboard.tsx
TeacherDashboard.tsx
hod
ApproveRequests.tsx
AssignClassTeacher.tsx
DepartmentAnalytics.tsx
HodDashboard.tsx
ManageClassTeacherCredentials.tsx
ManageCredentials.tsx
ManageTeachers.tsx
StudentOverview.tsx
TeacherDetails.tsx
parent
ChildPerformance.tsx
FeeManagement.tsx
ProgressReports.tsx
PTMSchedule.tsx
TeacherCommunication.tsx
staff
Attendance.tsx
Documents.tsx
LeaveManagement.tsx
Payroll.tsx
TaskManagement.tsx
Tasks.tsx
WorkLocations.tsx
student
AcademicChatbot.tsx
AcademicRecords.tsx
DocumentRequest.tsx
OUTLINE
TIMELINE
get_teacher_details.php U
update_user.php U
ProfileSettings.tsx U
Layout.tsx U
StudentOverview.tsx U
DepartmentAnalytics.tsx U
ApproveRequests.tsx U
ManageCredentials.tsx U
StudentDashboard.tsx 1
frontend > src > components > dashboards > StudentDashboard.tsx > StudentDashboard
{/* Upcoming Events */}

<h2 className="text-lg font-semibold text-gray-900 mb-4">Upcoming Events</h2>

{upcomingEvents.map((event, index) => (

<Calendar className="h-4 w-4 text-blue-600" />
<span className={`text-xs px-2 py-1 rounded-full bg-blue-100 text-blue-700'
}`}>
{event.type}

{event.title}
{new Date(event.start).toLocaleDateString()}

))}

{/* Quick Actions */}

<h2 className="text-lg font-semibold text-gray-900 mb-4">Quick Actions</h2>

{[
{ label: 'Pay Fees', icon: CreditCard, color: 'bg-blue-500' },
{ label: 'View Marks', icon: TrendingUp, color: 'bg-green-500' },
{ label: 'Download Books', icon: BookOpen, color: 'bg-purple-500' },
{ label: 'Request Document', icon: FileText, color: 'bg-orange-500' },
{ label: 'Apply Scholarship', icon: Award, color: 'bg-red-500' },
{ label: 'Academic Calendar', icon: Calendar, color: 'bg-indigo-500' }
].map((action, index) => {
const Icon = action.icon;
return (
<button
key={index}
className="flex flex-col items-center p-4 bg-gray-50 hover:bg-gray-100 rounded-lg transition-colors"
>
<div className={`p-3 rounded-full ${action.color} mb-2`}>
<Icon className="h-5 w-5 text-white" />

{action.label}
</button>
);
})}

Coding:

Testing is a crucial phase in the development of the Smart Campus Sphere project to ensure that all system modules function correctly, efficiently, and securely. A series of testing methodologies were applied to validate both the frontend (React + Vite) and backend (PHP–MySQL REST API) components. The following are the major tests conducted:

1. Unit Testing

Purpose:

- To verify that individual functions, modules, and components operate correctly.

Applied On:

- React Components
  Testing UI elements such as login forms, dashboards, validation functions, and API request handlers.
- PHP Backend Functions
  Verifying login authentication, database queries, request handling, and API responses.
- Database Queries (MySQL)
  Checking individual CRUD operations (Insert, Update, Select, Delete).
- Example Checks:
- Login form validation
- API JSON response format
- Session and token verification
- Dashboard rendering for each role

2. Integration Testing

- **Purpose:**
- To ensure smooth communication and data flow between combined modules.
- Integration Points Tested:
- React frontend communicating with PHP API
- PHP API interacting with MySQL
- Authentication flow (Login → Token/Session → Dashboard)
- Document upload flow (Frontend → Backend → Storage → Display)

Example Scenarios:

Student login → fetch dashboard → display academic records

Admin upload notice → students receive notification

Faculty update profile → backend saves → frontend reflects change

3. System Testing

Purpose:

- To evaluate the system as a whole and verify that all modules work together as per requirements.
- Areas Covered:
- Complete functionality of student, admin, and faculty dashboards
- Navigation and routing across pages
- Data consistency across modules
- Error handling and fallback pages

Example:

Checking whether a student can request documents and track status end-to-end

4. Acceptance Testing

Purpose:

- To validate that the final system meets user expectations and institutional requirements.
- Conducted With:
- Students
- Faculty
- Administrative staff
- Focus Areas:
- Ease of usage
- Clarity of UI
- Response time
- Accuracy of displayed academic/fee data
- This ensures the system is ready for real-world deployment.

5. Performance Testing

Purpose:

To assess system stability and responsiveness under load.

Key Aspects Tested:

- Page load time using Vite-optimized frontend
- Backend API response time under high traffic
- Database query speed for large records (users, notices, requests)
- Example:
- Stress testing login API for multiple parallel users

6. Security Testing

Purpose:

To ensure the platform is protected against common security threats.

Vulnerabilities Checked:

- SQL Injection attempts
- Session hijacking
- Unauthorized dashboard access
- File upload validation (document uploads)
- Measures Verified:
- Sanitized inputs before SQL queries
- Secure password hashing (PHP)
- Role-based access control enforcement

7. Usability Testing

Purpose:

To ensure the UI/UX is intuitive and user-friendly for all campus users.

Focus Areas:

- Mobile responsiveness
- Dashboard layout and clarity
- Accessibility for non-technical users
- Color contrast and readability
- Outcome:
- Improvements made to navigation bar, table readability, and notification visibility.

8. Regression Testing

Purpose:

To confirm that new updates do not break existing features.

Trigger Events:

- Adding new pages or components
- Updating API endpoints
- Modifying database schema
- Example:
- After implementing profile photo upload, testing login, dashboard, and request modules again.

User Manual

The following User Manual provides a clear guide for students, faculty, and administrators on how to use the OrbitCampus platform. It explains how to navigate the system, access features, and perform essential actions. This manual ensures that all users can operate the system efficiently and without confusion.

1. Introduction to OrbitCampus

OrbitCampus is a web-based campus management platform designed to streamline academic and administrative processes within an educational institution. The system provides separate dashboards for Students, Faculty, and Administrators, ensuring role-based access and security.

This user manual explains how to log in, navigate the system, perform tasks, and manage data effectively.

2. System Requirements:

Hardware Requirements

- Minimum **4 GB RAM** (8 GB recommended)
- Processor: **Intel i3/i5/i7 or AMD equivalent**
- Stable **internet connection**

Software Requirements

- **Operating System:** Windows / Linux / macOS

- **Browser:** Google Chrome, Microsoft Edge, or Firefox
- **XAMPP** installed (for local hosting)
- **Node.js + npm** (for running the React frontend)
- **Vite** development server (bundled with the frontend)

3. Getting Started

3.1 Accessing OrbitCampus

- Open your browser.
- Enter the official platform URL (provided by your institution).
- You will be directed to the Login Page.

4. Login & Authentication:

Steps to Log In

- Enter your **Email/Username**.
- Enter your **Password**.
- Click **Login**.

If credentials are correct, the user is taken to their role-based dashboard.

Forgot Password

If available:

- Click **Forgot Password**, enter your registered email, and follow instructions.

5. Role-Based Dashboards:

- After login, each user sees a dashboard tailored to their role.

6. Student Module – User Guide

6.1 Home Dashboard

Shows:

- Recent announcements
- Pending requests
- Profile summary

6.2 Profile Management:

Students can:

- Update basic profile details
- Upload/change profile photo

6.3 Academic Information:

- Includes:
- Academic records
- Course details
- Attendance (if enabled)
- Fee details

6.4 Document Requests:

Students can:

- Request bonafide certificates
- Request transcripts / leaves / academic documents
- Track request status
- *Pending*
- *In Process*
- *Approved*
- *Rejected*

6.5 Notifications:

- Displays all official notices posted by faculty/admin.

7. Faculty Module – User Guide

7.1 Dashboard:

Displays:

- Total students
- Pending document approvals
- Recent communications

7.2 Document Verification:

Faculty can:

- View student document requests
- Approve, Reject, or Mark as Processing
- Add comments or notes

7.3 Profile Section:

Faculty can manage:

- Personal details
- Contact information

7.4 Announcements:

Faculty can:

- Create announcements for students
- Edit or remove old notices

---

8. Admin Module – User Guide

Admins have full control over the system.

8.1 User Management:

Admin can:

- Add new students/faculty users
- Edit user details
- Reset passwords
- Remove inactive users

8.2 Document Management:

Admins can:

- Access all document requests
- Provide final approval
- Upload official documents when needed

8.3 Academic & Fee Data:

Admins can manage:

- Fee details
- Academic records
- Semester updates
- Department data

8.4 Notification System:

Admins post campus-wide notices and updates.

8.5 System Settings:

Admins can configure:

- Backend settings
- Modules visibility
- Security settings

9. Navigating the Interface:

Sidebar Menu

Contains:

- Dashboard
- Profile
- Academics
- Documents
- Notifications
- Settings (Admin)

Top Navigation Bar

Shows:

- User name
- Logout button
- Quick notification icon

10. How to Upload Documents:

- Go to **Document Upload** section.

- Click **Select File**.
- Choose a PDF/JPG/PNG file.
- Click **Upload**.
- The file is stored in the backend (PHP–MySQL).

11. How to Request a Document:

- Navigate to **Request Document**.
- Select the document type from the dropdown.
- Enter purpose (optional).
- Click **Submit Request**.
- Track progress in the **Request Status** section.

12. Logging Out:

- To safely log out:
- Click the **Logout** button in the top right corner.
- You will be redirected to the login page.

13. Troubleshooting Guide:

Login Issues

- Check email/password.
- Make sure Caps Lock is off.
- Contact Admin for password reset.

File Upload Error

- Ensure the file is within allowed size limit.
- Supported formats: PDF, JPG, PNG.

Page Not Loading

- Refresh the page.
- Clear browser cache.
- Try a different browser.

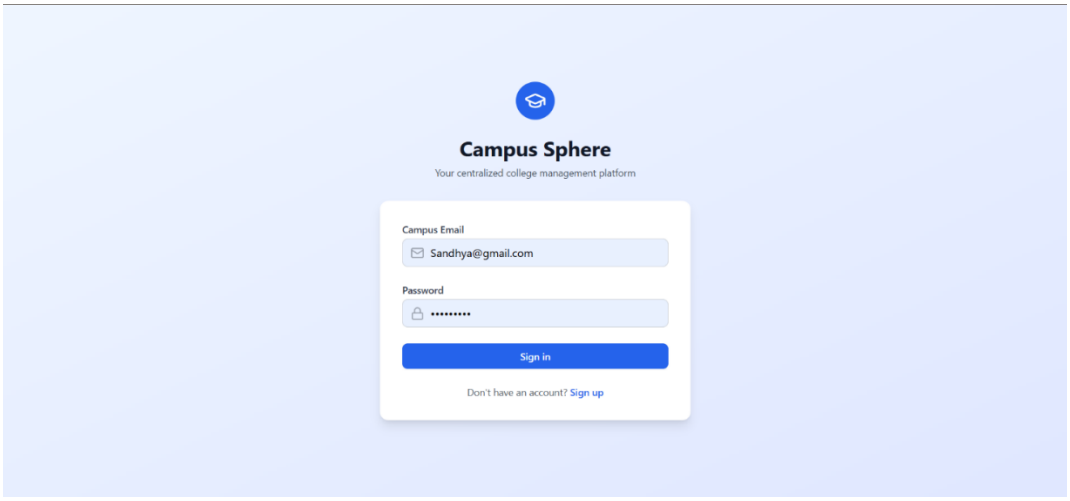
14. Safety & Security Tips:

- **Do not share your password with anyone.**

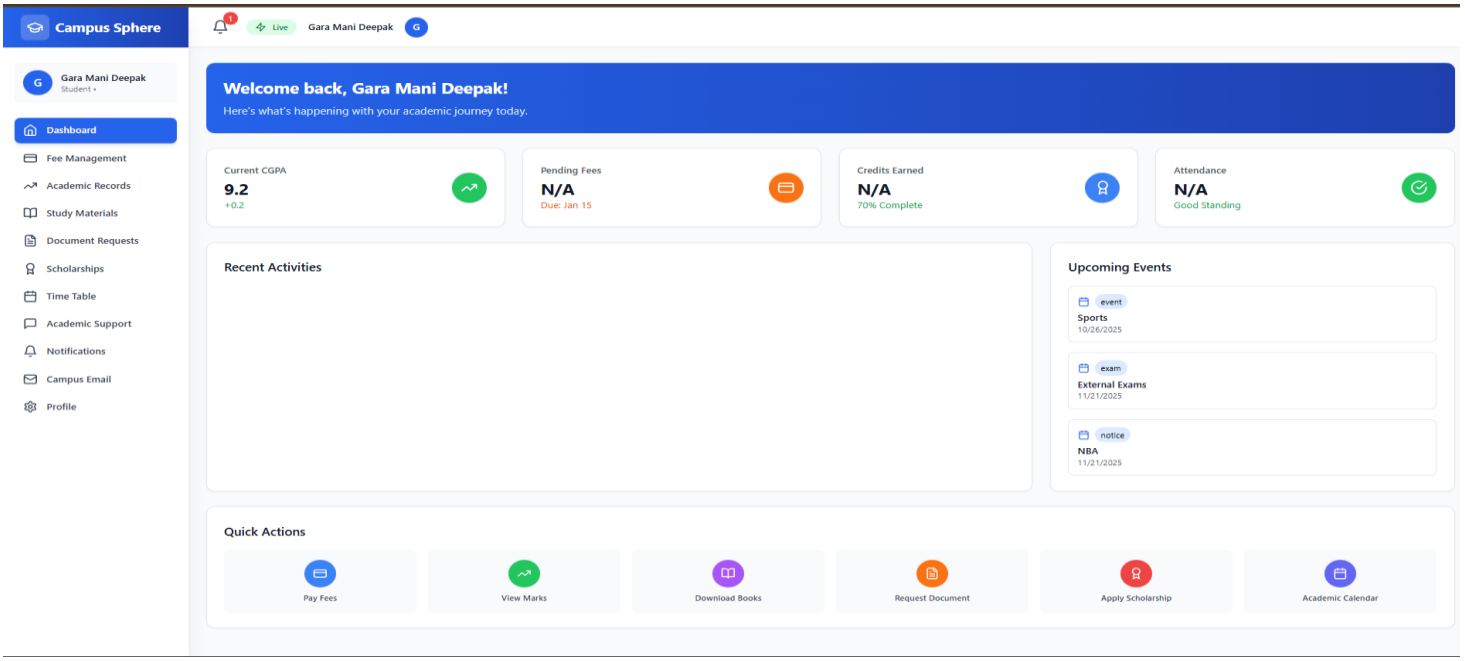
- **Always logout after use.**
- **Keep personal details updated.**
- **Avoid accessing the system on public computers.**

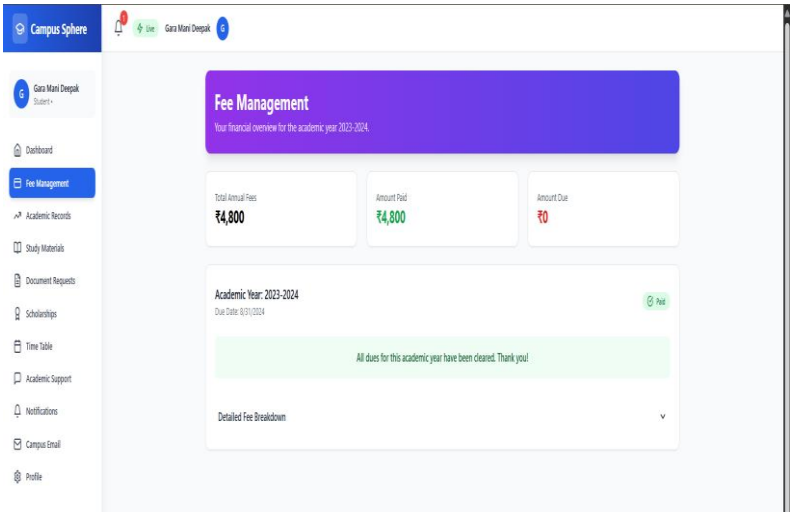
OUTPUT SCREEN’S

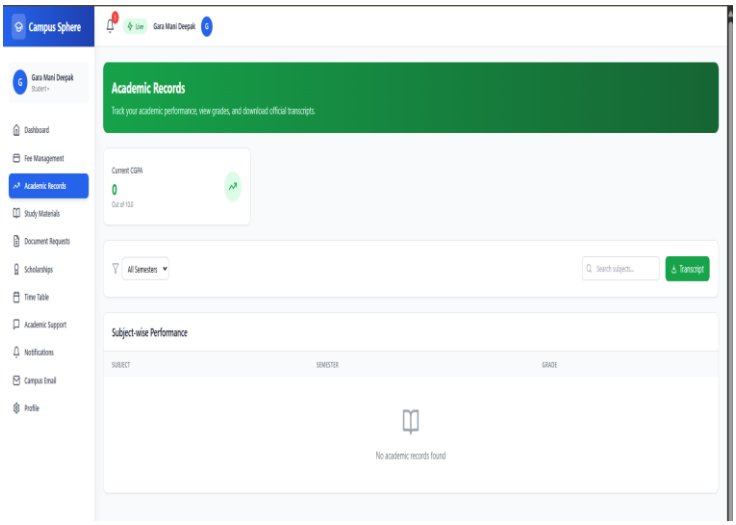
LOGIN PAGE

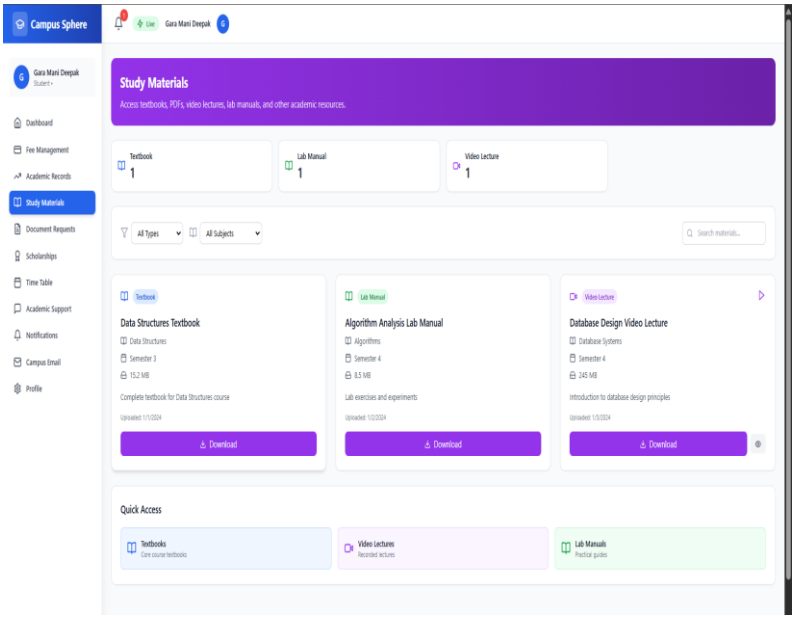


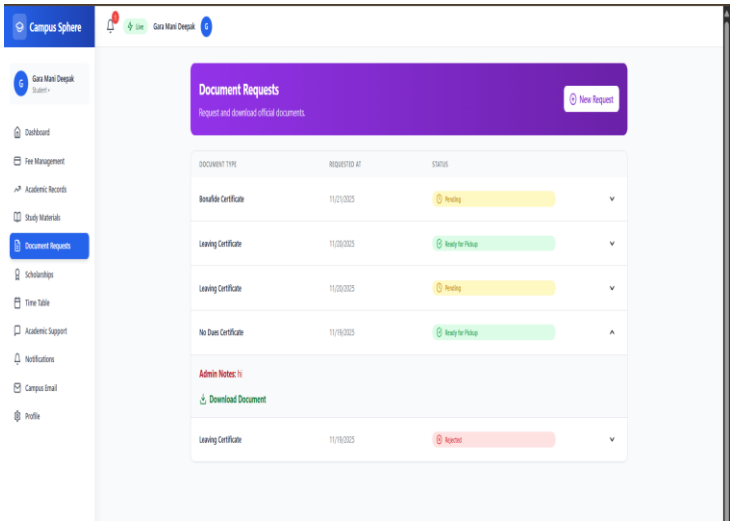
STUDENT SCREEN’S

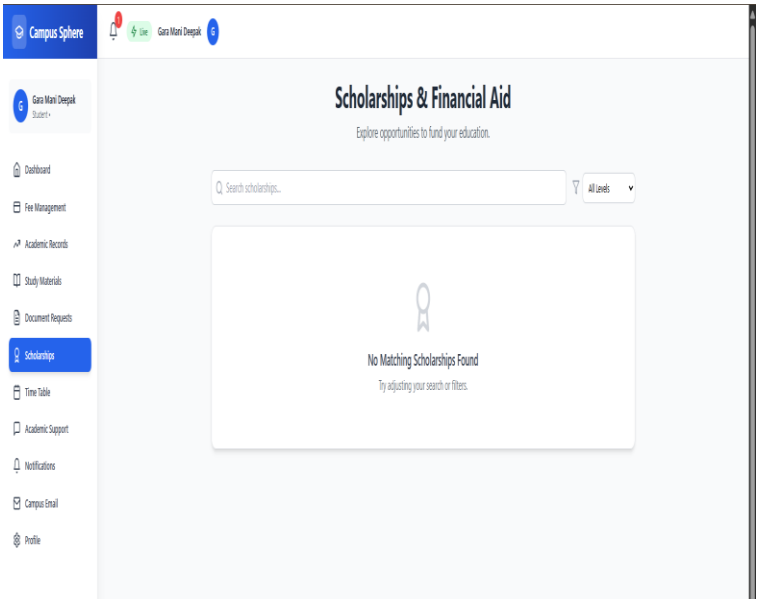


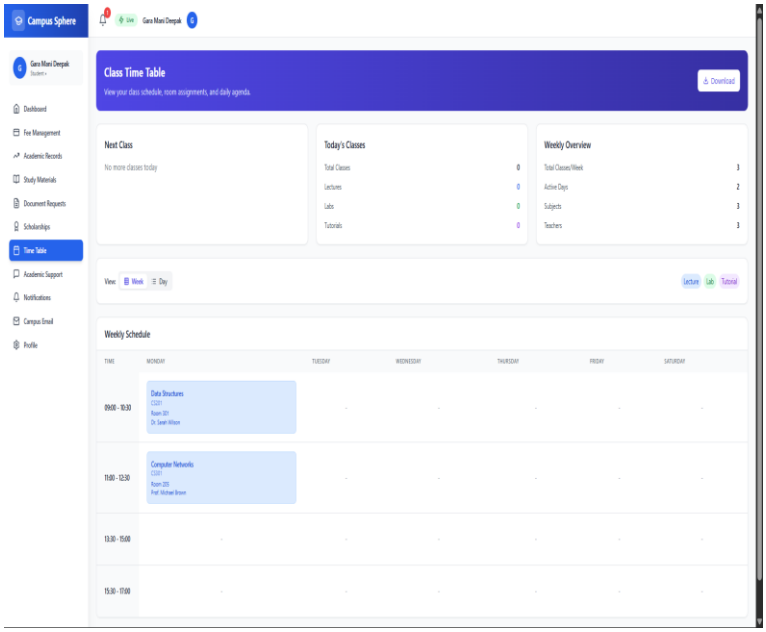

Campus Sphere
Live
Gara Mani Deepak
Fee Management
Your financial overview for the academic year 2023-2024.
Total Annual Fees
₹4,800
Amount Paid
₹4,800
Amount Due
₹0
Academic Year: 2023-2024
Paid
All dues for this academic year have been cleared. Thank you!
Detailed Fee Breakdown
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile

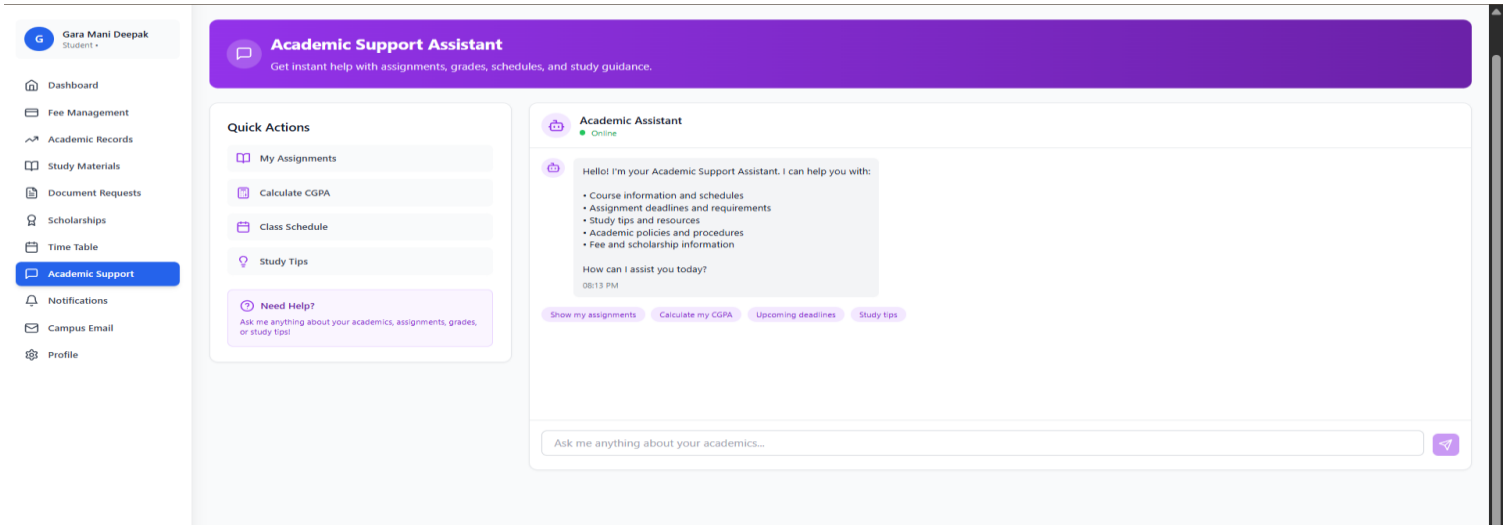

Campus Sphere
Live
Gara Mani Deepak
Academic Records
Track your academic performance, view grades, and download official transcripts.
Current CGPA
0
Out of 10.0
All Semesters
Search subjects...
Transcript
Subject-wise Performance
SUBJECT
SEMESTER
GRADE
No academic records found
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile

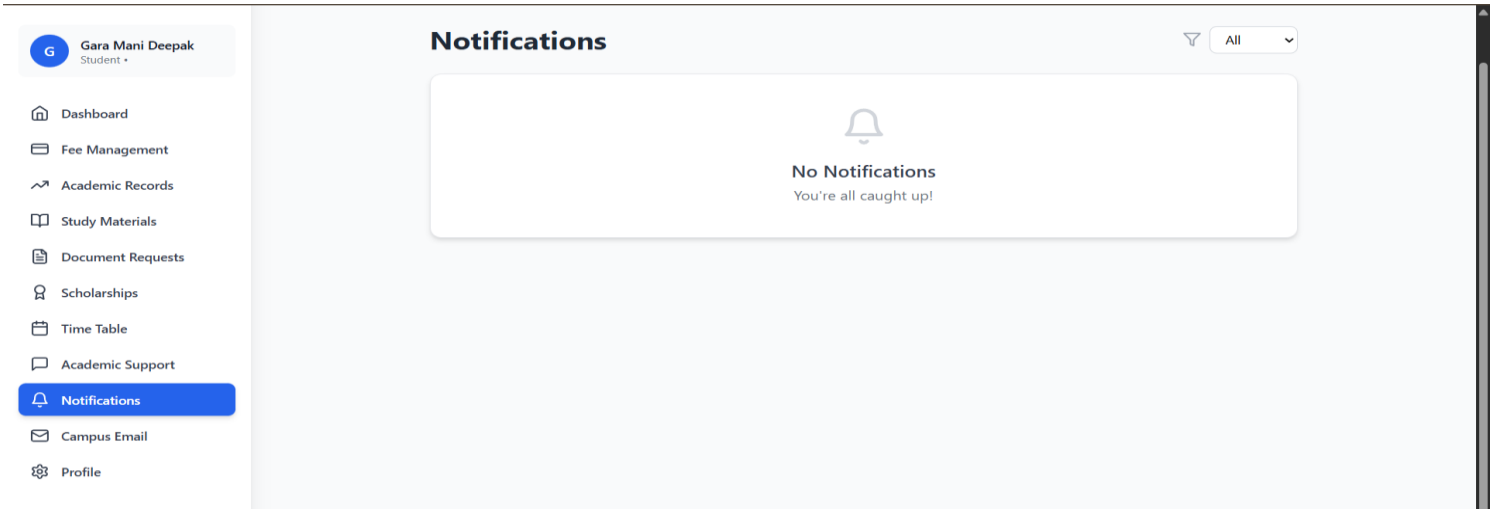

Campus Sphere
Live
Gara Mani Deepak
Study Materials
Access textbooks, PDFs, video lectures, lab manuals, and other academic resources.
Textbook
1
Lab Manual
1
Video Lecture
1
All Types
All Subjects
Search materials...
Data Structures Textbook
Data Structures
Semester 3
15.2 MB
Complete textbook for Data Structures course
Uploaded: 1/1/2024
Download
Algorithm Analysis Lab Manual
Algorithms
Semester 4
8.5 MB
Lab exercises and experiments
Uploaded: 1/2/2024
Download
Database Design Video Lecture
Database Systems
Semester 4
245 MB
Introduction to database design principles
Uploaded: 1/3/2024
Download
Quick Access
Textbooks
Core course textbooks
Video Lectures
Recorded lectures
Lab Manuals
Practical guides
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile

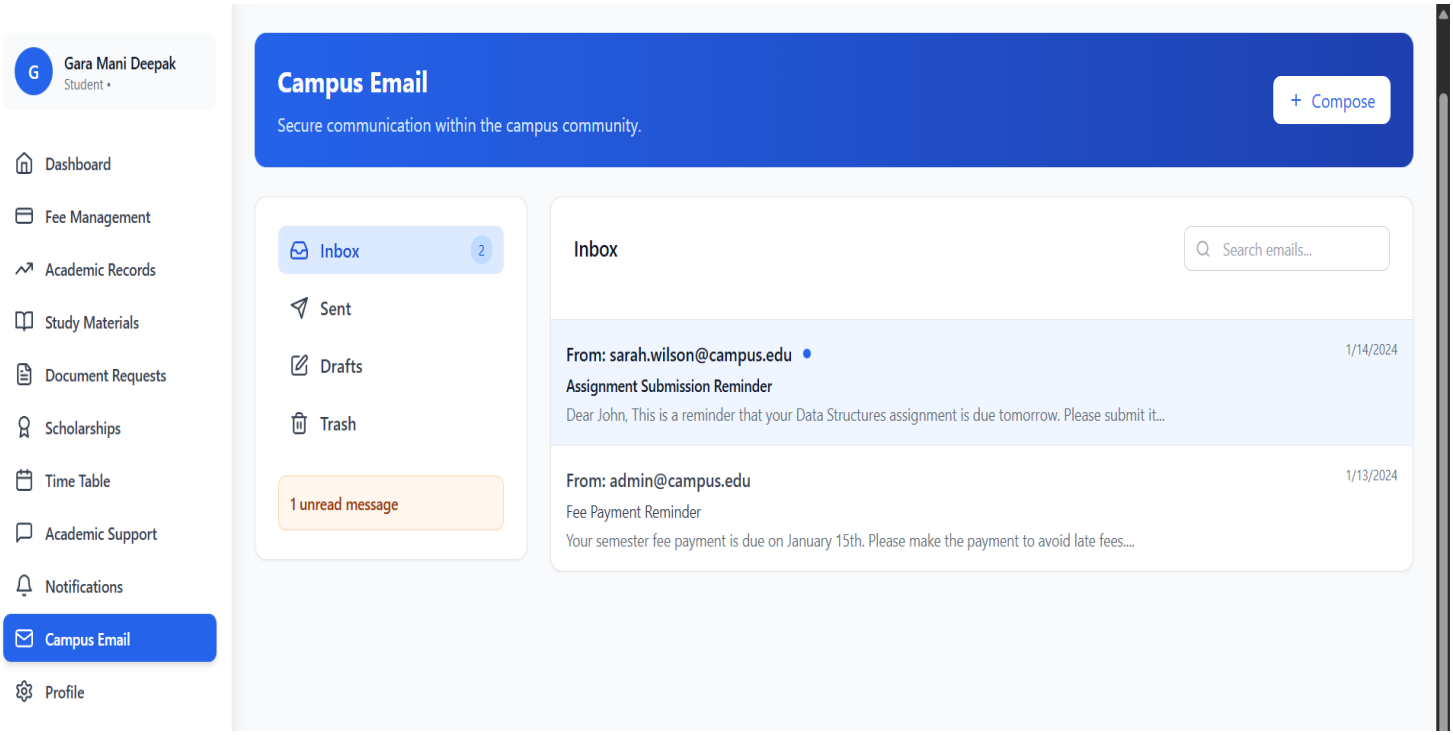

Campus Sphere
Live
Gara Mani Deepak
Document Requests
Request and download official documents.
New Request
DOCUMENT TYPE
REQUESTED AT
STATUS
Bonafide Certificate
11/21/2025
Pending
Leaving Certificate
11/20/2025
Ready for Pickup
Leaving Certificate
11/20/2025
Pending
No Dues Certificate
11/19/2025
Ready for Pickup
Admin Notes: hi
Download Document
Leaving Certificate
11/19/2025
Rejected
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile

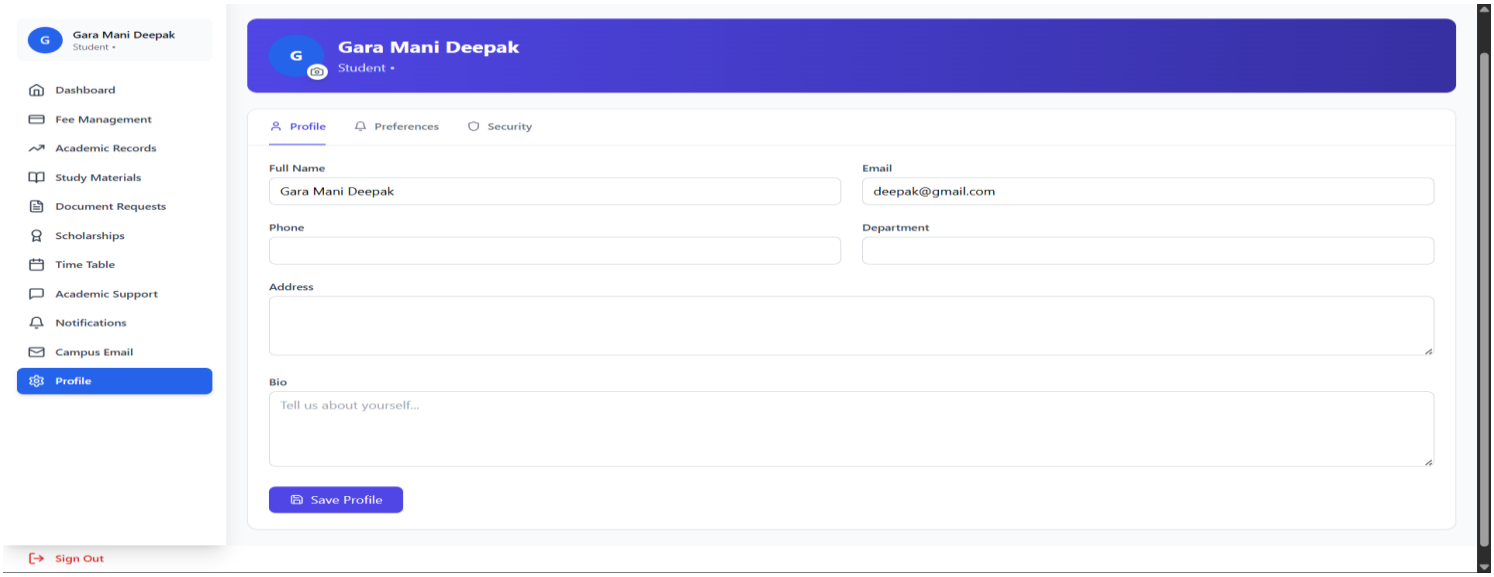

Campus Sphere
Live
Gara Mani Deepak
Scholarships & Financial Aid
Explore opportunities to fund your education.
Search scholarships...
All Levels
No Matching Scholarships Found
Try adjusting your search or filters.
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile


Campus Sphere
Live
Gara Mani Deepak
Class Time Table
View your class schedule, room assignments, and daily agenda.
Download
Next Class
No more classes today
Today's Classes
Total Classes
0
Lectures
0
Labs
0
Tutorials
0
Weekly Overview
Total Classes/Week
3
Active Days
2
Subjects
3
Teachers
3
View:
Week
Day
Lecture
Lab
Tutorial
Weekly Schedule
TIME
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
09:00 - 10:30
Data Structures
CS201
Room 101
Dr. Sarah Wilson
11:00 - 12:30
Computer Networks
CS401
Room 205
Prof. Michael Brown
13:30 - 15:00
15:30 - 17:00
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile

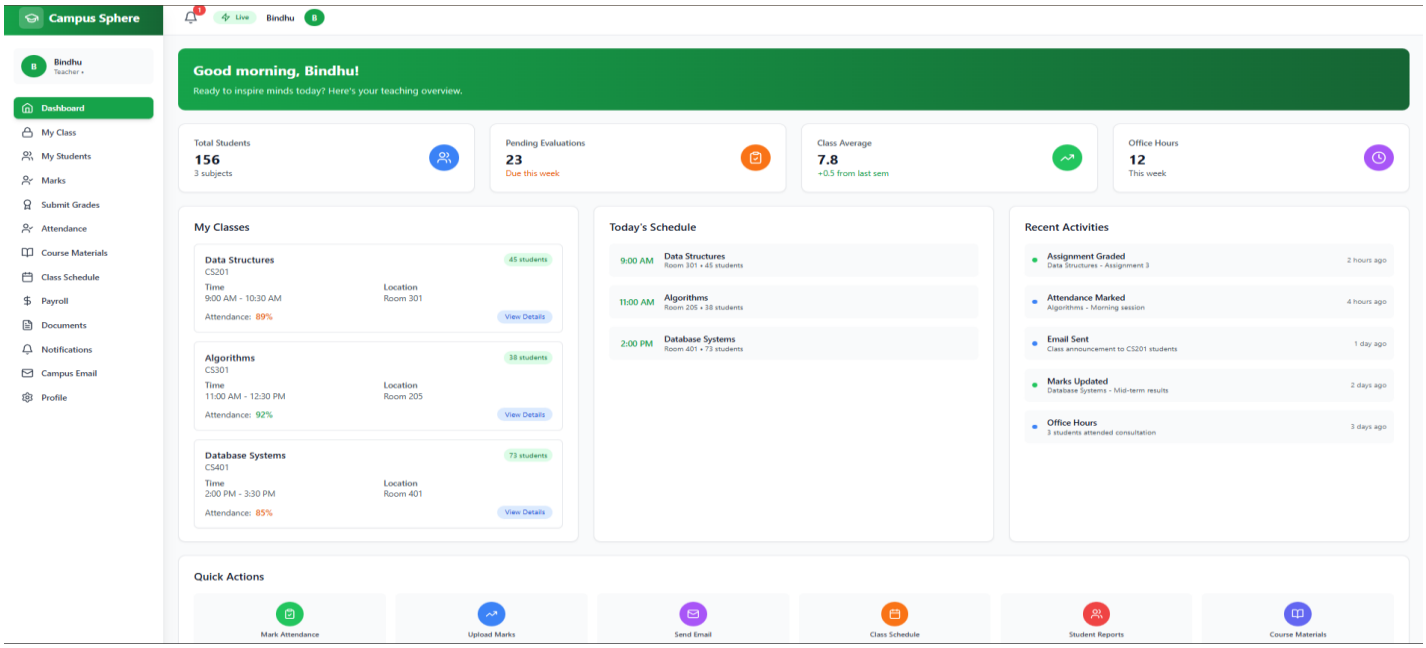

Gara Mani Deepak
Student •
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile
Academic Support Assistant
Get instant help with assignments, grades, schedules, and study guidance.
Quick Actions
My Assignments
Calculate CGPA
Class Schedule
Study Tips
Need Help?
Ask me anything about your academics, assignments, grades, or study tips!
Academic Assistant
Online
Hello! I'm your Academic Support Assistant. I can help you with:
• Course information and schedules
• Assignment deadlines and requirements
• Study tips and resources
• Academic policies and procedures
• Fee and scholarship information
How can I assist you today?
08:13 PM
Show my assignments
Calculate my CGPA
Upcoming deadlines
Study tips
Ask me anything about your academics...

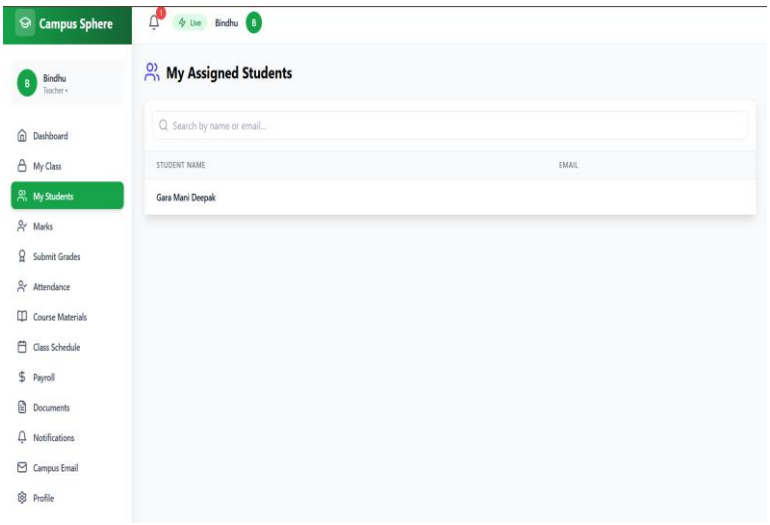

Gara Mani Deepak
Student •
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile
Notifications
All
No Notifications
You're all caught up!

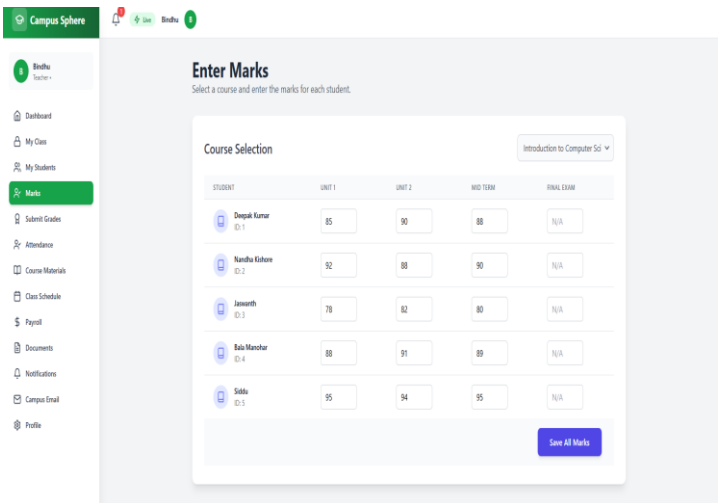

Gara Mani Deepak
Student •
Dashboard
Fee Management
Academic Records
Study Materials
Document Requests
Scholarships
Time Table
Academic Support
Notifications
Campus Email
Profile
Campus Email
Secure communication within the campus community.
+ Compose
Inbox 2
Sent
Drafts
Trash
1 unread message
Inbox
Search emails...
From: sarah.wilson@campus.edu
1/14/2024
Assignment Submission Reminder
Dear John, This is a reminder that your Data Structures assignment is due tomorrow. Please submit it...
From: admin@campus.edu
1/13/2024
Fee Payment Reminder
Your semester fee payment is due on January 15th. Please make the payment to avoid late fees....

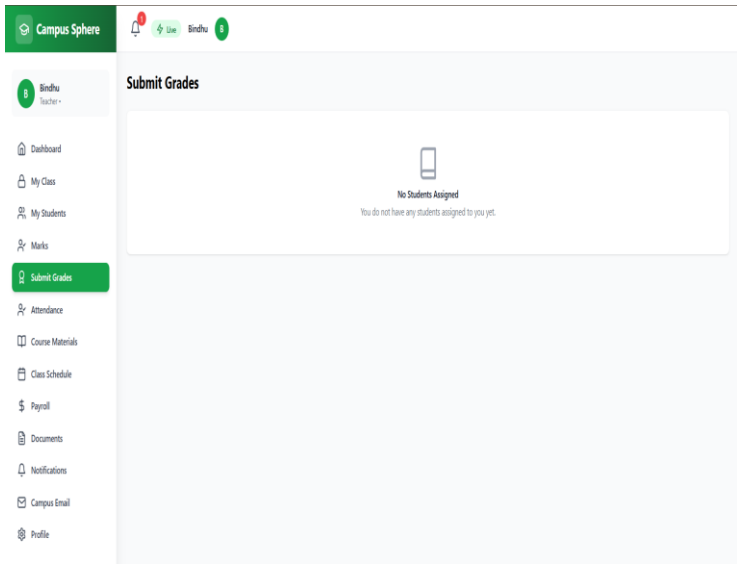


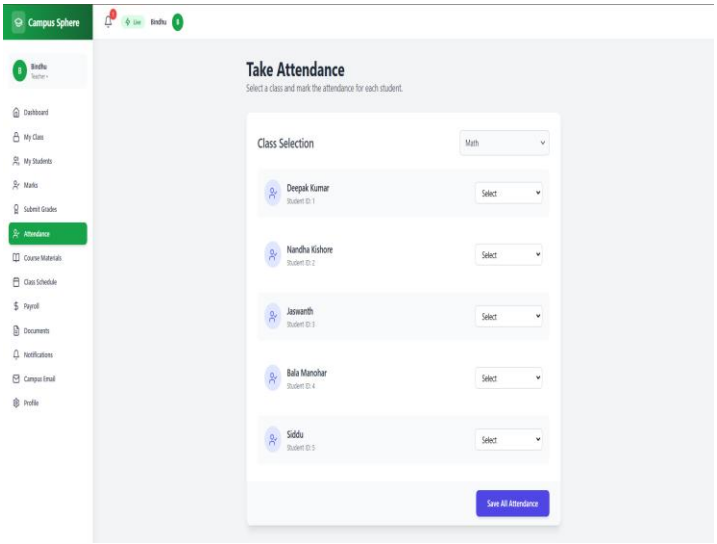
Teacher screen's

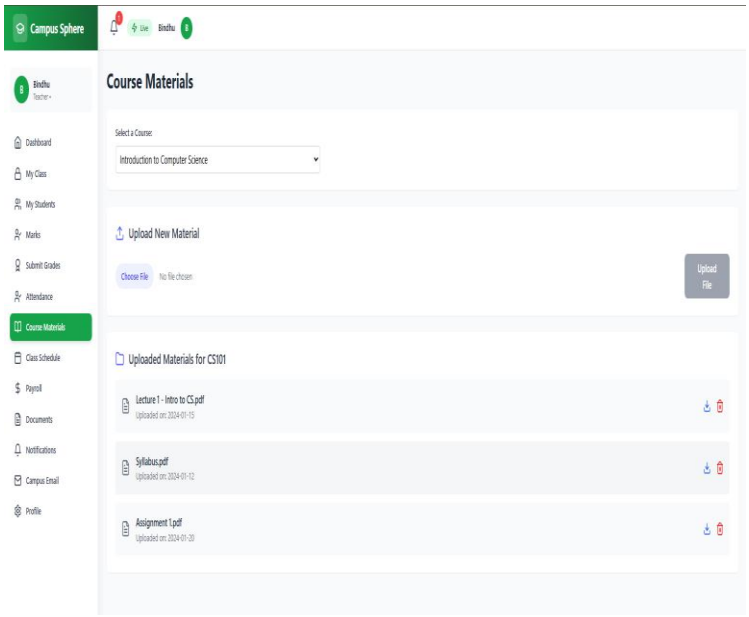


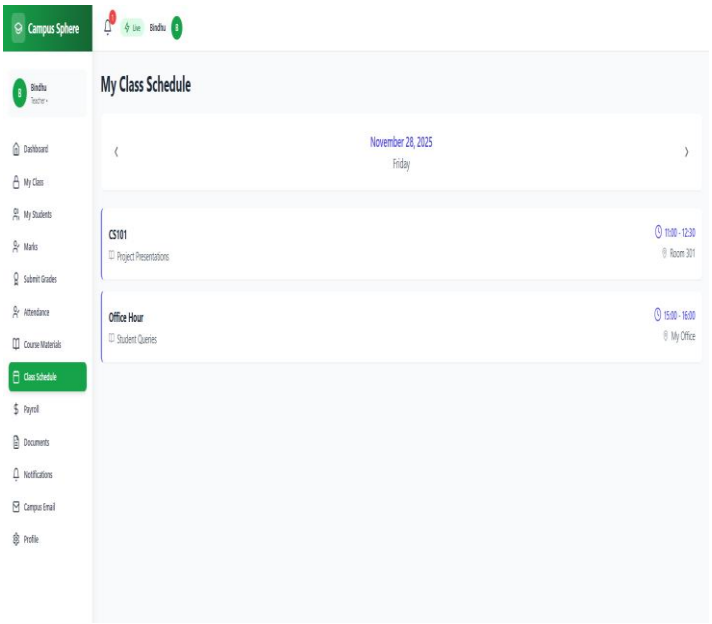


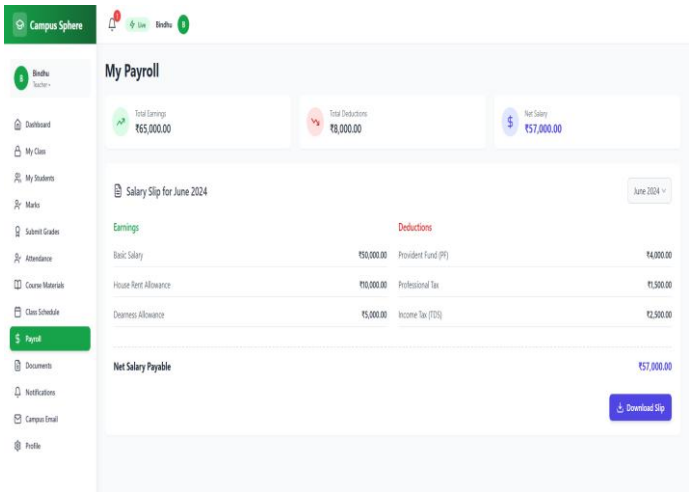


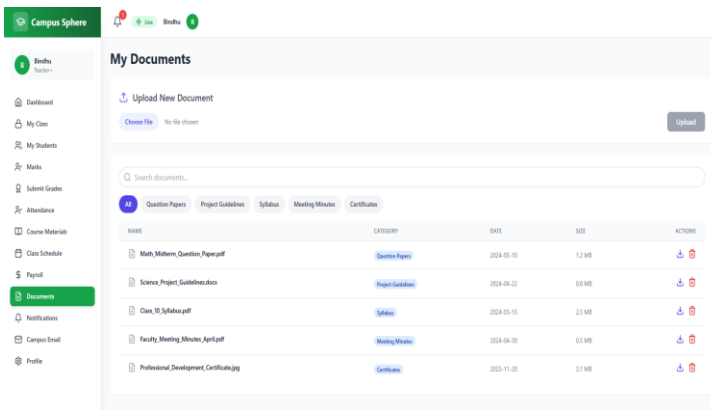

Campus Sphere
Live
Bindhu
Bindhu
Teacher
Dashboard
My Class
My Students
Marks
Submit Grades
Attendance
Course Materials
Class Schedule
Payroll
Documents
Notifications
Campus Email
Profile
Submit Grades
No Students Assigned
You do not have any students assigned to you yet.


Campus Sphere
Live
Bindhu
Bindhu
Teacher
Dashboard
My Class
My Students
Marks
Submit Grades
Attendance
Course Materials
Class Schedule
Payroll
Documents
Notifications
Campus Email
Profile
Take Attendance
Select a class and mark the attendance for each student.
Class Selection
Math
Deepak Kumar
Student ID: 1
Select
Nandha Kishore
Student ID: 2
Select
Jaswanth
Student ID: 3
Select
Bala Manohar
Student ID: 4
Select
Siddu
Student ID: 5
Select
Save All Attendance

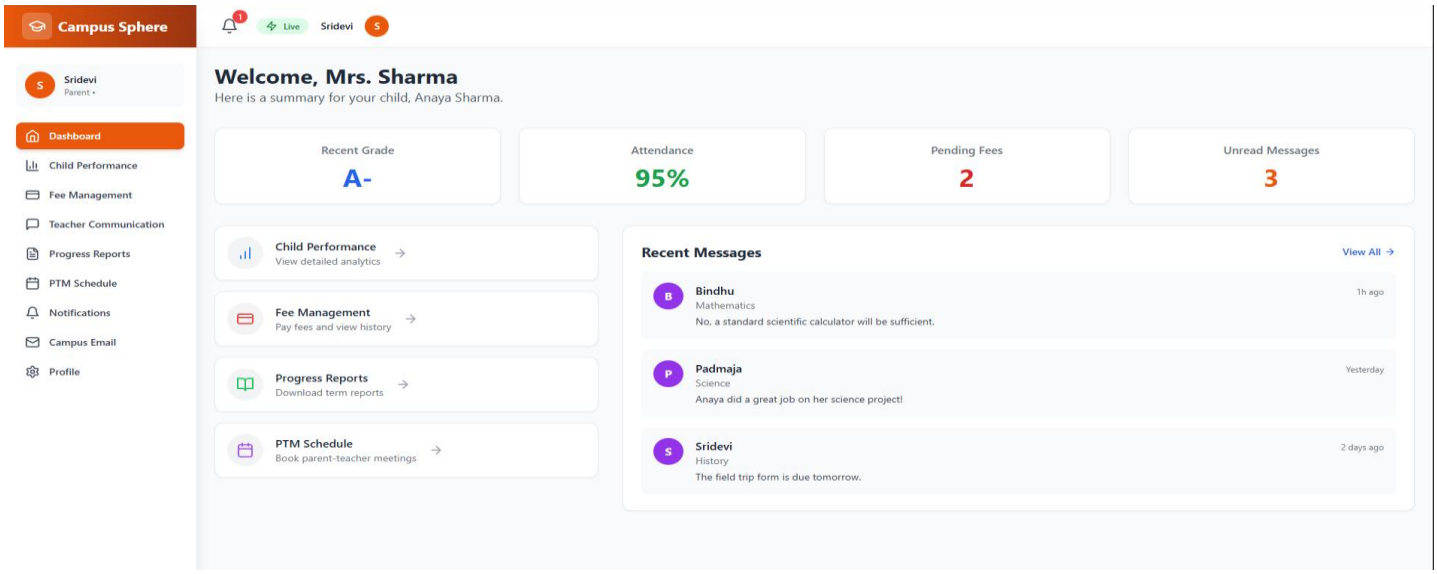

Campus Sphere
Live
Bindhu
Bindhu
Teacher
Dashboard
My Class
My Students
Marks
Submit Grades
Attendance
Course Materials
Class Schedule
Payroll
Documents
Notifications
Campus Email
Profile
Course Materials
Select a Course:
Introduction to Computer Science
Upload New Material
Choose File
No file chosen
Upload File
Uploaded Materials for CS101
Lecture 1 - Intro to CS.pdf
Uploaded on: 2024-01-15
Syllabus.pdf
Uploaded on: 2024-01-12
Assignment 1.pdf
Uploaded on: 2024-01-20

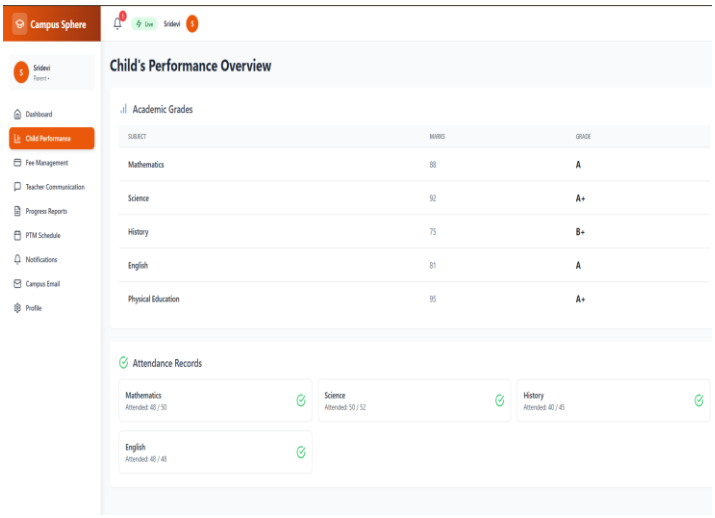

Campus Sphere
Live
Bindhu
Bindhu
Teacher
Dashboard
My Class
My Students
Marks
Submit Grades
Attendance
Course Materials
Class Schedule
Payroll
Documents
Notifications
Campus Email
Profile
My Class Schedule
November 28, 2025
Friday
CS101
Project Presentations
11:00 - 12:30
Room 301
Office Hour
Student Queries
15:00 - 16:00
My Office

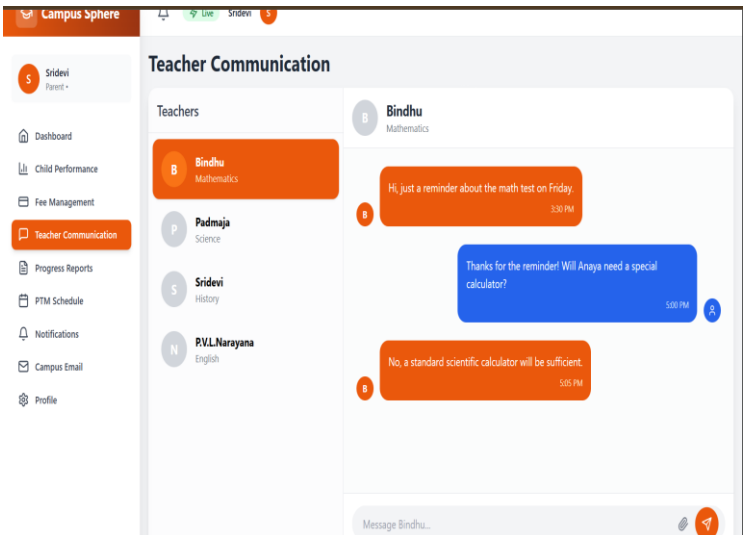

Campus Sphere
Live
Bindhu
Bindhu
Teacher
Dashboard
My Class
My Students
Marks
Submit Grades
Attendance
Course Materials
Class Schedule
Payroll
Documents
Notifications
Campus Email
Profile
My Payroll
Total Earnings
₹65,000.00
Total Deductions
₹8,000.00
Net Salary
₹57,000.00
Salary Slip for June 2024
June 2024
Earnings
Deductions
Basic Salary
₹50,000.00
Provident Fund (PF)
₹4,000.00
House Rent Allowance
₹10,000.00
Professional Tax
₹1,500.00
Dearness Allowance
₹5,000.00
Income Tax (TDS)
₹2,500.00
Net Salary Payable
₹57,000.00
Download Slip

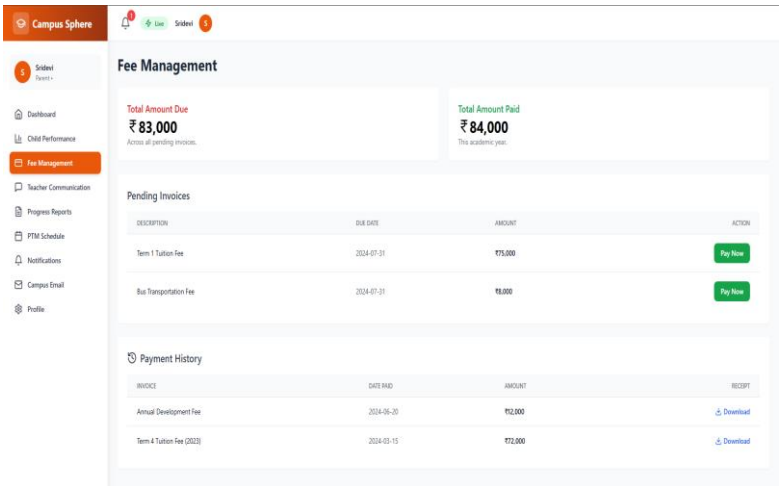

Campus Sphere
Live
Bindhu
Bindhu
Teacher
Dashboard
My Class
My Students
Marks
Submit Grades
Attendance
Course Materials
Class Schedule
Payroll
Documents
Notifications
Campus Email
Profile
My Documents
Upload New Document
Choose File
No file chosen
Upload
Search documents...
All
Question Papers
Project Guidelines
Syllabus
Meeting Minutes
Certificates
NAME
CATEGORY
DATE
SIZE
ACTIONS
Math_Midterm_Question_Paper.pdf
Question Papers
2024-05-10
1.2 MB
Science_Project_Guidelines.docx
Project Guidelines
2024-04-22
0.8 MB
Class_10_Syllabus.pdf
Syllabus
2024-03-15
2.5 MB
Faculty_Meeting_Minutes_April.pdf
Meeting Minutes
2024-04-30
0.5 MB
Professional_Development_Certificate.jpg
Certificates
2023-11-20
3.1 MB

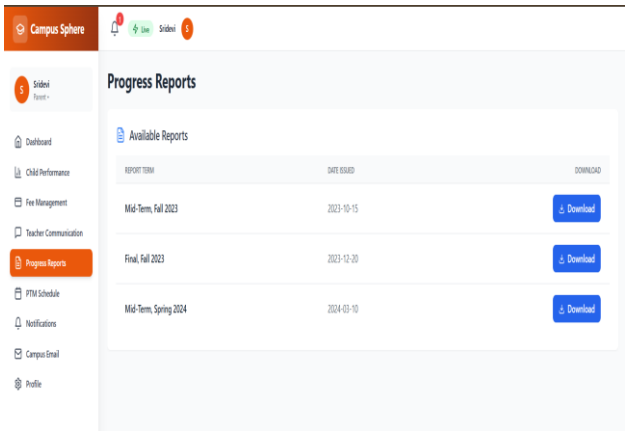
Parent screen’s

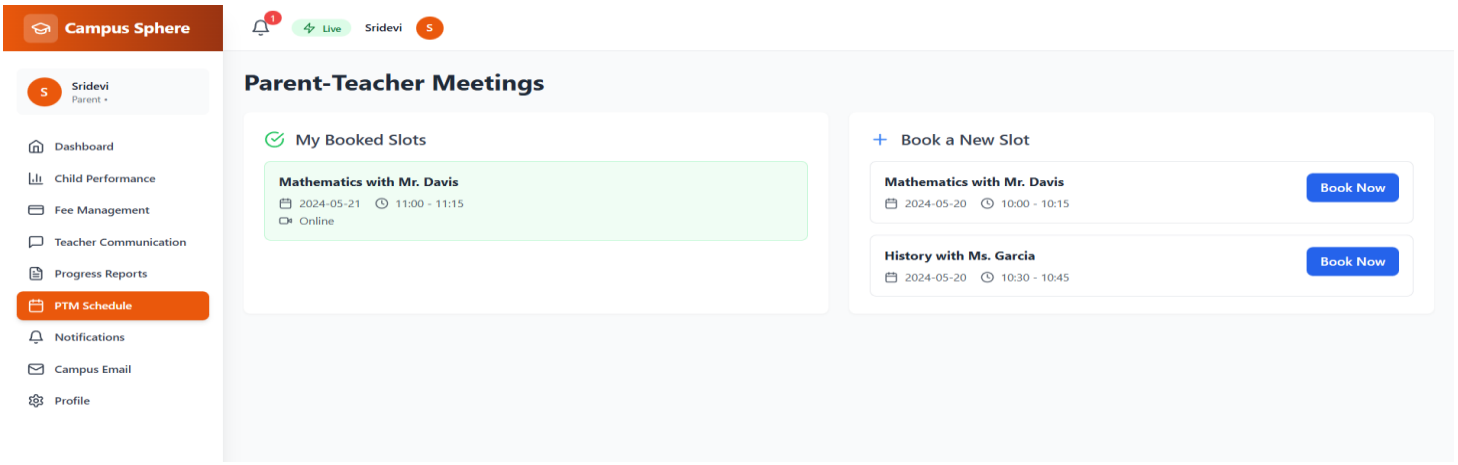




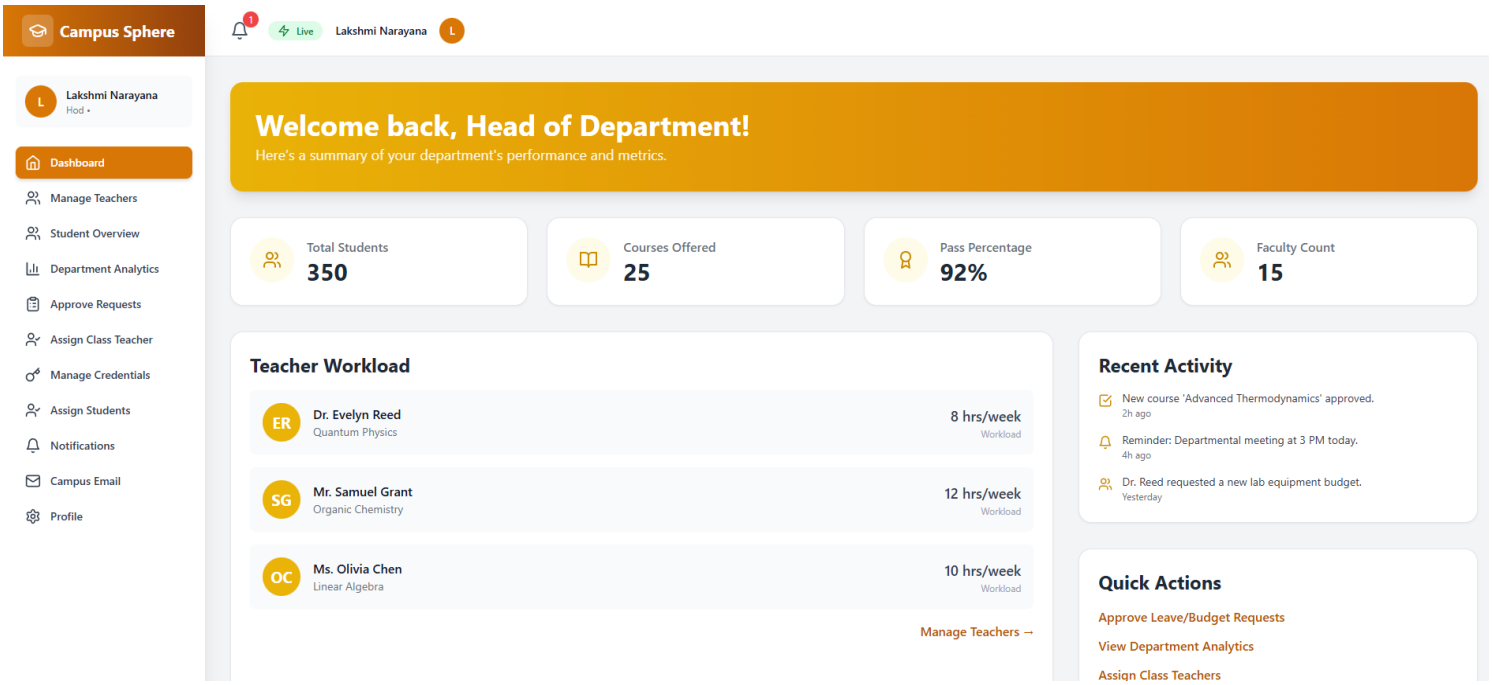


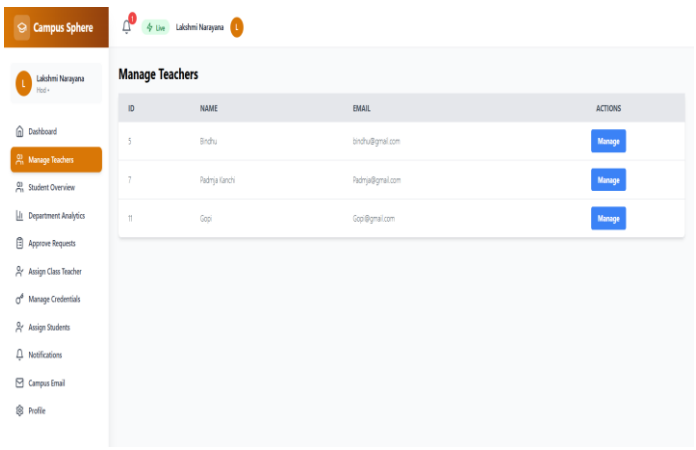


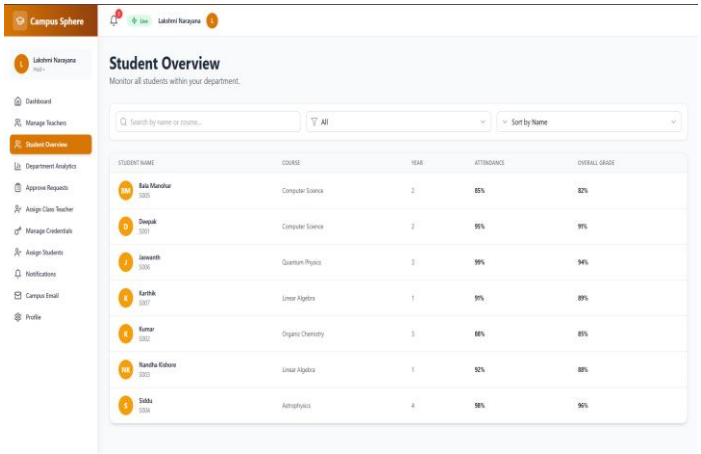


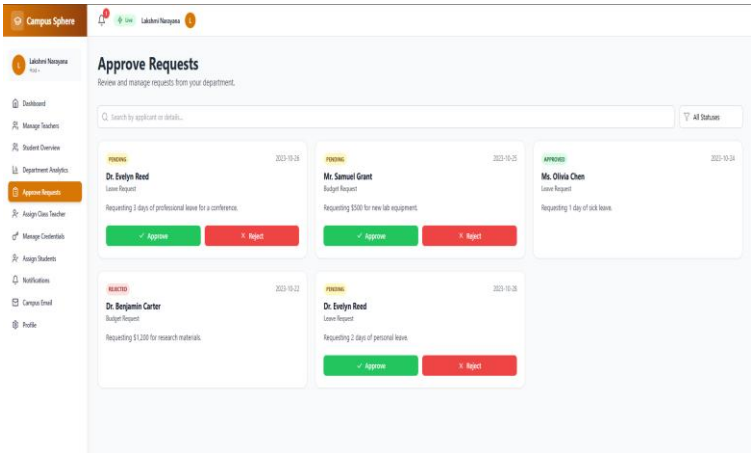

Campus Sphere
Live
Sridevi
S
Sridevi
Parent
Dashboard
Child Performance
Fee Management
Teacher Communication
Progress Reports
PTM Schedule
Notifications
Campus Email
Profile
Parent-Teacher Meetings
My Booked Slots
Mathematics with Mr. Davis
2024-05-21
11:00 - 11:15
Online
Book a New Slot
Mathematics with Mr. Davis
2024-05-20
10:00 - 10:15
Book Now
History with Ms. Garcia
2024-05-20
10:30 - 10:45
Book Now

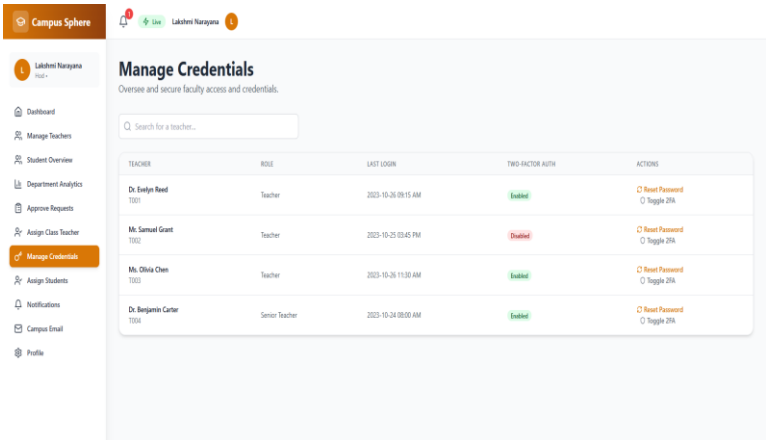
HOD Screen's

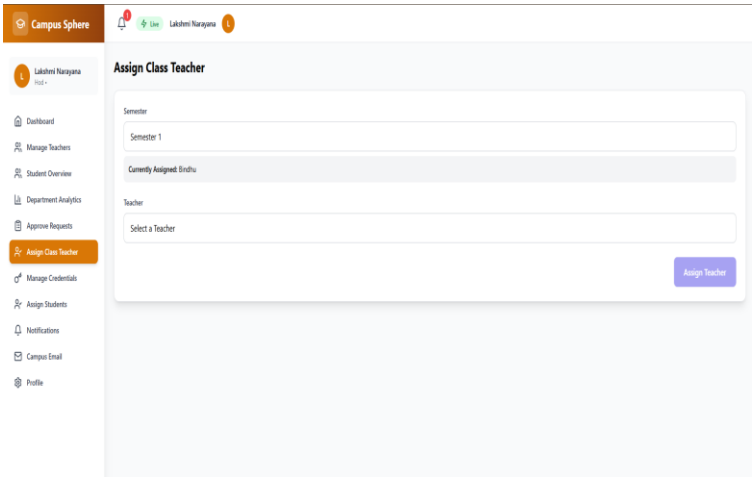

Campus Sphere
Live
Lakshmi Narayana
L
Lakshmi Narayana
Hod
Dashboard
Manage Teachers
Student Overview
Department Analytics
Approve Requests
Assign Class Teacher
Manage Credentials
Assign Students
Notifications
Campus Email
Profile
Welcome back, Head of Department!
Here's a summary of your department's performance and metrics.
Total Students
350
Courses Offered
25
Pass Percentage
92%
Faculty Count
15
Teacher Workload
ER
Dr. Evelyn Reed
Quantum Physics
8 hrs/week
Workload
SG
Mr. Samuel Grant
Organic Chemistry
12 hrs/week
Workload
OC
Ms. Olivia Chen
Linear Algebra
10 hrs/week
Workload
Manage Teachers →
Recent Activity
New course 'Advanced Thermodynamics' approved.
2h ago
Reminder: Departmental meeting at 3 PM today.
4h ago
Dr. Reed requested a new lab equipment budget.
Yesterday
Quick Actions
Approve Leave/Budget Requests
View Department Analytics
Assign Class Teachers

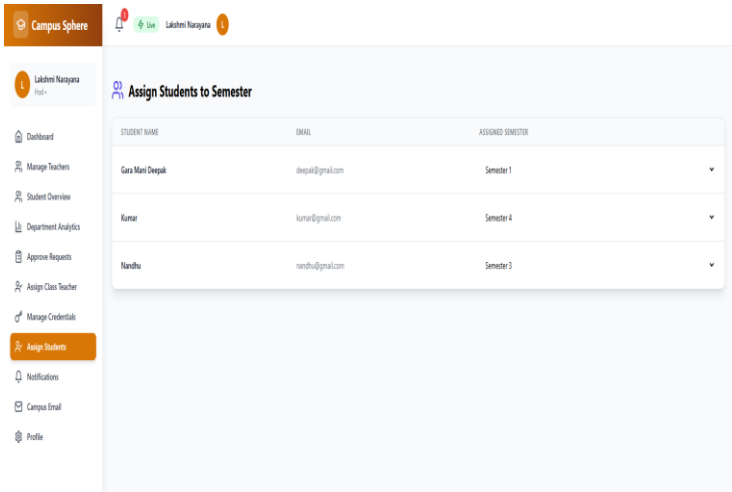

Campus Sphere
Manage Teachers
ID
NAME
EMAIL
ACTIONS
Manage

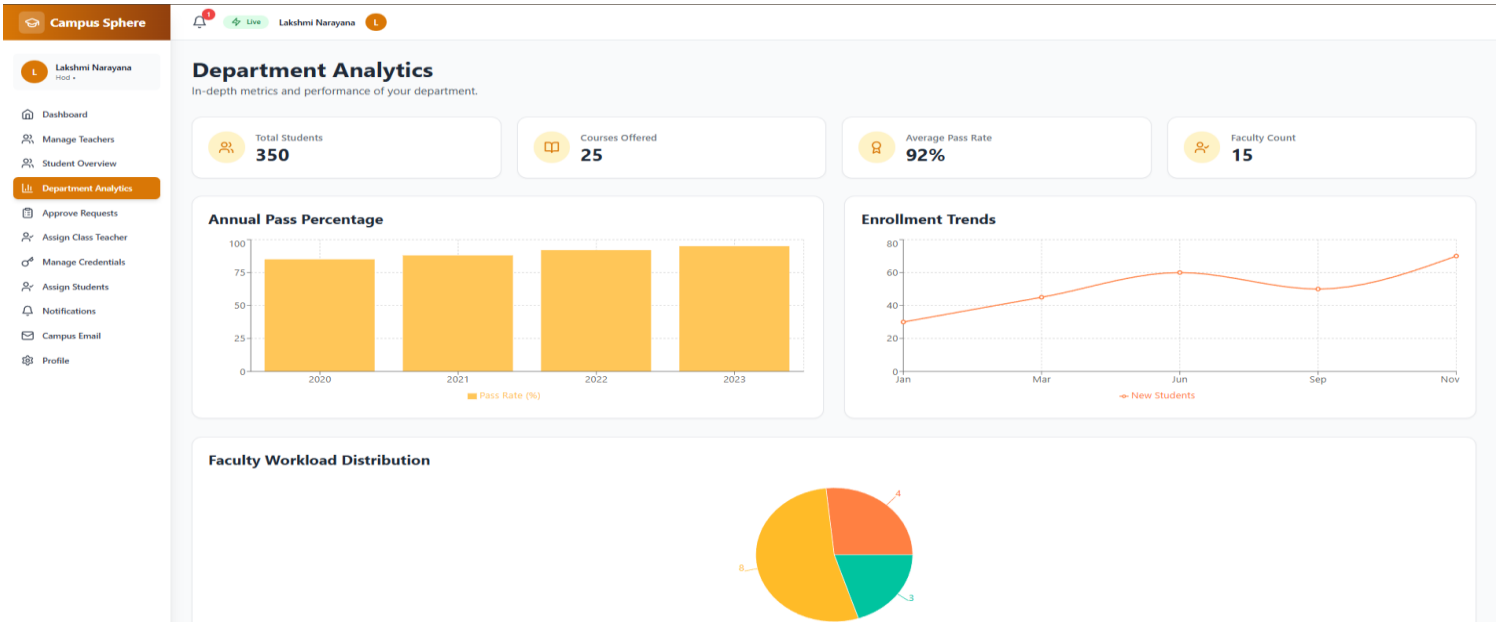

Campus Sphere
Student Overview
Monitor all students within your department.


Campus Sphere
Lakshmi Narayana
Approve Requests
Review and manage requests from your department.
Dr. Evelyn Reed
Mr. Samuel Grant
Ms. Olivia Chen
Dr. Benjamin Carter
Approve
Reject

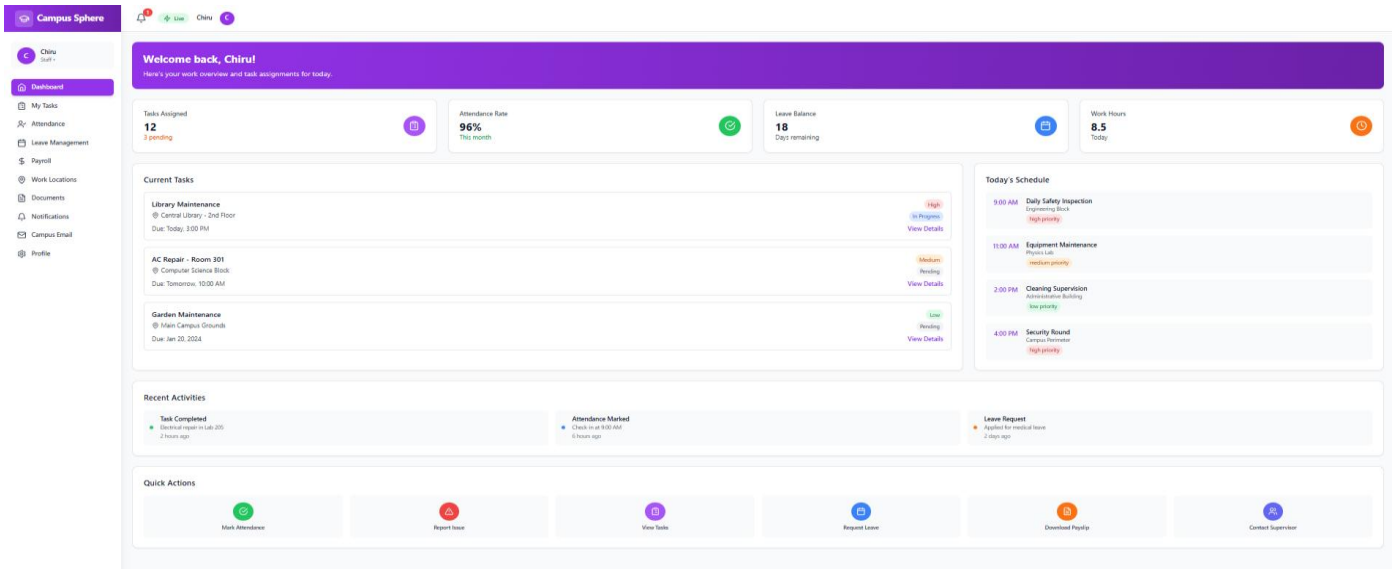

Campus Sphere
Lakshmi Narayana
Manage Credentials
Oversee and secure faculty access and credentials.
TEACHER
ROLE
LAST LOGIN
TWO-FACTOR AUTH
ACTIONS
Dr. Evelyn Reed
Mr. Samuel Grant
Ms. Olivia Chen
Dr. Benjamin Carter

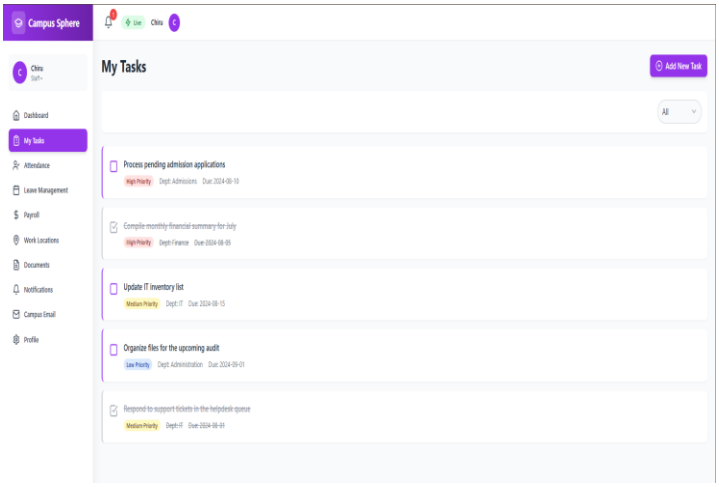

Campus Sphere
Lakshmi Narayana
Assign Class Teacher
Semester
Semester 1
Currently Assigned: Bindhu
Teacher
Select a Teacher
Assign Teacher

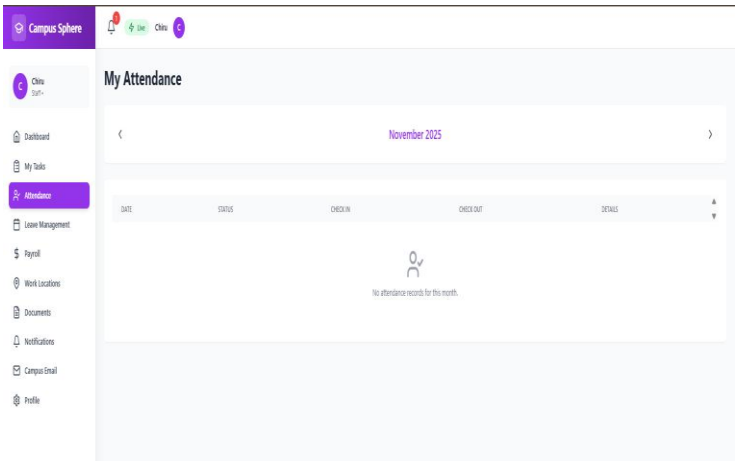

Campus Sphere
Lakshmi Narayana
Assign Students to Semester
STUDENT NAME
EMAIL
ASSIGNED SEMESTER
Gara Mani Deepak
Kumar
Nandhu

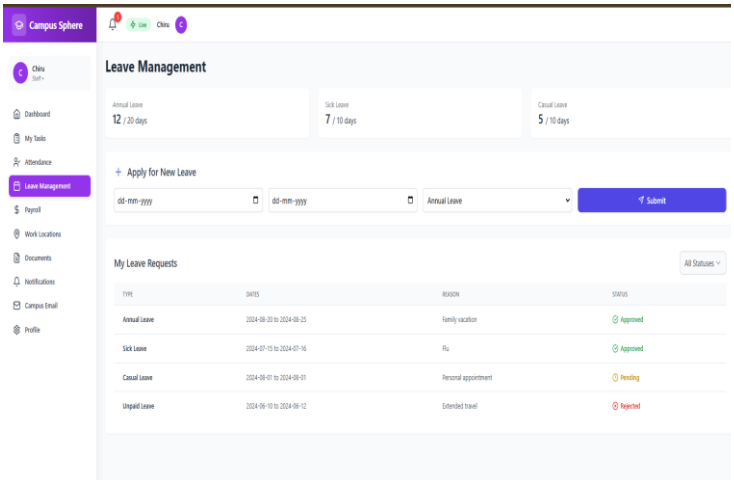

Campus Sphere
Lakshmi Narayana
Department Analytics
In-depth metrics and performance of your department.
Total Students
350
Courses Offered
25
Average Pass Rate
92%
Faculty Count
15
Annual Pass Percentage
Enrollment Trends
Faculty Workload Distribution

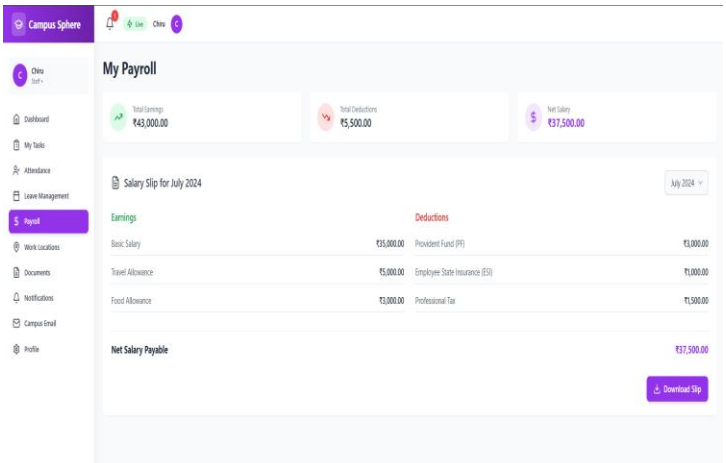
NON Teaching Staff Screen’s

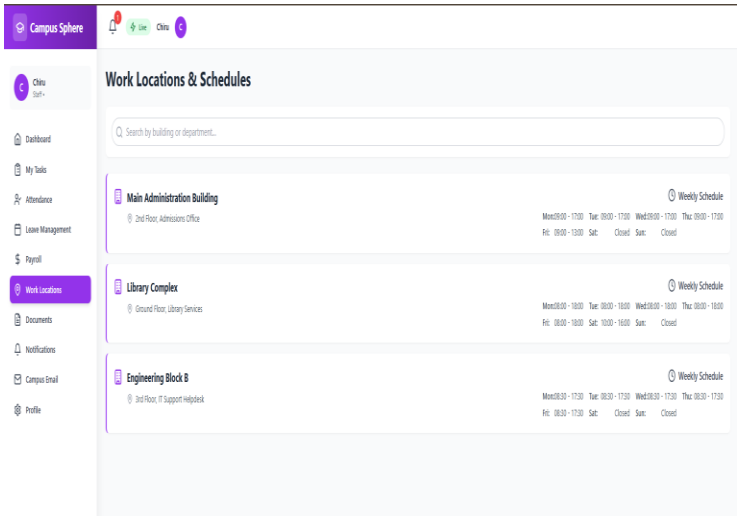


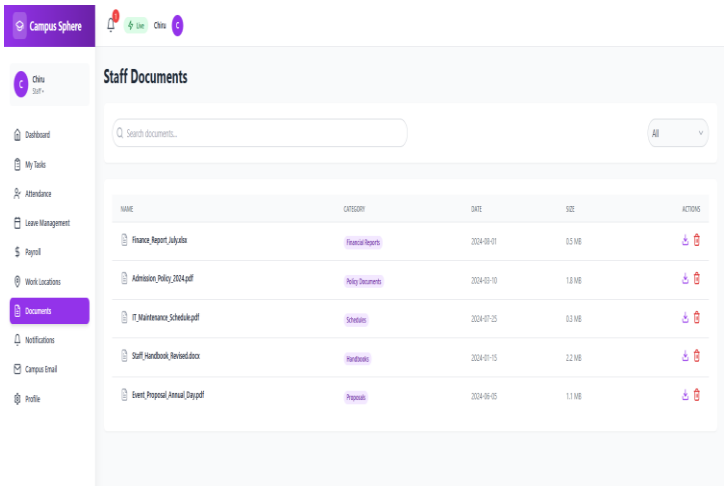




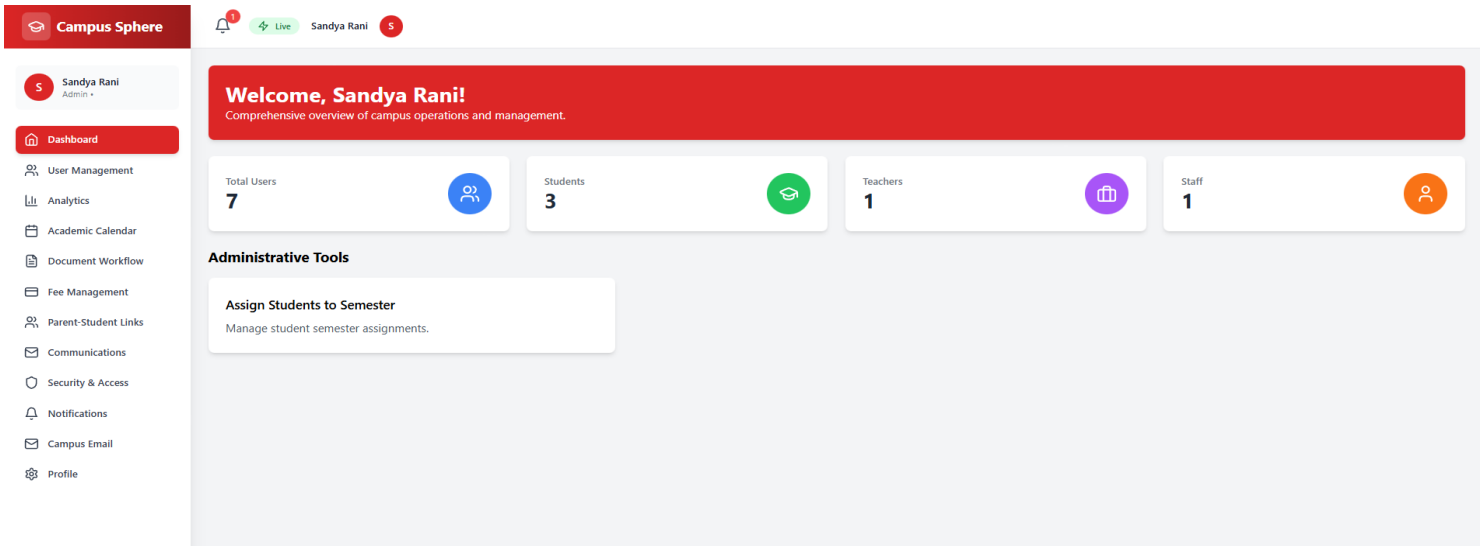


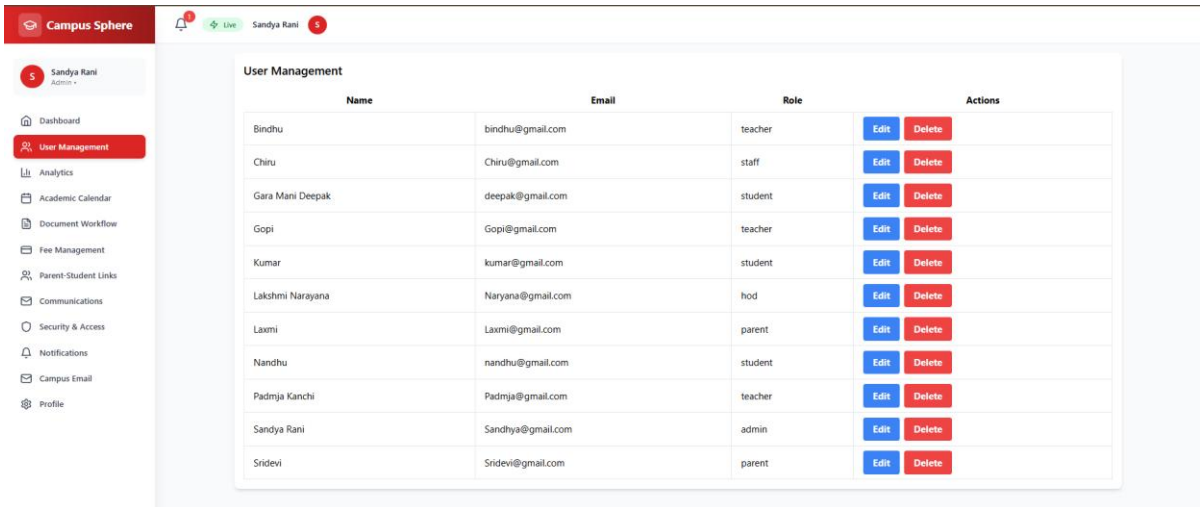


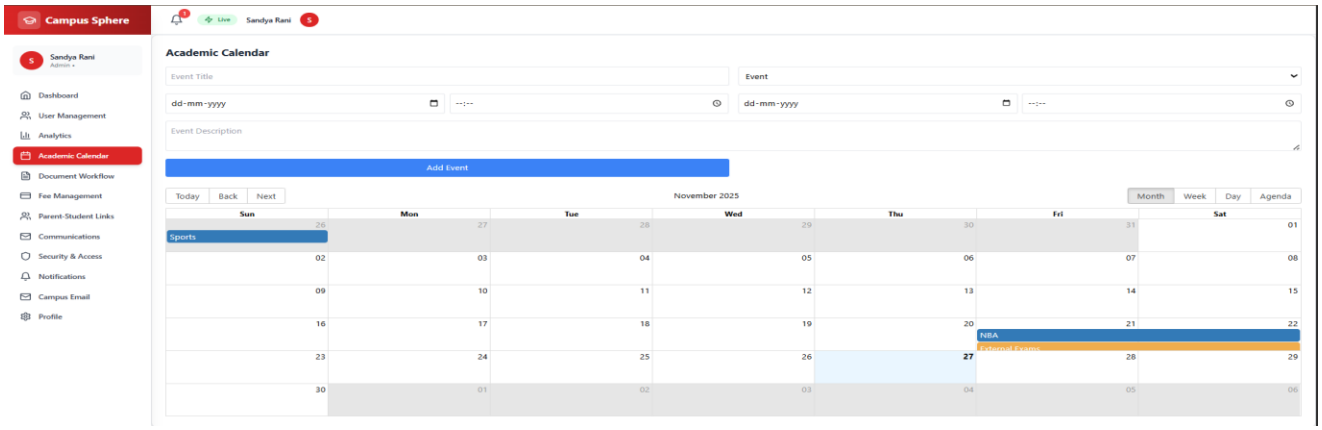


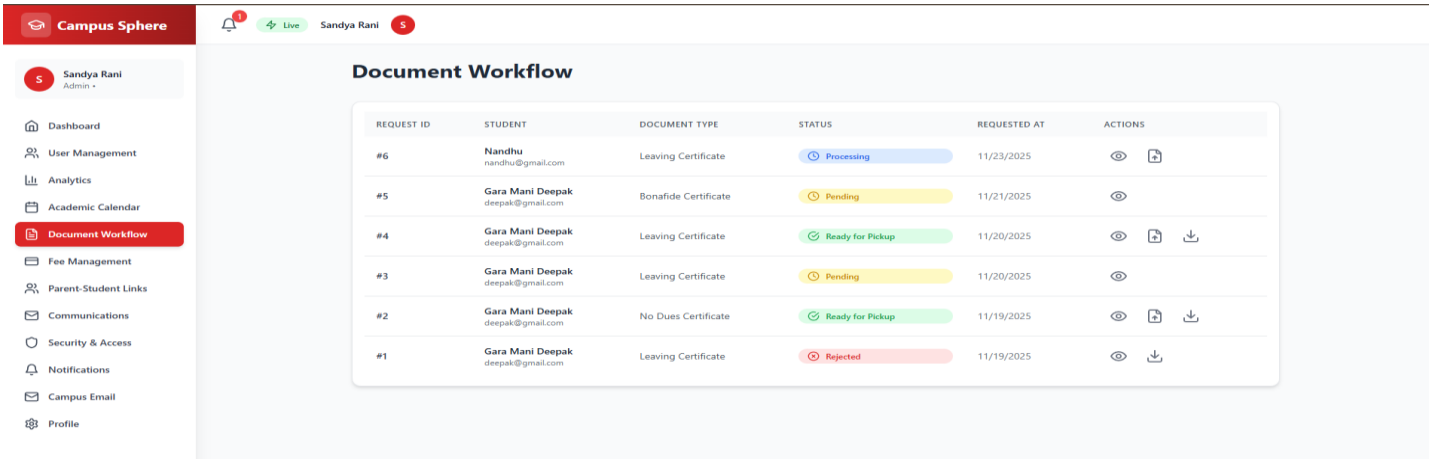


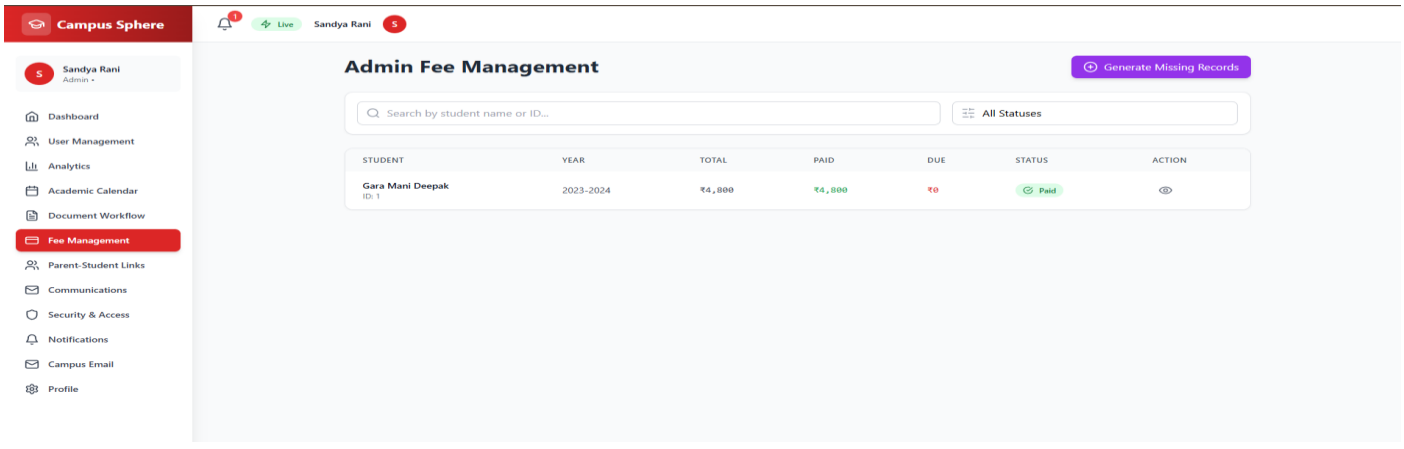
ADMIN Screen's

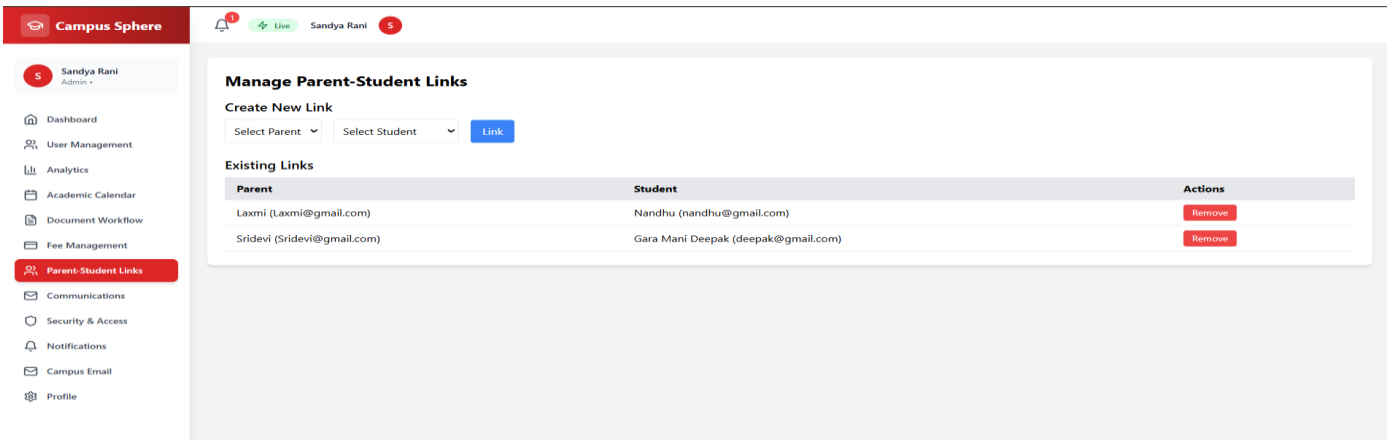


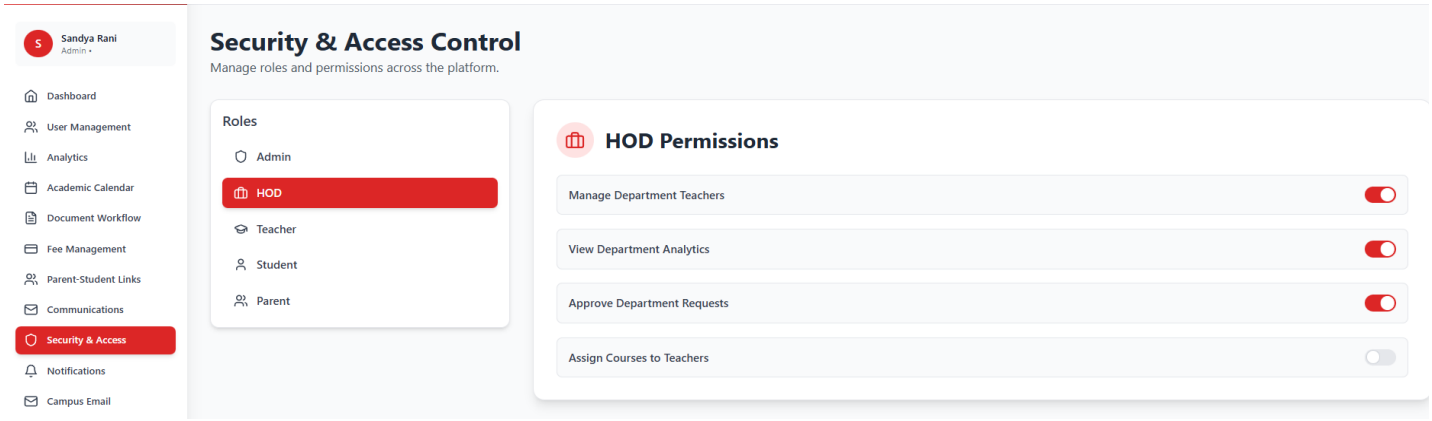


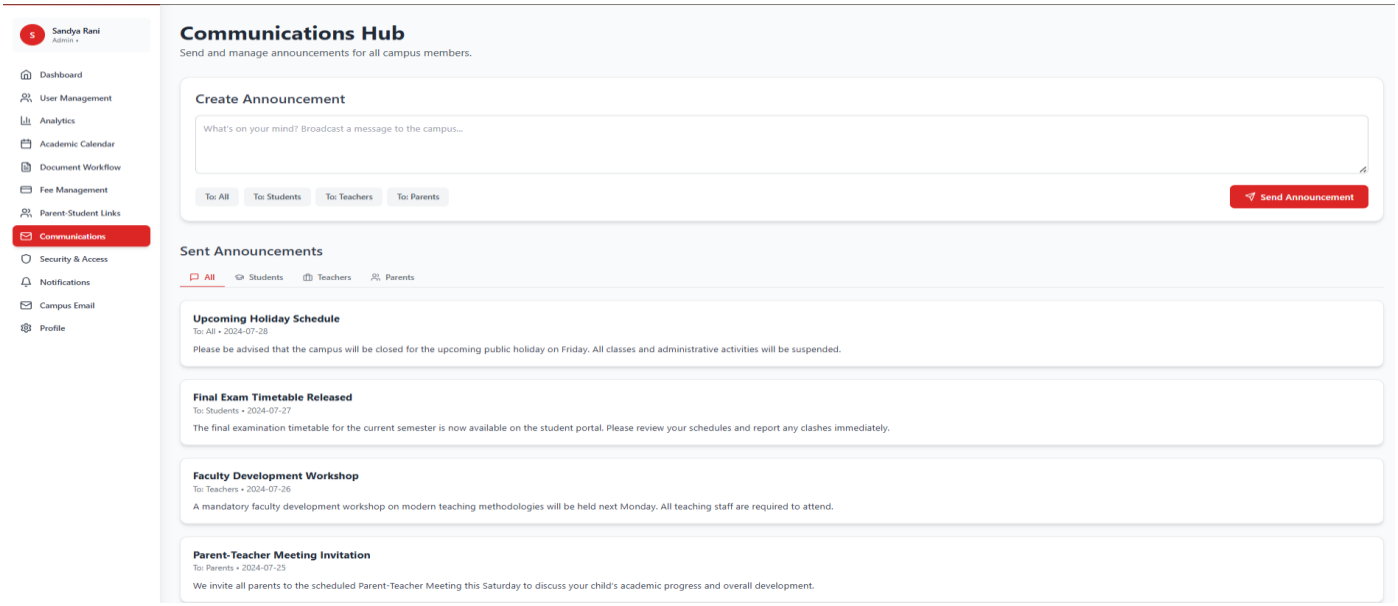

Campus Sphere
Live
Sandya Rani
Sandya Rani
Admin
Dashboard
User Management
Analytics
Academic Calendar
Document Workflow
Fee Management
Parent-Student Links
Communications
Security & Access
Notifications
Campus Email
Profile
Academic Calendar
Event Title
Event
dd-mm-yyyy
--:--
dd-mm-yyyy
--:--
Event Description
Add Event
Today
Back
Next
November 2025
Month
Week
Day
Agenda
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Sports
NBA
External Exams

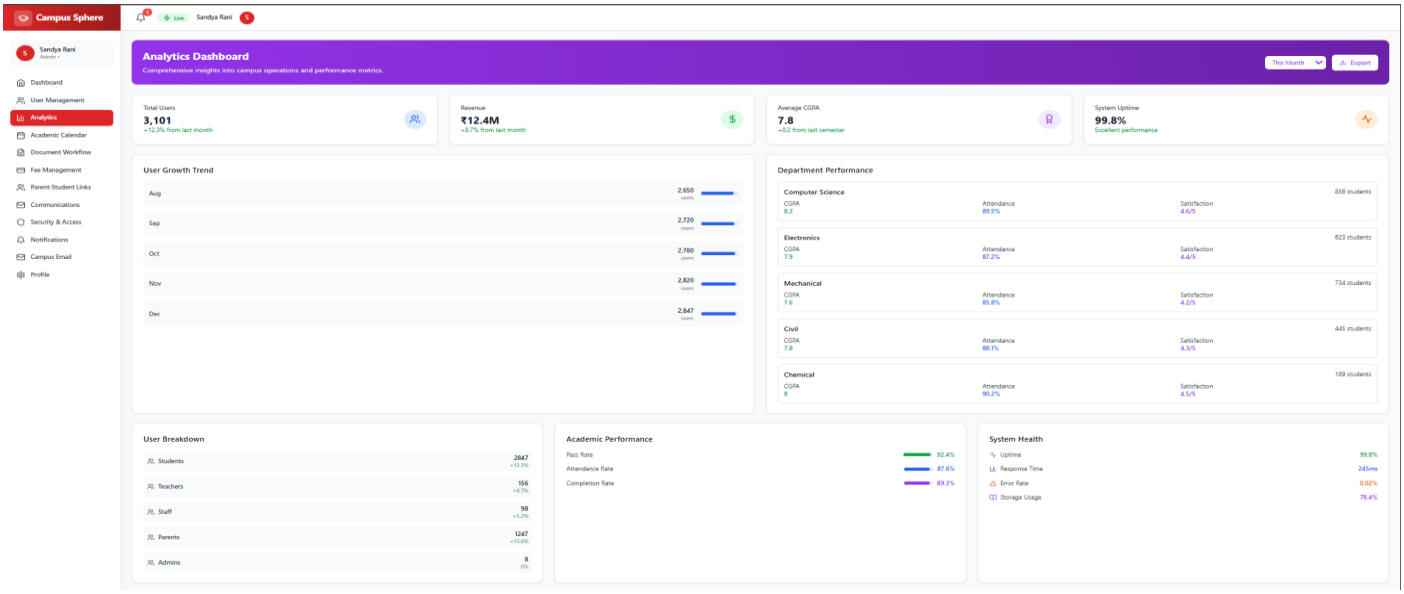

Campus Sphere
Live
Sandya Rani
Sandya Rani
Admin
Dashboard
User Management
Analytics
Academic Calendar
Document Workflow
Fee Management
Parent-Student Links
Communications
Security & Access
Notifications
Campus Email
Profile
Document Workflow
REQUEST ID
STUDENT
DOCUMENT TYPE
STATUS
REQUESTED AT
ACTIONS
#6
Nandhu
nandhu@gmail.com
Leaving Certificate
Processing
11/23/2025
#5
Gara Mani Deepak
deepak@gmail.com
Bonafide Certificate
Pending
11/21/2025
#4
Gara Mani Deepak
deepak@gmail.com
Leaving Certificate
Ready for Pickup
11/20/2025
#3
Gara Mani Deepak
deepak@gmail.com
Leaving Certificate
Pending
11/20/2025
#2
Gara Mani Deepak
deepak@gmail.com
No Dues Certificate
Ready for Pickup
11/19/2025
#1
Gara Mani Deepak
deepak@gmail.com
Leaving Certificate
Rejected
11/19/2025


Campus Sphere
Live
Sandya Rani
Sandya Rani
Admin
Dashboard
User Management
Analytics
Academic Calendar
Document Workflow
Fee Management
Parent-Student Links
Communications
Security & Access
Notifications
Campus Email
Profile
Admin Fee Management
Generate Missing Records
Search by student name or ID...
All Statuses
STUDENT
YEAR
TOTAL
PAID
DUE
STATUS
ACTION
Gara Mani Deepak
ID: 1
2023-2024
₹4,800
₹4,800
₹0
Paid


Campus Sphere
Live
Sandya Rani
Sandya Rani
Admin
Dashboard
User Management
Analytics
Academic Calendar
Document Workflow
Fee Management
Parent-Student Links
Communications
Security & Access
Notifications
Campus Email
Profile
Manage Parent-Student Links
Create New Link
Select Parent
Select Student
Link
Existing Links
Parent
Student
Actions
Laxmi (Laxmi@gmail.com)
Nandhu (nandhu@gmail.com)
Remove
Sridevi (Sridevi@gmail.com)
Gara Mani Deepak (deepak@gmail.com)
Remove


Sandya Rani
Admin
Dashboard
User Management
Analytics
Academic Calendar
Document Workflow
Fee Management
Parent-Student Links
Communications
Security & Access
Notifications
Campus Email
Security & Access Control
Manage roles and permissions across the platform.
Roles
Admin
HOD
Teacher
Student
Parent
HOD Permissions
Manage Department Teachers
View Department Analytics
Approve Department Requests
Assign Courses to Teachers


Sandya Rani
Admin
Dashboard
User Management
Analytics
Academic Calendar
Document Workflow
Fee Management
Parent-Student Links
Communications
Security & Access
Notifications
Campus Email
Profile
Communications Hub
Send and manage announcements for all campus members.
Create Announcement
What's on your mind? Broadcast a message to the campus...
To: All
To: Students
To: Teachers
To: Parents
Send Announcement
Sent Announcements
All
Students
Teachers
Parents
Upcoming Holiday Schedule
To: All • 2024-07-28
Please be advised that the campus will be closed for the upcoming public holiday on Friday. All classes and administrative activities will be suspended.
Final Exam Timetable Released
To: Students • 2024-07-27
The final examination timetable for the current semester is now available on the student portal. Please review your schedules and report any clashes immediately.
Faculty Development Workshop
To: Teachers • 2024-07-26
A mandatory faculty development workshop on modern teaching methodologies will be held next Monday. All teaching staff are required to attend.
Parent-Teacher Meeting Invitation
To: Parents • 2024-07-25
We invite all parents to the scheduled Parent-Teacher Meeting this Saturday to discuss your child's academic progress and overall development.


Campus Sphere
Sandya Rani
Analytics Dashboard
Comprehensive insights into campus operations and performance metrics.
This Month
Export
Total Users
3,101
+12.3% from last month
Revenue
₹12.4M
+8.7% from last month
Average CGPA
7.8
+0.2 from last semester
System Uptime
99.8%
Excellent performance
User Growth Trend
Aug
2,650
Sep
2,720
Oct
2,780
Nov
2,820
Dec
2,847
Department Performance
Computer Science
856 students
CGPA
8.2
Attendance
89.5%
Satisfaction
4.6/5
Electronics
623 students
CGPA
7.9
Attendance
87.2%
Satisfaction
4.4/5
Mechanical
734 students
CGPA
7.6
Attendance
85.8%
Satisfaction
4.2/5
Civil
445 students
CGPA
7.8
Attendance
88.1%
Satisfaction
4.3/5
Chemical
189 students
CGPA
8
Attendance
90.2%
Satisfaction
4.5/5
User Breakdown
Students
2847
Teachers
156
Staff
98
Parents
1247
Admins
8
Academic Performance
Pass Rate
92.4%
Attendance Rate
87.6%
Completion Rate
89.3%
System Health
Uptime
99.8%
Response Time
245ms
Error Rate
0.02%
Storage Usage
78.4%

BIBLIOGRAPHY

AND

CONCLUSION

Bibliography:

React Documentation – Official documentation for building interactive UIs.
https://react.dev/

Vite Official Guide – Documentation for fast frontend tooling and development server.
https://vitejs.dev/

PHP Manual – Reference for server-side scripting and backend logic.
https://www.php.net/manual/

MySQL Reference Manual – Official guide for relational database management and SQL querying.
https://dev.mysql.com/doc/

W3Schools Web Technologies Tutorials – General guidance for HTML, CSS, JavaScript.
https://www.w3schools.com/

MDN Web Docs – Standard reference for web technologies and frontend programming.
https://developer.mozilla.org/

Axios Documentation – For handling frontend-to-backend API communication.
https://axios-http.com/

Software Engineering Textbooks – For system design, SRS documentation, and testing methodologies:
Pressman, Roger. "Software Engineering: A Practitioner's Approach." McGraw-Hill.

Bootstrap & UI/UX Guidelines – For styling concepts referenced in frontend development.
https://getbootstrap.com/

General REST API Design References – Best practices for backend service structure. https://restfulapi.net/

Conclusion

The *Orbit Campus* platform successfully demonstrates how modern web technologies can digitalize and streamline core academic and administrative operations within an educational institution. By integrating a **React + Vite frontend** with a **PHP–MySQL backend**, the system delivers a highly responsive, secure, and scalable environment for students, faculty, and administrators.

The platform not only centralizes essential campus services—such as profile management, document handling, fee information, academic data, and notifications—but also significantly reduces manual workload, paperwork, and dependency on traditional processes. The role-based access model ensures that every user interacts only with the information relevant to their responsibilities, improving usability and security.

Through structured APIs, optimized database storage, and a user-friendly interface, Orbit Campus enhances transparency, minimizes errors, and accelerates campus workflows. It succeeds in creating a unified digital ecosystem that improves communication, accessibility, and operational efficiency for all stakeholders.

Overall, *Orbit Campus* stands as a scalable and future-ready solution capable of supporting the evolving digital needs of modern educational institutions.

THANK YOU