University Department Information Management System

Submitted By:
Team 18
Software Engineering
Rajib Mall, Suchi Kumari
Shiv Nadar University

Date of Submission – 05/05/2025

Table of Contents

Problem Statement

Universities today rely on fragmented, manual, and legacy systems to manage departmental information—ranging from student records and course scheduling to faculty workload and resource allocation—which leads to data inconsistencies, operational inefficiencies, and delayed decision-making. The lack of a unified platform forces department staff to maintain separate spreadsheets and disparate databases, resulting in redundant data entry, difficulty in tracking academic performance analytics, and increased likelihood of errors in reporting .

Moreover, existing solutions often lack real-time integration with central university services (such as the Student Information System, Financial Management System, and Library Management System), making it cumbersome for stakeholders—students, faculty, and administrators—to access up-to-date information. This fragmentation hampers timely course registration, grade submission, attendance tracking, and resource planning, ultimately affecting the quality of academic and administrative processes .

Therefore, there is a critical need for a web-based University Department Information Management System (UDIMS) that consolidates all departmental functions—student management, course and grade management, faculty workload tracking, inventory control, and analytics—into a single, role-based access platform. By automating workflows, enforcing standardized data handling, and integrating with existing university systems, UDIMS aims to eliminate manual redundancies, enhance data accuracy, and provide real-time insights to all stakeholders.

Software Requirements Specification (SRS)

Introduction

Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the University Department Information Management System (UDIMS). It details the system's objectives, features, interfaces, and design constraints. This document follows IEEE 830-1998 standards and serves as the primary reference for the development team.

Scope

The UDIMS project encompasses the development and implementation of a comprehensive information management system for university departments. This section defines the system boundaries, integration points, and project phases.

System Boundaries

- ***In Scope***
  - Student information management and academic records
  - Course management and scheduling
  - Faculty workload and research tracking
  - Department resource allocation and inventory
  - Academic performance analytics
  - Document management system
  - Automated reporting and analytics
  - Mobile-responsive web interface
  - Role-based access control
  - Integration with existing university systems
- ***Out of Scope***
  - University-wide financial management
  - Human resources management
  - Library management system
  - Campus security systems
  - Building management systems

- Third-party research databases

Project Phases and Deliverables

Phase	Deliverables	Timeline
Phase 1:	• Core system architecture • User authentication system • Basic student management • Department portal	Week 1
Phase 2:	• Course management system • Faculty dashboard • Grade management • System integration	Week 2-4
Phase 3:	• Advanced analytics • Mobile optimization • Attendance system • Performance optimization	Week 5-6
Phase 4:	• System testing • User training • Documentation • Deployment	Week 7

Definitions, Acronyms, and Abbreviations

Term	Definition

UDIMS	University Department Information Management System
API	Application Programming Interface
CRUD	Create, Read, Update, Delete operations
JWT	JSON Web Token

References

- IEEE 830-1998 Software Requirements Specification Standard
- Role-Based Access Control in Educational Systems
- University IT Security Policy Document v2.1
- Academic Information System Integration Guidelines 2023

Overview

The remaining sections of this document provide detailed requirements specifications. Section 2 gives a general description of the product, its functions, and user characteristics. Section 3 provides specific requirements including functional and non-functional requirements. Section 4 contains supporting information.

System Context

The UDIIMS will be integrated within the existing university IT infrastructure and will interface with other university systems including:

- Central Student Information System
- University Financial Management System
- Library Management System
- Research Grant Management Portal
- Alumni Portal

Stakeholders

The key stakeholders of the system include:

- Department Administration Staff
- Faculty Members

- Students
- Research Scholars
- Department Head
- University Administration
- External Auditors
- System Administrators

Terminology and Definitions

Term	**Definition**
SHALL	Mandatory requirement that must be implemented in the system
SHOULD	Recommended requirement that should be implemented unless there is a justified reason not to
MAY	Optional requirement that can be implemented if time and resources permit
UDIMS	University Department Information Management System
API	Application Programming Interface
CRUD	Create, Read, Update, Delete operations

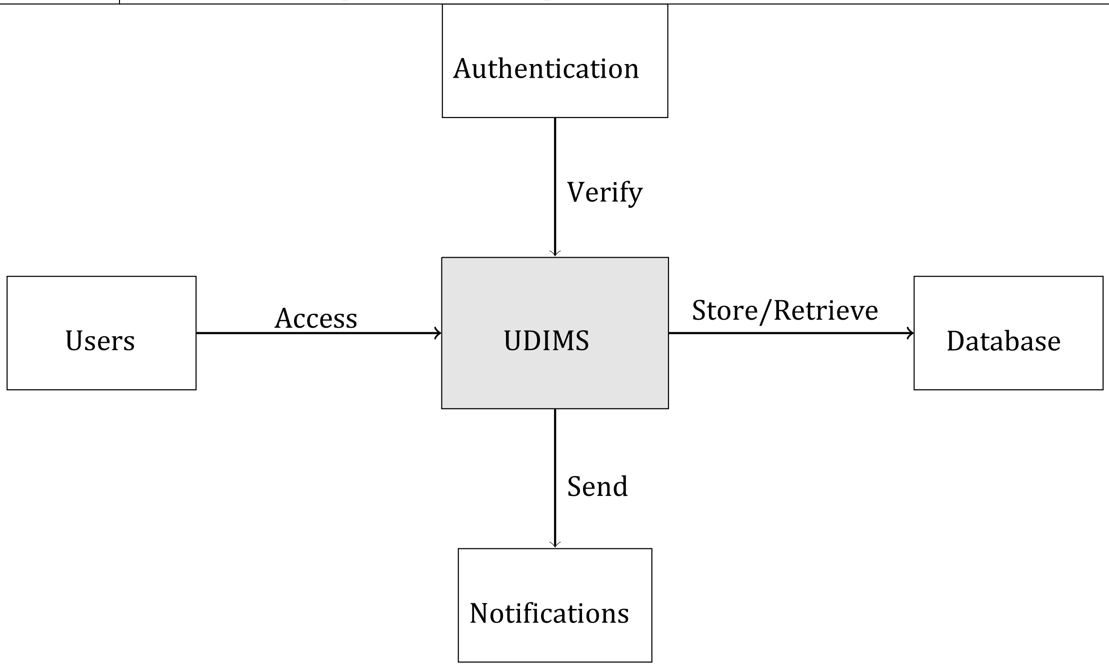


Figure1:SystemContextDiagram

Stakeholder	System Interaction	Potential Challenges
Students	• Course registration • Grade viewing • Document submission	• System availability during peak registration • Interface learning curve • Mobile accessibility
Faculty	• Grade management • Attendance tracking • Course material upload	• Bulk grade upload complexity • File size limitations • Real-time updates

Technology Considerations

- **Scalability Strategy**

  – ***Database Optimization***

  ∗ Implementation of database indexing

  ∗ Query
  optimization ∗
  Connection pooling

  ∗ Regular performance monitoring

  – ***Load Handling***

  ∗ Load balancing configuration

  ∗ Caching implementation

  ∗ Asynchronous processing for heavy tasks

- ***Error Handling and Recovery***

Error Type	Handling Strategy	Recovery Action
Database Connection	Retry with exponential backoff	Failover to backup database

Authentication Failure	Log attempt, notify user	Password reset option
File Upload Error	Chunk upload, progress tracking	Resume capability
Session Timeout	Auto-save feature	Session recovery

- **Performance Testing Strategy**
  - *Load Testing*
    - ∗ Simulate 100 concurrent users
    - ∗ Measure response times
    - ∗ Monitor system resources
    - ∗ Identify bottlenecks
  - *Stress Testing*
    - ∗ Test system limits
    - ∗ Verify graceful degradation
    - ∗ Recovery time measurement

Security Implementation

Component	**Implementation Details**
Password Policy	• Minimum 8 characters • Must include uppercase, lowercase, number, special character • Password history enforcement • Regular password change requirement
Session Management	• JWT token implementation • 30-minute session timeout • Secure cookie handling • CSRF protection

Access Control	• Role-based access control (RBAC) • Permission granularity • Activity logging • IP-based restrictions

Overall Description

UDIIMS is designed to streamline and automate the administrative processes of university departments. The system will handle student data management, course registration, grade processing, inventory management, financial tracking, and research documentation.

Product Perspective

UDIMS will be a new, self-contained web-based system designed to operate in the university's environment. The system will interface with:

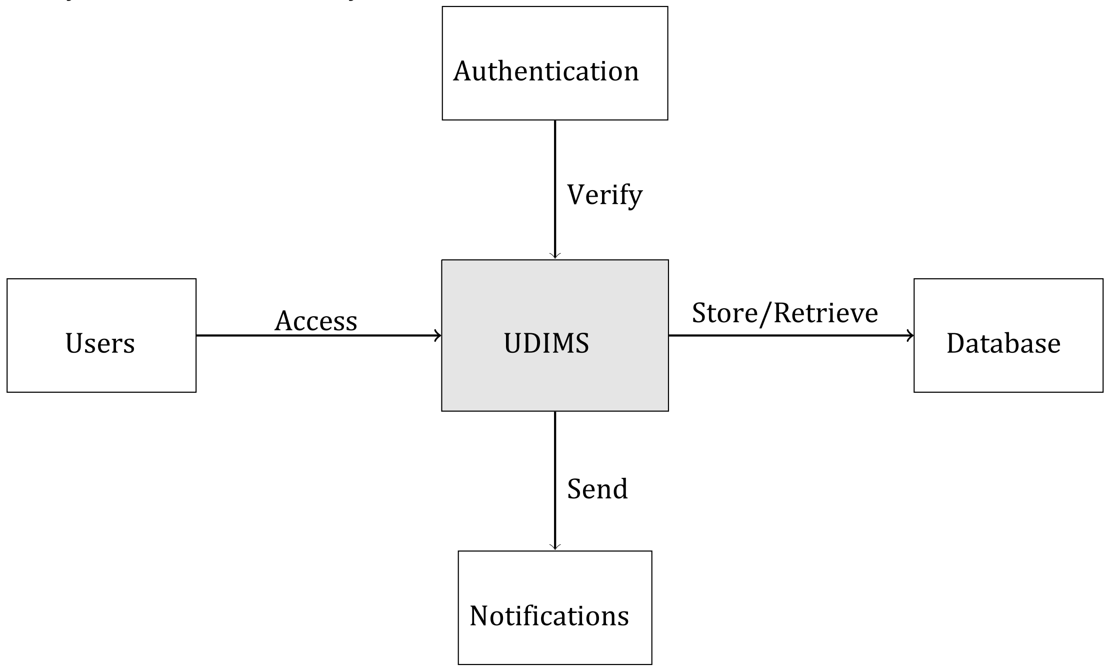


Figure2:SystemContextDiagram

Product Functions

The system SHALL provide the following core functions:

Function	**Description**

User Authentication	Secure login and role-based access control
Student Management	Registration, profile management, and academic record tracking
Course Management	Course creation, assignment, and scheduling
Grade Management	Grade entry, calculation, and report generation
Attendance Tracking	Student attendance recording and reporting

User Classes and Characteristics

User Class	**Role Description**	**Technical Expertise**
Administrator	System configuration and user management	High
Faculty	Course and grade management	Moderate
Students	Courseregistration and grade viewing	Basic
Staff	Administrative tasks and report generation	Moderate

Operating Environment

- Web-based application
- Compatible with standard web browsers
- Database server: MySQL/PostgreSQL
- Operating System: Platform independent

Design and Implementation Constraints

- Development using open-source technologies
- Must follow university security guidelines
- Must support concurrent user access
- Must maintain data privacy standards

Specific Requirements

External Interface Requirements

User Interfaces

The system SHALL provide:

- Responsive web interface
- Intuitive navigation menu
- Dashboard for each user role
- Form-based data entry
- Report generation interface

Hardware Interfaces

- Standard computer hardware
- Internet connectivity
- Minimum screen resolution: 1024x768

Software Interfaces

- Web Browser: Chrome, Firefox, Safari
- Database Management System
- Email Server Integration

Functional Requirements

ID	Requirement	Priority	Status
FR-1.1	System shall support single sign-on (SSO) authentication	High	Mandatory
FR-1.2	System shall provide role-based access control	High	Mandatory
FR-1.3	System shall enforce password policies and account security	High	Mandatory
FR-2.1	Faculty shall be able to create and manage courses	High	Mandatory

FR-2.2	Faculty shall be able to upload and manage course materials	Medium	Required
FR-2.3	Faculty shall be able to create and grade assignments	High	Mandatory
FR-3.1	Students shall be able to register for courses	High	Mandatory
FR-3.2	Students shall be able to submit assignments online	High	Required
FR-3.3	Students shall be able to view their grades and progress	High	Mandatory
FR-4.1	System shall generate automated attendance reports	Medium	Required
FR-4.2	System shall support bulk grade uploads via CSV	Medium	Required
FR-4.3	System shall provide analytics dashboard for administrators	Low	Optional
FR-5.1	System shall maintain audit logs of all critical operations	High	Mandatory
FR-5.2	System shall provide automated backup and recovery	High	Mandatory

Non-functional Requirements

ID	Requirement	Priority
NFR-1.1	System shall support 1000+ concurrent users	High
NFR-1.2	Page load time shall not exceed 2 seconds	High
NFR-1.3	API response time shall be under 500ms for 95% of requests	High
NFR-2.1	System shall maintain 99.9% uptime during academic year	High
NFR-2.2	All sensitive data shall be encrypted at rest and in transit	High
NFR-2.3	System shall implement rate limiting for API endpoints	High
NFR-3.1	System shall be compatible with major browsers (Chrome, Firefox, Safari)	Medium
NFR-3.2	System shall support responsive design for mobile devices	Medium
NFR-3.3	System shall provide accessibility features (WCAG 2.1)	Medium
NFR-4.1	System shall maintain automated daily backups	High
NFR-4.2	System recovery time objective (RTO) shall be under 4 hours	High
NFR-4.3	System shall support horizontal scaling	Medium
NFR-5.1	System shall log all user actions for audit purposes	High
NFR-5.2	System shall comply with GDPR and local data protection laws	High

Supporting Information

Please refer to the following documents:

1. System Design Document
2. Database Design Document
3. User Manual
4. Testing Plan

System Architecture

Architecture Overview

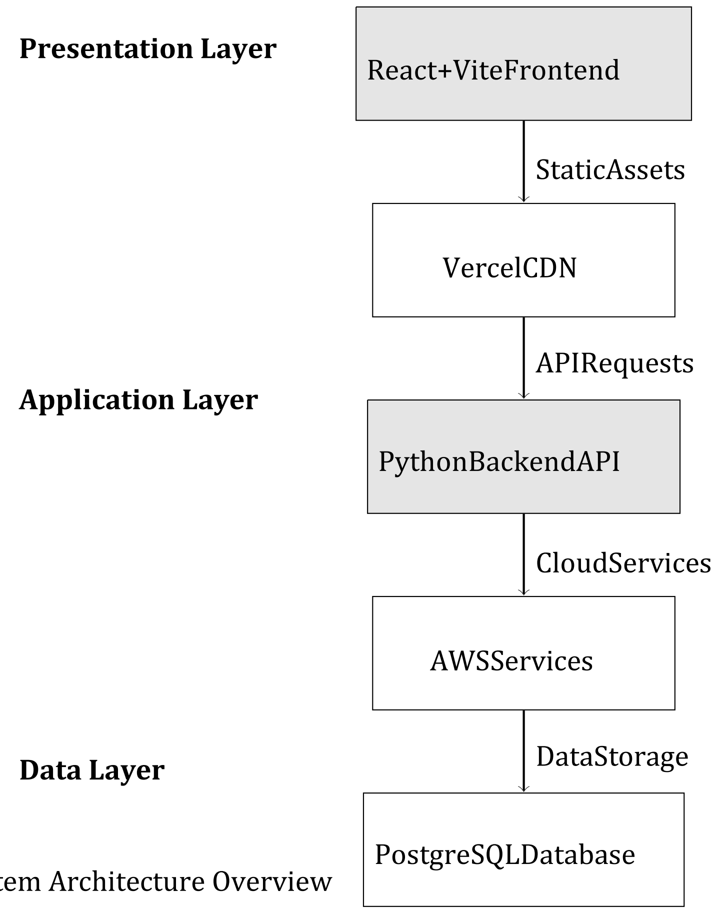


Figure 3: System Architecture Overview

Technology Stack

Component	Technology

Frontend	HTML5, CSS3, JavaScript, Bootstrap
Backend	PHP/Python with Laravel/Django
Database	MySQL/PostgreSQL
Server	Apache/Nginx
Version Control	Git

Data Requirements

Data Models

• **User Model**

- UserID (Primary Key)
- Username
- Password (Encrypted)
- Role
- Email – Last Login

• ***Student Model***

- StudentID (Primary Key)
- Name
- Department
- Batch
- Contact Information
- Academic Records

• ***Course Model***

- CourseID (Primary Key)
- Course Name
- Credits
- Department
- Prerequisites
- Syllabus

• ***Researcher Model***

- ResearcherID (Primary Key)

- Name
- Department
- Research Area
- Publications – Contact Information

- ***Teacher Model***
  - TeacherID (Primary Key)
  - Name
  - Department
  - Courses Taught
  - Office Hours
  - Contact Information

- ***Staff Model***
  - StaffID (Primary Key)
  - Name
  - Department
  - Role
  - Contact Information

Risk Management

Risk Assessment

Risk	**Impact**	**Probability**	**Mitigation Strategy**
Technical Complexity	High	Medium	Regular team training, consultation with experts
Schedule Delays	Medium	High	Buffer time in schedule, regular progress tracking
Integration Issues	High	Medium	Early integration testing, modular design

Resource Constraints	Medium	Low	Efficient resource allocation, prioritization

Appendices

Appendix A: Glossary

- UDIMS: University Department Information Management System
- SRS: Software Requirements Specification
- CRUD: Create, Read, Update, Delete
- UI: User Interface

Appendix B: Analysis Models

- Entity Relationship Diagrams
- Data Flow Diagrams
- Use Case Diagrams
- Sequence Diagrams
- Class Diagrams
- Activity Diagrams

Appendix C: Issues List

- Integration with legacy systems
- Data migration strategy
- Performance optimization
- Security compliance
- User training requirements
- System maintenance procedures

Appendix D: System Interfaces

- ***External Systems Integration***
  - Payment Gateway
  - Email Service
  - SMS Gateway – Cloud Storage
- **API Documentation**
- **Interface Specifications**

Appendix E: Data Dictionary

Field Name	**Data Type**	**Description**
StudentID	VARCHAR(10)	Unique identifier for students
CourseCode	VARCHAR(8)	Unique course identifier
GradePoint	DECIMAL(3,2)	Numeric grade value

Documentation Requirements

API Documentation Standards

- ***OpenAPI/Swagger Specification***
  - All APIs must be documented using OpenAPI 3.0
  - Include request/response examples
  - Document authentication requirements
  - Specify rate limits and quotas
- ***API Versioning***
  - Version number in URL path
  - Changelog maintenance
  - Deprecation notices
- ***Endpoint Documentation***
  - HTTP methods and status codes
  - Request parameters and data types
  - Response schemas

- Error handling

Code Documentation Requirements

Component	Documentation Requirements
Functions/Methods	• Purpose description • Parameter descriptions • Return value documentation • Usage examples • Exception handling
Classes	• Class purpose and responsibility • Constructor documentation • Public method descriptions • Property descriptions
Modules	• Module overview • Dependencies • Configuration options • Usage instructions

User Manual Specifications

- **Structure**
  - System overview
  - Getting started guide
  - Feature documentation
  - Troubleshooting guide
  - FAQ section

- ***Content Requirements***
    - Step-by-step instructions
    - Screenshots and diagrams
    - Video tutorials
    - Search functionality
    - Version history

Training Material Requirements

User Role	**Training Components**
Administrators	• System configuration • User management • Security protocols • Backup procedures
Faculty	• Course management • Grade entry • Report generation • Communication tools
Students	• Registration process • Course access • Assignment submission • Grade viewing

System Documentation Guidelines

- ***Technical Documentation***
    - System architecture

- Database schema
- Deployment procedures
- Security protocols – Performance metrics

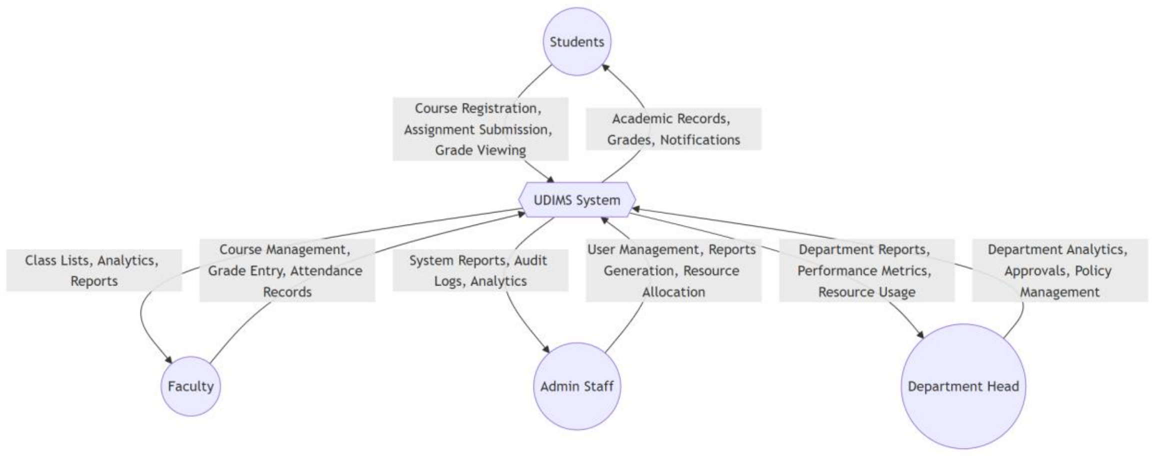
- ***Maintenance Documentation***
  - Backup procedures
  - Update protocols
  - Troubleshooting guides
  - System monitoring
  - Disaster recovery

- ***Version Control***
  - Documentation versioning
  - Change tracking
  - Review procedures – Approval workflow

Data Flow Diagram (DFD)

Level 0 DFD

Figure 1. Level 0 DFD for UDIMS


Students
Course Registration, Assignment Submission, Grade Viewing
Academic Records, Grades, Notifications
UDIMS System
Class Lists, Analytics, Reports
Course Management, Grade Entry, Attendance Records
System Reports, Audit Logs, Analytics
User Management, Reports Generation, Resource Allocation
Department Reports, Performance Metrics, Resource Usage
Department Analytics, Approvals, Policy Management
Faculty
Admin Staff
Department Head

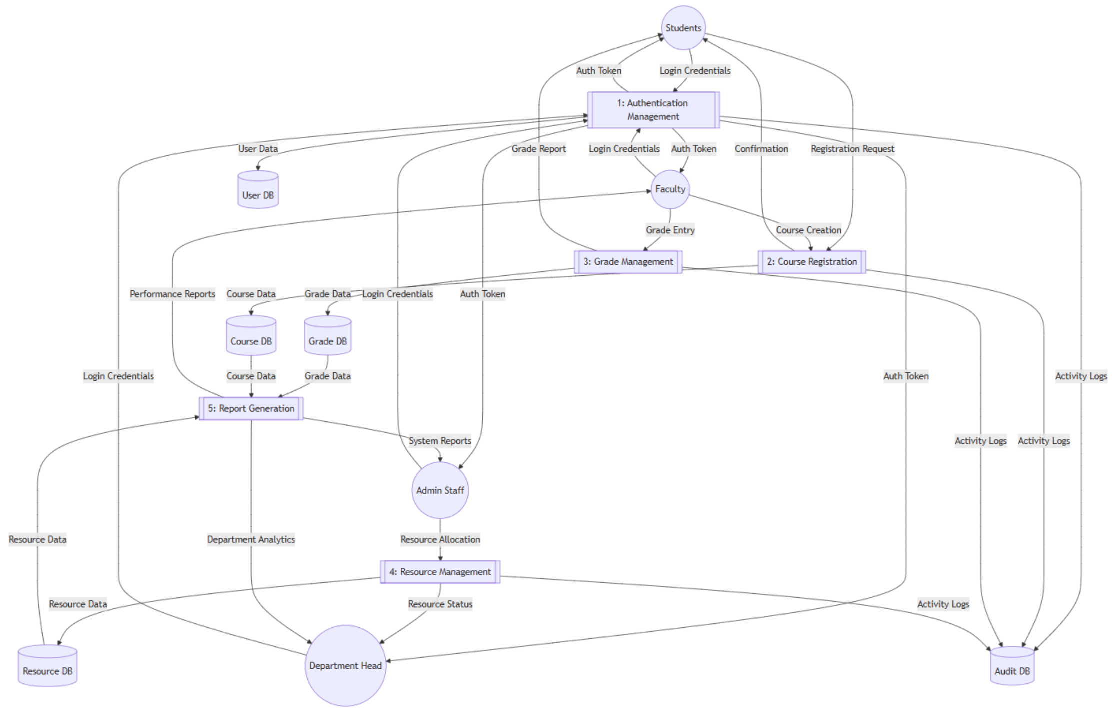
Description:
The Level 0 DFD (context diagram) shows the UDIMS system as a single process (the hexagon labeled "UDIMS System") interacting with three external entities—Students, Faculty, Admin Staff, and Department Head. Arrows indicate data flows:

- Students send course registration and receive academic records and grades.
- Faculty receive class lists and return grade entry and attendance records.
- Admin Staff receive system reports and send resource allocation requests.
- Department Head receives department analytics and sends policy management decisions.

This high-level view establishes the system boundary and major data exchanges.

Level 1 DFD

Figure 2. Level 1 DFD for UDIMS



Description:
The Level 1 DFD decomposes the UDIMS into five subprocesses:

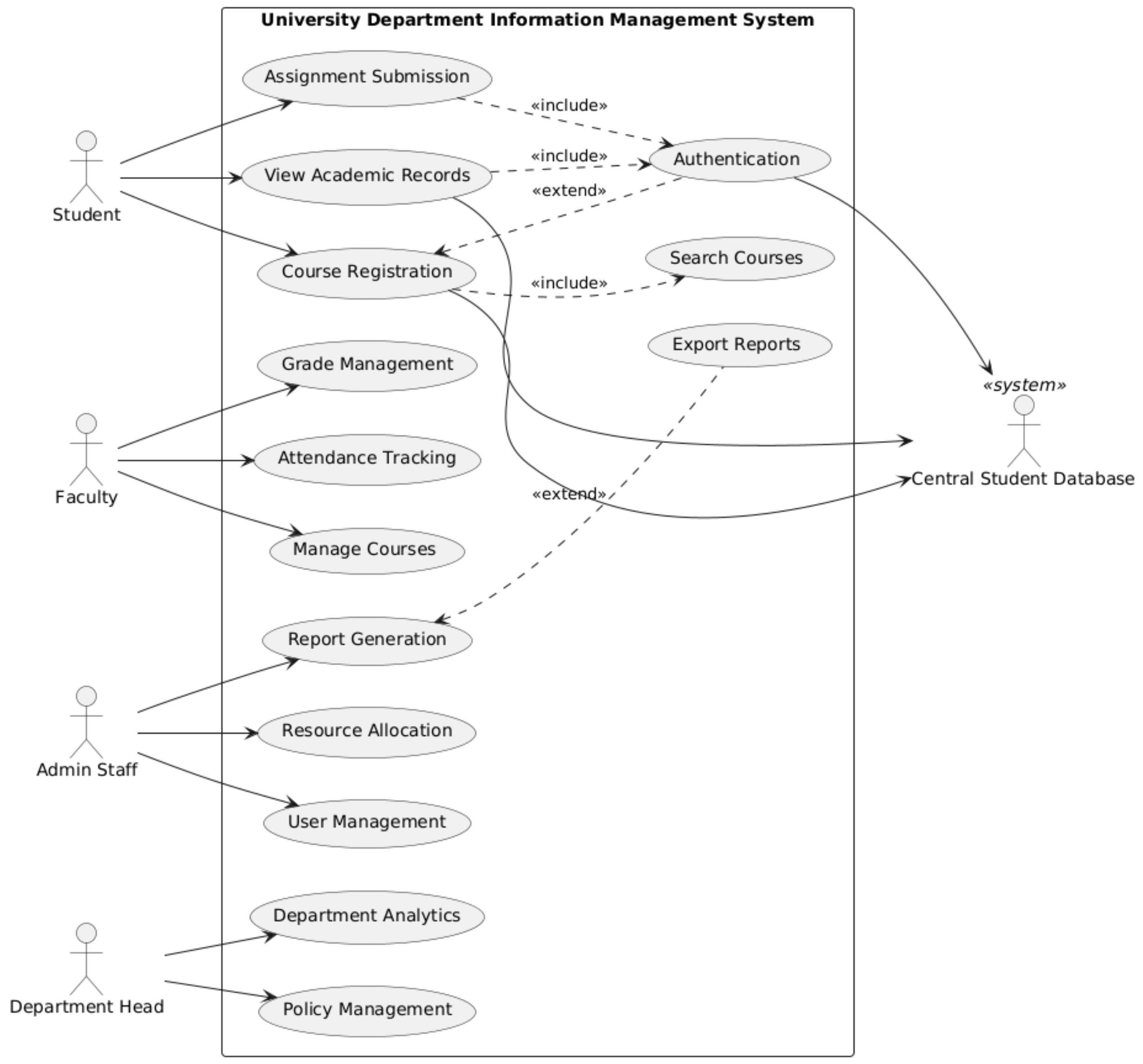
1. **Authentication Management** – Validates login credentials against the User DB and issues auth tokens to Students and Faculty.
2. **Course Registration** – Processes registration requests, updates Course DB, and confirms to Students.
3. **Grade Management** – Accepts grade entries from Faculty, updates Grade DB, and generates grade reports.
4. **Resource Management** – Manages resource data in Resource DB, allocates resources to Department Head, logs activity.
5. **Report Generation** – Aggregates data from Course DB, Grade DB, and logs to produce performance and system reports for Admin Staff and Department Head.

Data stores (User DB, Course DB, Grade DB, Resource DB, Audit DB) and data flows (e.g., “Auth Token,” “Grade Data,” “Activity Logs”) are labeled to show how information moves between processes, data stores, and external entities.

Use Case Diagram and Documentation

Use Case Diagram

Figure 3. Use Case Diagram for UDIMS



Description:
This diagram shows the primary actors—Student, Faculty, Admin Staff, Department Head—and the key system use-cases they invoke.

- **«include» relationships** (dashed arrows) indicate that Authentication is required before Course Registration, View Academic Records, and Assignment Submission.
- **«extend» relationships** (dotted arrows) show optional flows (e.g. Export Reports extends Report Generation).
- The external "Central Student Database" actor represents the university's master record system.

Use Case Documentation

1: User Authentication

ID: 1

Title: Authenticate Users

Description: The system verifies the identity of users (students, faculty, staff, etc.) using credentials such as username and password. It ensures that only authorized users can access the system based on their roles.

Primary Actor: User (Student, Faculty, Staff, Department Head)

Secondary Actor: Central Student Database / Authentication System

Preconditions: The user has entered their login credentials (username and password). The system is operational and connected to the authentication database. The user must have an active account in the system with valid credentials.

Postconditions: The user is authenticated and granted access to the system with rolebased permissions. If authentication fails, the user is prompted to retry or reset their password. **Dependency:** None

Generalization: None

Main Success Scenario: The user navigates to the login page and enters their login credentials, including their username and password. The system validates these credentials against the authentication database. If the credentials are valid, the system generates a session token, such as a JSON Web Token (JWT), and grants access to the user. The user is then redirected to their respective dashboard or home page based on their role. **Extensions or Alternate Flow:** If the credentials are invalid, the system denies access and prompts the user to retry or reset their password. If the user exceeds the maximum number of login attempts, the account is locked, and the user is notified via email to contact the administrator. In cases where the authentication database is unavailable, the system displays an error message and logs the issue for further investigation.

Frequency of Use: Very High (every time a user logs in)

Status: To be developed

Owner: Team 18

Priority: High

2: Course Registration

ID: 2

Title: Register for Courses

Description: Students can search for available courses, select courses to register for, and submit their registration requests. The system validates the student's eligibility and updates the course enrollment records.

Primary Actor: Student

Secondary Actor: Central Student Database

Preconditions: The student is authenticated and logged into the system. The registration period is open, and the student has not exceeded their course load limit. The system must have access to the course catalog and the student's academic records.

Postconditions: The student's course registration is successfully recorded in the system, and the student receives a confirmation of successful registration. If the registration fails, the system provides feedback and suggestions for alternative courses.

Dependency: Extends "Search Courses"

Generalization: None

Main Success Scenario: The student navigates to the "Course Registration" page and searches for available courses using filters such as department, semester, or keywords. The student selects courses to register for and adds them to their cart. After reviewing their selected courses, the student submits the registration request. The system validates the student's eligibility, including prerequisites and course load limits. If valid, the system processes the registration and updates the course enrollment records. The student receives a confirmation message with details of the registered courses.

Extensions or Alternate Flow: If the student exceeds their course load limit, the system displays an error message and prevents registration. If the course is full, the system notifies the student and allows them to choose alternative courses. If the student fails to meet prerequisites, the system denies registration and provides feedback. In cases where the course catalog is outdated, the system displays a warning message and logs the issue for review.

Frequency of Use: High (during registration periods)

Status: To be developed

Owner: Team 18

Priority: High

3: Search Courses

ID: 3

Title: Search for Courses

Description: Students can search for available courses based on various criteria such as department, semester, or keywords. The system retrieves and displays relevant course information.

Primary Actor: Student

Secondary Actor: Central Student Database

Preconditions: The student is authenticated and logged into the system. The course catalog is up-to-date and accessible. The system must have access to the course database to retrieve relevant information.

Postconditions: The student views a list of courses matching their search criteria. If no courses are found, the system provides suggestions for alternative search terms.

Dependency: Included by "Course Registration".

Generalization: None

Main Success Scenario: The student navigates to the "Search Courses" page and enters search criteria such as department, semester, or keywords. The system queries the course database and retrieves matching courses. The system displays the list of courses with details such as course name, credits, and prerequisites.

Extensions or Alternate Flow: If no courses match the search criteria, the system informs the student and suggests alternative search terms. If the course catalog is outdated, the system displays a warning message and logs the issue for further investigation.

Frequency of Use: High (during registration periods)

Status: To be developed

Owner: Team 18

Priority: Medium

4: View Academic Records

ID: 4

Title: View Academic Records

Description: Students can access their academic records, including grades, transcripts, and notifications.

Primary Actor: Student

Secondary Actor: Central Student Database

Preconditions: The student is authenticated and logged into the system. The student's academic records are available in the system and accessible to the student.

Postconditions: The student views their academic records, including grades, transcripts, and any pending notifications. If the records are unavailable, the system displays an error message. **Dependency:** None

Generalization: None

Main Success Scenario: The student navigates to the "View Academic Records" page. The system retrieves the student's academic records from the database and displays them, including grades, transcripts, and any pending notifications.

Extensions or Alternate Flow: If the student's academic records are unavailable, the system displays an error message and logs the issue for further investigation. If the student lacks permission to view certain records, the system denies access and displays an appropriate message.

Frequency of Use: High (regularly throughout the semester)

Status: To be developed

Owner: Team 18 **Priority:** High

5: Assignment Submission

ID: 5

Title: Submit Assignments Online

Description: Students can upload assignments and submit them for grading. The system validates the submission and stores it for faculty review.

Primary Actor: Student

Secondary Actor: Faculty

Preconditions: The student is authenticated and logged into the system. The assignment submission deadline has not passed, and the student has completed the assignment. The system must have access to the assignment database.

Postconditions: The assignment is successfully uploaded and marked as submitted. If the submission fails, the system provides feedback and allows the student to retry.

Dependency: None

Generalization: None

Main Success Scenario: The student navigates to the "Assignment Submission" page and selects the assignment they wish to submit. The student uploads the assignment file and submits it. The system validates the file format and size. If valid, the system stores the assignment and notifies the faculty. The student receives a confirmation of successful submission.

Extensions or Alternate Flow: If the submission deadline has passed, the system denies submission and displays an error message. If the file format or size is invalid, the system rejects the submission and prompts the student to correct it. If the system encounters a technical issue during submission, it logs the error and notifies the student to retry later.

Frequency of Use: Moderate (periodically during the semester)

Status: To be developed

Owner: Team 18

Priority: High

6: Grade Management

ID: 6

Title: Enter and Manage Grades

Description: Faculty can enter grades for students, calculate grade points, and generate reports.

Primary Actor: Faculty

Secondary Actor: Central Student Database

Preconditions: The faculty member is authenticated and logged into the system. The faculty member has permission to manage grades, and the grading period is open. The system must have access to the student database.

Postconditions: Grades are entered and saved in the system. Grade reports are generated and accessible. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The faculty member navigates to the "Grade Management" page and selects a course and accesses the grade entry interface. The faculty member enters grades for students and saves them. The system calculates grade points and generates reports. The faculty member reviews and approves the grade report. The system updates the student's academic records and sends notifications.

Extensions or Alternate Flow: If the grading period is closed, the system denies grade entry and displays an error message. If the faculty member lacks permission to manage grades, the system denies access. If the system encounters a technical issue during grade entry, it logs the error and notifies the faculty member to retry later.

Frequency of Use: High (at the end of each semester)

Status: To be developed

Owner: Team 18

Priority: High

7: Attendance Tracking

ID: 7

Title: Track Student Attendance

Description: Faculty can mark student attendance for classes and generate attendance reports.

Primary Actor: Faculty

Secondary Actor: Central Student Database

Preconditions: The faculty member is authenticated and logged into the system. The faculty member has permission to track attendance, and the class session is active. The system must have access to the attendance database.

Postconditions: Attendance records are updated in the system. Attendance reports are generated and accessible. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The faculty member navigates to the "Attendance Tracking" page and selects a class session. The faculty member marks attendance for students, and the system updates the attendance records in real-time. The faculty member generates an attendance report for the class, which is saved and made available for review. **Extensions or Alternate Flow:** If the class session is not active, the system denies attendance tracking. If the faculty member lacks permission to track attendance, the system denies access. If the system encounters a technical issue during attendance tracking, it logs the error and notifies the faculty member to retry later.

Frequency of Use: High (daily during classes)

Status: To be developed

Owner: Team 18

Priority: High

8: Manage Courses

ID: 8

Title: Manage Courses

Description: Faculty can create, edit, and manage courses, including setting prerequisites, syllabus, and schedules. **Primary Actor:** Faculty

Secondary Actor: Central Student Database

Preconditions: The faculty member is authenticated and logged into the system. The faculty member has permission to manage courses. The system must have access to the course database.

Postconditions: The course details are updated in the system. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The faculty member navigates to the "Manage Courses" page and selects a course to edit or creates a new course. The faculty member enters course details such as course name, credits, prerequisites, and syllabus. The system validates the input data. If valid, the system saves the course details and updates the course catalog. The faculty member receives a confirmation of successful update.

Extensions or Alternate Flow: If the faculty member lacks permission to manage courses, the system denies access. If the input data is invalid (e.g., missing fields), the system rejects the update and prompts the faculty member to correct it. If the system encounters a technical issue during course management, it logs the error and notifies the faculty member to retry later.

Frequency of Use: Moderate (before each semester)

Status: To be developed

Owner: Team 18

Priority: High

9: Report Generation

ID: 9

Title: Generate Reports

Description: Admin staff can generate various reports such as system activity logs, audit logs, and performance metrics. **Primary Actor:** Admin Staff

Secondary Actor: Audit DB

Preconditions: The admin staff member is authenticated and logged into the system. The admin staff member has permission to generate reports. Relevant data is available in the system.

Postconditions: Reports are generated and saved in the system. If the process fails, the system provides feedback and logs the issue.

Dependency: Extends "Export Reports"

Generalization: None

Main Success Scenario: The admin staff member navigates to the "Report Generation" page and selects the type of report to generate, such as system activity logs or audit logs. The system retrieves the necessary data from the database and generates the report in the specified format, such as PDF or CSV. The admin staff member reviews the report and saves it.

Extensions or Alternate Flow: If the admin staff member lacks permission to generate reports, the system denies access. If the report generation fails due to data retrieval issues, the system displays an error message. If the system encounters a technical issue during report generation, it logs the error and notifies the admin staff member to retry later.

Frequency of Use: Moderate (periodically)

Status: To be developed

Owner: Team 18

Priority: Medium

10: Resource Allocation

ID: 10

Title: Allocate Resources

Description: Admin staff can allocate resources such as classrooms, labs, and equipment to departments and courses. **Primary Actor:** Admin Staff

Secondary Actor: Resource DB

Preconditions: The admin staff member is authenticated and logged into the system. The admin staff member has permission to manage resource allocation. Available resources are listed in the system.

Postconditions: Resource allocation records are updated in the system. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The admin staff member navigates to the "Resource Allocation" page and selects a resource to allocate, such as a classroom or lab. The admin staff member assigns the resource to a department or course. The system validates the allocation, ensuring there are no conflicts with existing bookings. If valid, the system saves the allocation and updates the resource status. The admin staff member receives a confirmation of successful allocation.

Extensions or Alternate Flow: If the admin staff member lacks permission to allocate resources, the system denies access. If the resource is already allocated, the system displays a conflict message. If the system encounters a technical issue during resource allocation, it logs the error and notifies the admin staff member to retry later.

Frequency of Use: Moderate (before each semester)

Status: To be developed

Owner: Team 18

Priority: Medium

11: User Management

ID: 11

Title: Manage Users

Description: Admin staff can add, edit, and deactivate user accounts, including students, faculty, and staff.

Primary Actor: Admin Staff

Secondary Actor: Central Student Database

Preconditions: The admin staff member is authenticated and logged into the system. The admin staff member has permission to manage users. The system must have access to the user database.

Postconditions: User accounts are updated in the system. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The admin staff member navigates to the "User Management" page and selects a user account to edit or creates a new account. The admin staff member enters user details such as username, role, and permissions. The system validates the input data. If valid, the system saves the user account and updates the user database. The admin staff member receives a confirmation of successful update.

Extensions or Alternate Flow: If the admin staff member lacks permission to manage users, the system denies access. If the input data is invalid (e.g., duplicate username), the system rejects the update and prompts the admin staff member to correct it. If the system encounters a technical issue during user management, it logs the error and notifies the admin staff member to retry later. **Frequency of Use:** Moderate (as needed)

Status: To be developed

Owner: Team 18

Priority: High

12: Department Analytics

ID: 12

Title: Analyze Department Performance

Description: Department heads can view analytics and performance metrics for their departments, including student performance, resource utilization, and research output.

Primary Actor: Department Head

Secondary Actor: Data Storage

Preconditions: The department head is authenticated and logged into the system. The department head has permission to view analytics. Relevant data is available in the system.
Postconditions: The department head views analytics and performance metrics. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The department head navigates to the "Department Analytics" page. The system retrieves relevant data for the department, such as student performance, resource utilization, and research output. The system displays analytics in the form of charts, graphs, and reports. The department head reviews the analytics and takes necessary actions.

Extensions or Alternate Flow: If the department head lacks permission to view analytics, the system denies access. If the data is incomplete or unavailable, the system displays a warning message. If the system encounters a technical issue during analytics generation, it logs the error and notifies the department head to retry later.

Frequency of Use: Moderate (periodically)

Status: To be developed

Owner: Team 18

Priority: High

13: Policy Management

ID: 13

Title: Manage Department Policies

Description: Department heads can create, edit, and approve policies related to department operations, such as resource usage, research guidelines, and student conduct.

Primary Actor: Department Head

Secondary Actor: Data Storage

Preconditions: The department head is authenticated and logged into the system. The department head has permission to manage policies. The system must have access to the policy database.

Postconditions: Policies are updated in the system. If the process fails, the system provides feedback and logs the issue.

Dependency: None

Generalization: None

Main Success Scenario: The department head navigates to the "Policy Management" page and selects a policy to edit or creates a new policy. The department head enters policy details and saves them. The system validates the policy content. If valid, the system saves the policy and updates the policy database. The department head receives a confirmation of successful update.

Extensions or Alternate Flow: If the department head lacks permission to manage policies, the system denies access. If the policy content is invalid (e.g., missing fields), the system rejects the update and prompts the department head to correct it. If the system encounters a technical issue during policy management, it logs the error and notifies the department head to retry later. **Frequency of Use:** Low (as needed)

Status: To be developed

Owner: Team 18

Priority: Low

14: Export Reports

ID: 14

Title: Export Reports

Description: Admin staff or department heads can export generated reports in various formats (e.g., PDF, CSV) for further analysis or record-keeping.

Primary Actor: Admin Staff / Department Head

Secondary Actor: Report Generation Module

Preconditions: The user is authenticated and has permission to export reports. Reports have been generated and are available for export. The system must have access to the report database.

Postconditions: The report is exported in the desired format (e.g., PDF, CSV). The user downloads the report successfully. If the process fails, the system provides feedback and logs the issue.

Dependency: Extends "Report Generation"

Generalization: None

Main Success Scenario: The user navigates to the "Reports" section and selects a report to export, such as an attendance report or grade report. The user chooses the export format, such as PDF or CSV. The system generates the report in the selected format. The user downloads the report to their local device.

Extensions or Alternate Flow: If the user lacks permission to export reports, the system denies access. If the report generation fails due to data retrieval issues, the system displays an error message. If the system encounters a technical issue during report export, it logs the error and notifies the user to retry later.

Frequency of Use: Moderate (periodically)

Status: To be developed

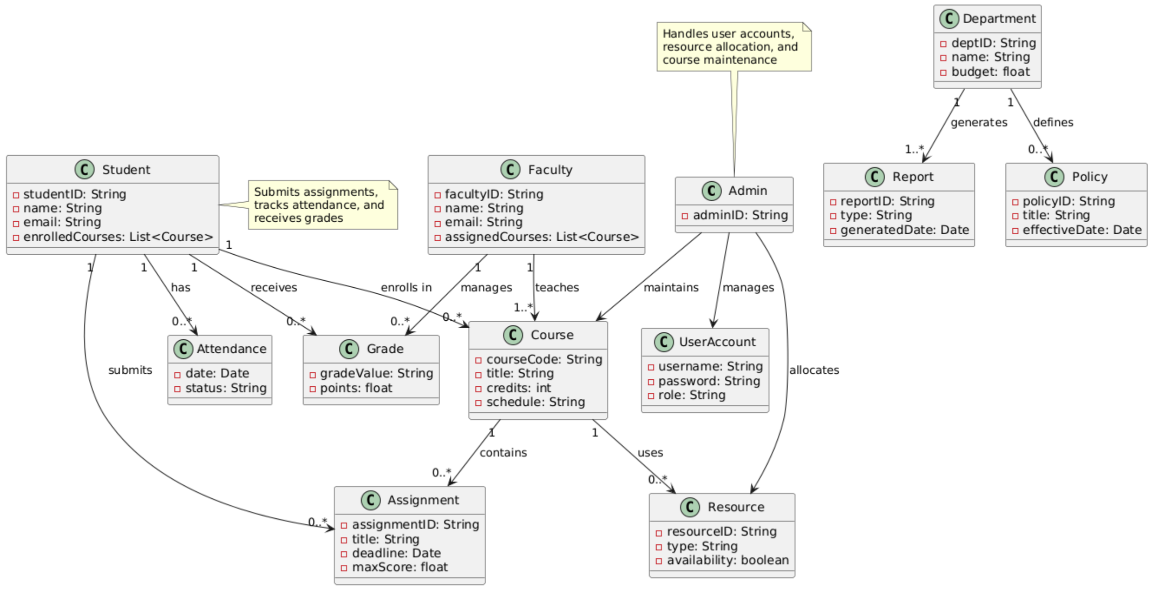
Owner: Team 18

Priority: Medium

Domain Model

Domain Model Diagram

Figure 5. Domain Model for UDIMS



Description:
The domain model identifies the key entities (classes), their attributes, and the relationships (associations with multiplicities) between them. It serves as the conceptual backbone for your system's data design.

Class	Key Attributes	Notes
Student	studentID: String name: String email: String enrolledCourses: List<Course>	Submits assignments, tracks attendance, receives grades.
Faculty	facultyID: String name: String email: String assignedCourses: List<Course>	Manages and teaches courses; records grades & attendance.
Course	courseCode: String title: String	Central entity linking students, faculty, assignments.

Class	Key Attributes	Notes
	credits: int schedule: String	
Assignment	assignmentID: String title: String deadline: Date maxScore: float	Belongs to a course; submitted by students.
Grade	gradeValue: String points: float	Awarded to Student for an Assignment; linked to Course.
Attendance	date: Date status: String	Records student presence per course session.
Resource	resourceID: String type: String availability: boolean	E.g. lab equipment; allocated by Admin.
UserAccount	username: String password: String role: String	Authentication credentials for all actors.
Admin	adminID: String	Manages user accounts, resources, course maintenance.
Report	reportID: String type: String generatedDate: Date	Generated for performance metrics and audits.
Policy	policyID: String title: String effectiveDate: Date	Defines department rules; created by Dept Head.
Department	deptID: String name: String budget: float	Oversees courses, reports, and policies.

Relationships and Multiplicities

- **Student – Course**:
  – A Student *enrolls in* 0..* Courses; each Course has 1..* Students.
- **Faculty – Course**:
  – A Faculty *teaches* 1..* Courses; each Course is *managed by* 1 Faculty.
- **Course – Assignment**:
  – A Course *contains* 0..* Assignments; each Assignment belongs to exactly 1 Course.
- **Student – Assignment – Grade**:
  – A Student *submits* 0..* Assignments; each submission yields 0..* Grade records.
- **Student – Attendance**:
  – A Student *has* 0..* Attendance entries; each entry ties to exactly 1 Student.
- **Course – Resource**:
  – A Course *uses* 0..* Resources; each Resource may support 0..* Courses.
- **Admin – UserAccount / Resource**:
  – Admin *allocates* Resources and *maintains* UserAccounts.
- **Department – Report / Policy**:
  – Department *generates* 1..* Reports; *defines* 0..* Policies.

Narrative Summary

The domain model lays out the static structure of UDIMS. At its center is the **Course** class, which connects **Students**, **Faculty**, **Assignments**, **Grades**, and **Resources**. **Students** enroll in courses, submit assignments, attend sessions, and receive grades. **Faculty** members manage courses by creating assignments, taking attendance, and entering grades. The **Admin** actor maintains the system's **UserAccounts** and allocates **Resources** needed for courses. **Department Heads** generate **Reports** for analytics and establish **Policies** to govern departmental operations. This model ensures clear data ownership, enforces multiplicity constraints, and guides the subsequent database schema design.

Sequence Diagram

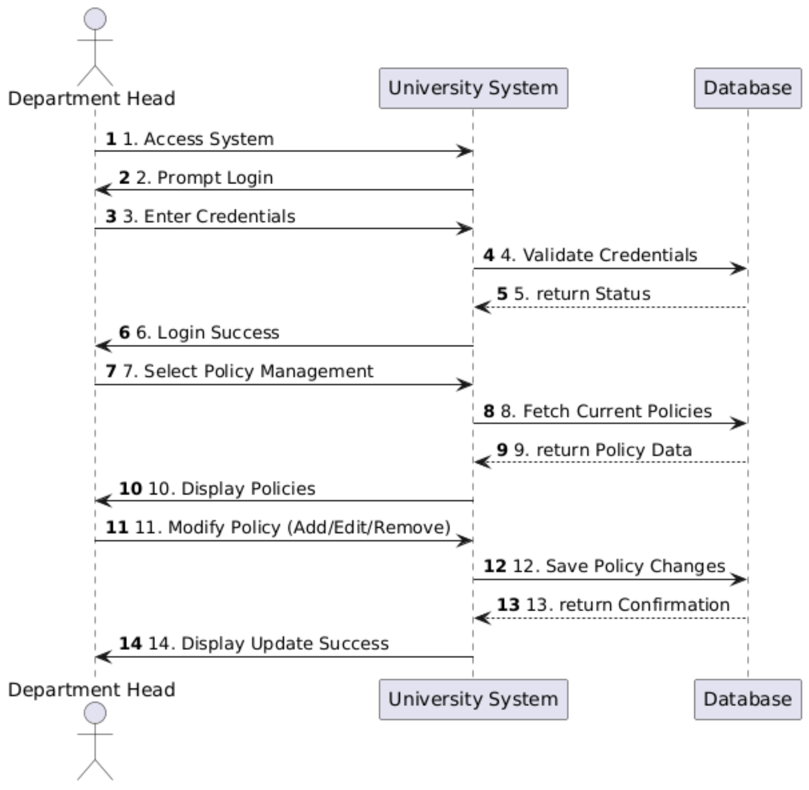
1. Policy Management

Actors: Department Head, University System, Database
Flow:

1. Department Head logs in and selects "Policy Management."
2. System fetches current policies from the database.
3. Department Head modifies policies (add/edit/remove).
4. System saves changes and confirms success.
   Connection to SRS: Aligns with *Functional Requirement 2* (Admin can manage policies).

Diagram Reference:

Policy Management Sequence Diagram



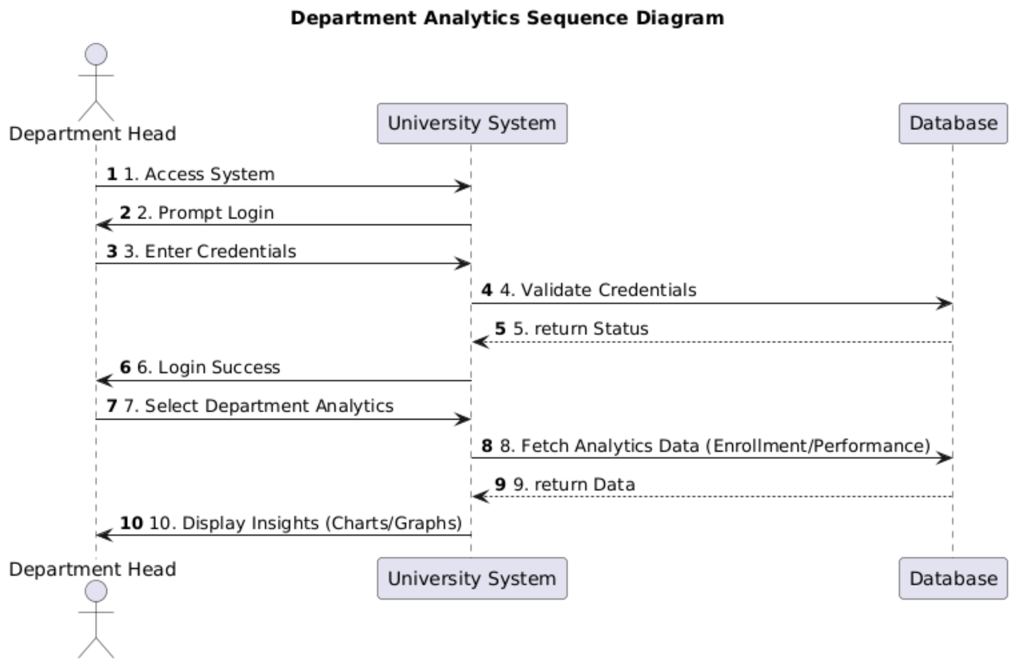
Explanation: Highlights policy retrieval, modification, and confirmation steps.

2. Department Analytics

Actors: Department Head, University System, Database
Flow:

1. Department Head logs in and selects "Analytics."
2. System fetches enrollment/performance data from the database.
3. Data is displayed as charts/graphs.
   Connection to SRS: Supports *Non-Functional Requirement* (Performance: <2s response time).

Diagram Reference:



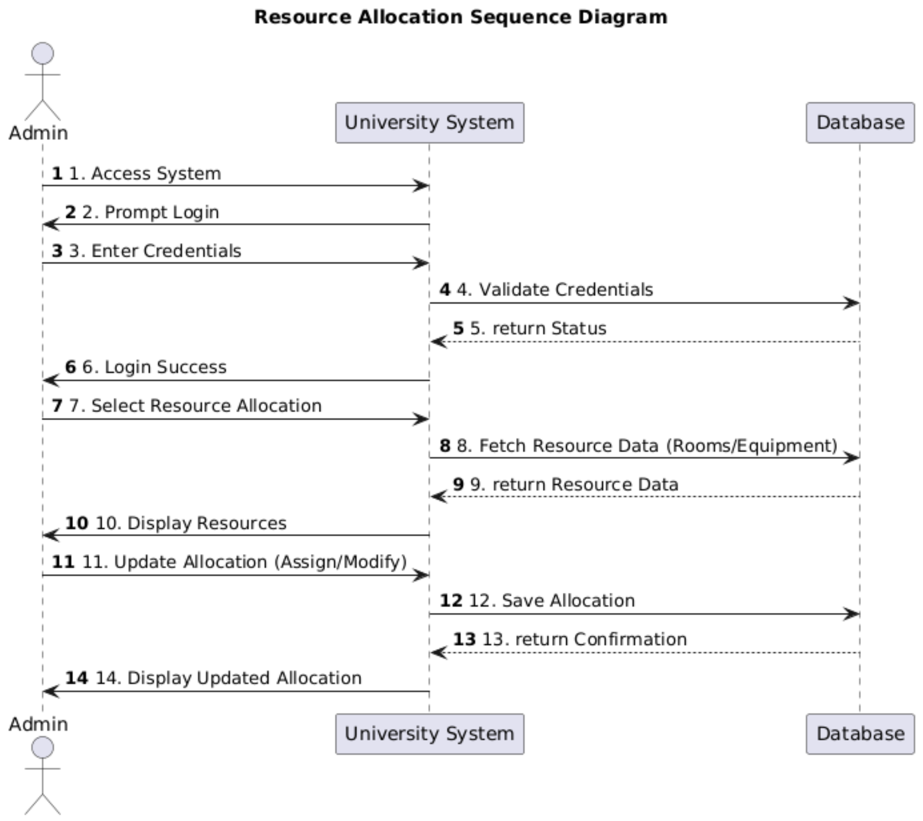
Explanation: Focuses on data retrieval and visualization.

3. Resource Allocation

Actors: Admin, University System, Database
Flow:

1. Admin logs in and selects "Resource Allocation."
2. System fetches resource data (rooms/equipment).
3. Admin updates allocations, and changes are saved.
   Connection to SRS: Maps to *Product Function* (Resource tracking).

Diagram Reference:


Resource Allocation Sequence Diagram
Admin
University System
Database
1 1. Access System
2 2. Prompt Login
3 3. Enter Credentials
4 4. Validate Credentials
5 5. return Status
6 6. Login Success
7 7. Select Resource Allocation
8 8. Fetch Resource Data (Rooms/Equipment)
9 9. return Resource Data
10 10. Display Resources
11 11. Update Allocation (Assign/Modify)
12 12. Save Allocation
13 13. return Confirmation
14 14. Display Updated Allocation
Admin
University System
Database

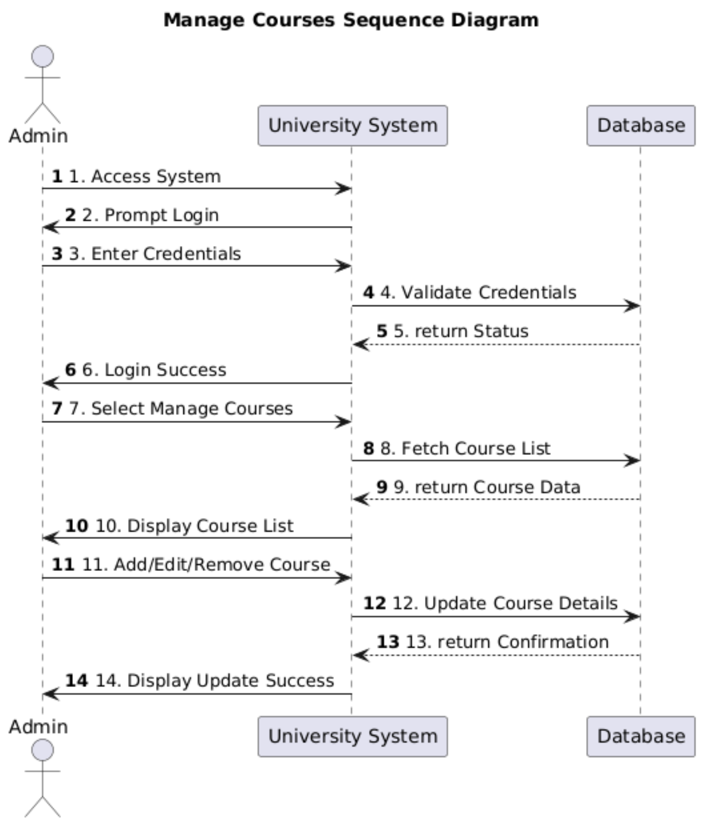
Explanation: Emphasizes real-time updates and confirmation.

4. Course Management

Actors: Admin, University System, Database
Flow:

1. Admin logs in and selects "Manage Courses."
2. System retrieves course list.
3. Admin adds/edits/removes courses; changes are saved.
   Connection to SRS: Validates *Functional Requirement 2* (CRUD operations).

Diagram Reference:


Manage Courses Sequence Diagram
Admin
University System
Database
1 1. Access System
2 2. Prompt Login
3 3. Enter Credentials
4 4. Validate Credentials
5 5. return Status
6 6. Login Success
7 7. Select Manage Courses
8 8. Fetch Course List
9 9. return Course Data
10 10. Display Course List
11 11. Add/Edit/Remove Course
12 12. Update Course Details
13 13. return Confirmation
14 14. Display Update Success
Admin
University System
Database

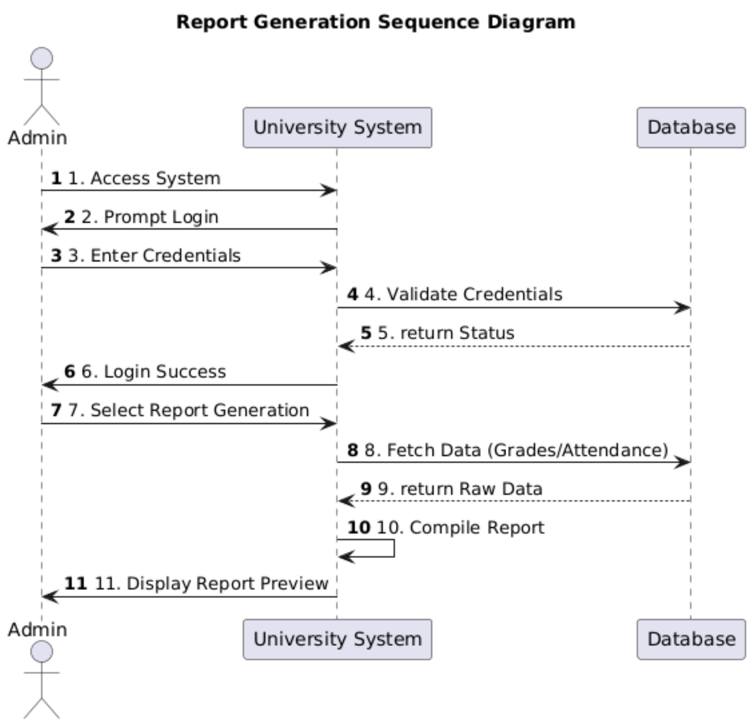
Explanation: Demonstrates course lifecycle management.

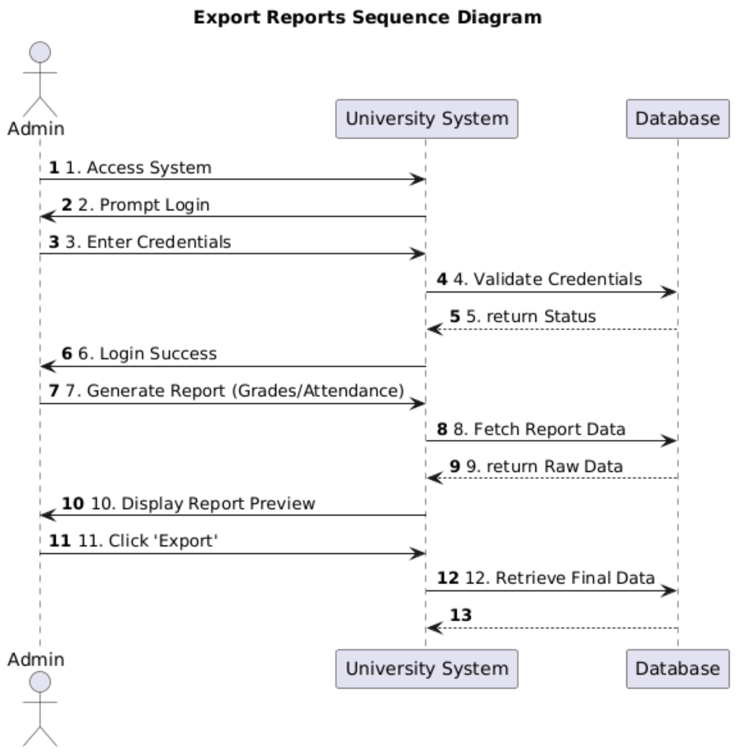
5. Report Generation & Export

Actors: Admin, University System, Database
Flow:

1. Admin logs in and selects "Generate Report."
2. System compiles data (grades/attendance) and displays a preview.
3. Admin exports the report in desired format.
   Connection to SRS: Ties to *External Interface Requirements* (UI for reports).

Diagram Reference:





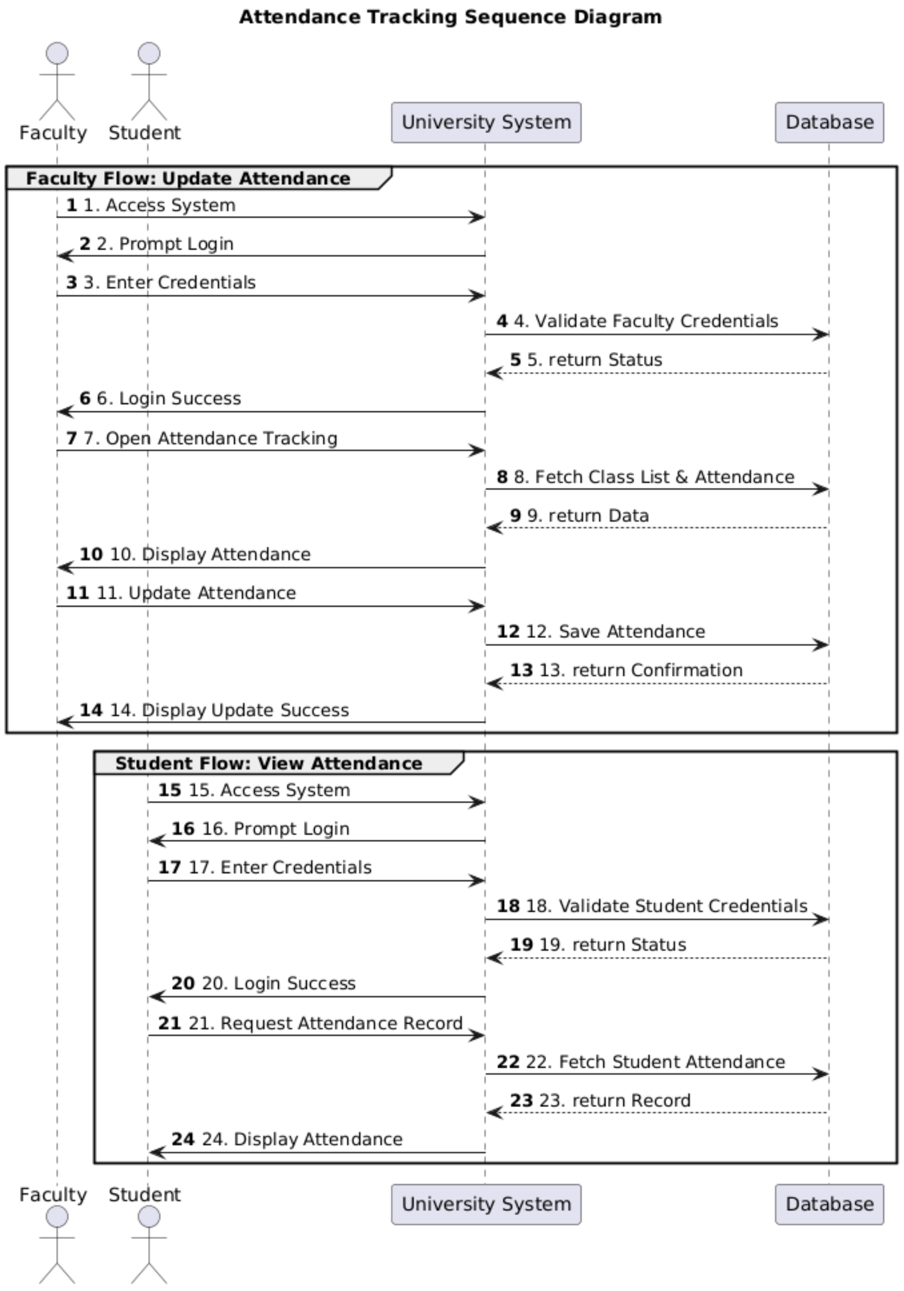
Explanation: Shows end-to-end report creation and export.

6. Attendance Tracking

Actors: Faculty, Student, University System, Database
Flow:

- **Faculty:** Logs in, updates attendance, and saves changes.
- **Student:** Logs in and views attendance records.
  Connection to SRS: Supports *User Characteristics* (Dual roles for faculty/students).

Diagram Reference:


Attendance Tracking Sequence Diagram
Faculty
Student
University System
Database
Faculty Flow: Update Attendance
1 1. Access System
2 2. Prompt Login
3 3. Enter Credentials
4 4. Validate Faculty Credentials
5 5. return Status
6 6. Login Success
7 7. Open Attendance Tracking
8 8. Fetch Class List & Attendance
9 9. return Data
10 10. Display Attendance
11 11. Update Attendance
12 12. Save Attendance
13 13. return Confirmation
14 14. Display Update Success
Student Flow: View Attendance
15 15. Access System
16 16. Prompt Login
17 17. Enter Credentials
18 18. Validate Student Credentials
19 19. return Status
20 20. Login Success
21 21. Request Attendance Record
22 22. Fetch Student Attendance
23 23. return Record
24 24. Display Attendance
Faculty
Student
University System
Database

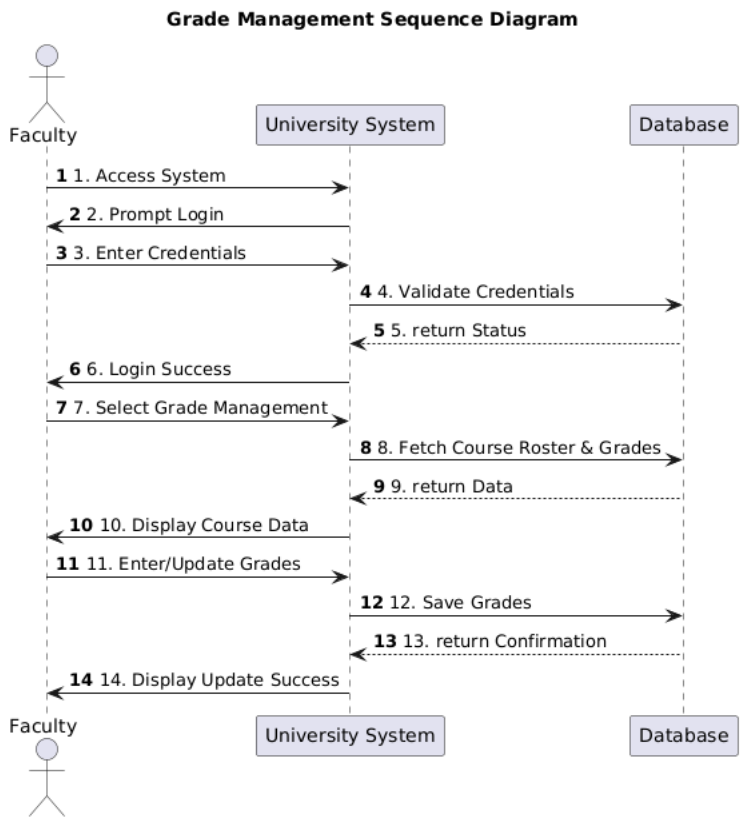
Explanation: Dual flows for updating and viewing attendance.

7. Grade Management

Actors: Faculty, University System, Database
Flow:

1. Faculty logs in and selects "Grade Management."
2. System fetches course roster and grades.
3. Faculty updates grades, which are saved to the database.
   Connection to SRS: Aligns with *Security Requirements* (Role-based access).

Diagram Reference:


Grade Management Sequence Diagram
Faculty
University System
Database
1 1. Access System
2 2. Prompt Login
3 3. Enter Credentials
4 4. Validate Credentials
5 5. return Status
6 6. Login Success
7 7. Select Grade Management
8 8. Fetch Course Roster & Grades
9 9. return Data
10 10. Display Course Data
11 11. Enter/Update Grades
12 12. Save Grades
13 13. return Confirmation
14 14. Display Update Success
Faculty
University System
Database

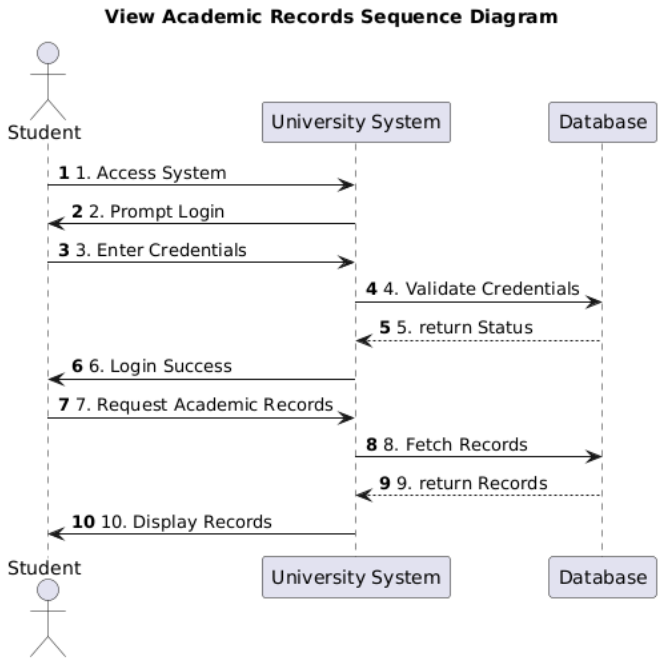
Explanation: Focuses on secure grade entry and storage.

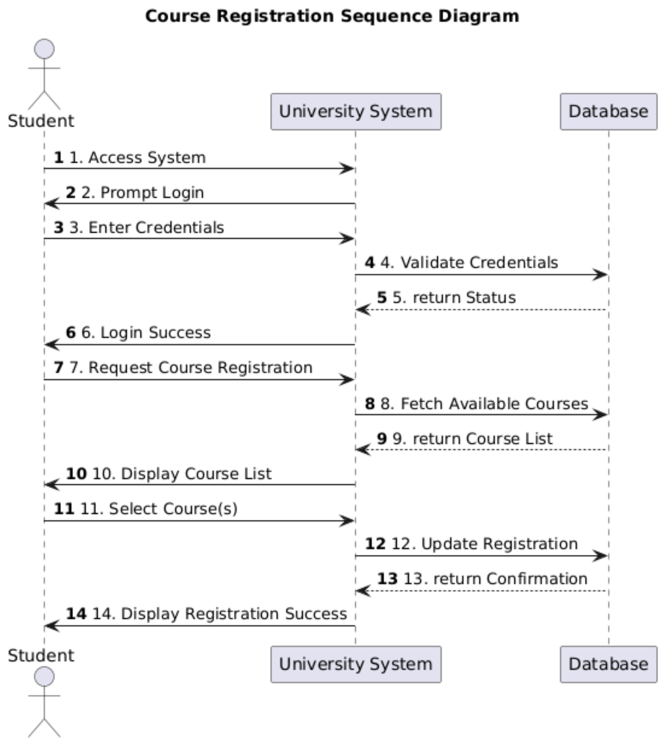
8. Academic Records & Course Registration

Actors: Student, University System, Database
Flow:

- **Academic Records:** Student views their records post-login.
- **Course Registration:** Student selects courses, and the system updates registrations.
  Connection to SRS: Validates *User Characteristics* (Student privileges).

Diagram Reference:




Course Registration Sequence Diagram
Student
University System
Database
1 1. Access System
2 2. Prompt Login
3 3. Enter Credentials
4 4. Validate Credentials
5 5. return Status
6 6. Login Success
7 7. Request Course Registration
8 8. Fetch Available Courses
9 9. return Course List
10 10. Display Course List
11 11. Select Course(s)
12 12. Update Registration
13 13. return Confirmation
14 14. Display Registration Success
Student
University System
Database

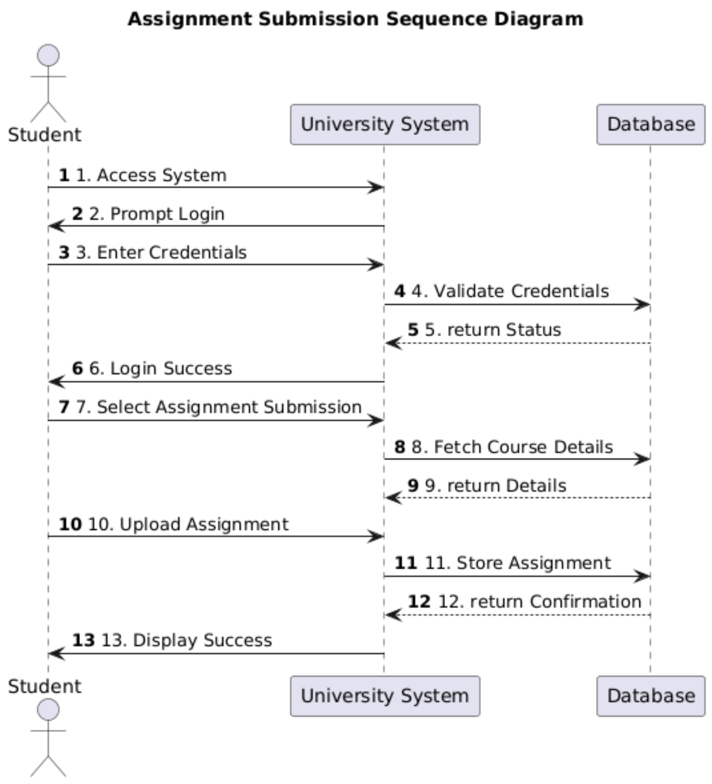
Explanation: Streamlines student self-service actions.

9. Assignment Submission

Actors: Student, University System, Database
Flow:

1. Student logs in and submits an assignment.
2. System validates and stores the submission.
3. Confirmation is displayed.
   Connection to SRS: Maps to *Functional Requirement 3* (Student actions).

Diagram Reference:



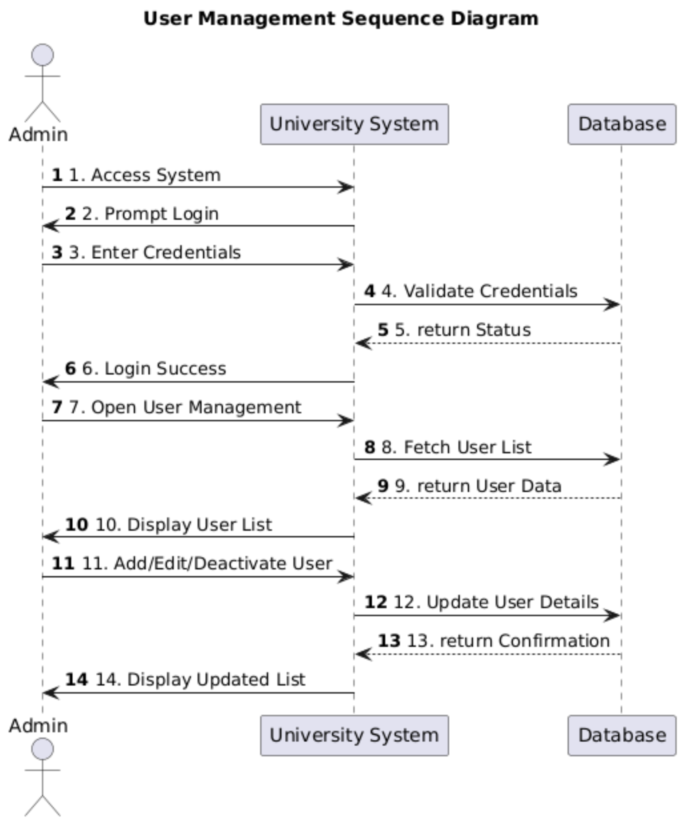
Explanation: Highlights submission validation and storage.

10. User Management (Not Shown)

Actors: Admin, University System, Database
Flow:

1. Admin logs in and accesses "User Management."
2. System retrieves user data.
3. Admin adds/edits/deletes users; changes are saved.
   Connection to SRS: Critical for *Functional Requirement 2* (Admin roles).

Diagram Reference:



Explanation: Highlights user management.

Project Setup

Project Setup Requirements:

Technologies Used:

Frontend:

- React (v18.3.1)
- TypeScript
- Vite
- TailwindCSS
- Ant Design
- Zustand (state management)
- React Router DOM

Backend:

- Python
- Flask (v2.3.3)
- SQLAlchemy
- JWT Authentication
- SQLite database

Installation Steps:

- Clone repo
- Install dependencies:
- Frontend: npm install
- Backend:
- cd Backend
- pip install -r requirements.txt

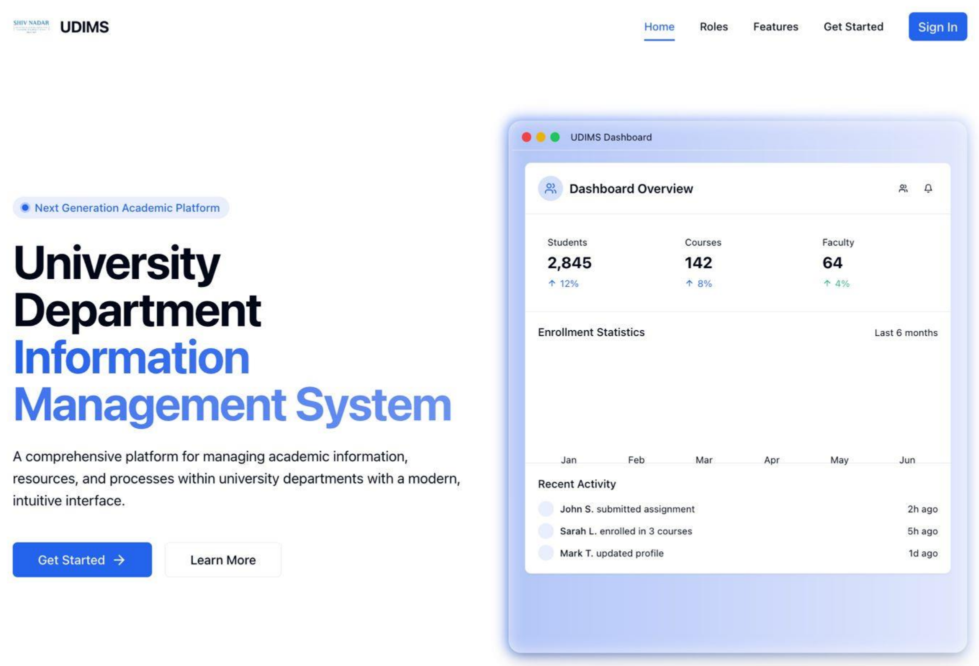
Start servers:

- Frontend: npm run dev
- Backend: cd Backend && python app.py
- System Requirements:
- Node.js (v16+)
- Python (v3.8+)
- NPM or Yarn
- 4GB RAM minimum
- 1GB disk space

Results

This section showcases key interfaces of the **University Department Information Management System (UDIMS)**, demonstrating its functionality and alignment with the SRS.

Figure 1: UDIMS Dashboard



Next Generation Academic Platform

University Department Information Management System

A comprehensive platform for managing academic information, resources, and processes within university departments with a modern, intuitive interface.

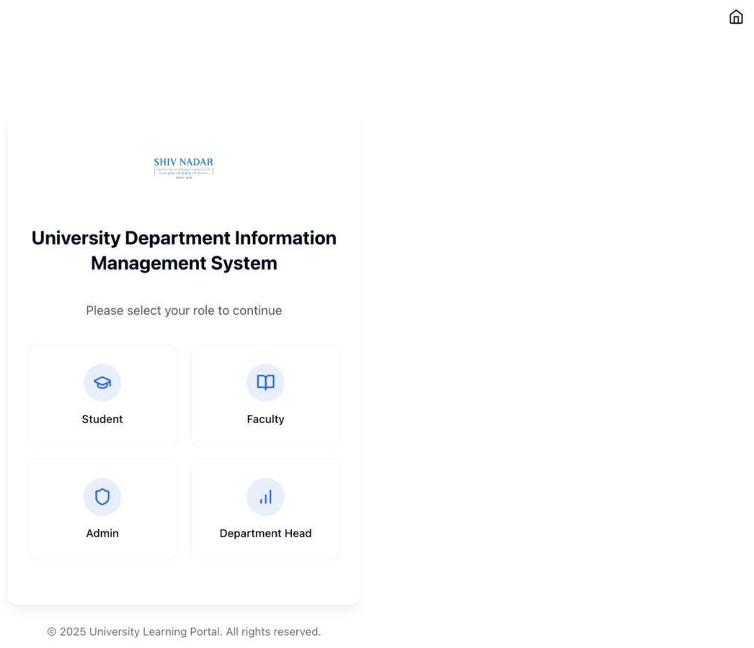
John S. submitted assignment	2h ago
Sarah L. enrolled in 3 courses	5h ago
Mark T. updated profile	1d ago

Description:

- **Dashboard Layout:** Centralized view displaying critical metrics:
    - **Students (2,845):** Real-time enrollment count with trend analysis (+12% growth).
    - **Courses (142):** Tracked course inventory with percentage increase.
    - **Faculty (64):** Faculty engagement statistics.
- **Enrollment Trends:** Line chart visualizes enrollment data for the last 6 months.
- **Recent Activity:** Logs user actions (e.g., assignment submissions, course enrollments).

Relevance: Validates *Functional Requirement 3.1* (Real-time data access) and *Non-Functional Requirement 3.2* (Performance metrics).

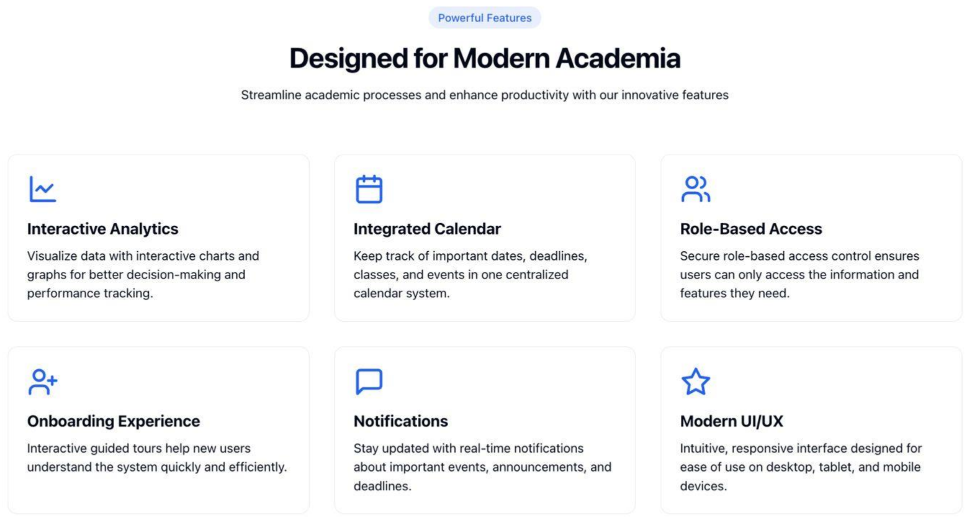
Figure 2: Role Selection Screen


SHIV NADAR
University Department Information Management System
Please select your role to continue
Student
Faculty
Admin
Department Head
© 2025 University Learning Portal. All rights reserved.

Description:

- **Role Options:** Users select their role (Student, Faculty, Admin, Department Head) to access tailored features.
- **Security Alignment:** Implements *SRS 3.3* (Role-Based Access Control) to restrict unauthorized actions.
  Relevance: Supports *User Characteristics* defined in Section 4.2.3 (Admins manage resources; Students submit assignments).

Figure 3: Key Features Overview



Powerful Features

Designed for Modern Academia

Streamline academic processes and enhance productivity with our innovative features

Interactive Analytics

Visualize data with interactive charts and graphs for better decision-making and performance tracking.

Integrated Calendar

Keep track of important dates, deadlines, classes, and events in one centralized calendar system.

Role-Based Access

Secure role-based access control ensures users can only access the information and features they need.

Onboarding Experience

Interactive guided tours help new users understand the system quickly and efficiently.

Notifications

Stay updated with real-time notifications about important events, announcements, and deadlines.

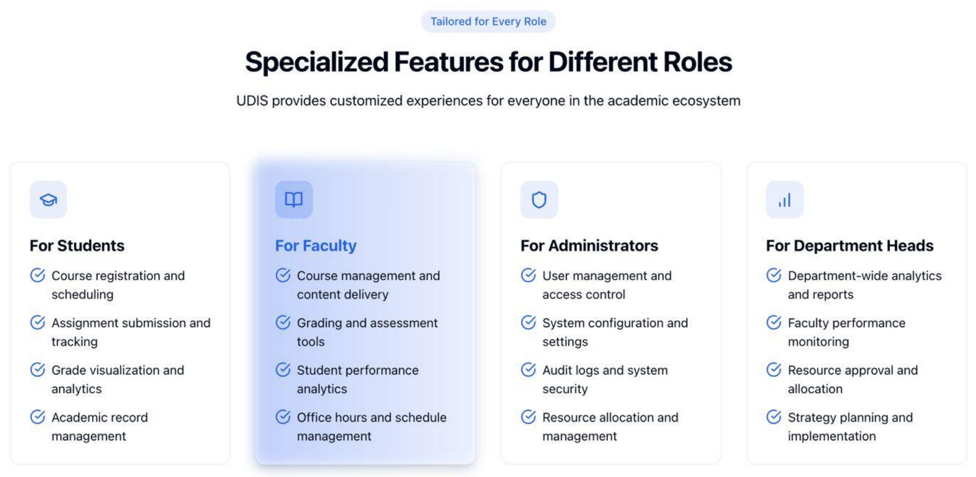
Modern UI/UX

Intuitive, responsive interface designed for ease of use on desktop, tablet, and mobile devices.

Description:

- **Interactive Analytics:** Enables data-driven decision-making (e.g., enrollment trends).
- **Integrated Calendar:** Centralizes deadlines, classes, and events, addressing *SRS 2.2* (Product Functions).
- **Notifications:** Real-time alerts for deadlines/updates, fulfilling *Non-Functional Requirement 3.2* (Usability).
  Relevance: Demonstrates adherence to *External Interface Requirements* (Modern UI/UX).

Figure 4: Role-Specific Features


Tailored for Every Role

Specialized Features for Different Roles

UDIS provides customized experiences for everyone in the academic ecosystem

For Students
- Course registration and scheduling
- Assignment submission and tracking
- Grade visualization and analytics
- Academic record management

For Faculty
- Course management and content delivery
- Grading and assessment tools
- Student performance analytics
- Office hours and schedule management

For Administrators
- User management and access control
- System configuration and settings
- Audit logs and system security
- Resource allocation and management

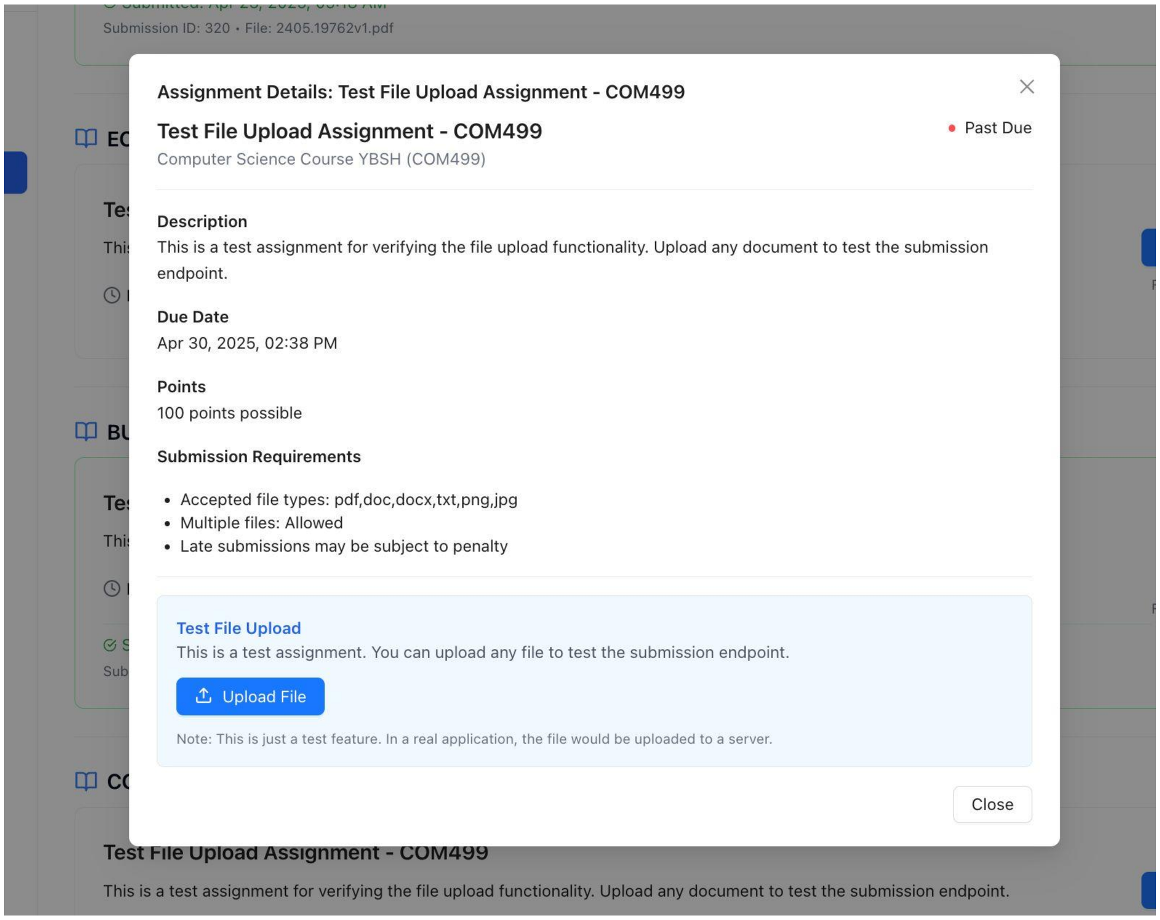
For Department Heads
- Department-wide analytics and reports
- Faculty performance monitoring
- Resource approval and allocation
- Strategy planning and implementation

Description:

- **Students:** Course registration, assignment submission, grade tracking (*Use Case: Course Registration*).
- **Faculty:** Grading tools, performance analytics (*Use Case: Grade Management*).
- **Admins:** User management, resource allocation (*Use Case: Resource Allocation*).
- **Department Heads:** Strategic planning via department-wide reports (*Use Case: Policy Management*).
  Relevance: Validates *Functional Requirements* (Section 4.3.1) and aligns with actor-specific workflows.

Student Portal

Figure 5: Assignment Submission Interface



Submission ID: 320 • File: 2405.19762v1.pdf

Assignment Details: Test File Upload Assignment - COM499

Test File Upload Assignment - COM499

● Past Due

Computer Science Course YBSH (COM499)

Description

This is a test assignment for verifying the file upload functionality. Upload any document to test the submission endpoint.

Due Date

Apr 30, 2025, 02:38 PM

Points

100 points possible

Submission Requirements

- Accepted file types: pdf,doc,docx,txt,png,jpg
- Multiple files: Allowed
- Late submissions may be subject to penalty

Test File Upload

This is a test assignment. You can upload any file to test the submission endpoint.

Upload File

Note: This is just a test feature. In a real application, the file would be uploaded to a server.

Close

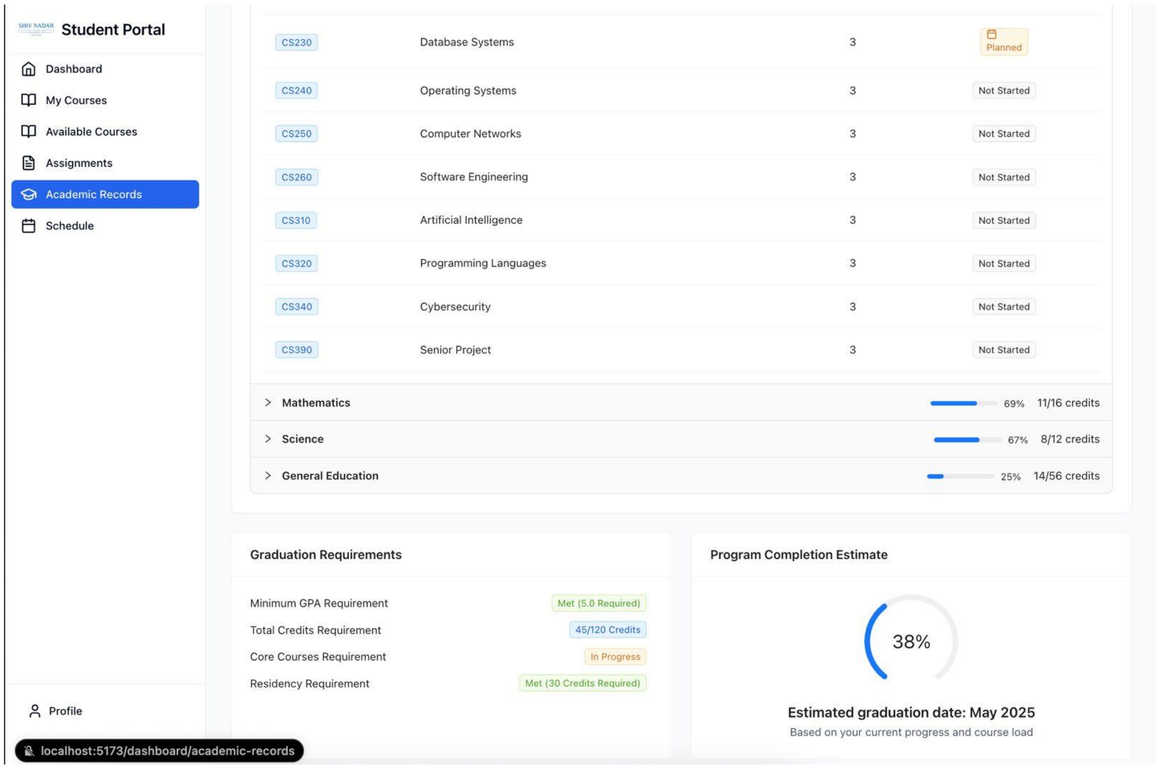
Test File Upload Assignment - COM499

This is a test assignment for verifying the file upload functionality. Upload any document to test the submission endpoint.

Description:

- **Assignment Details:** Displays course code (COM499), due date (Apr 30, 2025), points (100), and accepted file types (PDF, DOC, PNG, etc.).
- **Submission Workflow:** Students can upload multiple files to test the submission endpoint (*Use Case: Assignment Submission*).
  Relevance: Validates *Functional Requirement 3.1* (Student actions: assignment upload) and *Non-Functional Requirement 3.2* (Usability with clear instructions).

Figure 6: Academic Progress Dashboard



CS230	Database Systems	3	Planned
CS240	Operating Systems	3	Not Started
CS250	Computer Networks	3	Not Started
CS260	Software Engineering	3	Not Started
CS310	Artificial Intelligence	3	Not Started
CS320	Programming Languages	3	Not Started
CS340	Cybersecurity	3	Not Started
CS390	Senior Project	3	Not Started

Mathematics	69%	11/16 credits
Science	67%	8/12 credits
General Education	25%	14/56 credits

Graduation Requirements

Minimum GPA Requirement	Met (5.0 Required)
Total Credits Requirement	45/120 Credits
Core Courses Requirement	In Progress
Residency Requirement	Met (30 Credits Required)

Program Completion Estimate

38%

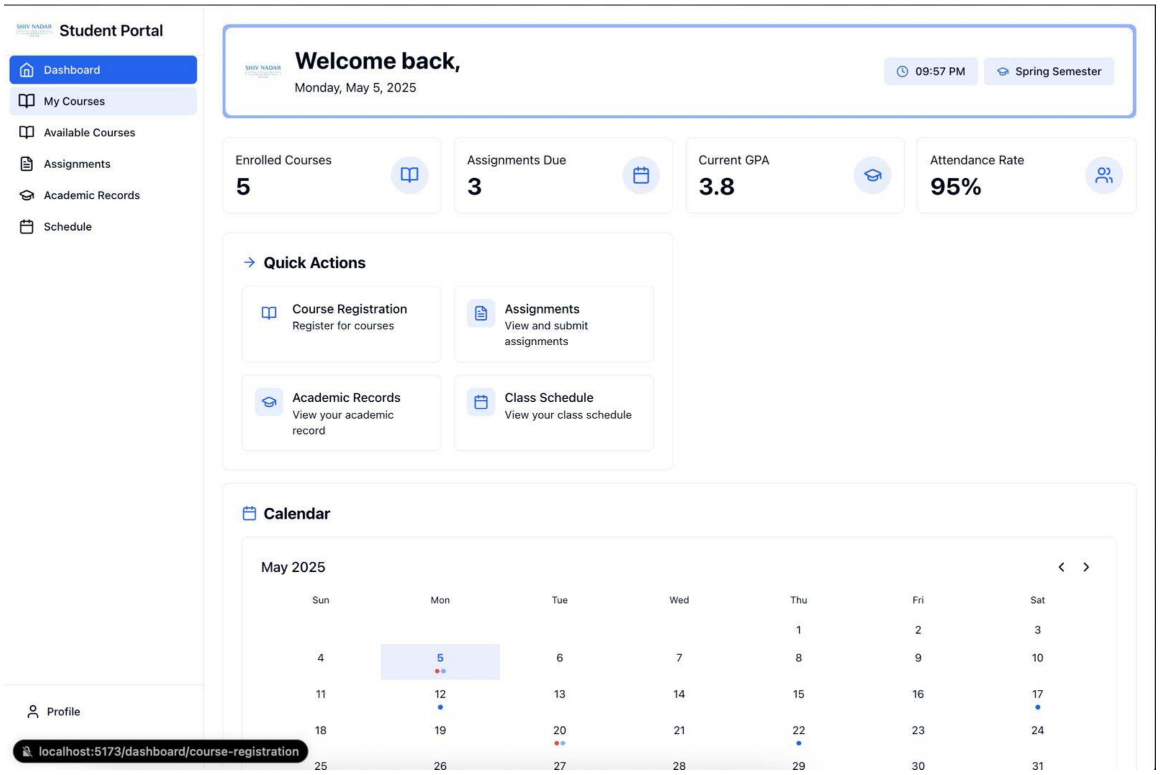
Estimated graduation date: May 2025

Based on your current progress and course load

Description:

- **Course Schedule:** Lists enrolled courses (e.g., Database Systems, Operating Systems) with status indicators (Planned/Not Started).
- **Graduation Requirements:** Tracks GPA (3.8), credits (45/120), and residency status.
- **Progress Visualization:** Progress bar (38%) and estimated graduation date (May 2025).
  Relevance: Aligns with *Functional Requirement 3.3* (Academic record management) and *SRS 2.2* (Product Functions: tracking student progress).

Figure 7: Student Dashboard Overview

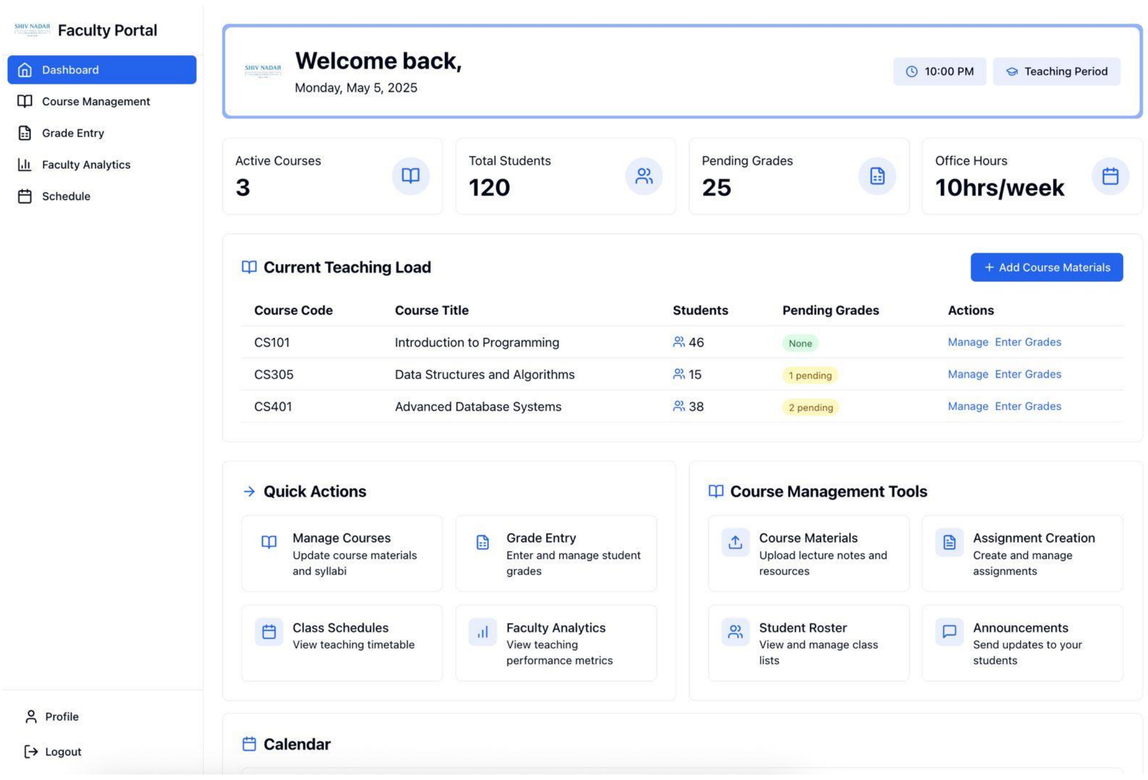


Description:

- **Summary Metrics:** Displays enrolled courses (5), assignments due (3), current GPA (3.8), and attendance rate (95%).
- **Quick Actions:** Direct links for course registration, assignments, and academic records (*Use Case: Course Registration*).
- **Integrated Calendar:** Centralized view of May 2025 schedule with key dates. **Relevance:** Demonstrates *External Interface Requirements* (UI/UX: intuitive navigation) and supports *User Characteristics* (Student self-service).

Faculty Portal

Figure 8: Faculty Dashboard Overview



Welcome back,
Monday, May 5, 2025

Active Courses 3 | Total Students 120 | Pending Grades 25 | Office Hours 10hrs/week

Current Teaching Load

Course Code	Course Title	Students	Pending Grades	Actions
CS101	Introduction to Programming	46	None	Manage Enter Grades
CS305	Data Structures and Algorithms	15	1 pending	Manage Enter Grades
CS401	Advanced Database Systems	38	2 pending	Manage Enter Grades

Quick Actions

- Manage Courses — Update course materials and syllabi
- Grade Entry — Enter and manage student grades
- Class Schedules — View teaching timetable
- Faculty Analytics — View teaching performance metrics

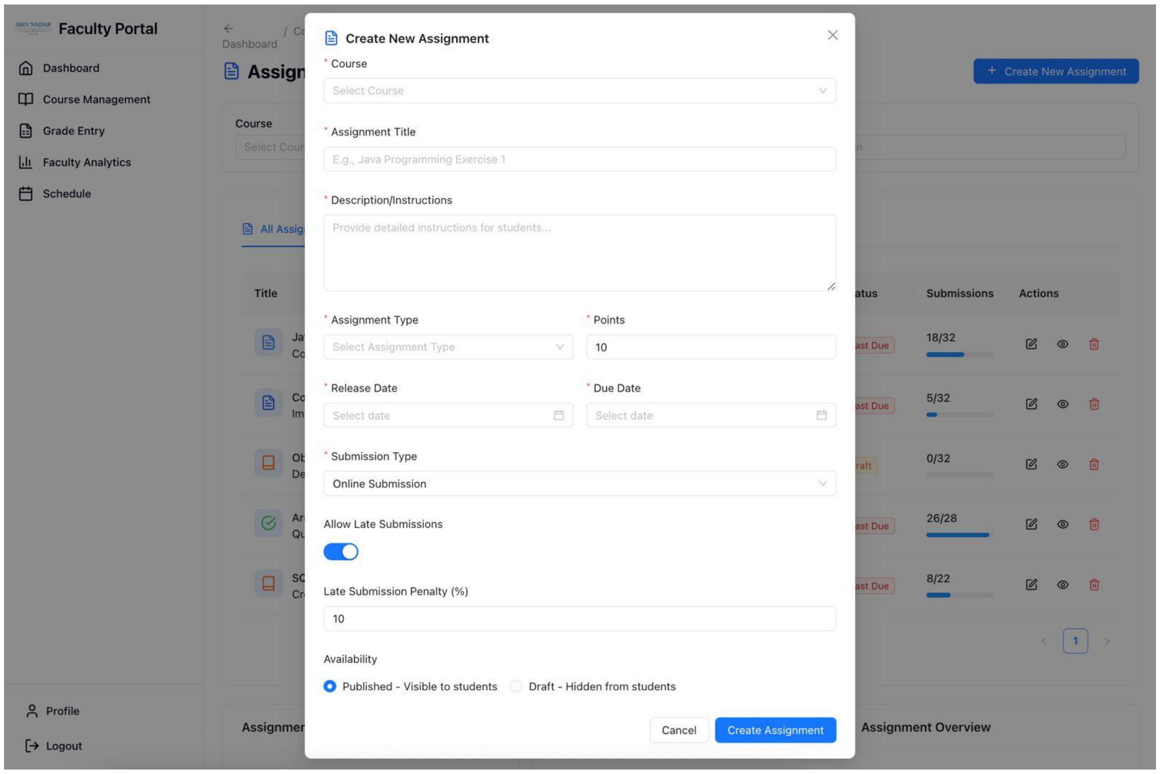
Course Management Tools

- Course Materials — Upload lecture notes and resources
- Assignment Creation — Create and manage assignments
- Student Roster — View and manage class lists
- Announcements — Send updates to your students

Description:

- **Key Metrics:** Displays total students (120), pending grades (25), and office hours (10hrs/week).
- **Course Management:** Lists active courses (e.g., CS101, CS305) with pending grades and quick actions (*Use Case: Grade Management*).
- **Quick Actions:** Direct access to grade entry, assignment creation, and analytics. **Relevance:** Validates *Functional Requirement 3.1* (Faculty actions: grade entry, course management) and *SRS 2.2* (Product Functions: real-time data access).

Figure 9: Assignment Creation Interface



Create New Assignment

* Course: Select Course

* Assignment Title: E.g., Java Programming Exercise 1

* Description/Instructions: Provide detailed instructions for students...

* Assignment Type: Select Assignment Type

* Points: 10

* Release Date: Select date

* Due Date: Select date

* Submission Type: Online Submission

Allow Late Submissions: [✓]

Late Submission Penalty (%): 10

Availability: [✓ Published - Visible to students] Draft - Hidden from students

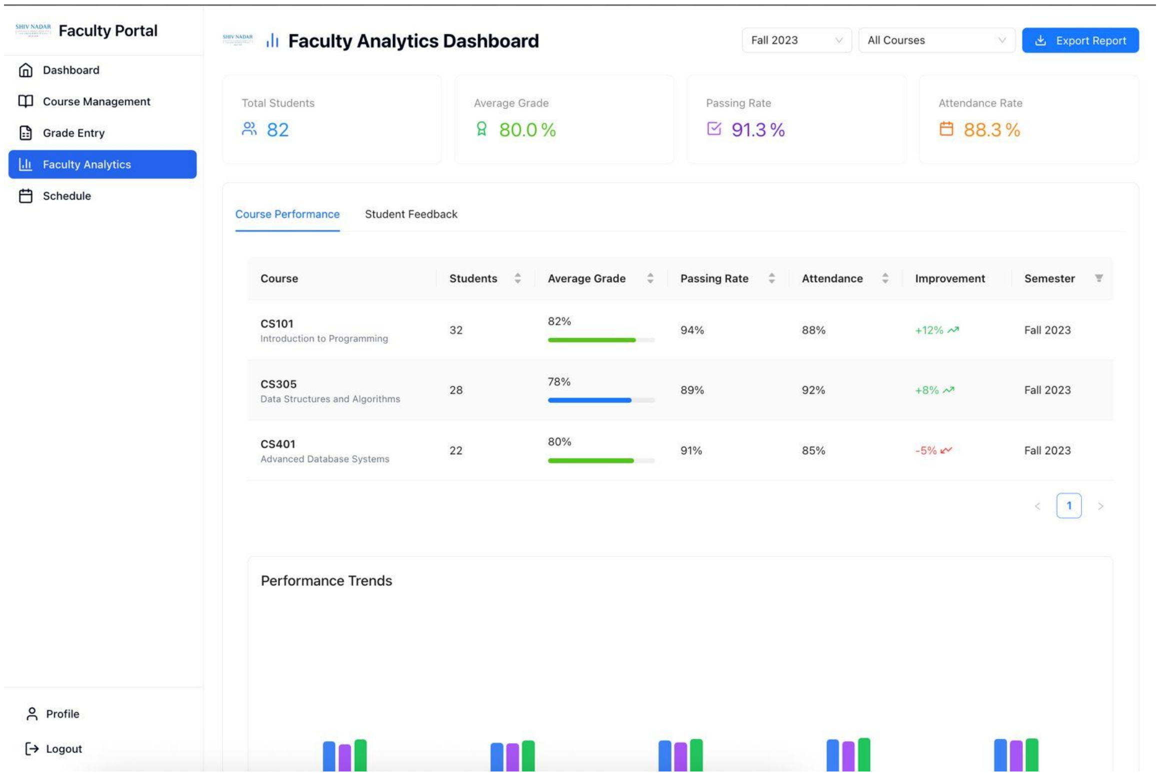
Cancel | Create Assignment

Submissions
18/32
5/32
0/32
26/28
8/22

Description:

- **Assignment Details:** Faculty can define title, type (e.g., Java Programming Exercise 1), points (10), due dates, and late penalties (10%).
- **Submission Settings:** Supports online submissions with file type restrictions (PDF, DOCX, etc.) and draft/publish visibility.
  Relevance: Aligns with *Functional Requirement 3.2* (Non-functional: usability) and *Use Case: Assignment Submission*.

Figure 10: Faculty Analytics Dashboard



Total Students 82

Average Grade 80.0 %

Passing Rate 91.3 %

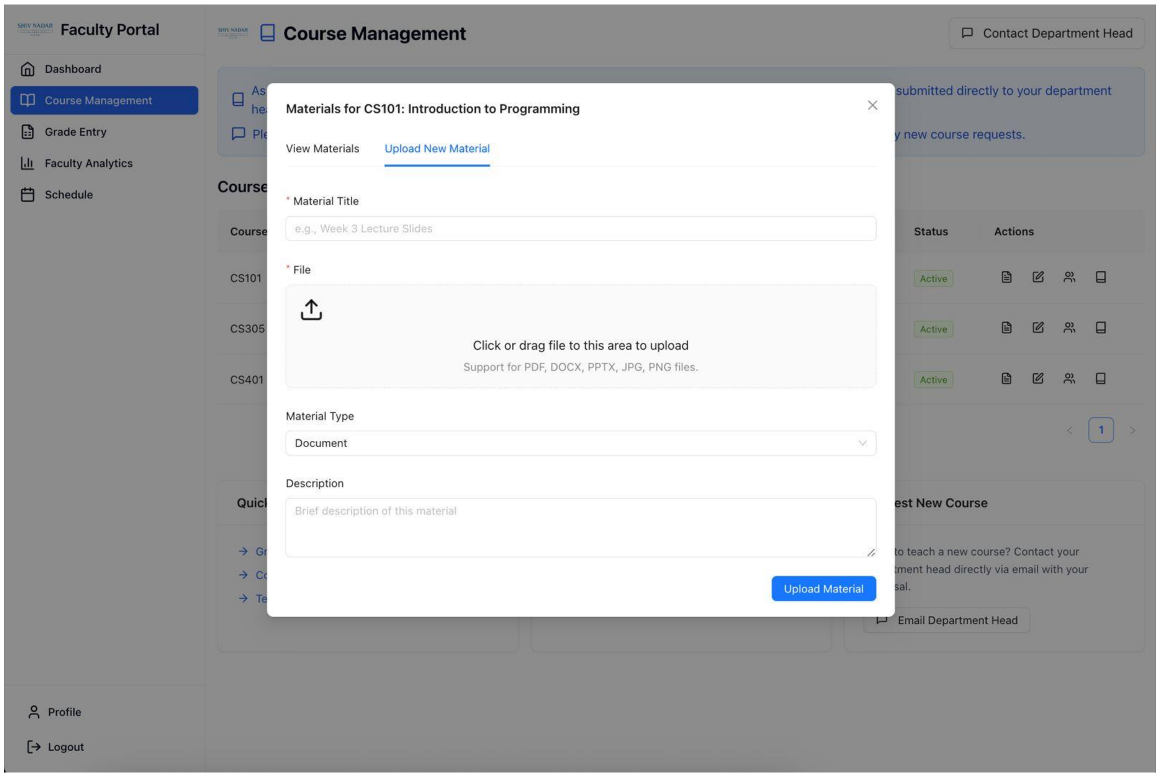
Attendance Rate 88.3 %

Course	Students	Average Grade	Passing Rate	Attendance	Improvement	Semester
CS101 Introduction to Programming	32	82%	94%	88%	+12%	Fall 2023
CS305 Data Structures and Algorithms	28	78%	89%	92%	+8%	Fall 2023
CS401 Advanced Database Systems	22	80%	91%	85%	-5%	Fall 2023

Description:

- **Course Performance:** Tracks average grades (80%), passing rates (91.3%), and attendance (88.3%) across courses like CS101 and CS305.
- **Trend Analysis:** Visualizes semester-wise improvement (e.g., +12% in CS101) and student feedback.
  Relevance: Supports *Non-Functional Requirement 3.2* (Performance: <2s response time for analytics) and *Product Perspective* (data-driven decision-making).

Figure 11: Course Management Interface


Faculty Portal

Course Management

Contact Department Head

Materials for CS101: Introduction to Programming

View Materials | Upload New Material

* Material Title

e.g., Week 3 Lecture Slides

* File

Click or drag file to this area to upload

Support for PDF, DOCX, PPTX, JPG, PNG files.

Material Type

Document

Description

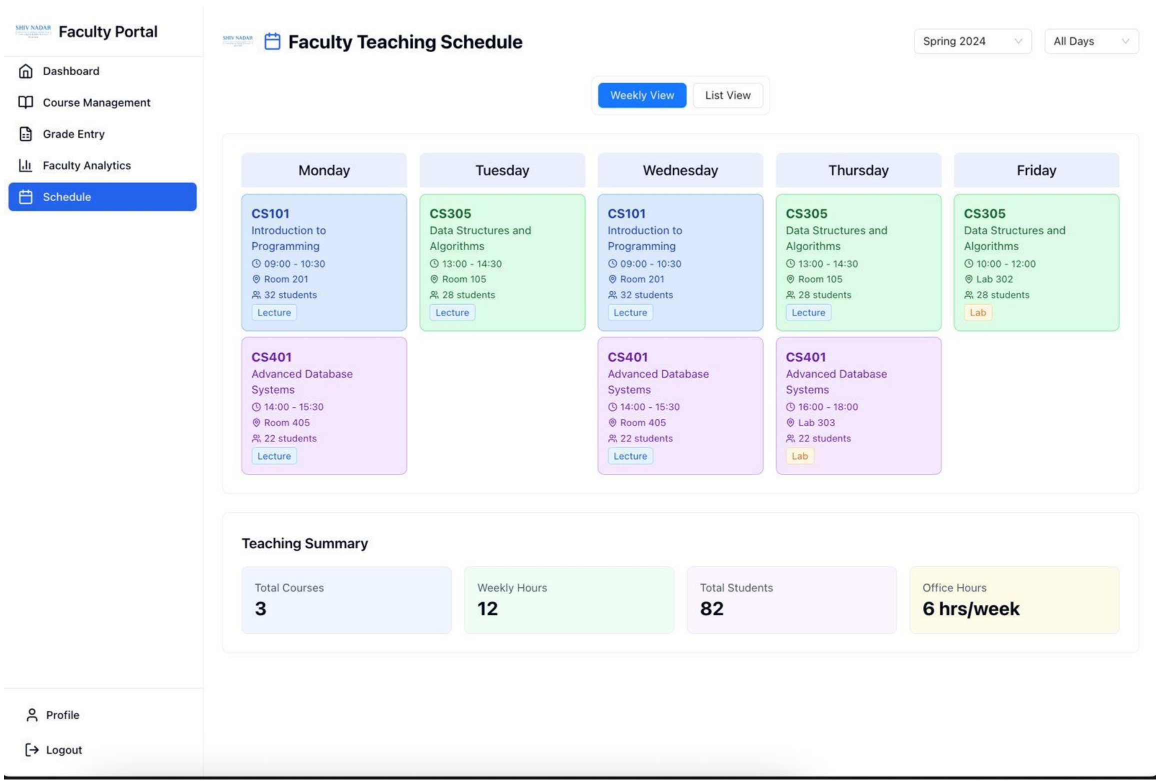
Brief description of this material

Upload Material

Description:

- **Material Upload:** Faculty can upload lecture slides (PDF, PPTX) and manage course content for classes like CS101.
- **Course Requests:** Interface to propose new courses or update syllabi via department head approval.
  Relevance: Maps to *Functional Requirement 2.2* (Product Functions: CRUD operations for courses).

Figure 12: Teaching Schedule & Workload



Faculty Teaching Schedule

Monday	Tuesday	Wednesday	Thursday	Friday
CS101 Introduction to Programming; 09:00 - 10:30; Room 201; 32 students; Lecture	CS305 Data Structures and Algorithms; 13:00 - 14:30; Room 105; 28 students; Lecture	CS101 Introduction to Programming; 09:00 - 10:30; Room 201; 32 students; Lecture	CS305 Data Structures and Algorithms; 13:00 - 14:30; Room 105; 28 students; Lecture	CS305 Data Structures and Algorithms; 10:00 - 12:00; Lab 302; 28 students; Lab
CS401 Advanced Database Systems; 14:00 - 15:30; Room 405; 22 students; Lecture		CS401 Advanced Database Systems; 14:00 - 15:30; Room 405; 22 students; Lecture	CS401 Advanced Database Systems; 16:00 - 18:00; Lab 303; 22 students; Lab	

Teaching Summary

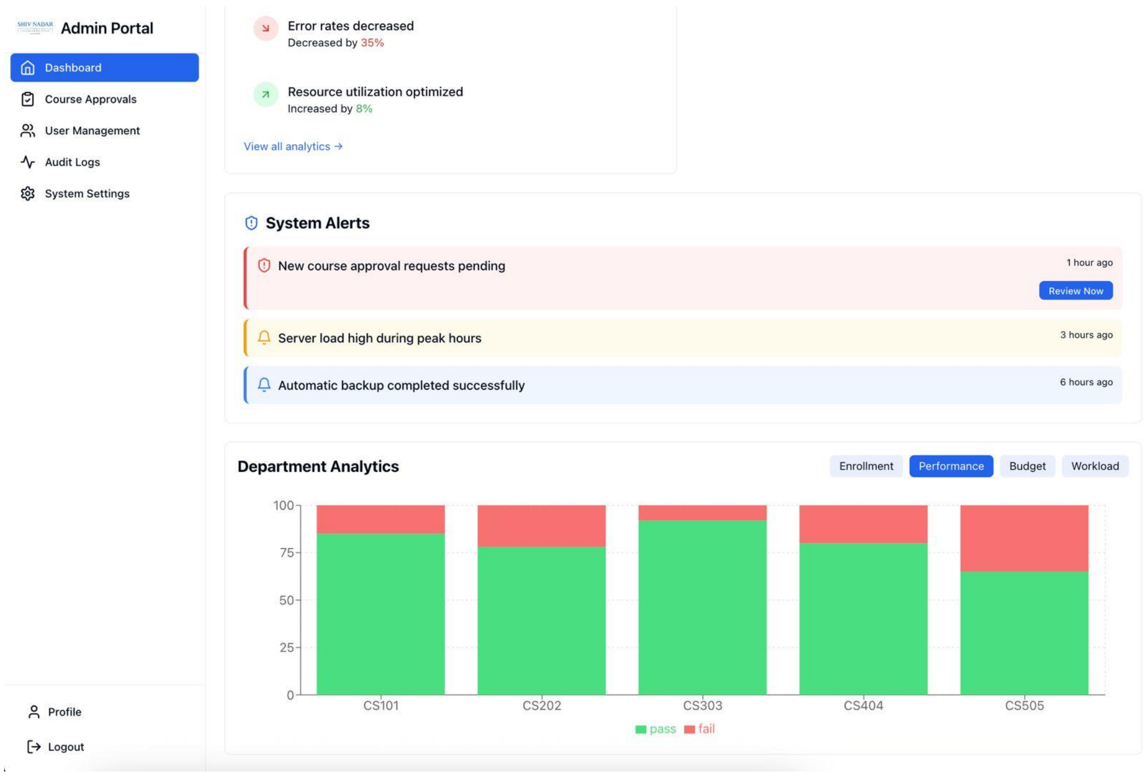
Total Courses	Weekly Hours	Total Students	Office Hours
3	12	82	6 hrs/week

Description:

- **Weekly Schedule:** Lists class timings (e.g., CS101 at 09:00 AM), room numbers, and session types (lecture/lab).
- **Workload Summary:** Tracks total courses (3), weekly hours (12), and office hours (6hrs/week).
  Relevance: Demonstrates *External Interface Requirements* (Integrated Calendar) and *User Characteristics* (Faculty workload management).

Admin Portal

Figure 13: Admin Dashboard Overview



Error rates decreased
Decreased by 35%

Resource utilization optimized
Increased by 8%

View all analytics →

System Alerts

New course approval requests pending — 1 hour ago — Review Now

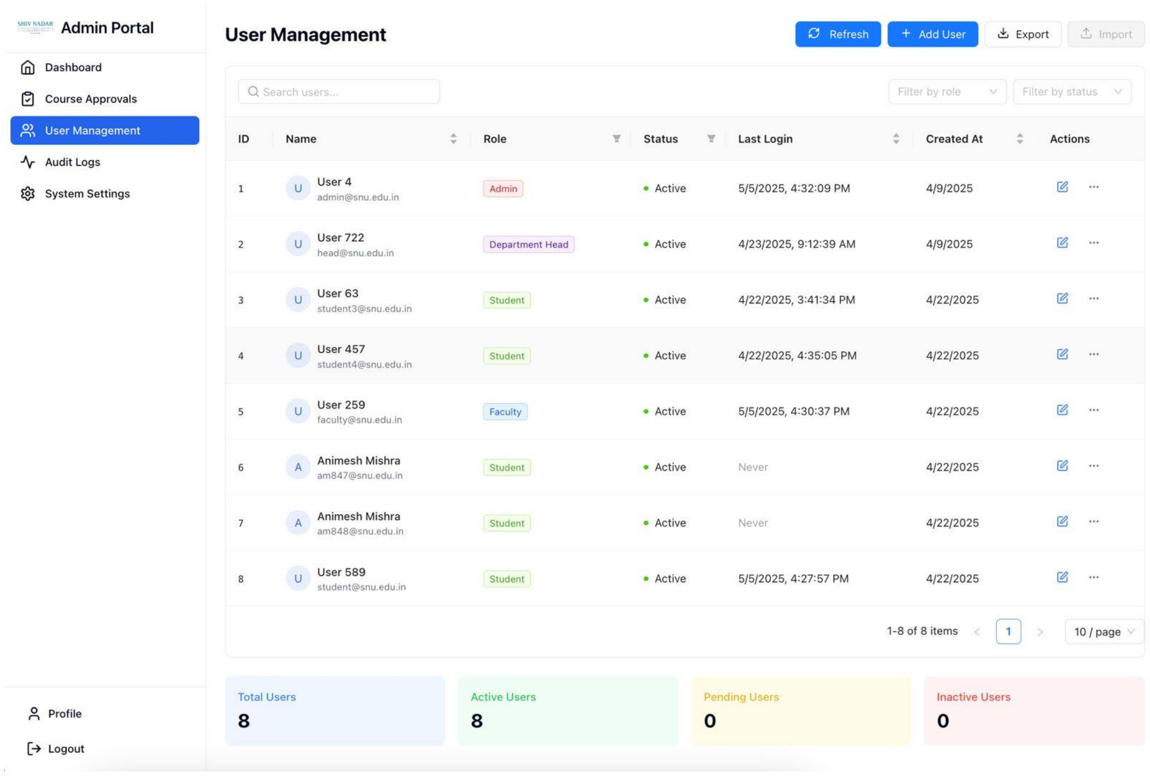
Server load high during peak hours — 3 hours ago

Automatic backup completed successfully — 6 hours ago

Description:

- **System Health Metrics:** Tracks error rate reduction (35%) and resource utilization improvement (8%).
- **Alerts:** Highlights pending course approvals, server load warnings, and backup status (*Use Case: Report Generation*).
- **Department Analytics:** Visualizes workload, budget, and performance metrics for departments like Computer Science.
  Relevance: Validates *Functional Requirement 3.1* (Admin actions: system monitoring) and *Non-Functional Requirement 3.2* (Performance optimization).

Figure 14: User Management Interface



User Management

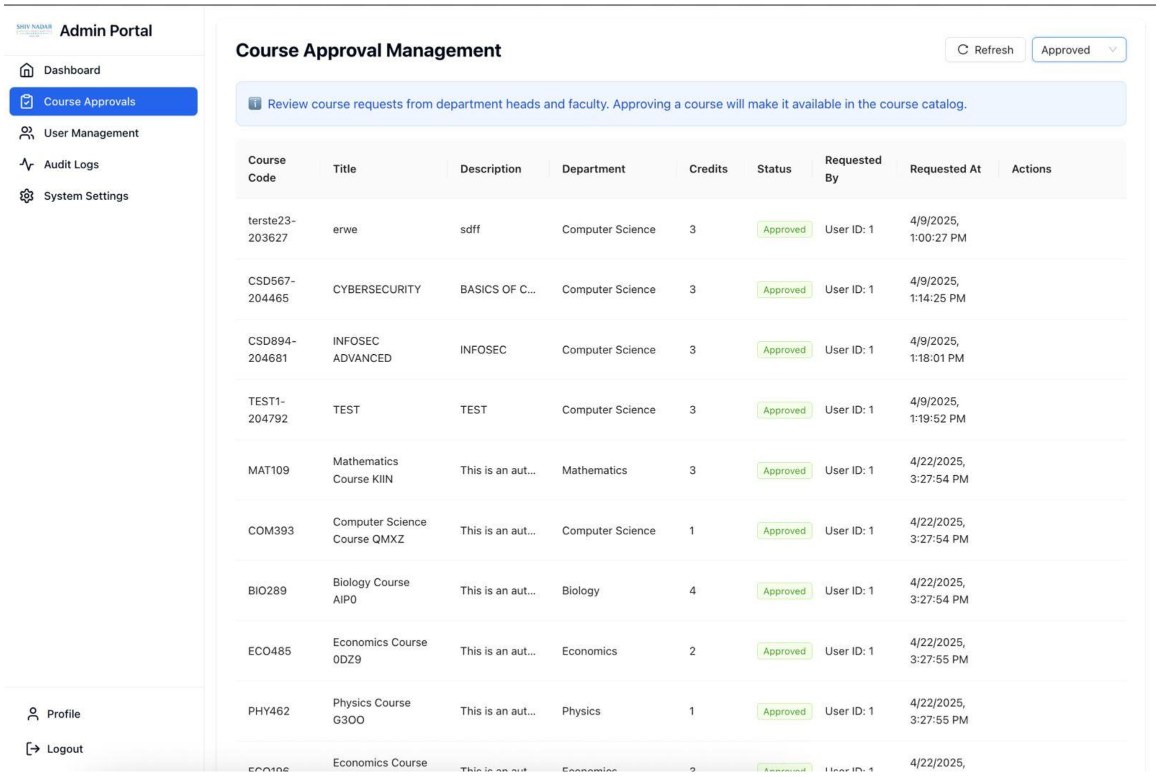
ID	Name	Role	Status	Last Login	Created At	Actions
1	User 4 admin@snu.edu.in	Admin	Active	5/5/2025, 4:32:09 PM	4/9/2025	
2	User 722 head@snu.edu.in	Department Head	Active	4/23/2025, 9:12:39 AM	4/9/2025	
3	User 63 student3@snu.edu.in	Student	Active	4/22/2025, 3:41:34 PM	4/22/2025	
4	User 457 student4@snu.edu.in	Student	Active	4/22/2025, 4:35:05 PM	4/22/2025	
5	User 259 faculty@snu.edu.in	Faculty	Active	5/5/2025, 4:30:37 PM	4/22/2025	
6	Animesh Mishra am847@snu.edu.in	Student	Active	Never	4/22/2025	
7	Animesh Mishra am848@snu.edu.in	Student	Active	Never	4/22/2025	
8	User 589 student@snu.edu.in	Student	Active	5/5/2025, 4:27:57 PM	4/22/2025	

Total Users	Active Users	Pending Users	Inactive Users
8	8	0	0

Description:

- **User List:** Displays user roles (Admin, Faculty, Student), statuses (Active/Inactive), and actions (edit/delete).
- **Search Functionality:** Allows filtering users by name, role, or email.
  Relevance: Aligns with *Functional Requirement 2.2* (Product Functions: CRUD operations for users) and *SRS 3.3* (Role-Based Access Control).

Figure 15: Course Approval Management



Course Approval Management

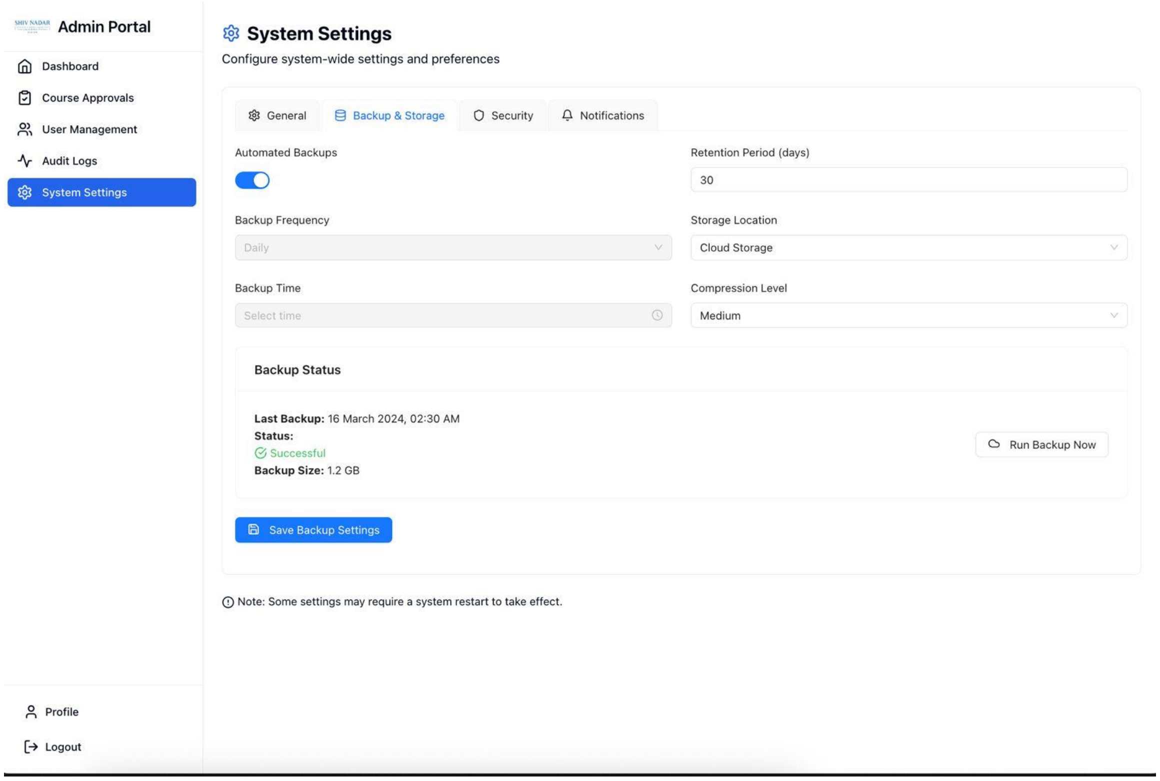
Review course requests from department heads and faculty. Approving a course will make it available in the course catalog.

Course Code	Title	Description	Department	Credits	Status	Requested By	Requested At	Actions
terste23-203627	erwe	sdff	Computer Science	3	Approved	User ID: 1	4/9/2025, 1:00:27 PM	
CSD567-204465	CYBERSECURITY	BASICS OF C...	Computer Science	3	Approved	User ID: 1	4/9/2025, 1:14:25 PM	
CSD894-204681	INFOSEC ADVANCED	INFOSEC	Computer Science	3	Approved	User ID: 1	4/9/2025, 1:18:01 PM	
TEST1-204792	TEST	TEST	Computer Science	3	Approved	User ID: 1	4/9/2025, 1:19:52 PM	
MAT109	Mathematics Course KIIN	This is an aut...	Mathematics	3	Approved	User ID: 1	4/22/2025, 3:27:54 PM	
COM393	Computer Science Course QMXZ	This is an aut...	Computer Science	1	Approved	User ID: 1	4/22/2025, 3:27:54 PM	
BIO289	Biology Course AIP0	This is an aut...	Biology	4	Approved	User ID: 1	4/22/2025, 3:27:54 PM	
ECO485	Economics Course 0DZ9	This is an aut...	Economics	2	Approved	User ID: 1	4/22/2025, 3:27:55 PM	
PHY462	Physics Course G3OO	This is an aut...	Physics	1	Approved	User ID: 1	4/22/2025, 3:27:55 PM	
[illegible]	Economics Course	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	4/22/2025,	

Description:

- **Course Requests:** Lists pending/approved courses (e.g., CYBERSECURITY, INFOSEC ADVANCED) with details like department, credits, and requester.
- **Approval Actions:** Admins can approve/reject courses, making them available in the catalog (*Use Case: Course Management*).
  Relevance: Supports *Functional Requirement 3.1* (Admin roles: course oversight) and *Domain Model* (Course class).

Figure 16: System Settings & Backup Configuration



System Settings

Configure system-wide settings and preferences

Automated Backups: on

Retention Period (days): 30

Backup Frequency: Daily

Storage Location: Cloud Storage

Backup Time: Select time

Compression Level: Medium

Backup Status

Last Backup: 16 March 2024, 02:30 AM
Status:
Successful
Backup Size: 1.2 GB

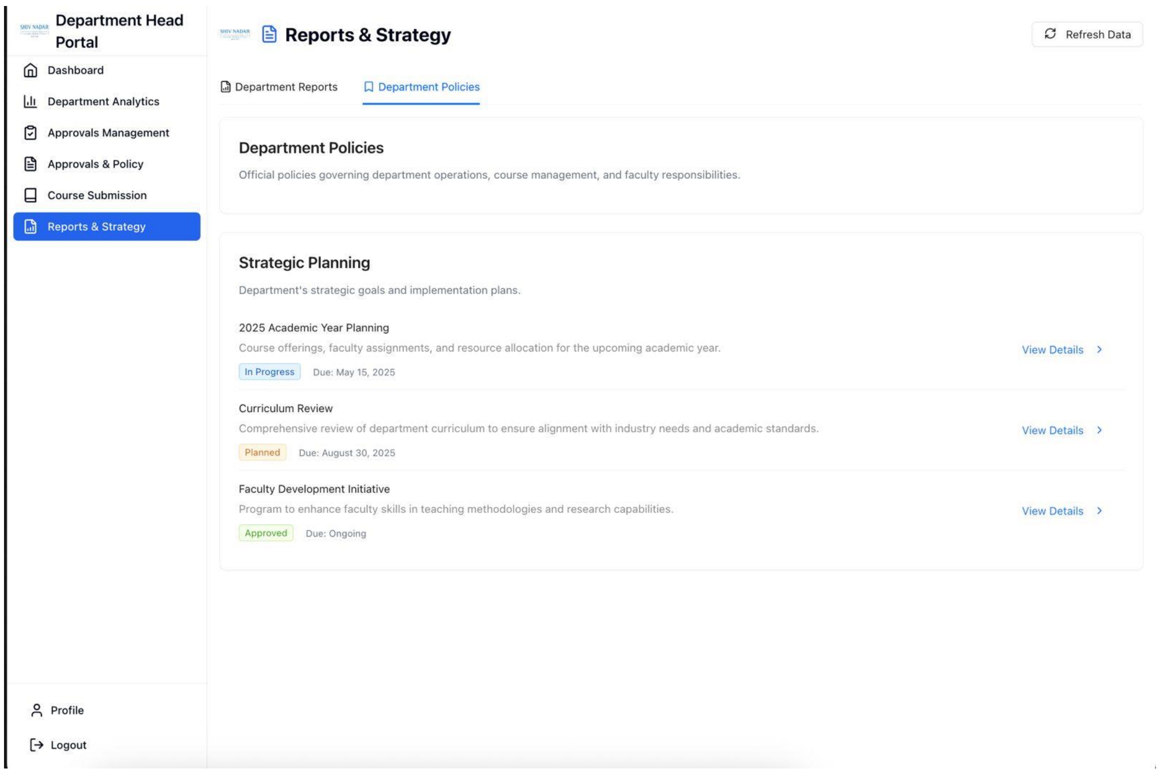
Note: Some settings may require a system restart to take effect.

Description:

- **Automated Backups:** Configures frequency (daily), retention period (30 days), and storage location (cloud).
- **Security Settings:** Includes encryption levels and notification preferences. **Relevance:** Maps to *Non-Functional Requirements* (Security: encrypted backups) and *External Interface Requirements* (System configuration UI).

Department Head Portal

Figure 17: Strategic Planning & Reports Dashboard



Department Head Portal

Dashboard
Department Analytics
Approvals Management
Approvals & Policy
Course Submission
Reports & Strategy

Profile
Logout

Reports & Strategy

Refresh Data

Department Reports | Department Policies

Department Policies

Official policies governing department operations, course management, and faculty responsibilities.

Strategic Planning

Department's strategic goals and implementation plans.

2025 Academic Year Planning
Course offerings, faculty assignments, and resource allocation for the upcoming academic year.
In Progress Due: May 15, 2025
View Details

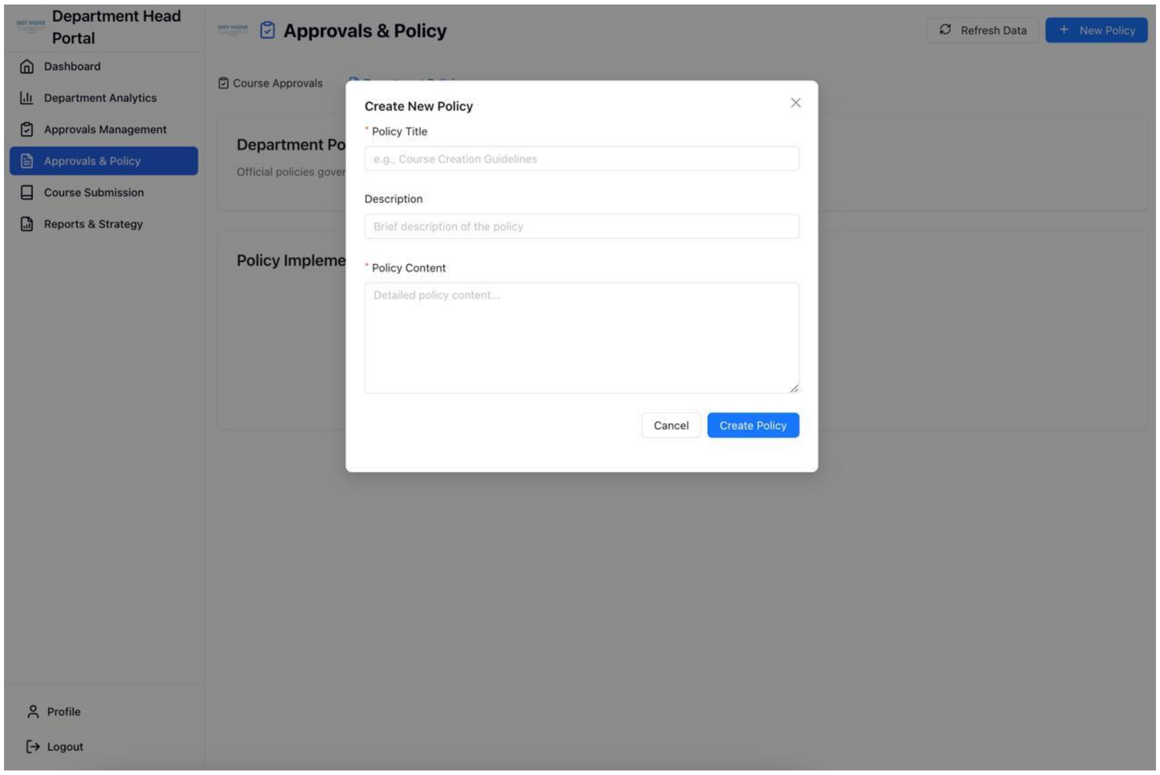
Curriculum Review
Comprehensive review of department curriculum to ensure alignment with industry needs and academic standards.
Planned Due: August 30, 2025
View Details

Faculty Development Initiative
Program to enhance faculty skills in teaching methodologies and research capabilities.
Approved Due: Ongoing
View Details

Description:

- **Strategic Initiatives:** Lists 2025 Academic Year Planning (course offerings, resource allocation), Curriculum Review, and Faculty Development programs.
- **Progress Tracking:** Status indicators (In Progress/Approved) and deadlines for each initiative.
  Relevance: Aligns with *Functional Requirement 3.1* (Department Head roles: strategic planning) and *SRS 2.2* (Product Functions: resource allocation).

Figure 18: Policy Creation Interface



Create New Policy

* Policy Title

e.g., Course Creation Guidelines

Description

Brief description of the policy

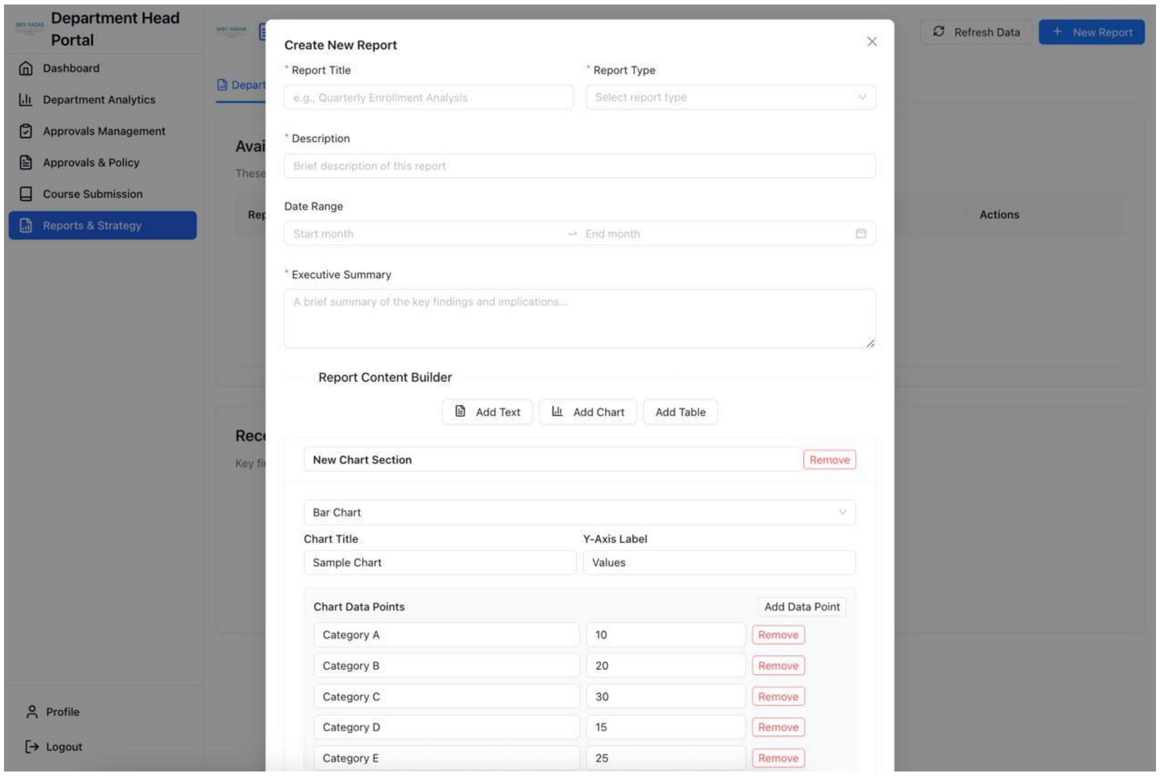
* Policy Content

Detailed policy content...

Description:

- **Policy Drafting:** Allows creation of policies (e.g., "Course Creation Guidelines") with titles, descriptions, and detailed content.
- **Workflow Integration:** Policies are saved and implemented department-wide after approval.
  Relevance: Validates *Functional Requirement 3.1* (Policy management) and *Use Case: Policy Management*.

Figure 19: Custom Report Generation



Department Head Portal

Dashboard

Department Analytics

Approvals Management

Approvals & Policy

Course Submission

Reports & Strategy

Profile

Logout

Refresh Data

New Report

Create New Report

* Report Title

e.g., Quarterly Enrollment Analysis

* Report Type

Select report type

* Description

Brief description of this report

Date Range

Start month → End month

* Executive Summary

A brief summary of the key findings and implications...

Report Content Builder

Add Text | Add Chart | Add Table

New Chart Section — Remove

Bar Chart

Chart Title

Sample Chart

Y-Axis Label

Values

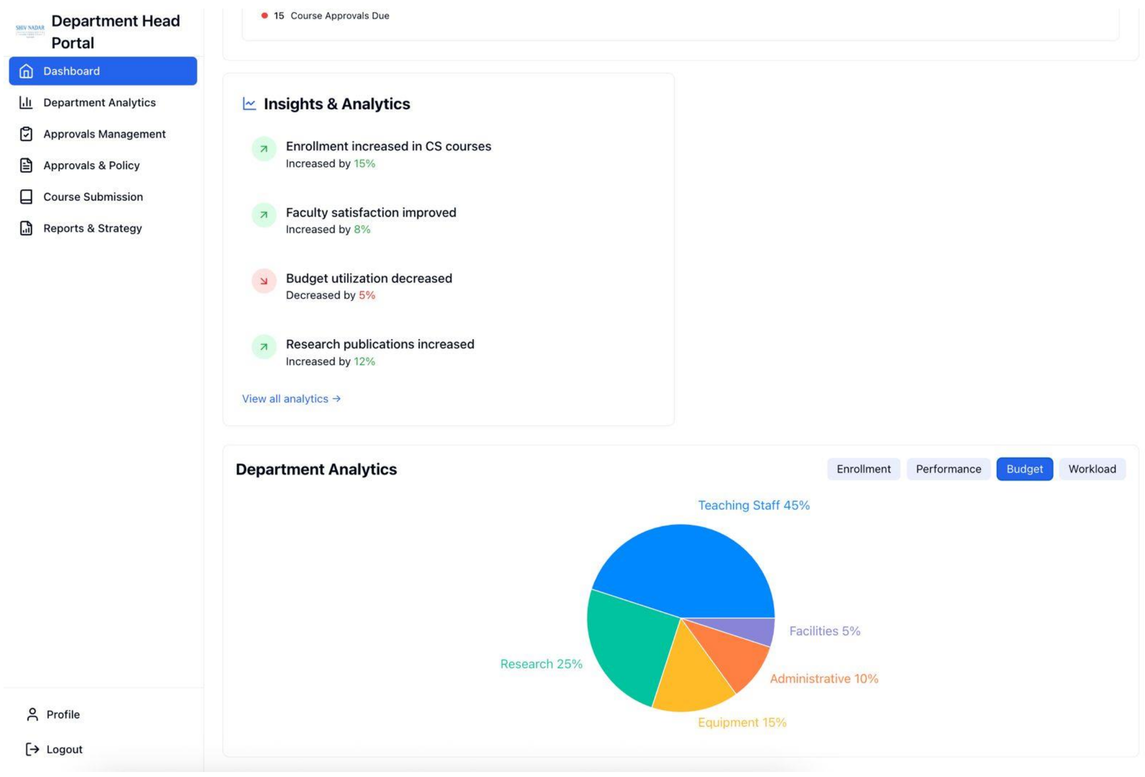
Chart Data Points — Add Data Point

Category A	10	Remove
Category B	20	Remove
Category C	30	Remove
Category D	15	Remove
Category E	25	Remove

Description:

- **Report Customization:** Tools to build reports with text, charts (bar/pie), and tables.
- **Data Visualization:** Example chart showing enrollment categories (Category A: 10, Category B: 20).
  Relevance: Supports *Non-Functional Requirement 3.2* (Performance: real-time analytics) and *SRS 3.3* (External Interfaces: interactive dashboards).

Figure 20: Department Insights & Analytics



Department Head Portal

15 Course Approvals Due

Insights & Analytics

Enrollment increased in CS courses
Increased by 15%

Faculty satisfaction improved
Increased by 8%

Budget utilization decreased
Decreased by 5%

Research publications increased
Increased by 12%

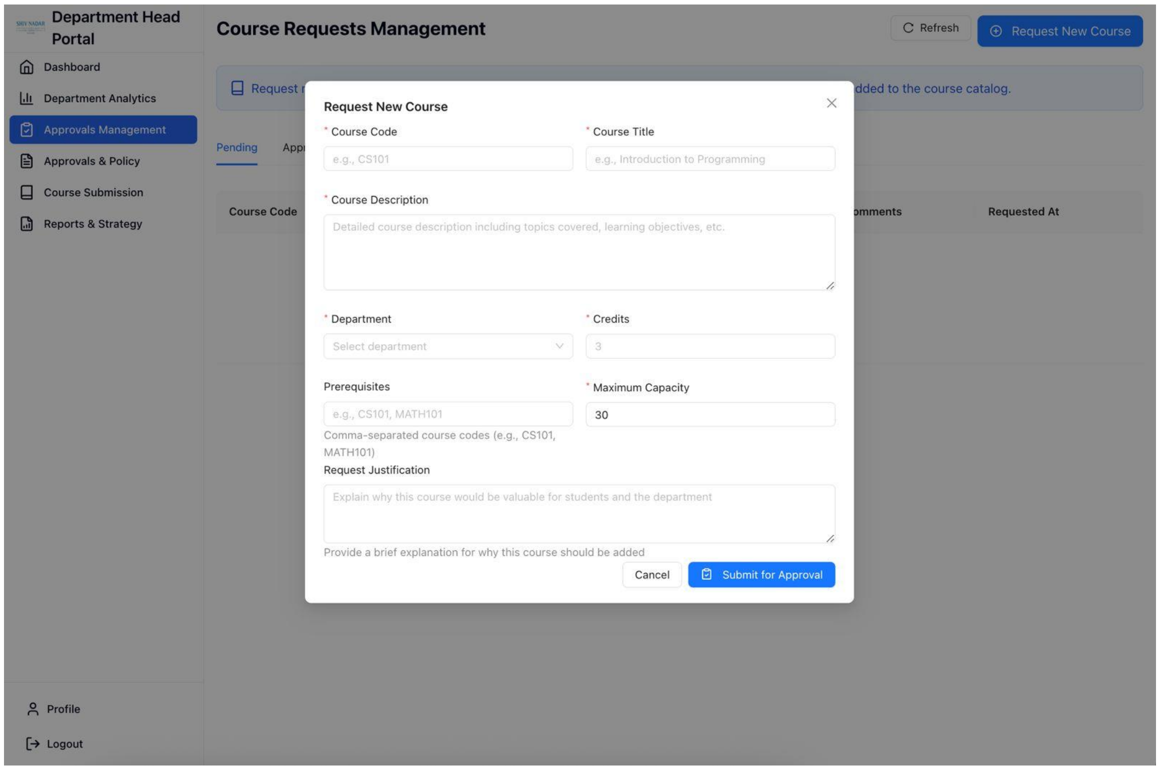
View all analytics →

Department Analytics

Description:

- **Key Metrics:** Highlights enrollment growth (15% in CS courses), faculty satisfaction (+8%), and research publications (+12%).
- **Budget Allocation:** Pie chart breaks down departmental spending (Teaching Staff: 45%, Facilities: 5%).
  Relevance: Demonstrates *Product Perspective* (data-driven decision-making) and *User Characteristics* (Department Head oversight).

Figure 21: Course Submission Workflow



Department Head Portal

- Dashboard
- Department Analytics
- Approvals Management
- Approvals & Policy
- Course Submission
- Reports & Strategy
- Profile
- Logout

Course Requests Management

Refresh | Request New Course

Request New Course

* Course Code: e.g., CS101
* Course Title: e.g., Introduction to Programming
* Course Description: Detailed course description including topics covered, learning objectives, etc.
* Department: Select department
* Credits: 3

Prerequisites: e.g., CS101, MATH101

Comma-separated course codes (e.g., CS101, MATH101)

* Maximum Capacity: 30

Request Justification: Explain why this course would be valuable for students and the department

Provide a brief explanation for why this course should be added

Cancel | Submit for Approval

Description:

- **Course Proposal:** Fields for course code (e.g., CS101), title, description, prerequisites, and justification.
- **Approval Process:** Requests are submitted to admins for review (*Use Case: Course Management*).
  Relevance: Maps to *Functional Requirement 2.2* (Course CRUD operations) and *Domain Model* (Course class attributes).

Summary

In this project, we have designed and developed a comprehensive University Department Information Management System (UDIMS) to address the shortcomings of fragmented, manual departmental processes. Beginning with a clear **Problem Statement**, we identified key pain-points—data inconsistency, redundant workflows, and lack of real-time integration—that motivated a unified, web-based solution. The **Software Requirements Specification (SRS)** established both functional and non-functional requirements, ensuring that UDIMS would meet stakeholder needs for user management, course registration, grade processing, resource allocation, and reporting.

Our **Data Flow Diagrams (Level 0 and Level 1)** provided a top-down view of information movement between external actors (students, faculty, admin staff, department head) and core system processes (authentication, course registration, grade management, resource management, report generation). The **Use Case Diagram** and detailed use-case tables translated those processes into concrete user interactions, while the **Domain Model** clarified the underlying data structures—entities, attributes, and relationships—that drive the application's database schema. A representative **Sequence Diagram** then illustrated the step-by-step object interactions for a critical scenario, reinforcing the design's coherence.

The **Project Setup** section documented the technology stack (e.g., React frontend, Node.js/Express backend, MongoDB), installation steps, and system requirements, enabling straightforward deployment and evaluation. Finally, the **Results** section showcased screenshots of key modules—login, dashboard, course management, report generation—demonstrating that UDIMS meets its objectives for usability, security, and performance.

Overall, UDIMS successfully streamlines departmental operations by automating workflows, enforcing standardized data handling, and providing real-time analytics. Future enhancements could include integration with additional university services (library, finance), advanced data-mining for predictive analytics, and mobile-friendly interfaces to further improve accessibility and decision-support for all stakeholders.