

**UNIVERSITY MANAGEMENT SYSTEM**

PROJECT REPORT

23AID215 – USER INTERFACE DESIGN

Submitted by

ABHISHTA H MALLAYA - CB.AI.U4AID23102

GOWTHAM S D - CB.AI.U4AID23113

KANISHKA S J - CB.AI.U4AID23121

VYSHNAV KUMAR S - CB.AI.U4AID23151

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

Artificial Intelligence and Data Science

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112 (INDIA)

APRIL - 2025

# AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

# AMRITA VISHWA VIDYAPEETHAM

# COIMBATORE - 641 112

# BONAFIDE CERTIFICATE

# This is to certify that the thesis entitled “UNIVERRSITY MANAGEMENT SYSTEM” submitted by ABHISHTA H MALLAYA - CB.AI.U4AID23102, GOWTHAM S D - CB.AI.U4AID23113, KANISHKA S J - CB.AI.U4AID23121, VYSHNAV KUMAR S - CB.AI.U4AID23151 for the award of the Degree of Bachelor of Technology in the “AI-DS” is a Bonafide record of the work carried out by her under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

# Ms. Sreelakshmi K

# Project Guide

## Acknowledgement

We would like to express our special thanks of gratitude to our teacher (MS. SREELAKSHMI K ma’am), who gave us the golden opportunity to do this wonderful project on the topic ( UNIVERSITY MANAGEMENT SYSTEM), which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given.

We would also like to thank our group members, as without their cooperation, we would not have been able to complete the project within the prescribed time.

# Contents

[Acknowledgement iv](#_TOC_250033)

[List of Pages v](#_TOC_250032)

[List of Tables vi](#_TOC_250031)

[Abstract viii](#_TOC_250029)

1

[1.1 Literature Survey 4](#_TOC_250028)

[1.2 Problem statement 9](#_TOC_250027)

[1.3 Objectives 9](#_TOC_250026)

[1.4 Organization of the thesis 10](#_TOC_250025)

[References 46](#_TOC_250001)

# List of Figures

* 1. ADMIN-DASHBOARD
  2. ADMIN-BOOKMANAGEMENT
  3. ADMIN-AUDITORIUM
  4. ADMIN-STUDENTMANAGEMENT
  5. ADMIN- TEACHERMANAGEMENT
  6. ADMIN-TIMETABLE
  7. STUDENT-DASHBOARD
  8. STUDENT-BOOKS
  9. STUDENT ATTENDANCE
  10. STUDENT-COURSES
  11. STUDENT-FEEDBACK
  12. STUDENT-GRADE
  13. STUDENT-PROFILE
  14. TEACHER-DASHBOARD
  15. TEACHER-ATTENDANCE
  16. TEACHER-COURSES
  17. TEACHER-FEEDBACK
  18. TEACHER-TIMETABLE
  19. TEACHER-PROFILE
  20. TEACHER AUDITORIUM

**List of Tables**

* 1. User
  2. student
  3. teacher
  4. admin
  5. department
  6. course
  7. enrollment
  8. teacher course
  9. attendance
  10. feedback
  11. book
  12. borrowed book
  13. auditorium
  14. auditorium booking
  15. auditorium availability
  16. timeslot
  17. academic year
  18. schedule
  19. section
  20. role
  21. day
  22. schedule type

**1.1 Literature Survey**

The development of college management systems has developed significantly over the past few years, from standalone applications to integrated web-based solutions. Sharma and Patel conducted a comprehensive review of educational management systems, highlighting the shift toward cloud-based architectures that support multi-device access and real-time data synchronization. Their study shows that modern educational institutions require systems capable of handling complex administrative workflows while providing intuitive interfaces for diverse user groups [1].

Research by Martinez et al. demonstrated that integrated management systems in higher education can reduce administrative workload by up to 35% while improving data accuracy by eliminating redundant data entry [2].

Similarly, Wang and Ibrahim found that institutions implementing comprehensive management systems reported significant improvements in student satisfaction rates, particularly regarding transparency in academic processes and resource availability [3].

The technological foundations for educational management systems have also been extensively studied. Kumar and Zhao evaluated various technology stacks for academic information systems, concluding that React-based frontends paired with robust ORM solutions like Prisma offered optimal performance and maintainability for educational contexts [4].

Their findings indicated that PostgreSQL databases provided the necessary flexibility and transactional integrity required for complex academic data relationships.

Security considerations in educational management systems were addressed by Okonkwo and Garcia, who proposed a multi-layered security framework incorporating role-based access control and advanced authentication methods [5].

Their work emphasized the increasing importance of data protection in systems handling sensitive student information and academic records.

Recent work by Thompson et al. has focused on the integration of analytics capabilities within management systems, enabling data-driven decision-making at institutional levels [6]. Their research demonstrated how properly structured database schemas could support both operational functions and analytical reporting without compromising system performance.

**1.2 Problem Statement**

Despite technological advancements, many educational institutions continue to struggle with fragmented management systems that operate in isolation, creating data silos and inefficient workflows. This fragmentation leads to several critical issues:

1. **Data Inconsistency:** Information exists in multiple independent systems, maintaining data consistently becomes challenging, resulting in discrepancies that affect decision-making and operational efficiency.
2. **Workflow Inefficiency:** The lack of integration between academic, administrative, and resource management systems makes it tough for users to navigate multiple interfaces and often duplicate efforts when performing related tasks
3. **Reporting Complexity:** Generating comprehensive reports across departmental boundaries requires manual data aggregation from disparate sources, increasing the likelihood of errors and delaying critical insights
4. **Security Vulnerabilities:** Maintaining proper security protocols across multiple independent systems increases the potential attack surface and complicates the implementation of consistent security policies

This project addresses these challenges by developing a unified college management system that integrates all essential academic and administrative functions within a single, secure platform accessible to administrators, faculty, and students.

**1.3 Objectives**

The primary aim of this project is to design and implement a comprehensive web-based college management system that seamlessly integrates all aspects of institutional operations. The specific objectives are:

1. To develop a unified database architecture that effectively models complex relationships between academic entities while maintaining data integrity and supporting efficient queries across all system modules.
2. To create role-specific interfaces that provide administrators, teachers, and students with intuitive access to their respective functions while maintaining a consistent user experience.
3. To implement a secure authentication framework utilizing Next.js authentication capabilities with role-based access control to ensure data protection and appropriate feature access.
4. To design modular system components for academic management, library services, facility booking, and extracurricular activities that function both independently and as an integrated whole. This modular approach builds on the architectural principles established by Nguyen and Roberts for scalable educational platforms [7].
5. To optimize system performance through efficient database design, appropriate indexing strategies, and optimized frontend rendering to support concurrent users without degradation in response times.
6. To ensure systemextensibility through well-documented APIs and modular architecture that will accommodate future enhancements and integration with external systems.

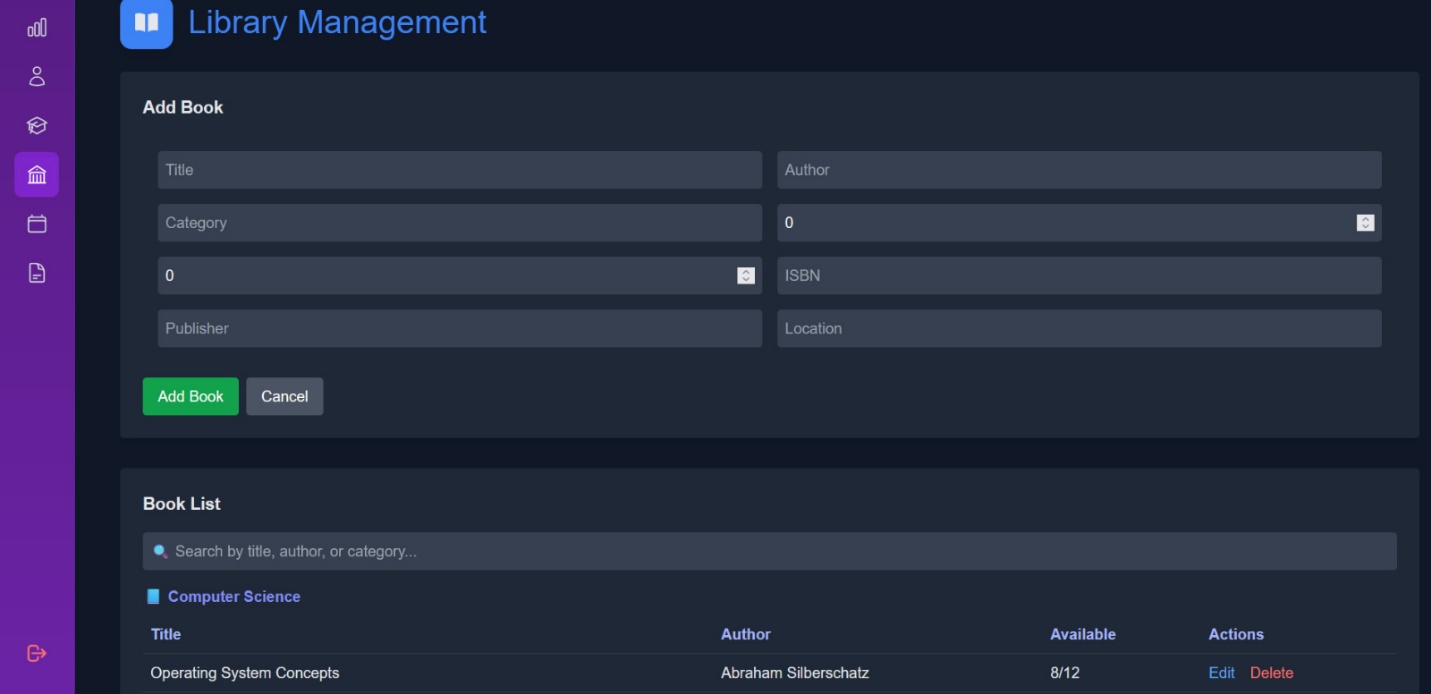
**ADMIN UI:**

* 1. ADMIN-DASHBOARD



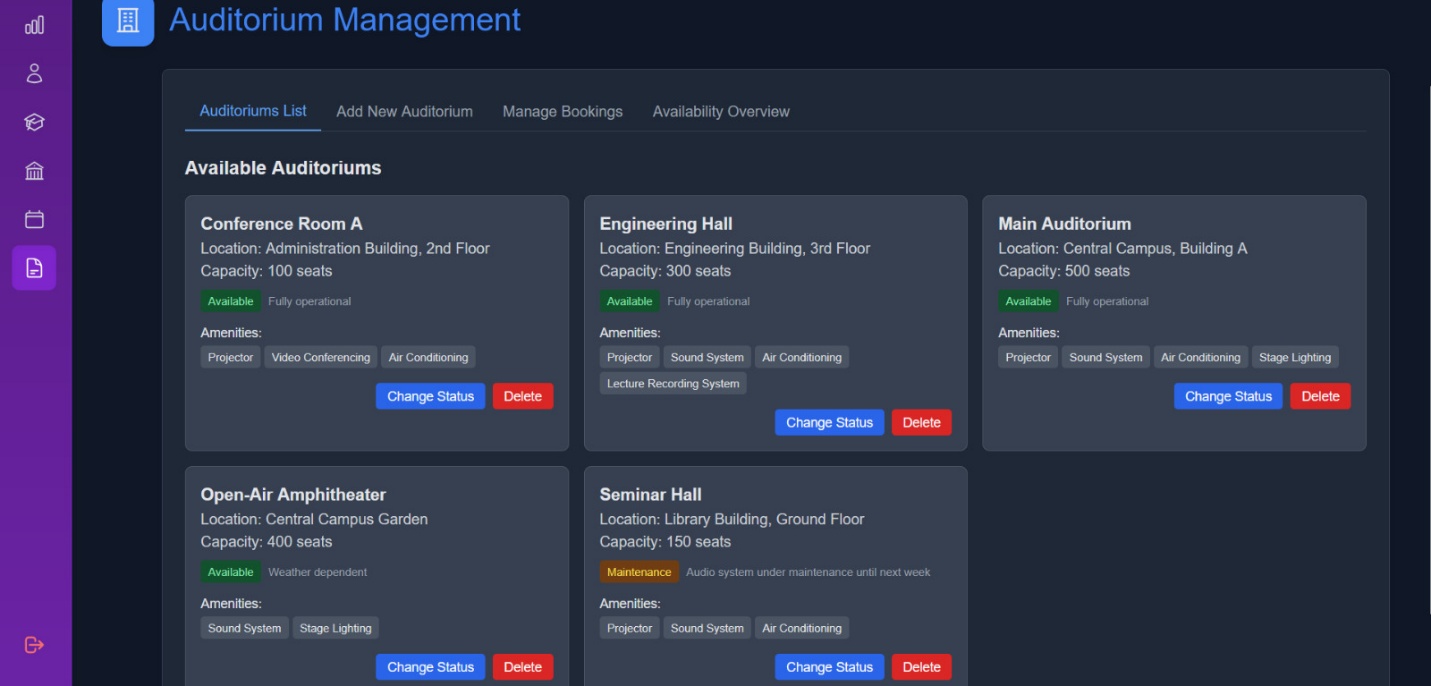
The administrative dashboard presents a comprehensive overview of the educational institution's key performance indicators via an intuitive graphical interface. This central command center displays real-time analytics including enrollment metrics, attendance statistics, resource utilization rates, and system notifications, enabling administrators to monitor institutional operations efficiently.

* 1. ADMIN-BOOKMANAGEMENT



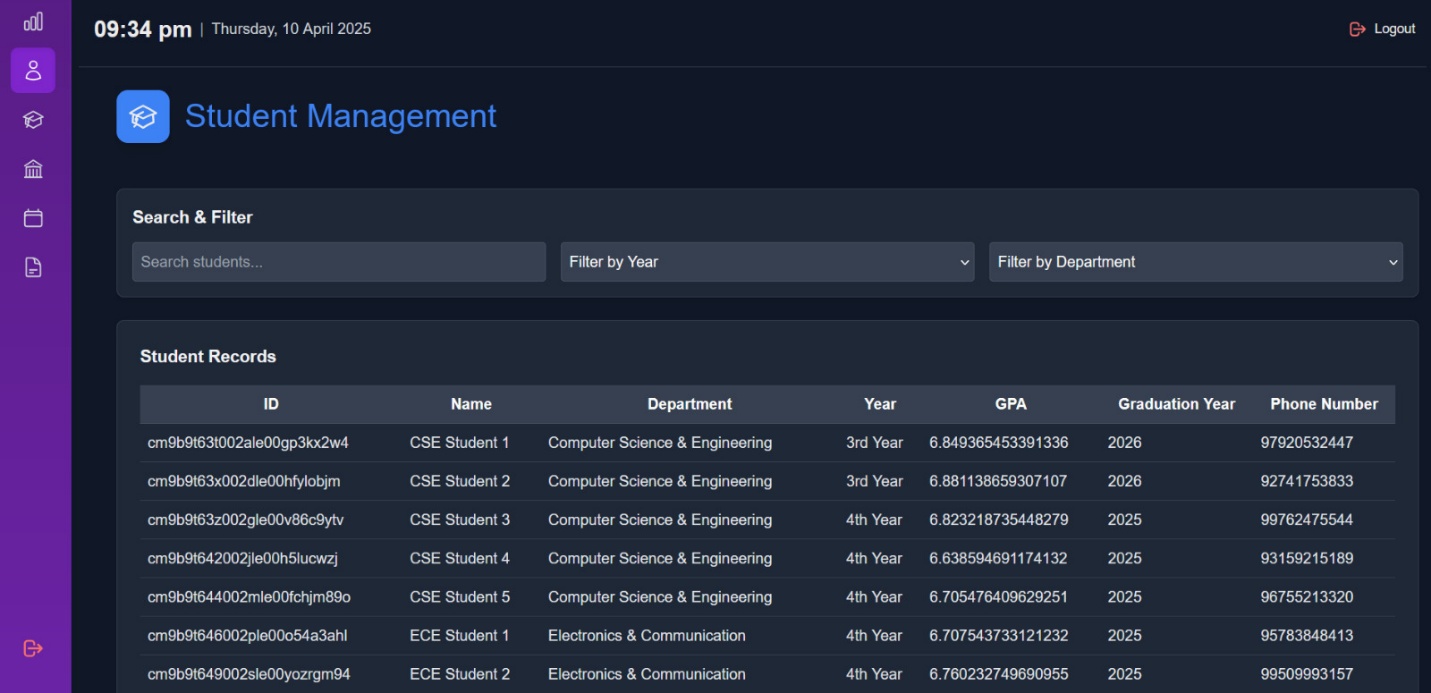
The book management interface facilitates comprehensive library asset control through a structured database presentation. Administrators can access full bibliographic records, implement cataloging protocols, monitor circulation statistics, manage acquisition processes, and generate inventory reports through this centralized literary resource management system.

* 1. ADMIN-AUDITORIUM



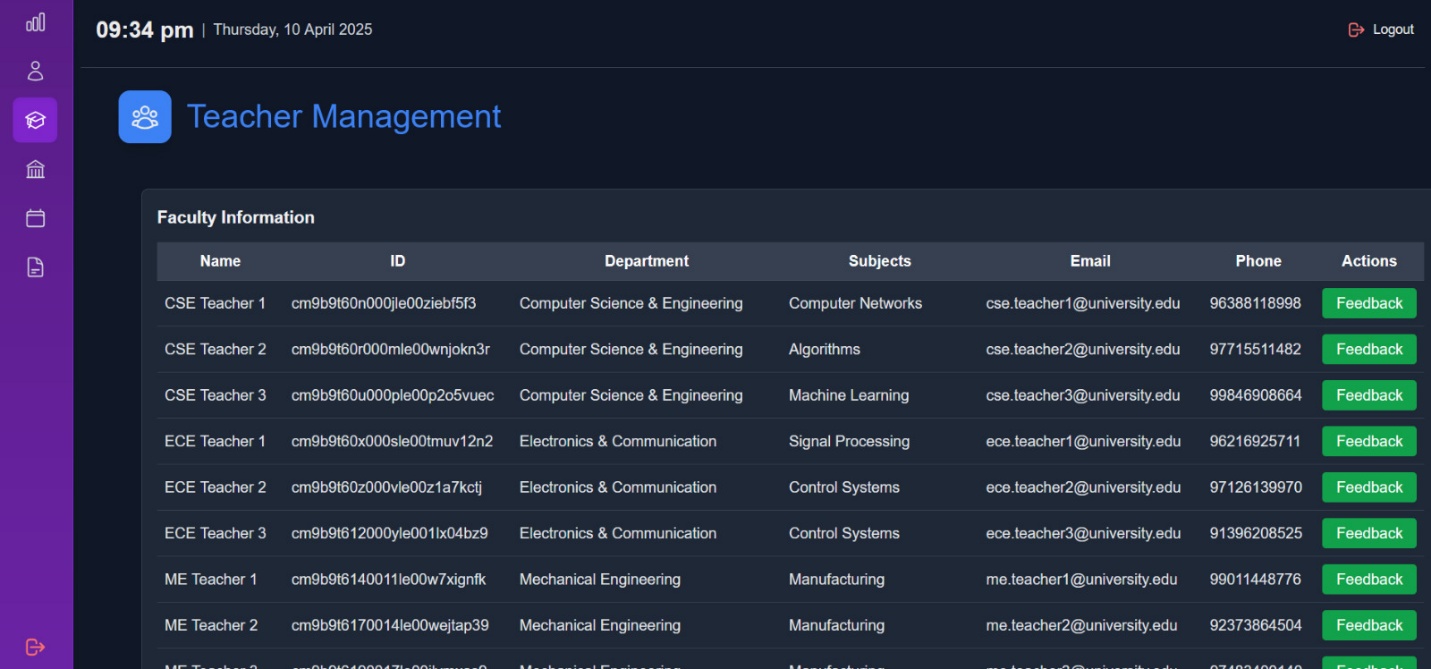
This specialized facility management module provides administrators with scheduling capabilities for the institution's auditorium resources. The interface displays reservation statuses, booking requests, capacity utilization metrics, and conflict detection tools, supporting optimal space allocation for academic and extracurricular events.

* 1. ADMIN-STUDENTMANAGEMENT



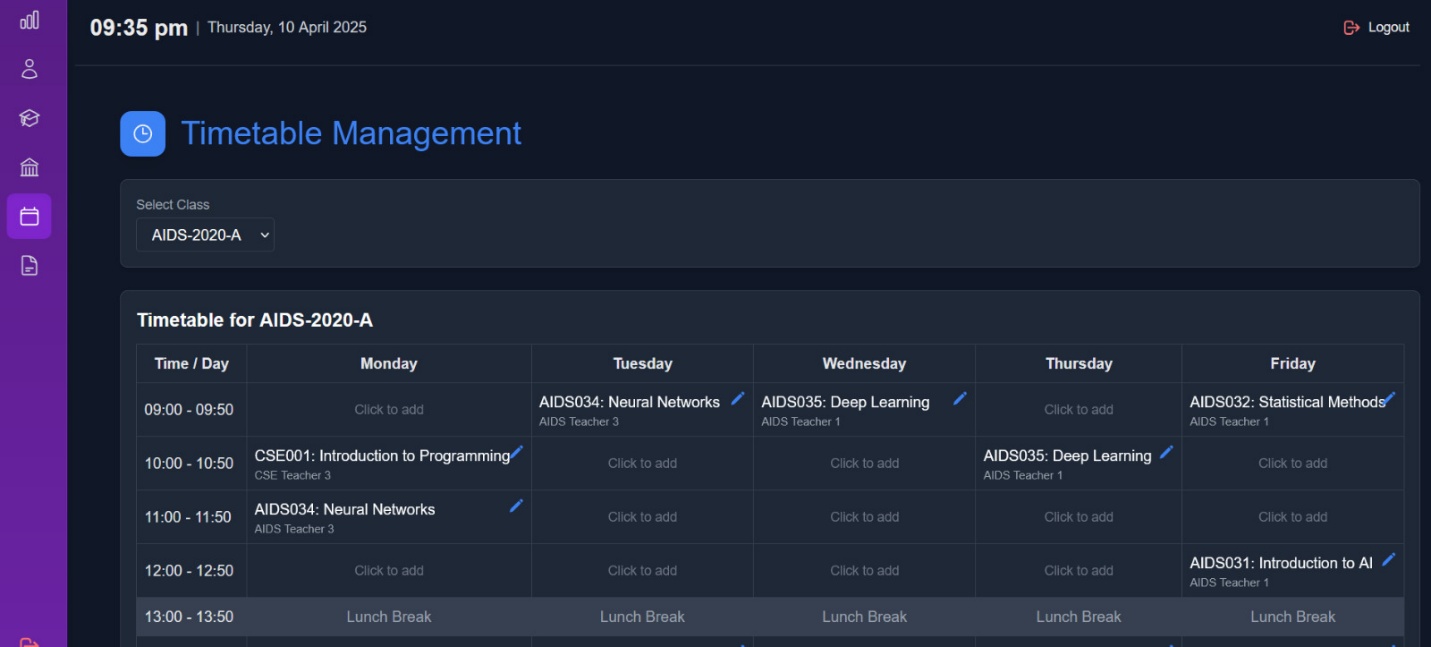
The student management system presents a sophisticated database interface for maintaining comprehensive student records. Administrators can access demographic information, academic histories, enrollment statuses, and communication logs while implementing batch processing operations for registration, promotion, and academic tracking purposes.

* 1. ADMIN- TEACHERMANAGEMENT



This faculty administration interface provides comprehensive human resource management capabilities for teaching staff. The system facilitates credential verification, workload allocation, performance evaluation documentation, and professional development tracking within an integrated personnel management framework.

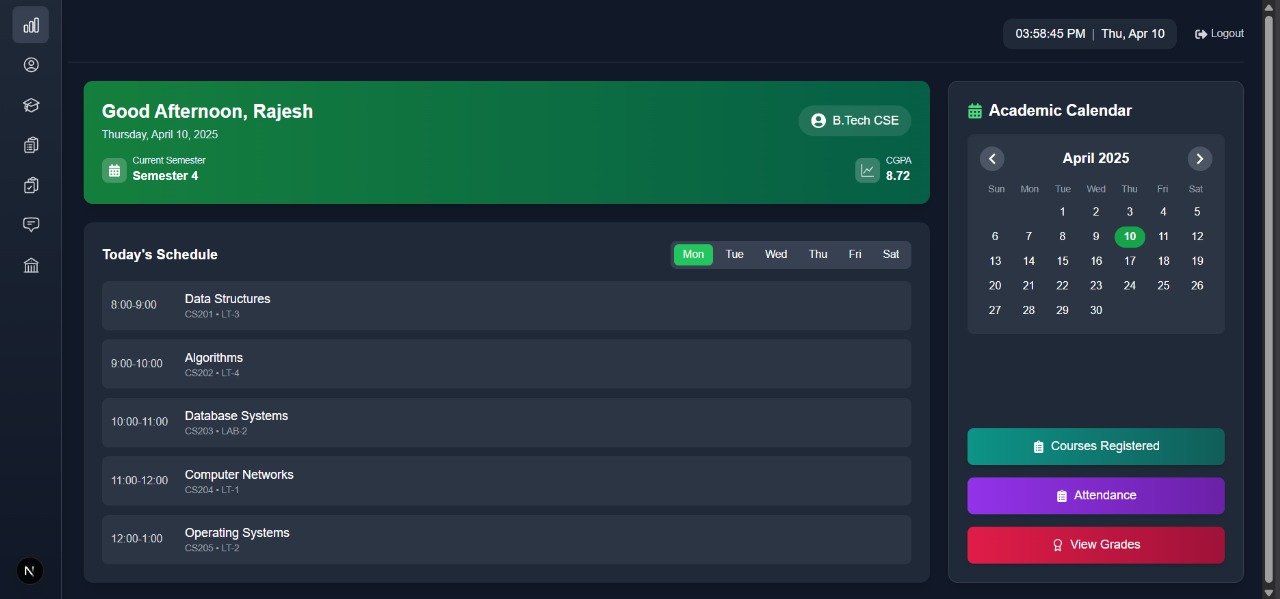
* 1. ADMIN-TIMETABLE



The timetable administration module offers master scheduling functionality with algorithm-assisted resource optimization. Administrators can configure academic periods, allocate classrooms, assign faculty, and resolve scheduling conflicts while maintaining institutional scheduling policies across academic terms.

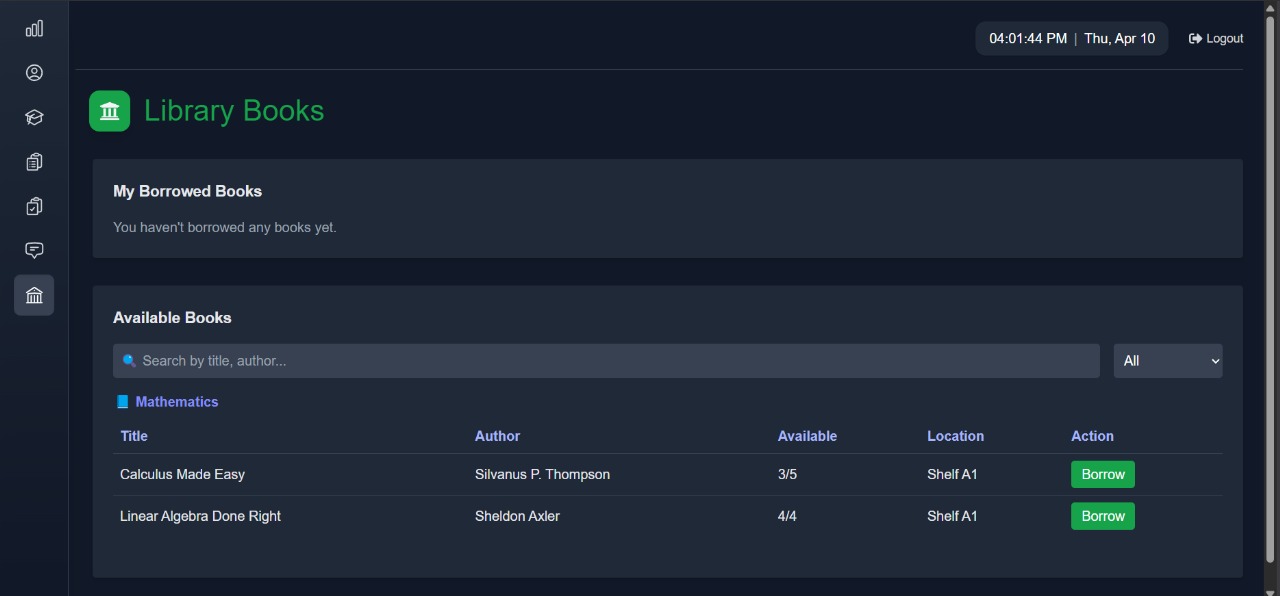
**STUDENT UI:**

* 1. STUDENT-DASHBOARD



The personalized student portal presents a customized academic overview with dynamic content based on individual enrollment parameters. Students receive a consolidated display of course schedules, pending assignments, assessment results, and institutional announcements tailored to their specific academic programs.

* 1. STUDENT-BOOKS



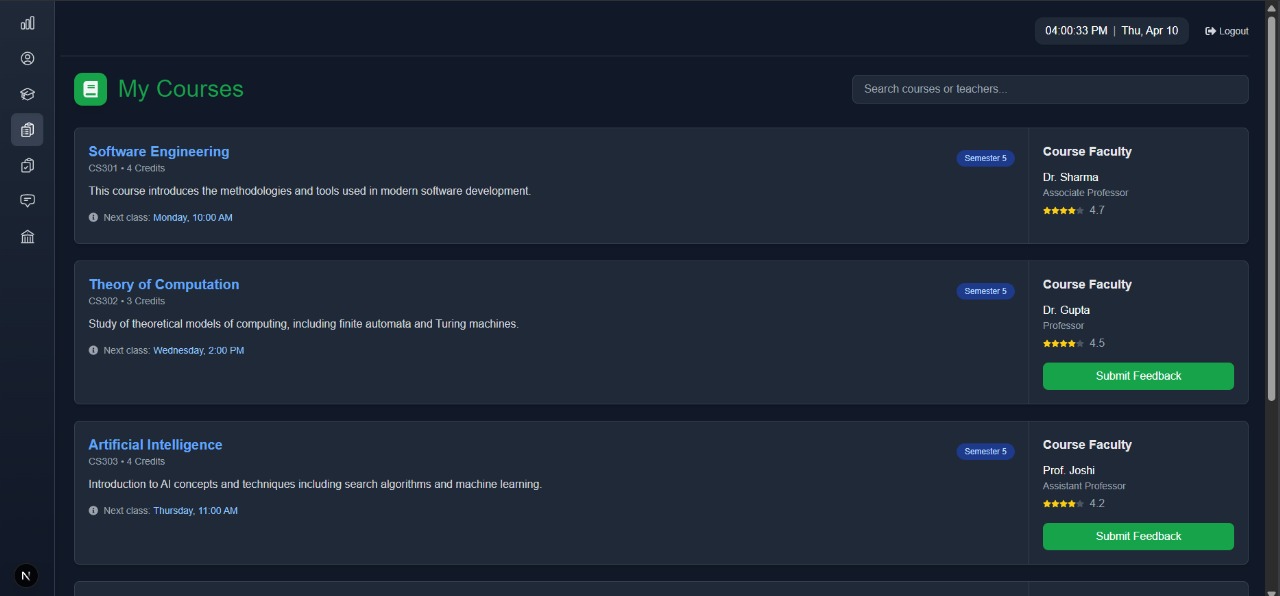
The student library interface provides learners with digital catalog access to the institution's literary resources. Students can search bibliographic records, check availability status, place reservation requests, and monitor their borrowing history through this dedicated literary access portal.

* 1. STUDENT ATTENDANCE



This attendance tracking module displays individual attendance records across enrolled courses. Students can view their participation metrics, absence documentation, attendance performance analytics, and any attendance-related policies or notifications from the institution.

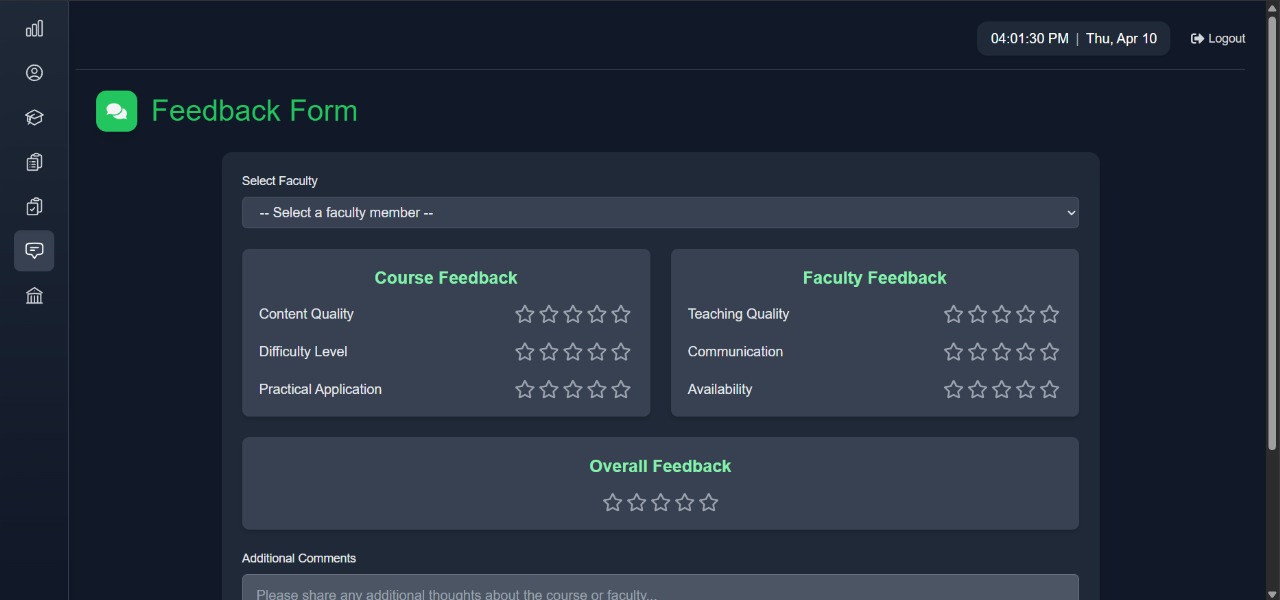
2.10 STUDENT-COURSES





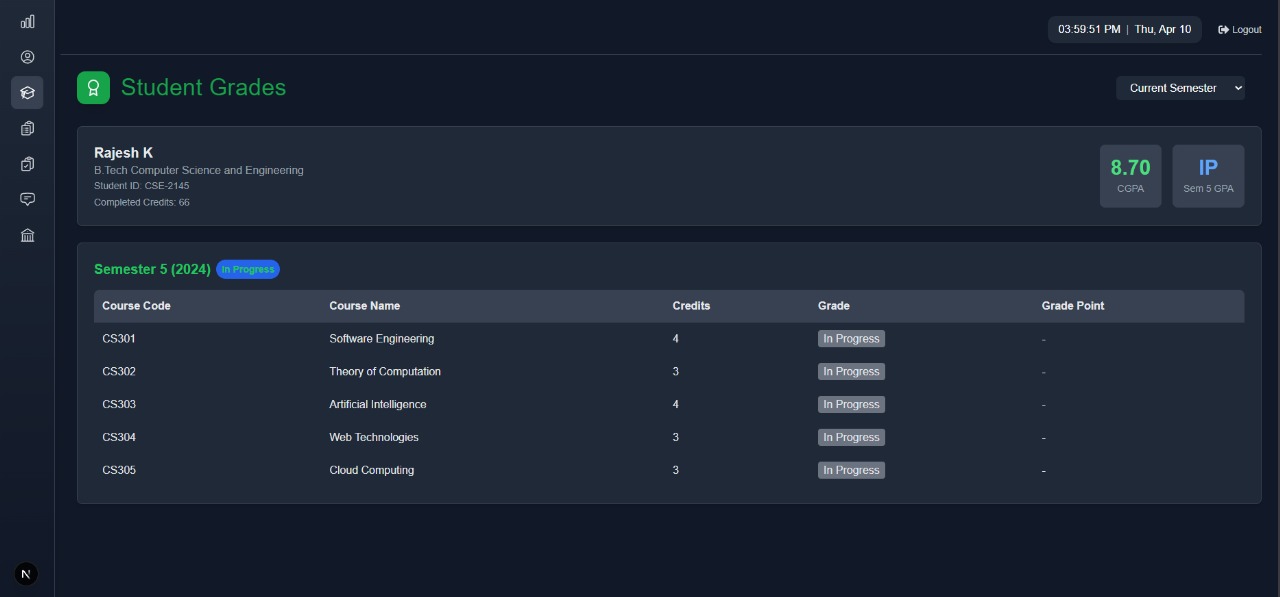
The course management interface provides students with detailed academic program information. The system displays enrolled courses, syllabus documentation, instructor information, and assessment schedules while facilitating access to digital learning resources associated with each course.

2.11 STUDENT-FEEDBACK



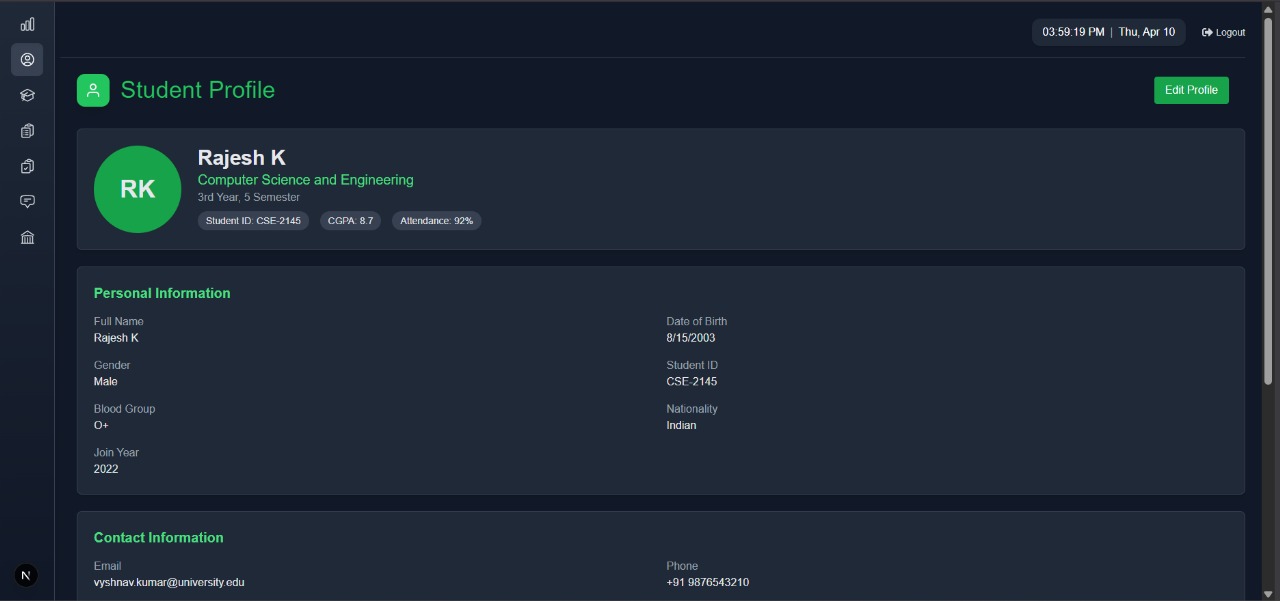
The institutional feedback mechanism allows students to submit structured evaluations regarding academic programs, teaching quality, and institutional services. This interface facilitates anonymous feedback submission, displays previous evaluation submissions, and may include institutional responses to collective feedback.

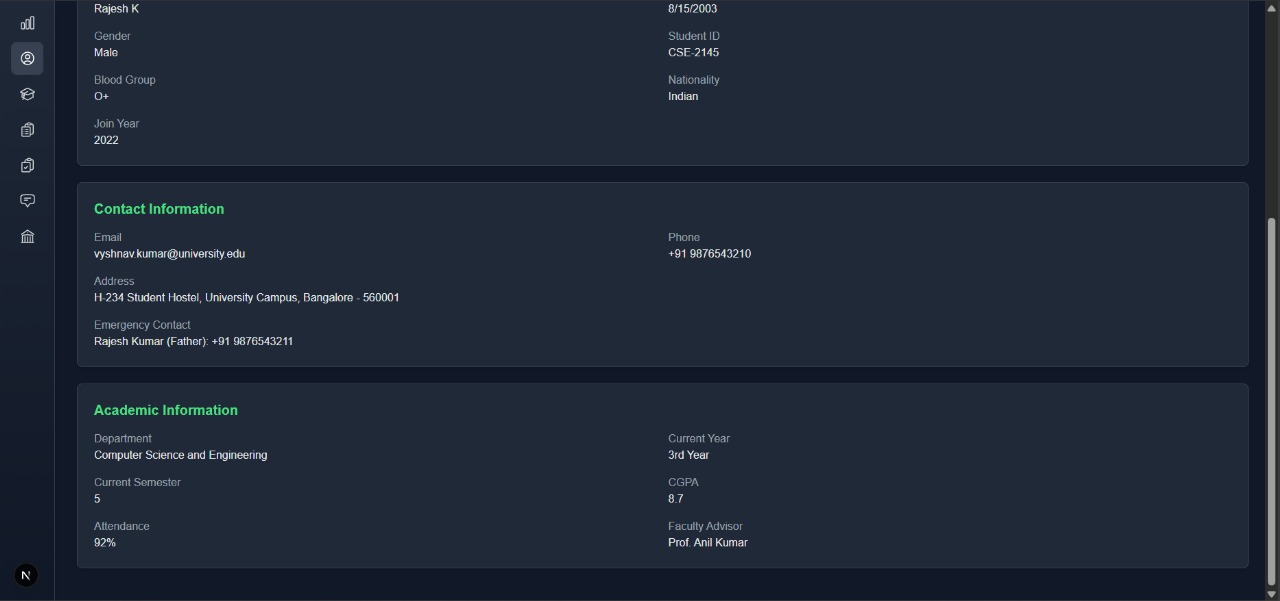
2.12 STUDENT-GRADE



This academic performance module presents comprehensive assessment results across enrolled courses. Students can view detailed grade breakdowns, cumulative performance metrics, assessment feedback, and progress toward degree requirements through this secure grade reporting interface.

2.13 STUDENT-DASHBOARD



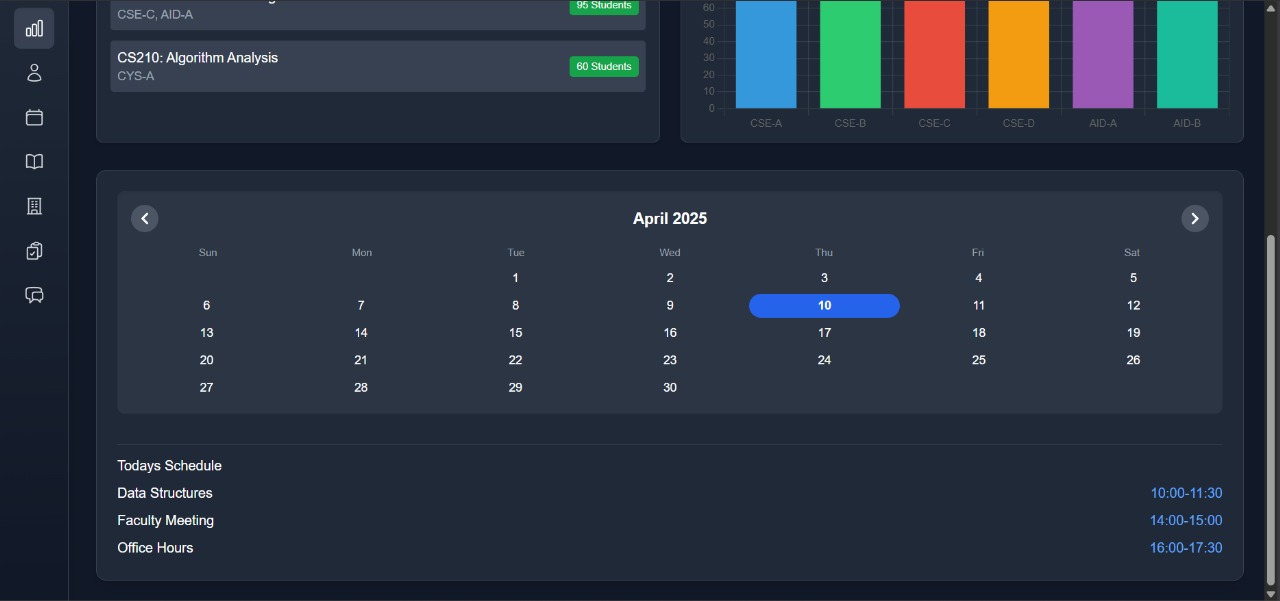


The student profile management interface allows learners to maintain their personal and academic information. Students can update contact details, communication preferences, privacy settings, and personal identifiers while viewing their institutional status information.

**TEACHER UI:**

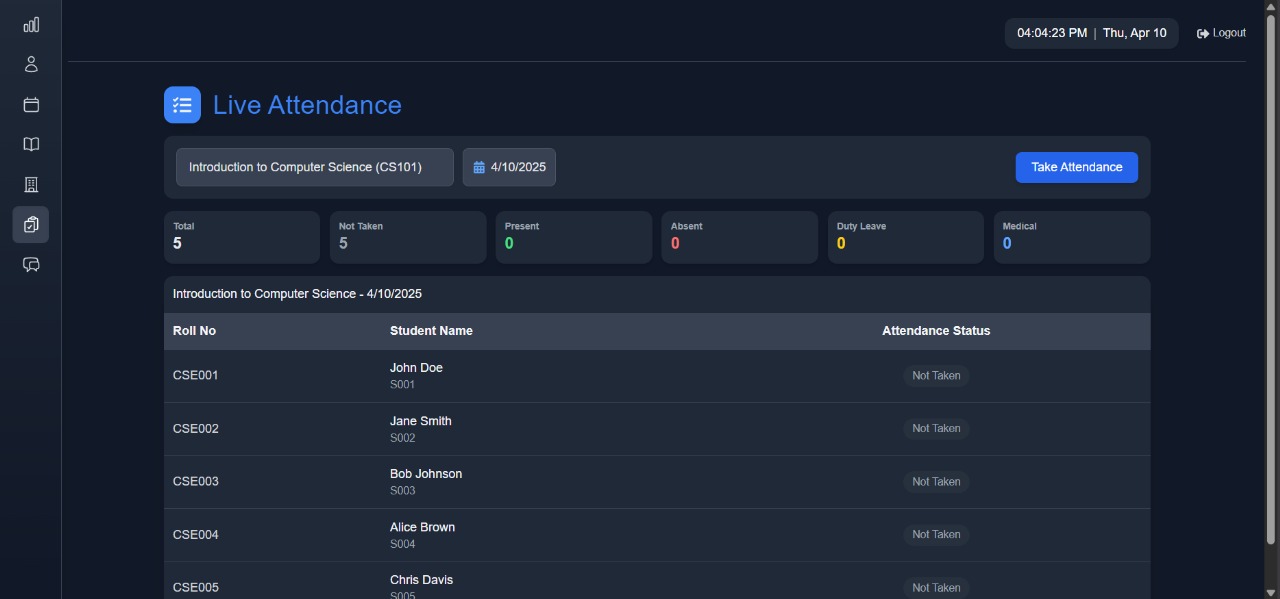
TEACHER-DASHBOARD





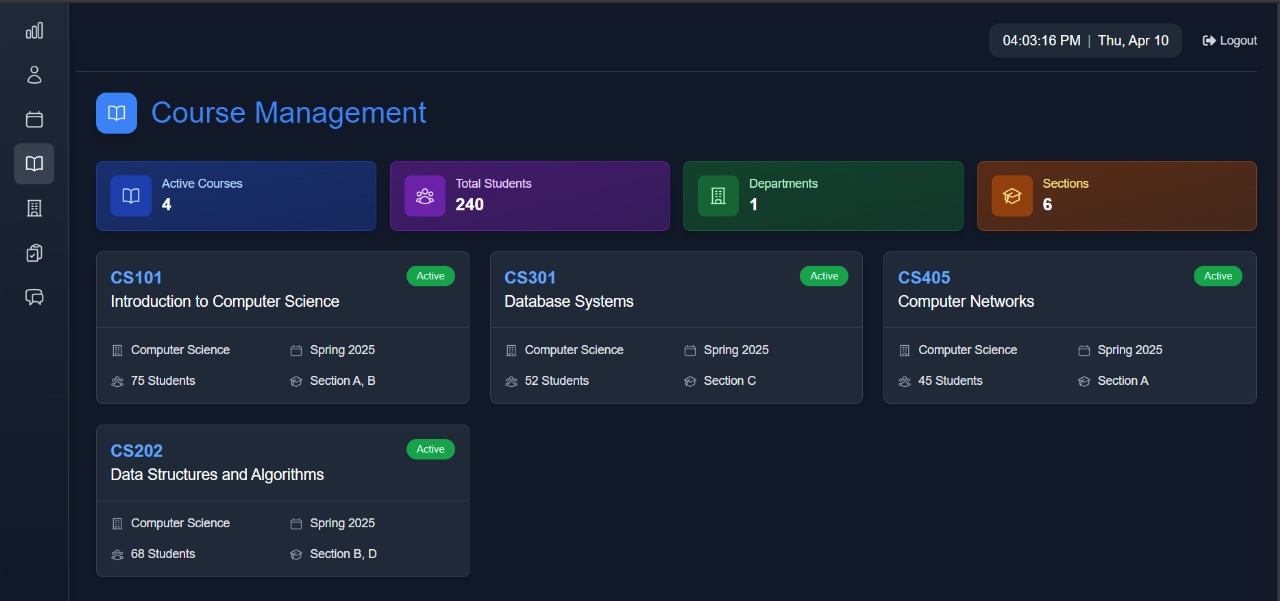
The faculty portal presents instructors with a consolidated view of their teaching responsibilities and institutional communications. This interface displays course assignments, student rosters, upcoming deadlines, departmental notifications, and quick-access links to teaching resources.

TEACHER-ATTENDANCE



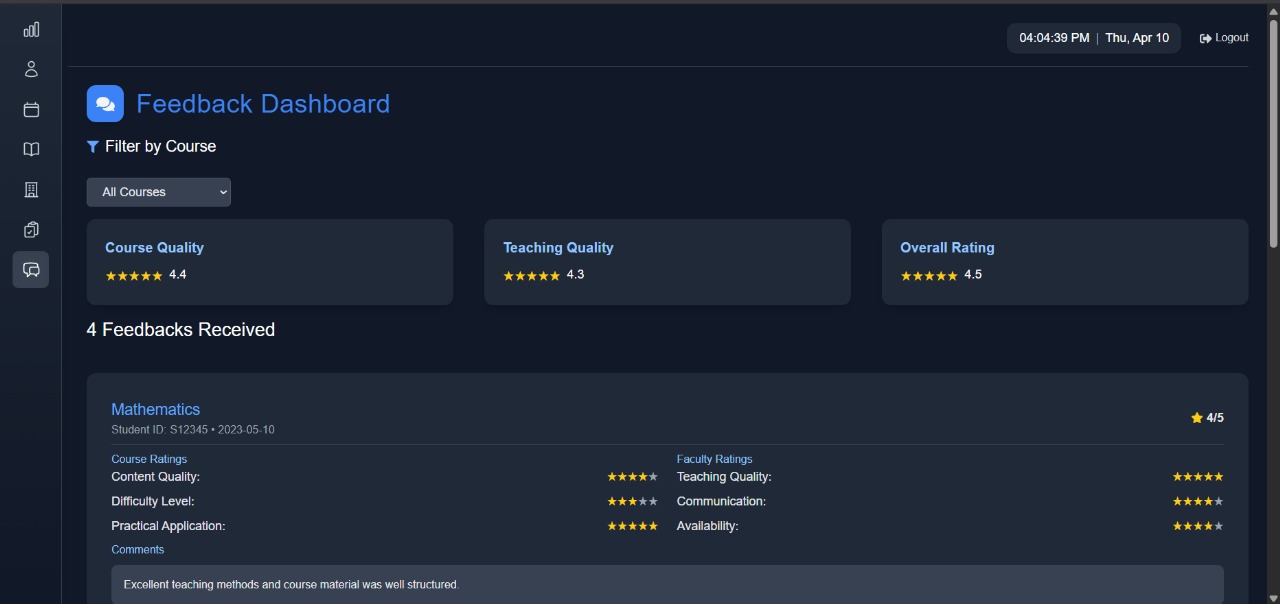
This attendance management system allows faculty to record and monitor student participation across assigned courses. Teachers can input attendance data, generate participation reports, identify attendance patterns, and communicate with students regarding attendance concerns.

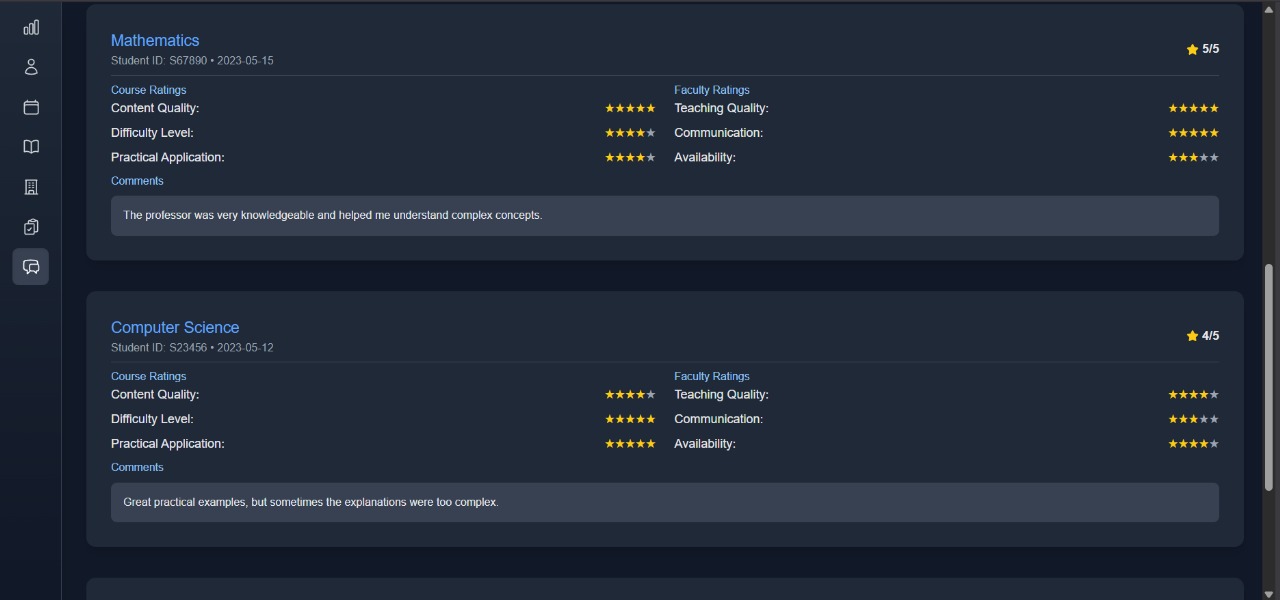
TEACHER-COURSES



The course administration interface provides faculty with comprehensive management tools for their assigned courses. Instructors can access syllabus documents, student enrollment lists, assessment schedules, and digital learning resource repositories for each course under their supervision.

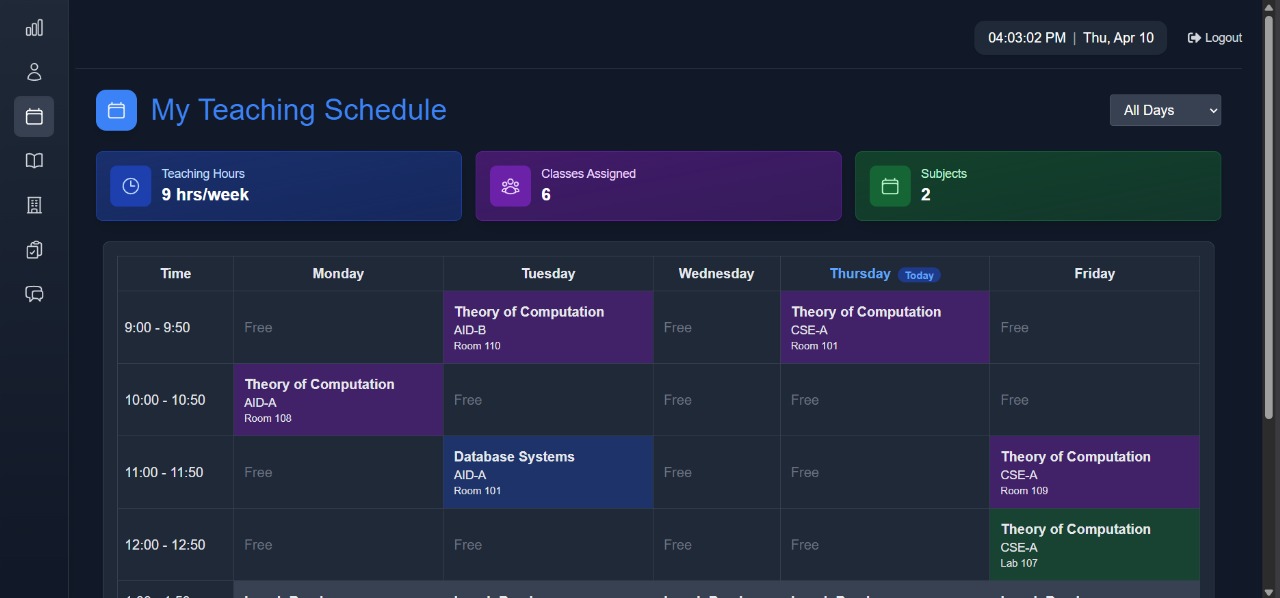
TEACHER-FEEDBACK





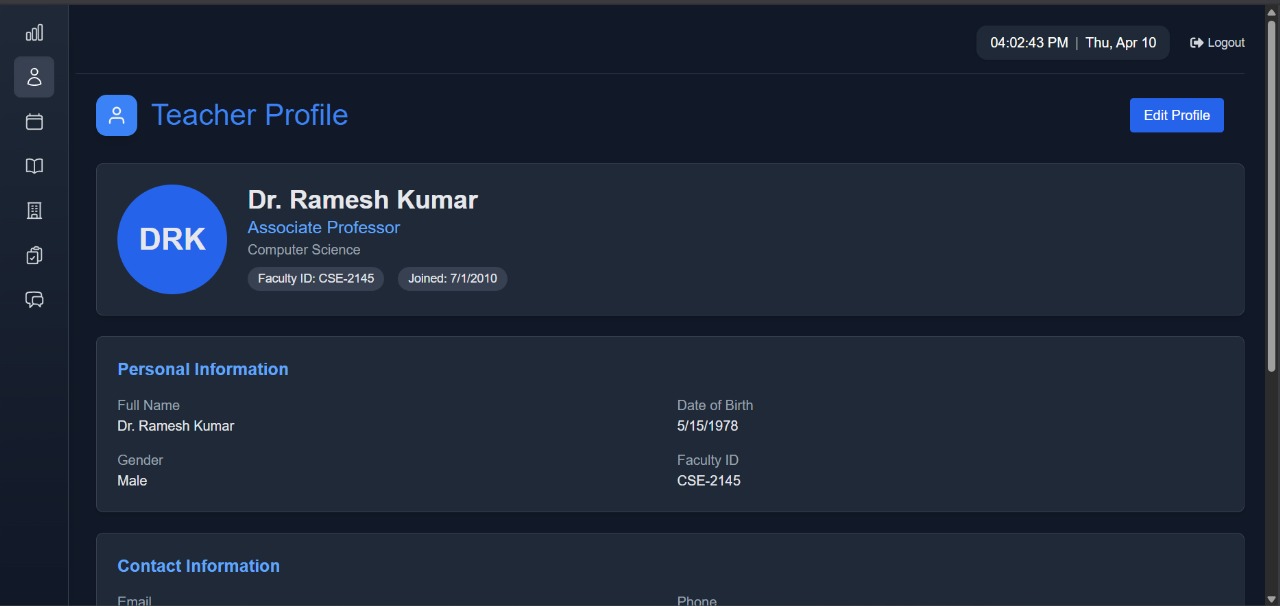
This faculty feedback system facilitates the collection and review of student evaluations. Instructors can access anonymized student feedback, historical evaluation trends, qualitative comments, and institutional benchmarks regarding teaching effectiveness.

TEACHER-TIMETABLE



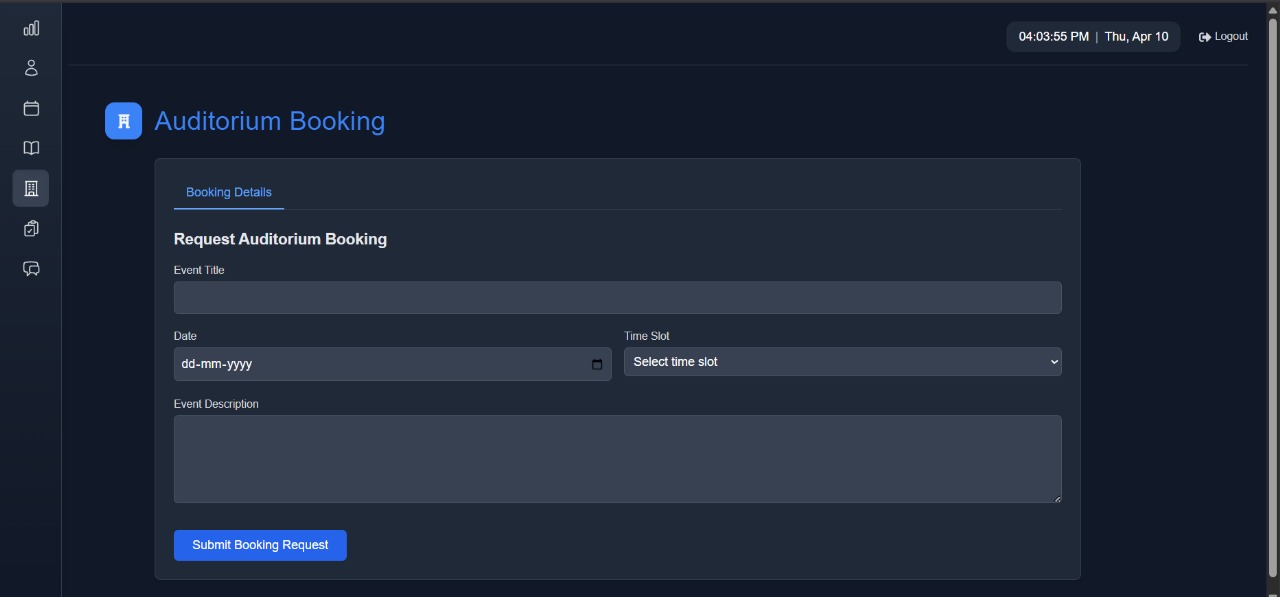
The faculty scheduling interface displays personalized teaching assignments, classroom allocations, and institutional calendar information. Teachers can view their scheduled courses, non-teaching responsibilities, and potential scheduling conflicts within this personal timetable system.

TEACHER-PROFILE



The faculty profile management module allows instructors to maintain their professional information within the institutional database. Teachers can update credentials, research interests, office hours, contact preferences, and biographical information visible to students and colleagues.

TEACHER-AUDITORIUM



This specialized reservation interface allows faculty members to request and manage auditorium bookings for academic purposes. Teachers can view availability calendars, submit facility requests, specify technical requirements, and track the approval status of their auditorium reservations.

**List of Tables**

* 1. **User**

The User model serves as the foundation for authentication and authorization within the system. It stores essential user credentials and links to specific roles through relationships with Student, Teacher, and Admin models.

### Schema

model User {

id String @id @default(cuid())

name String?

email String @unique

password String

role Role @default(STUDENT)

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

admin Admin?

student Student?

teacher Teacher?

}

* 1. **Student**

The Student model contains detailed information about enrolled students. It maintains academic records, personal details, and relationships with courses, attendance, and borrowed books.

### Schema

model Student {

id String @id @default(cuid())

userId String @unique

studentId String @unique

name String

dob DateTime

gender String @db.Char(1) // Using single character M/F/O

departmentId String

batch String @db.VarChar(10)

semester Int @db.SmallInt

joinDate DateTime

contact String @db.VarChar(15) // Changed to String to handle phone numbers properly

academicInfo Json? // Efficient storage of complex data

attendance Attendance[]

borrowedBooks BorrowedBook[]

enrollments Enrollment[]

department Department @relation(fields: [departmentId], references: [id])

user User @relation(fields: [userId], references: [id], onDelete: Cascade)

}

* 1. **Teacher**

The Teacher model stores faculty information including personal details, professional background, and teaching assignments. It links teachers to courses, schedules, and departments.

### Schema

model Teacher {

id String @id @default(cuid())

userId String @unique

teacherId String @unique @map("facultyId")

name String

dob DateTime

gender String @db.Char(1) // Using single character M/F/O

departmentId String

designation String @db.VarChar(50)

specialization String @db.VarChar(100)

joinDate DateTime

contact String @db.VarChar(15) // Changed to String for phone numbers

professional Json?

auditoriumBookings AuditoriumBooking[]

feedback Feedback[]

schedules Schedule[]

department Department @relation(fields: [departmentId], references: [id])

user User @relation(fields: [userId], references: [id], onDelete: Cascade)

teachingCourses TeacherCourse[]

}

* 1. **Admin**

The Admin model represents system administrators with elevated privileges to manage the university system. Each admin is linked to a User record for authentication.

### Schema

model Admin {

id String @id @default(cuid())

userId String @unique

name String

adminId String @unique

user User @relation(fields: [userId], references: [id], onDelete: Cascade)

}

* 1. **Department**

The Department model organizes the academic structure of the university. It groups related courses, sections, and faculty members under specific academic disciplines.

### Schema

model Department {

id String @id @default(cuid())

name String @unique

code String @unique

courses Course[]

sections Section[]

students Student[]

teachers Teacher[]

}

* 1. **Course**

The Course model contains information about academic offerings. It stores details about course content, credits, and relationships with departments, enrollments, and schedules.

### Schema

model Course {

id String @id @default(cuid())

code String @unique

name String

credits Int

departmentId String

semester Int

attendance Attendance[]

department Department @relation(fields: [departmentId], references: [id])

enrollments Enrollment[]

feedback Feedback[]

schedules Schedule[]

teacherCourses TeacherCourse[]

}

* 1. **Enrollment**

The Enrollment model tracks student registration in courses. It manages academic progress by storing grades, status, and semester information for each student-course relationship.

### Schema

model Enrollment {

id String @id @default(cuid())

studentId String

courseId String

year Int

semester Int

grade String?

gradePoint Float?

status String @default("In Progress")

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

course Course @relation(fields: [courseId], references: [id])

student Student @relation(fields: [studentId], references: [id])

@@unique([studentId, courseId, year, semester])

}

* 1. **Teacher Course**

The TeacherCourse model establishes the many-to-many relationship between teachers and courses. It specifies which teacher instructs which course for a particular section, year, and semester.

### Schema

model TeacherCourse {

id String @id @default(cuid())

teacherId String

courseId String

section String

year Int

semester Int

course Course @relation(fields: [courseId], references: [id])

teacher Teacher @relation(fields: [teacherId], references: [id])

@@unique([teacherId, courseId, section, year, semester])

}

* 1. **Attendance**

The Attendance model records student presence in courses. It tracks attendance status by date, allowing for detailed attendance monitoring throughout the academic term.

### Schema

model Attendance {

id String @id @default(cuid())

studentId String

courseId String

date DateTime

status String

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

course Course @relation(fields: [courseId], references: [id])

student Student @relation(fields: [studentId], references: [id])

@@unique([studentId, courseId, date])

}

* 1. **Feedback**

The Feedback model captures student evaluations of courses and teachers. It stores detailed ratings across multiple dimensions and allows for qualitative comments.

### Schema

model Feedback {

id String @id @default(cuid())

studentId String? // Using studentId consistently

teacherId String

courseId String

date DateTime @default(now())

courseRating Json // { content: float, assignments: float, difficulty: float }

teacherRating Json // { knowledge: float, teaching: float, accessibility: float }

overallRating Float

comments String?

course Course @relation(fields: [courseId], references: [id])

teacher Teacher @relation(fields: [teacherId], references: [id])

}

* 1. **Book**

The Book model manages the university library inventory. It tracks book details, availability, and location within the library system.

### Schema

model Book {

id String @id @default(cuid())

title String

author String

category String

isbn String @unique

publisher String

copies Int

available Int

location String

borrowedBooks BorrowedBook[]

}

* 1. **Borrowed Book**

The BorrowedBook model tracks library lending activities. It records when books are borrowed and returned, maintaining the status of each transaction.

### Schema

model BorrowedBook {

id String @id @default(cuid())

bookId String

studentId String

borrowDate DateTime @default(now())

dueDate DateTime

returnDate DateTime?

status String

book Book @relation(fields: [bookId], references: [id])

student Student @relation(fields: [studentId], references: [id])

@@unique([bookId, studentId, borrowDate])

}

* 1. **Auditorium**

The Auditorium model catalogs venues for academic and extracurricular events. It stores details about location, capacity, and available amenities.

### Schema

model Auditorium {

id String @id @default(cuid())

name String @unique

location String

capacity Int

hasWhiteboard Boolean @default(false)

status String

statusNote String?

amenities String[]

bookings AuditoriumBooking[]

slots AvailabilitySlot[]

}

* 1. **Auditorium Booking**

The AuditoriumBooking model manages reservations for auditorium spaces. It tracks who has booked a venue, for what purpose, and when.

### Schema

model AuditoriumBooking {

id String @id @default(cuid())

title String

auditoriumId String

teacherId String

date DateTime

timeSlot String

description String

status String

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

auditorium Auditorium @relation(fields: [auditoriumId], references: [id])

teacher Teacher @relation(fields: [teacherId], references: [id])

}

* 1. **Auditorium Availability**

The AvailabilitySlot model tracks when auditoriums are available for booking. It helps prevent scheduling conflicts by maintaining a record of available time slots.

### Schema

model AvailabilitySlot {

id String @id @default(cuid())

auditoriumId String

date DateTime

timeSlot String

isAvailable Boolean @default(true)

auditorium Auditorium @relation(fields: [auditoriumId], references: [id])

@@unique([auditoriumId, date, timeSlot])

}

* 1. **Timeslot**

The TimeSlot model defines standard time periods. It provides consistent time blocks that can be referenced for timetable.

### Schema

model TimeSlot {

id String @id @default(cuid())

startTime String

endTime String

name String?

schedules Schedule[]

}

* 1. **Academic Year**

The AcademicYear model defines the university's academic calendar. It establishes the timeframe for scheduling classes and other academic activities.

### Schema

model AcademicYear {

id String @id @default(cuid())

name String

startDate DateTime

endDate DateTime

isActive Boolean @default(false)

schedules Schedule[]

}

* 1. **Schedule**

The Schedule model organizes when and where classes take place. It coordinates the complex relationships between courses, teachers, rooms, and time slots.

### Schema

model Schedule {

id String @id @default(cuid())

day Day

timeSlotId String

sectionId String

courseId String

teacherId String

academicYearId String

semester Int

roomNumber String

type ScheduleType @default(LECTURE)

isBreak Boolean @default(false)

academicYear AcademicYear @relation(fields: [academicYearId], references: [id])

course Course @relation(fields: [courseId], references: [id])

section Section @relation(fields: [sectionId], references: [id])

teacher Teacher @relation(fields: [teacherId], references: [id])

timeSlot TimeSlot @relation(fields: [timeSlotId], references: [id])

@@unique([day, timeSlotId, sectionId, academicYearId, semester])

}

* 1. **Section**

The Section model organizes students into specific class groups. It facilitates targeted scheduling and academic management for cohorts within departments.

### Schema

model Section {

id String @id @default(cuid())

name String

departmentId String

batch String

schedules Schedule[]

department Department @relation(fields: [departmentId], references: [id])

@@unique([name, departmentId, batch])

}

* 1. **Role**

The Role enum defines user permission levels within the system. It establishes a clear hierarchy for access control.

### Schema

enum Role {

STUDENT

TEACHER

ADMIN

}

* 1. **Day**

The Day enum represents days of the week for scheduling purposes. It provides a standardized way to reference weekdays in the schedule.

### Schema

enum Day {

MONDAY

TUESDAY

WEDNESDAY

THURSDAY

FRIDAY

SATURDAY

SUNDAY

}

* 1. **Schedule Type**

The ScheduleType enum categorizes different types of academic sessions. It distinguishes between various teaching and non-teaching activities.

### Schema

enum ScheduleType {

LECTURE

LAB

TUTORIAL

PROJECT

BREAK

}

**RESULT & CONCLUSION :**

Thus an elegant fully-functional website has been created which supports the collaboration of the student , teacher and admins.

**REFERENCES :**

GITHUB REPOSITORY OF THE PROJECT :

[https://github.com/GowthamDhanaraju/Sem4\_UMS](https://github.com/GowthamDhanaraju/Sem4_UMS%0c)

**Bibliography :**

[1] Sharma, R., & Patel, V. (2020). Evolution of Educational Management Systems: A Comprehensive Review. Journal of Educational Technology, 42(3), 215-231.

[2] Martinez, C., Johnson, P., & Lee, S. (2022). Quantifying Administrative Efficiency Gains Through Integrated Management Systems in Higher Education. International Journal of Educational Management, 36(2), 178-193.

[3] Wang, L., & Ibrahim, M. (2021). Impact of Integrated Management Systems on Student Satisfaction and Academic Transparency. Higher Education Quarterly, 75(1), 42-58.

[4] Kumar, A., & Zhao, Y. (2023). Evaluating Technology Stacks for Modern Academic Information Systems. Journal of Computing in Higher Education, 35(2), 312-327.

[5] Okonkwo, C., & Garcia, R. (2021). Multi-layered Security Frameworks for Educational Management Systems. Journal of Information Security Applications, 58, 102717.

[6] Thompson, K., Davis, M., & Wilson, J. (2024). Analytics Integration in Educational Management Systems: Enabling Data-Driven Institutional Decision Making. Education and Information Technologies, 29(1), 45-63.

[7] Nguyen, T., & Roberts, P. (2023). Modular Architecture Principles for Scalable Educational Platforms. Software Engineering for Education, 12(3), 215-230.