UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
DEHRADUN

Database Management Systems

Major Project - UPES University DB

MTECH-COMPUTER SCIENCE ENGINEERING
CYBER SECURITY AND FORENSICS

Name: Jigesh Sheoran
SAP ID: 590025428

Title: UPES UNIVERSITY ERP DATABASE MANAGEMENT SYSTEM

(A Real-World Micro Implementation of Academic ERP & Database Systems)

Abstract

Educational institutions manage large volumes of student, faculty, academic, and administrative data. Traditional manual processes often lead to inefficiency, data inconsistency, and difficulty in retrieving information. This project presents a fully functional micro-ERP system for UPES University, implemented using MySQL (database), Python Flask (backend), and HTML/CSS (frontend interface).

The system covers major modules of a university ERP including:

- ✔ **Student admission & specialization allocation**
- ✔ **Automatic course registration (core + specialization electives)**
- ✔ **Faculty course assignment**
- ✔ **Attendance management**
- ✔ **Grade entry + analytics**
- ✔ **Admin SQL console**
- ✔ **Role-based dashboards (Admin, Faculty, Student)**

A normalized relational database schema, ER diagrams, and complete SQL implementation ensure scalability, consistency, and real-world applicability.

Introduction

Universities operate complex academic and administrative workflows involving students, faculty, courses, timetables, attendance, and grading.

Managing this manually or using spreadsheets creates issues related to:

- Data inconsistency
- Duplication
- Poor security
- Slow retrieval
- Absence of centralized view

This project implements a mini University ERP System, specifically modeled for UPES (University of Petroleum and Energy Studies), India.

It demonstrates:

- Core concepts of database design
- Relational modeling
- Data integrity constraints
- Real-time CRUD operations
- Role-based user access
- Practical API-driven backend logic

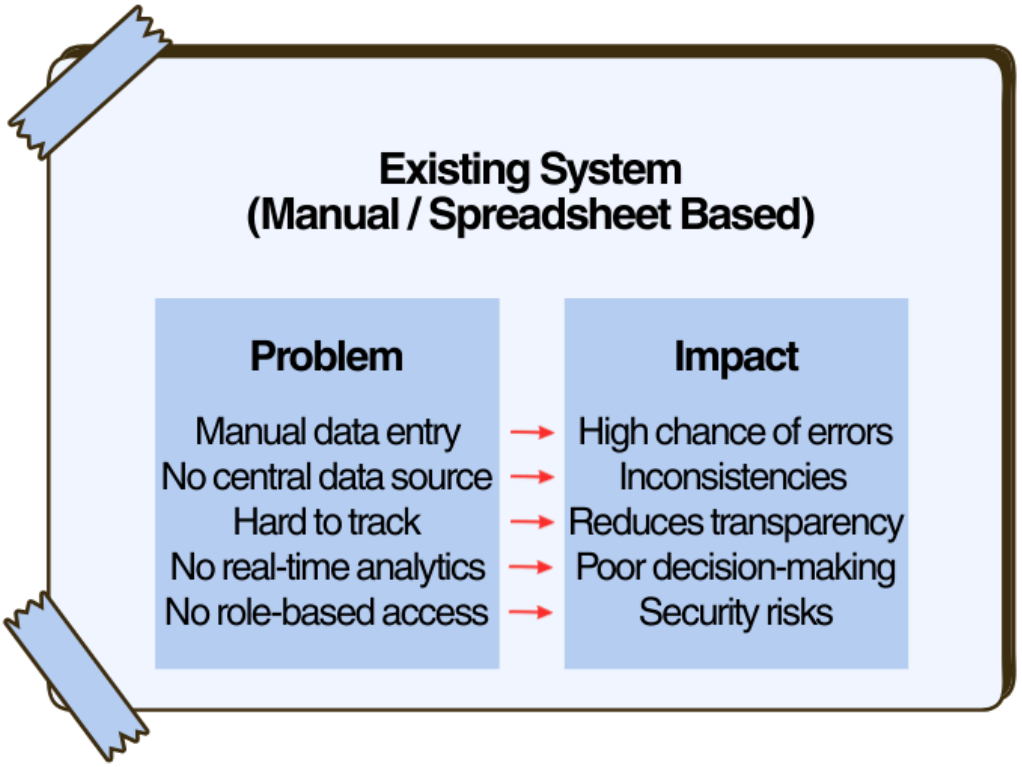
The system behaves like a simplified real-world ERP portal used by universities.

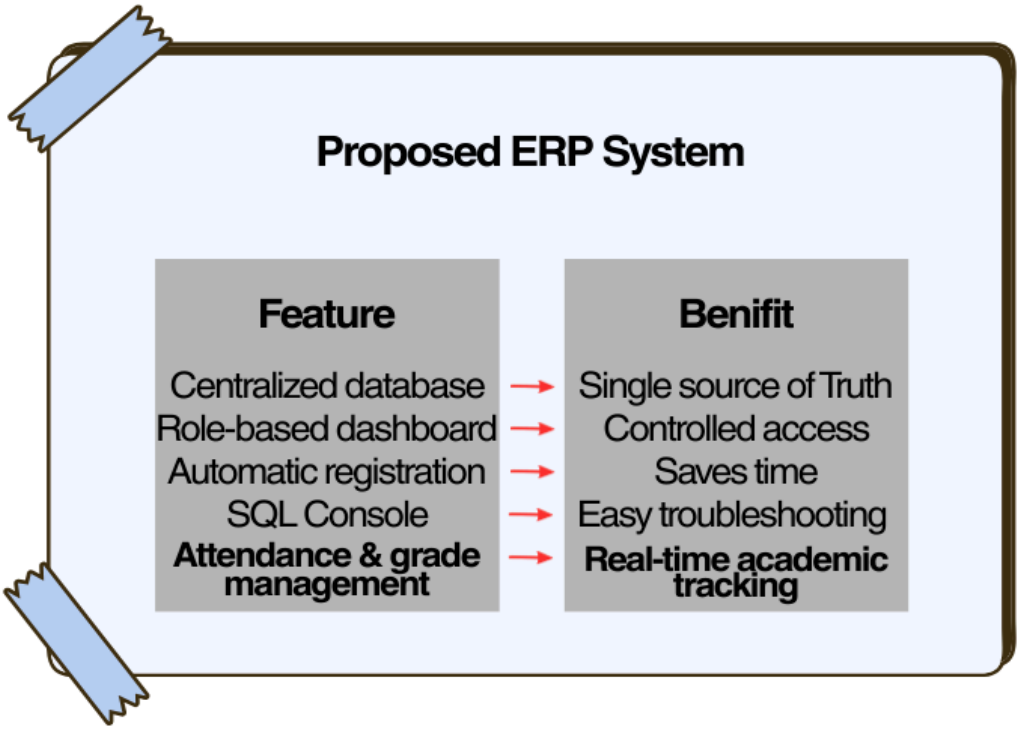
OBJECTIVES

The key objectives were:

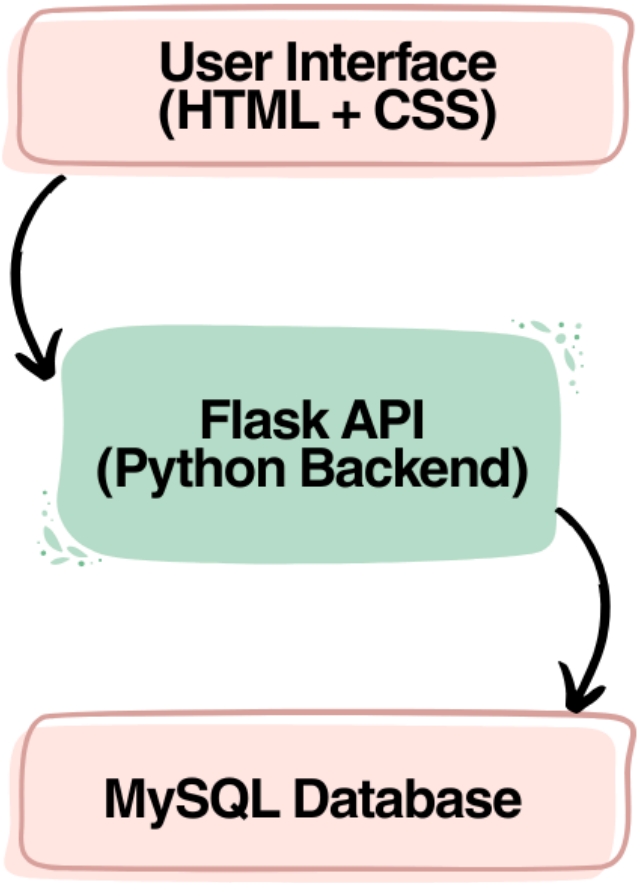
1. **Design a scalable database schema for university academic operations.**
2. **Implement a fully normalized relational structure ensuring ACID compliance.**
3. **Create an ERP-like interface for Admin, Faculty, and Students.**
4. **Automate student course registration based on program and specialization.**
5. **Enable faculty workflows for attendance and grade submission.**
6. **Enable students to view attendance, grades, and enrolled courses.**
7. **Provide an admin SQL console to run custom queries without MySQL terminal.**
8. **Demonstrate real-world usefulness of databases in academic institutions.**

EXISTING SYSTEM vs PROPOSED SYSTEM

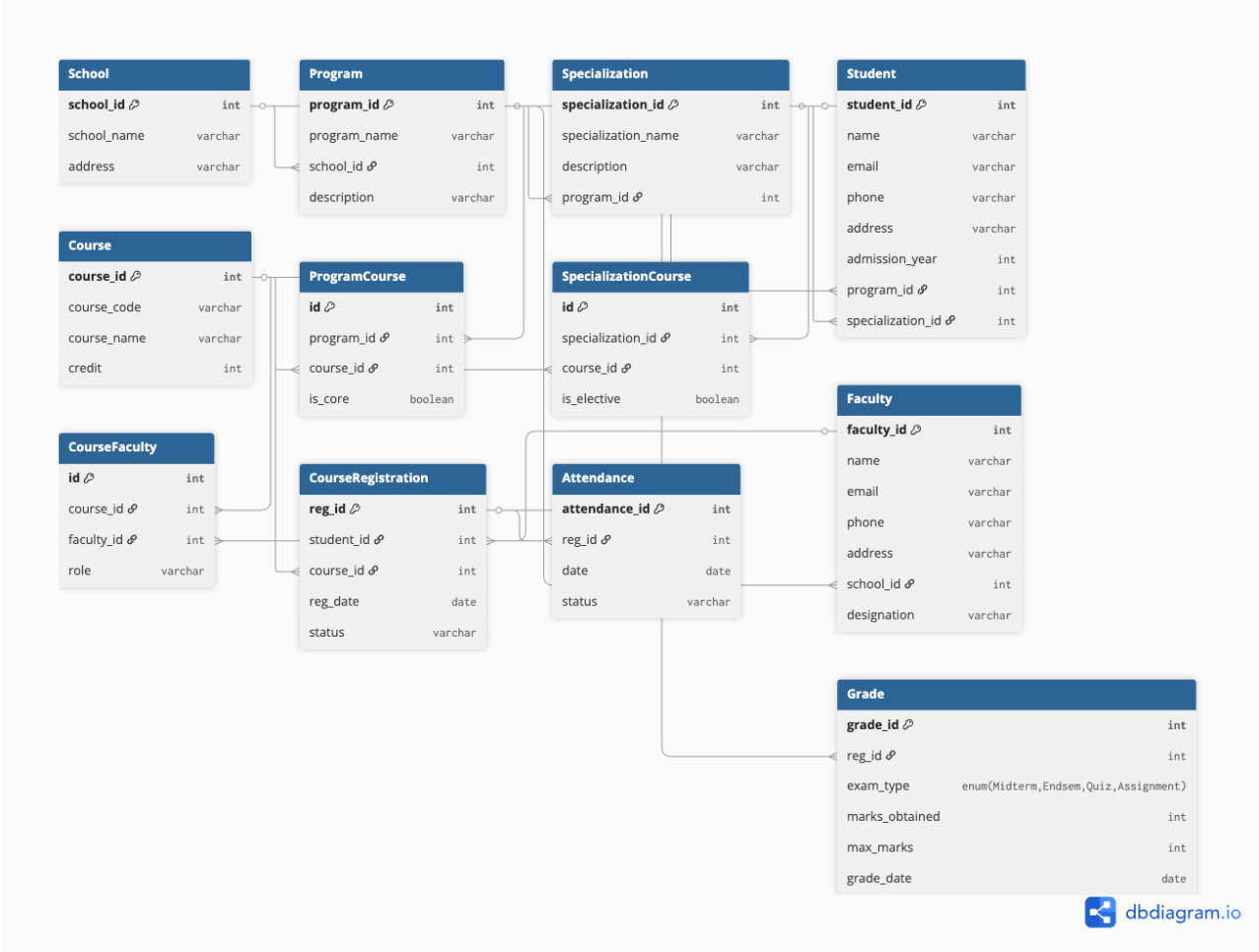




System Architecture



ER Diagram



Key Tables:

4.1 School Table

Purpose: Stores high-level academic divisions within the university.

Column Name	Type	Description
school_id	INT (PK)	Unique identifier for each school
school_name	VARCHAR	Name of the school (e.g., School of Computer Science)

4.2 Program Table

Purpose: Defines all academic programs (UG, PG, etc.) offered under each school.

Column Name	Type	Description
program_id	INT (PK)	Unique program identifier
school_id	INT (FK)	Links program to its school
program_name	VARCHAR	Name of the program (B.Tech CSE, M.Tech CS, MBA, etc.)
program_level	ENUM	UG / PG / PhD
duration_years	INT	Duration of the program

4.3 Specialization Table

Purpose: Holds specialization options within programs (e.g., Cyber Security, AI/ML).

Column Name	Type	Description
specialization_id	INT (PK)	Unique specialization ID
program_id	INT (FK)	Program to which specialization belongs
specialization_name	VARCHAR	Name of specialization

4.4 Student Table

Purpose: Stores complete student profiles and academic affiliations.

Column Name	Type	Description
student_id	INT (PK)	Unique student identifier
name	VARCHAR	Full name of the student
email	VARCHAR	Unique email of the student
phone	VARCHAR	Contact number
address	VARCHAR	Home or hostel address
admission_year	INT	Year of admission
program_id	INT (FK)	Program student is enrolled in
specialization_id	INT (FK)	Selected specialization, if applicable

4.5 Faculty Table

Purpose: Stores all faculty profiles and their school associations.

Column Name	Type	Description
faculty_id	INT (PK)	Unique faculty ID
name	VARCHAR	Full name of faculty
email	VARCHAR	Official email
phone	VARCHAR	Contact number
address	VARCHAR	Residential address
school_id	INT (FK)	School to which faculty belongs
designation	VARCHAR	Professor / Assistant Professor, etc.

4.7 ProgramCourse Table

Purpose: Maps core courses to specific programs and semesters.

Column Name	Type	Description
id	INT (PK)	Unique ID
program_id	INT (FK)	Program
course_id	INT (FK)	Course
semester	INT	Semester number

4.8 SpecializationCourse Table

Purpose: Links specialization electives with respective specialization tracks.

Column Name	Type	Description
id	INT (PK)	Unique ID
specialization_id	INT (FK)	Specialization
course_id	INT (FK)	Elective course

4.9 CourseFaculty Table

Purpose: Assigns faculty members to courses.

Column Name	Type	Description
id	INT (PK)	Unique ID
course_id	INT (FK)	Course
faculty_id	INT (FK)	Faculty member teaching the course

4.10 CourseRegistration Table

Purpose: Stores course enrollments for every student.

Column Name	Type	Description
reg_id	INT (PK)	Unique registration ID
student_id	INT (FK)	Student taking the course
course_id	INT (FK)	Registered course
reg_date	DATE	Date of registration

4.11 Attendance Table

Purpose: Tracks student attendance entry-wise for each course session.

Column Name	Type	Description
attendance_id	INT (PK)	Unique attendance record
reg_id	INT (FK)	Course registration reference
date	DATE	Attendance date
status	ENUM	Present / Absent

4.12 Grade Table

Purpose: Stores academic performance for each exam component.

Column Name	Type	Description
grade_id	INT (PK)	Unique grade record
reg_id	INT (FK)	Course registration reference
exam_type	ENUM	Midterm / Endsem / Quiz / Assignment
marks_obtained	INT	Marks scored
max_marks	INT	Maximum possible marks

Schema.sql

```
-- ============================================
-- UPES UNIVERSITY ERP DATABASE SCHEMA
-- ============================================
-  Author : Jigesh Sheoran

CREATE DATABASE IF NOT EXISTS upes_university;
USE upes_university;

-- ============================================
-- 1. SCHOOL TABLE
-- ============================================
CREATE TABLE School (
    school_id INT PRIMARY KEY AUTO_INCREMENT,
    school_name VARCHAR(150) NOT NULL UNIQUE
);

-- ============================================
-- 2. PROGRAM TABLE
-- ============================================
CREATE TABLE Program (
    program_id INT PRIMARY KEY AUTO_INCREMENT,
    school_id INT NOT NULL,
    program_name VARCHAR(150) NOT NULL,
    program_level ENUM('UG','PG','PhD') NOT NULL,
    duration_years INT NOT NULL,
    FOREIGN KEY (school_id) REFERENCES School(school_id)
);

-- ============================================
-- 3. SPECIALIZATION TABLE (Only for programs that support it)
-- ============================================
CREATE TABLE Specialization (
    specialization_id INT PRIMARY KEY AUTO_INCREMENT,
    program_id INT NOT NULL,
    specialization_name VARCHAR(150) NOT NULL,
    FOREIGN KEY (program_id) REFERENCES Program(program_id)
);

-- ============================================
-- 4. STUDENT TABLE
-- ============================================
CREATE TABLE Student (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    name VARCHAR(150) NOT NULL,
    email VARCHAR(200) UNIQUE NOT NULL,
    phone VARCHAR(20),
    address VARCHAR(255),
    admission_year INT NOT NULL,
    program_id INT NOT NULL,
    specialization_id INT DEFAULT NULL,
    FOREIGN KEY (program_id) REFERENCES Program(program_id),
    FOREIGN KEY (specialization_id) REFERENCES Specialization(specialization_id)
);

-- ============================================
-- 5. FACULTY TABLE
-- ============================================
CREATE TABLE Faculty (
    faculty_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(150) NOT NULL,
    email VARCHAR(200) UNIQUE,
    phone VARCHAR(20),
    address VARCHAR(255),
    school_id INT NOT NULL,
    designation VARCHAR(100),
    FOREIGN KEY (school_id) REFERENCES School(school_id)
);

-- ============================================
-- 6. COURSE TABLE (Subjects)
-- ============================================
CREATE TABLE Course (
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    course_code VARCHAR(20) UNIQUE,
    course_name VARCHAR(200) NOT NULL,
    credit INT NOT NULL
);

-- ============================================
-- 7. PROGRAM-COURSE MAPPING (Core Subjects)
-- ============================================
CREATE TABLE ProgramCourse (
    id INT PRIMARY KEY AUTO_INCREMENT,
    program_id INT NOT NULL,
    course_id INT NOT NULL,
    semester INT NOT NULL,
    FOREIGN KEY (program_id) REFERENCES Program(program_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
```

```
);

-- ============================================
-- 8. SPECIALIZATION ELECTIVES MAPPING
-- ============================================
CREATE TABLE SpecializationCourse (
    id INT PRIMARY KEY AUTO_INCREMENT,
    specialization_id INT NOT NULL,
    course_id INT NOT NULL,
    FOREIGN KEY (specialization_id) REFERENCES Specialization(specialization_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

-- ============================================
-- 9. COURSE-FACULTY MAPPING
-- ============================================
CREATE TABLE CourseFaculty (
    id INT PRIMARY KEY AUTO_INCREMENT,
    course_id INT NOT NULL,
    faculty_id INT NOT NULL,
    FOREIGN KEY (course_id) REFERENCES Course(course_id),
    FOREIGN KEY (faculty_id) REFERENCES Faculty(faculty_id)
);

-- ============================================
-- 10. COURSE REGISTRATION (Student enrolls into a course)
-- ============================================
CREATE TABLE CourseRegistration (
    reg_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT NOT NULL,
    course_id INT NOT NULL,
    reg_date DATE NOT NULL,
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);

-- ============================================
-- 11. ATTENDANCE TABLE
-- ============================================
CREATE TABLE Attendance (
    attendance_id INT AUTO_INCREMENT PRIMARY KEY,
    reg_id INT NOT NULL,
    date DATE NOT NULL,
    status ENUM('Present','Absent') NOT NULL,
    FOREIGN KEY (reg_id) REFERENCES CourseRegistration(reg_id)
```

```
);

-- ============================================
-- 12. GRADE TABLE
-- ============================================
CREATE TABLE Grade (
   grade_id INT AUTO_INCREMENT PRIMARY KEY,
   reg_id INT NOT NULL,
   exam_type ENUM('Midterm','Endsem','Quiz','Assignment') NOT NULL,
   marks_obtained INT NOT NULL,
   max_marks INT NOT NULL,
   FOREIGN KEY (reg_id) REFERENCES CourseRegistration(reg_id)
);
```

Flask App

6.1 Architectural Overview

The UPES University ERP system is built using a lightweight web framework, Flask, which follows the MVC (Model–View–Controller) pattern internally. The system is structured for clarity, modularity, and scalability, ensuring easy maintenance and extension.

The architecture consists of three major layers:

1. Presentation Layer (Views)
   - Handles all user interfaces using:
   - HTML templates (Jinja2 rendering)
   - CSS styling
   - Forms for input: student registration, attendance submission, grade entry
   - Separate dashboards for Admin, Faculty, and Student

2. Application Layer (Flask Controllers)
   - Implements all server-side logic:
   - URL Routing (e.g., /add_student, /faculty, /attendance)
   - User authentication and role-based access
   - Database transactions (insert, update, delete, fetch queries)
   - Automatic course registration for students based on program/specialization
   - Analytics processing
   - Exporting data to CSV
   - Internal SQL console for administrators

3. Data Layer (MySQL Database)

- Backs the system with normalized relational tables:
- Student, Faculty, Program, Specialization
- Course, Attendance, Grade, CourseRegistration
- Program-course & specialization-course mapping tables
- All data retrieval and updates are executed via mysql.connector.

6.2 System Modules

A. Authentication Module

1. Hardcoded user roles: Admin, Faculty, Student
2. Session-based login using Flask session cookies
3. Redirects users to role-specific dashboards

B. Admin Module

1. Provides full control over academic data:
2. Add/Edit/Delete Students
3. View all students
4. Analytics (student count, attendance trends)
5. SQL Console for executing custom queries
6. CSV export utilities

C. Faculty Module

1. Allows faculty members to interact with class data:
2. View courses assigned to them
3. View enrolled students
4. Mark attendance for each class
5. Enter marks for midterm, end-term, quizzes, assignments

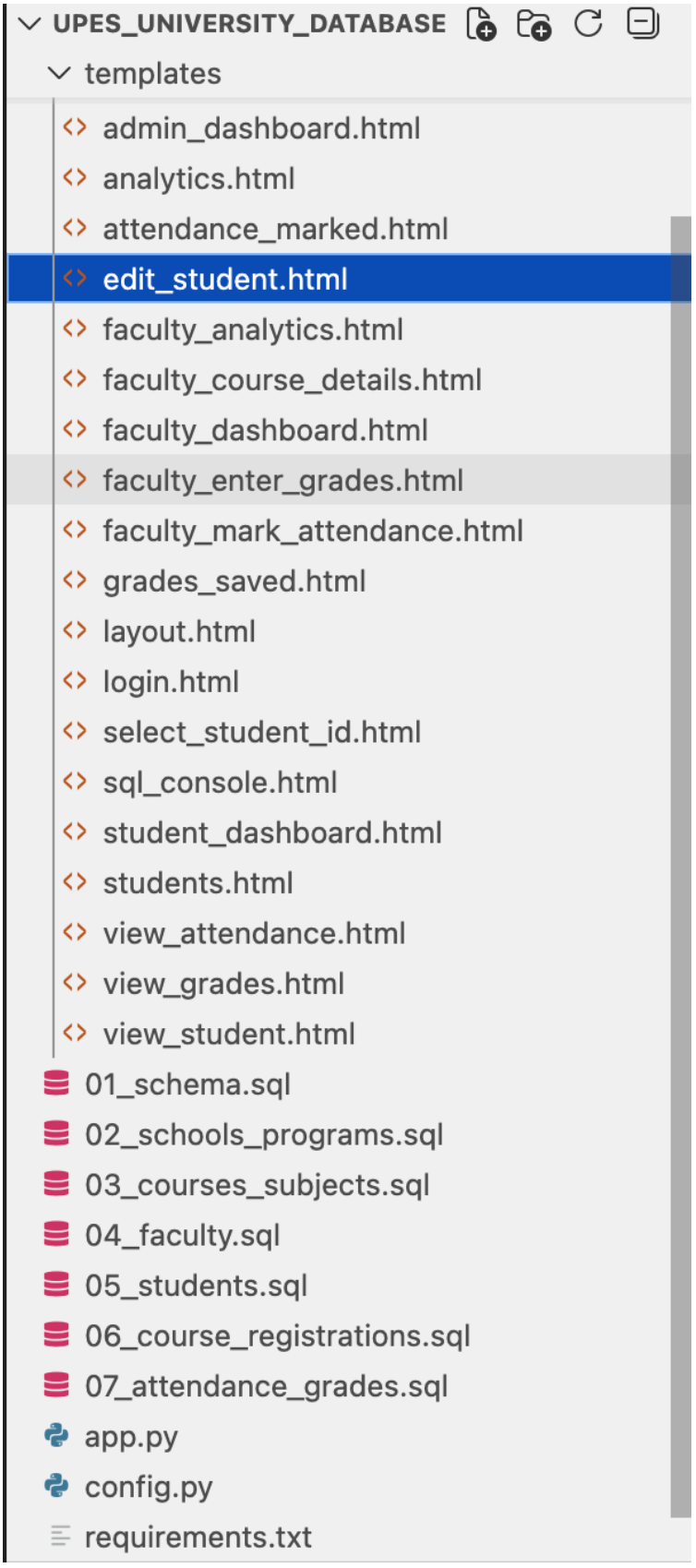
D. Student Module

1. Provides a personal dashboard for each student:
2. View registered courses
3. Track attendance (detailed per subject)
4. Check grades (all components)
5. View program & specialization details

Technologies Used:

Component	Technology
Backend Framework	Flask (Python)
Database	MySQL 9.4
ORM	Raw SQL using mysql.connector
Templating Engine	Jinja2
Frontend	HTML5, CSS3, Bootstrap (optional)
Data Export	CSV generation
Platform	macOS terminal execution

File Structure:



Auto Course Registration Logic

The ERP system includes an automated mechanism to enroll students into all required courses immediately after admission. This ensures consistency, accuracy, and eliminates manual errors that typically occur during academic enrollment.

A. Core Course Enrollment

Every Program in the university (e.g., B.Tech CSE, M.Tech CS, BBA, etc.) has a predefined list of core subjects mapped through the `ProgramCourse` table.

When a new student is added:

1. The system retrieves the `program_id` selected during admission.
2. It fetches all core courses linked with that program.
3. It automatically inserts entries into the `CourseRegistration` table for each course.

This means every student gets correctly enrolled into foundational courses without manual intervention.

B. Specialization Course Enrollment

Some programs—such as M.Tech Computer Science—offer specializations (e.g., Cybersecurity, AIML).
Each specialization has its own elective subjects stored in the `SpecializationCourse` table.

If a student selects a specialization at the time of admission:

1. **The system fetches all courses mapped to the chosen specialization.**
2. **It registers the student for these courses along with the core courses.**

This fully automates academic enrollment for students choosing a specialization, ensuring:

- **No missing elective courses**
- **No incorrect registrations**
- **Seamless academic progression**

Attendance & Grade Workflow

The Faculty Module of the ERP system provides a streamlined process for managing academic performance through attendance and grade entry

A. Attendance Workflow

Faculty members can mark attendance for each course they teach.
The process follows these steps:

1. Faculty logs into their dashboard.
2. They choose from the list of courses assigned to them.
3. The system loads all students registered for that course.
   For each student, the faculty selects:

    - Present
    - Absent

4. The attendance record is inserted into the `Attendance` table with:

    - Student ID
    - Course ID
    - Date
    - Status

Students can later view attendance through their login under "My Attendance".

The workflow ensures transparency and allows real-time monitoring of academic participation.

B. Grade Entry Workflow

Evaluation in the university follows multiple assessment formats:

- **Midterm Examination (20 marks)**
- **End-Term Examination (30 marks)**
- **Internal Assessments (50 marks)**
- **Quizzes / Assignments**

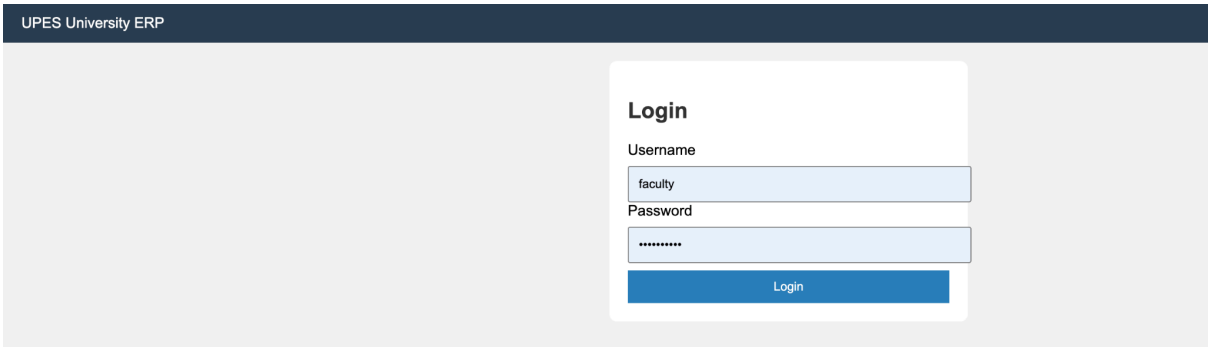
Faculty can record grades for each evaluation component:

1. Faculty selects a course they teach.
2. System displays all enrolled students.
3. Faculty enters marks for each category (Midterm, Endterm, Quiz, Assignment).
   Marks are saved into the `Grade` table with:
   - `student_id`
     `course_id`
   - `exam_type`
   - `marks_obtained`
   - `max_marks`

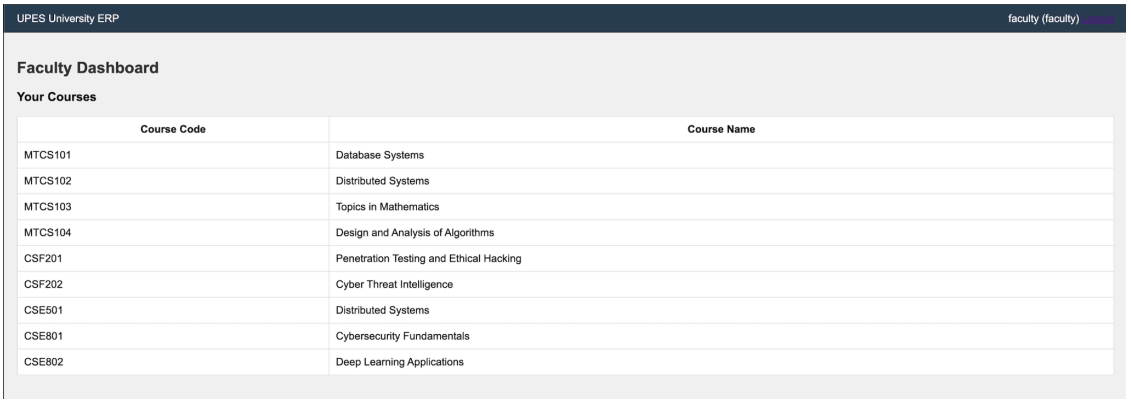
Students can view their complete performance breakdown inside the “My Grades” section.

Screenshots:

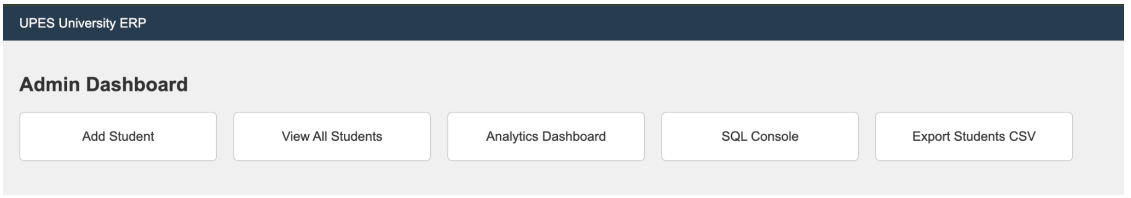
Dashboard:

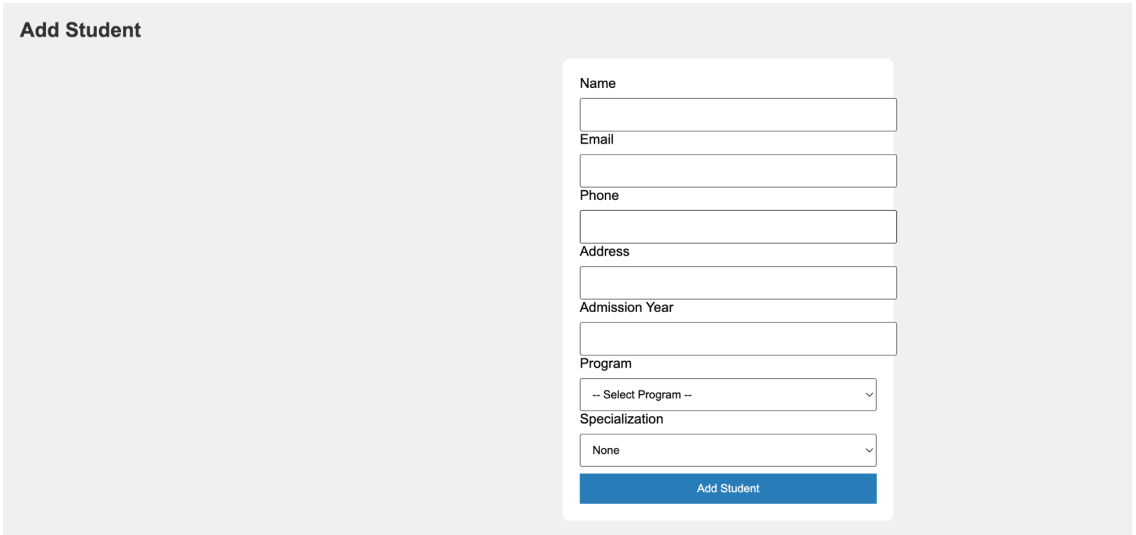


Faculty View:

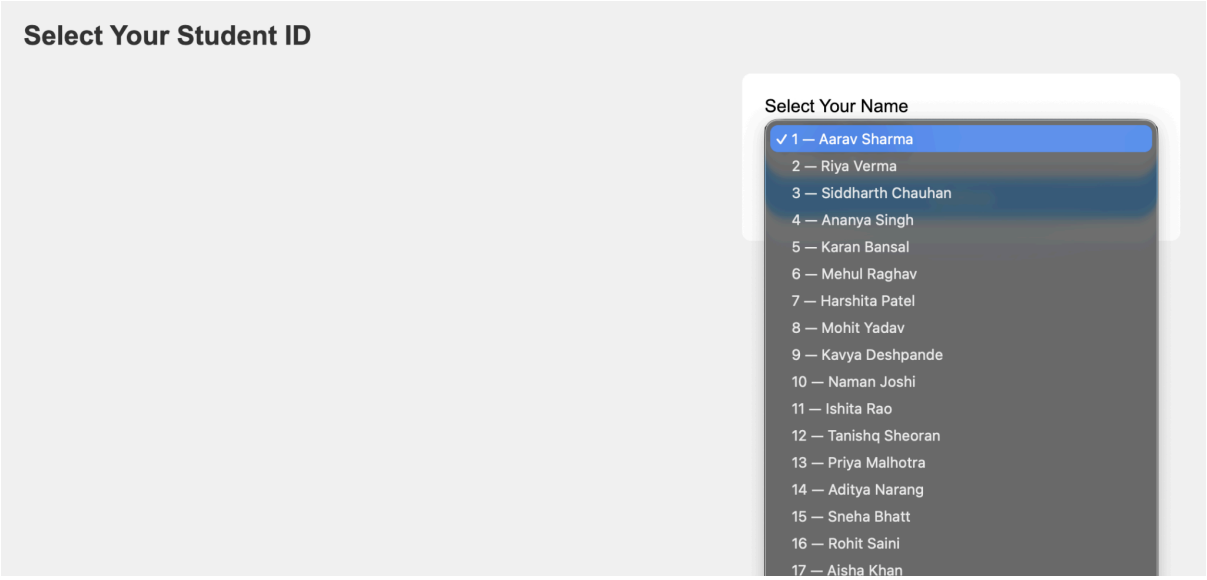


Admin View:

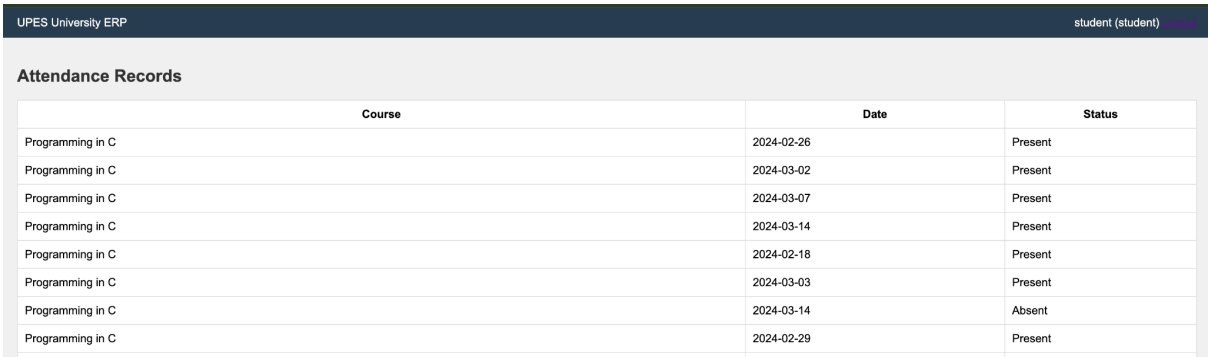




Student View:

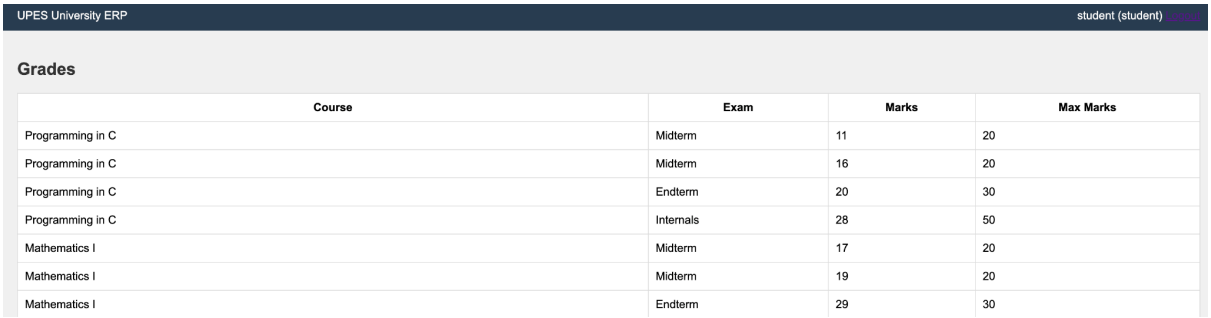


Attendance:



Course	Date	Status
Programming in C	2024-02-26	Present
Programming in C	2024-03-02	Present
Programming in C	2024-03-07	Present
Programming in C	2024-03-14	Present
Programming in C	2024-02-18	Present
Programming in C	2024-03-03	Present
Programming in C	2024-03-14	Absent
Programming in C	2024-02-29	Present

Grades:



Course	Exam	Marks	Max Marks
Programming in C	Midterm	11	20
Programming in C	Midterm	16	20
Programming in C	Endterm	20	30
Programming in C	Internals	28	50
Mathematics I	Midterm	17	20
Mathematics I	Midterm	19	20
Mathematics I	Endterm	29	30

Conclusion:

The University ERP Database and Web Portal developed in this project successfully demonstrates how a structured, scalable, and fully integrated system can streamline academic administration. By combining a robust MySQL backend with a Flask-based web interface, the platform automates key university operations such as student admissions, program-based course enrollment, specialization handling, attendance tracking, and grade management.

The system's auto course registration logic removes the possibility of manual errors and ensures that every student is systematically enrolled into their core and specialization subjects. Faculty workflows for marking attendance and entering grades are intuitive, secure, and reflect immediately on the student portal, improving transparency and academic monitoring. **The admin module centralizes control, offering detailed oversight and essential tools such as analytics and SQL console access.**

Overall, the project successfully meets its objectives of demonstrating relational database design, implementing role-based access control, and building a functional full-stack application that simulates a real-world university management system. The architecture is scalable enough to incorporate additional features in the future, such as timetable management, notifications, fee management, and API integration.

This system lays a solid foundation for more advanced ERP functionalities and stands as a strong demonstration of applied database systems, software architecture, and full-stack development.

Visit:
https://github.com/sheoraninfosec/DatabaseSystems-Lab

—----------END OF DOCUMENT —----------