# THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Deemed to be University)
Patiala, Punjab

## A Mini Project On

**"University Management System"**

**For the partial complement of Database Management System 2025**
**Under the supervision of**
**Department of Computer Science and Engineering**


**Submitted By:**

| | |
|---|---|
| **"Sajid Miya"** | **102367013** |
| **"Nikshitha"** | **102317043** |
| **"Arshdeep Singh"** | **102317064** |
| **"Manjot Singh Gill"** | **102317068** |


**Submitted To:**

**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY**
**Department of Computer Science and Engineering**
**Patiala, Punjab, India**
**"June 2025"**

# Table of Contents

# INTRODUCTION

The University Management System (UMS) is designed to efficiently manage various aspects of a university's operations, including student enrolment, faculty details, course management, exams, fee payments, and results processing. The system ensures data integrity, reduces redundancy, and improves administrative efficiency by organizing university-related data in a structured and secure manner.

This system is database-driven, making it easier for universities to manage academic records, track student progress, and streamline administrative processes. It is essential for maintaining accurate student and faculty records, ensuring smooth examination processes, and managing financial transactions related to tuition and fees.

# OBJECTIVE

The objective of the University Management System project is to develop a comprehensive web-based platform that streamlines and automates the administrative and academic processes of a university. This system aims to provide an efficient and user-friendly interface for managing student registrations, faculty assignments, course enrollments, exams, results, and fee payments. By integrating various functionalities into a single platform, the system seeks to enhance operational efficiency, reduce manual workload, and improve communication between students, faculty, and administrators.

# PROBLEM STATEMENT

Develop an efficient and robust university management system to address the comprehensive needs of academic institutions. The system should streamline operations by offering functionalities such as student and faculty management, course administration, enrollment handling, examination scheduling, fee tracking, and administrative oversight.

The solution must be scalable to accommodate the growth of students, faculty, and courses over time. It should prioritize security by incorporating authentication mechanisms and role-based access control. Performance optimization is essential to ensure quick data retrieval and minimal response times, while reliability measures are required to prevent data loss and maintain accuracy. Additionally, the system must feature a user-friendly interface for seamless interaction across diverse roles, including students, faculty, and administrators.

This project aims to deliver a centralized platform that enhances efficiency, simplifies management, and provides a secure and reliable framework for university operations.

# SCOPE OF THE PROJECT

The **University Management System** aims to provide a comprehensive, efficient, and user-friendly platform for managing various aspects of a university's operations, including student enrollment, faculty management, course administration, examination scheduling, fee processing, and result management. The scope of the project encompasses the following aspects:

**1. Functional Scope**

- **Student Management:**
  - Registration of new students with verification and approval by administrators.
  - Managing student profiles, enrollment status, and course registrations.
  - Displaying personalized dashboards with exam schedules, results, and fee status.
- **Faculty Management:**
  - Registration of faculty members and verification by administrators.
  - Managing faculty profiles, assigned courses, and examination responsibilities.
  - Providing personalized dashboards with relevant information.
- **Course Management:**
  - Creation, modification, and deletion of courses by administrators.
  - Assignment of faculty members to courses.
  - Allowing students to register for courses.
- **Examination Management:**
  - Scheduling exams and notifying students and faculty members.
  - Evaluating and storing exam results.
  - Displaying results to students and allowing administrators to update them.
- **Fee Management:**
  - Generating fee records for students based on enrollment status and courses.
  - Allowing administrators to update, view, or delete fee records.
  - Displaying pending or completed fee transactions on student dashboards.
- **Department Management:**
  - Creating, updating, or deleting departments.
  - Appointing Heads of Departments (HODs) by administrators.
  - Associating faculty members with relevant departments.
- **User Authentication and Authorization:**
  - Implementing role-based access control for administrators, faculty, and students.
  - Ensuring secure login and logout functionalities.
- **Email Notifications:**
  - Sending automated emails to users for actions such as approval, rejection, or restrictions.

**2. Technical Scope**

- **Backend Development:** Using **Flask** (Python) as the backend framework.
- **Database Management:** Implementing **MySQL** for handling relational data.
- **Frontend Development:** Utilizing **HTML, CSS, JavaScript** for designing user interfaces.

- **Deployment:** Deploying the application using tools like **Gunicorn**, **uWSGI**, and **Nginx**.
- **Security:** Incorporating standard web security practices to prevent common vulnerabilities.

**3. Limitations**

- The system is designed primarily for **university-level operations** and may not be suitable for other educational institutions without modification.
- The current scope excludes advanced features such as **attendance tracking, hostel management, and library management**.
- Integration with external learning platforms (e.g., Moodle, Blackboard) is not covered in this phase.

**4. Future Enhancements**

- Adding modules for **attendance tracking**, **library management**, and **hostel management**.
- Implementing **report generation** for analytics and performance tracking.
- Integrating **SMS notifications** along with email alerts for improved communication.

# METHODOLOGY

The development of the University Management System (UMS) adheres to a systematic approach utilizing secondary research methods. This methodology involves studying existing resources, technologies, frameworks, and best practices for creating web-based management systems. The steps are as follows:

1. **Requirement Analysis**
    - Collect and examine system requirements using existing documentation, research papers, and online materials.
    - Identify core functionalities such as student registration, faculty management, course administration, exam scheduling, fee processing, and result generation.
    - Define user roles, including Admin, Faculty, and Student.
2. **Technology Research**
    - Evaluate suitable technologies, including Flask (Python framework) for backend development and HTML, CSS, and JavaScript for the frontend.
    - Assess database systems like MySQL for efficient data management and retrieval.
3. **System Design Analysis**
    - Study existing University Management Systems to recognize common design patterns, architectures, and interfaces.
    - Create a relational database schema to support functionalities like enrollment, course assignment, exam scheduling, and fee management.
4. **UI/UX Design Analysis**
    - Examine modern web design principles to ensure the system is user-friendly and accessible.
    - Analyze interface designs of similar systems to develop intuitive dashboards tailored for Admin, Faculty, and Student roles.
5. **Testing Approaches**
    - Explore testing methodologies for Flask applications, including Unit Testing, Integration Testing, and End-to-End Testing.
6. **Deployment and Maintenance Research**
    - Investigate hosting platforms and deployment options suitable for Flask applications.
    - Review continuous integration and deployment practices to ensure effective system maintenance and updates.

# TECHNOLOGIES USED

1. Frontend:
    - **HTML5:** For structuring content and creating user interfaces.
    - **CSS3:** For styling the application and ensuring responsive design.
    - **JavaScript:** For enhancing interactivity and client-side validation.

2. **Backend:**
    - **Flask (Python):** A lightweight, micro web framework for building the application's backend.
    - **Jinja2:** Templating engine for rendering dynamic web pages.

3. **Database:**
    - **MySQL :** Relational database systems for data storage and management.

4. **Security:**
    - **Flask-Login:** For user authentication and session management.

5. **Email Handling:**
    - **Flask-Mail:** For sending email notifications to users (students and faculty).

6. **Version Control:**
    - **Git:** For code management and collaboration.
    - **GitHub:** For maintaining version history and collaborative development.

# EXPECTED OUTUT

The expected output of our University Management System (UMS) project will be a fully functional, user-friendly web application built using Flask, designed to efficiently manage students, faculty, courses, exams, results, fees, and departments. The application will feature distinct login systems for students, faculty, and administrators, ensuring role-based access control and data security. Administrators will have comprehensive control through a central dashboard, enabling them to add, edit, and delete records related to students, faculty, courses, departments, exams, results, and fees. Authentication for students and faculty members will be handled via their work mail, with password validation requiring a minimum length of eight characters to enhance security.

The system will facilitate seamless management of student enrollment, course assignments, exam scheduling, and result processing. Students will have access to their academic progress, exam schedules, and fee payment status, while faculty members will be able to view their assigned courses and departments, with the ability to update and publish exam results. Additionally, the application will support department management, associating faculty members with their respective departments and assigning Heads of Departments (HoDs).

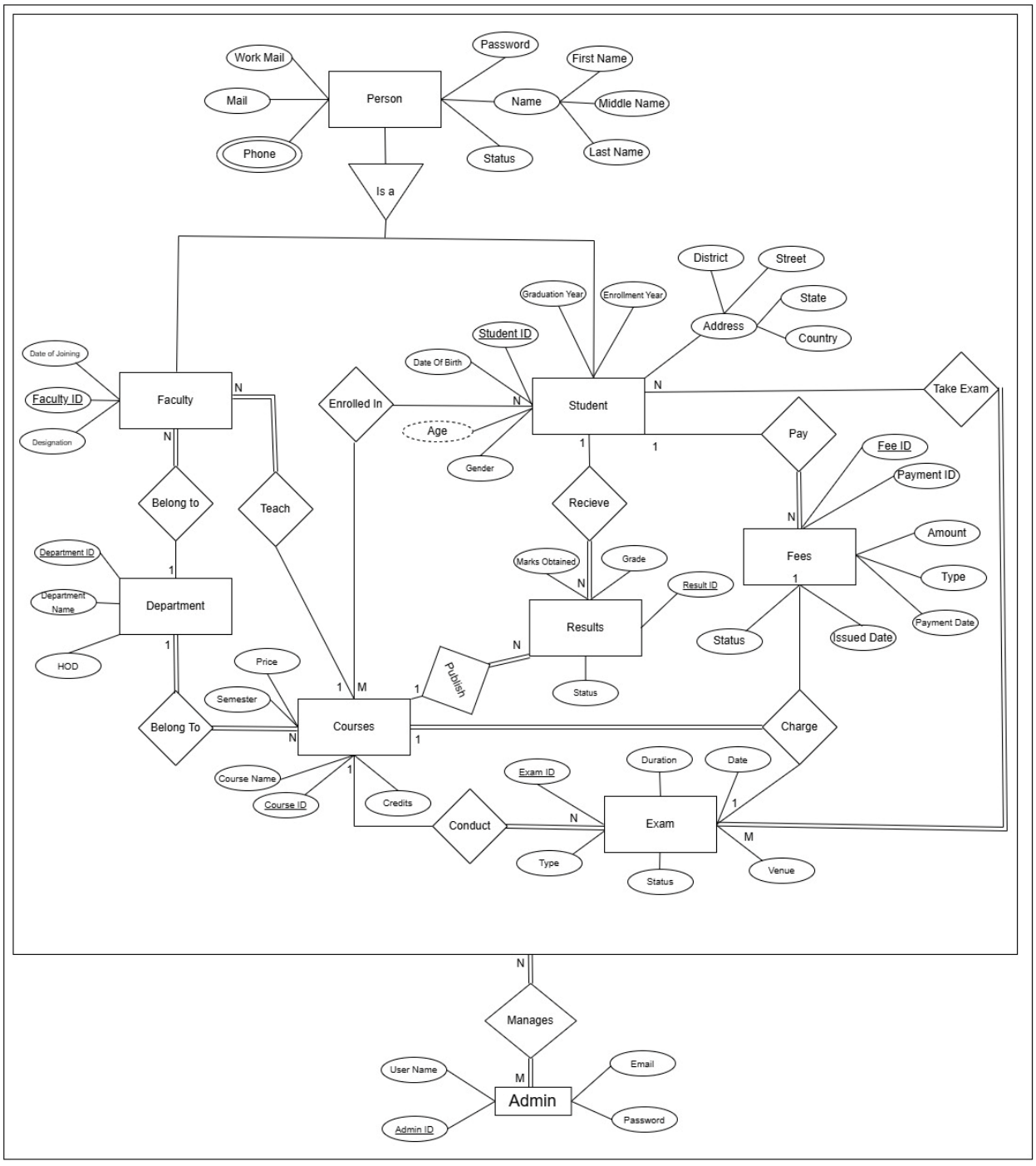
A robust fee management system will be integrated, allowing fees to be issued, payments processed, and receipts generated, with status indicators showing whether a fee is pending or paid. The result management system will offer features to record, update, and display exam results, with grading status classified as Unevaluated, Evaluated, or Locked.

The user interface is expected to be clean, responsive, and intuitive, ensuring a positive user experience. Security measures will be implemented to safeguard sensitive data, and thorough error handling will be incorporated to maintain smooth operation. Scalability, integrity, and consistency will be prioritized throughout the development process, ensuring the application remains reliable as it grows.

Furthermore, all database constraints will be enforced at the application level before reaching the database level to maintain data integrity. The application is expected to use a local MySQL server via XAMPP Server for database management.

Overall, the project aims to streamline university administration processes, delivering a seamless experience for students, faculty, and administrators alike.

# ER DIAGRAM


Work Mail
Password
First Name
Mail
Person
Name
Middle Name
Phone
Status
Last Name
Is a
District
Street
Graduation Year
Enrollment Year
State
Address
Country
Student ID
Date Of Birth
Date of Joining
Faculty ID
Faculty
Enrolled In
Student
Take Exam
Age
Pay
Designation
Gender
Fee ID
Payment ID
Belong to
Teach
Recieve
Department ID
Amount
Marks Obtained
Grade
Fees
Department Name
Department
Result ID
Type
Results
Payment Date
HOD
Status
Issued Date
Price
Publish
Semester
Belong To
Courses
Charge
Course Name
Duration
Date
Exam ID
Course ID
Credits
Conduct
Exam
Type
Status
Venue
Manages
User Name
Email
Admin
Admin ID
Password

# ER To Table

The University Management System requires a well-structured database with 12 (including two tables for multivalued attributes of students and faculty) tables to handle various entities and their relationships. The relationships between entities are outlined below:

**1. Students and Courses (N:M Relationship)**

- **Tables Required:** students, courses, enrollment
- Students can enroll in multiple courses, and a course can have multiple students.
- The enrollment table establishes the relationship, with a composite primary key comprising student_id and course_id.
- An additional attribute, enrollment_date, is included.

**2. Faculty and Department (1:N Relationship)**

- **Tables Required:** faculty, department
- A faculty member belongs to a single department, but a department can have multiple faculty members.
- The department table is independent, while the faculty table includes department_id as a foreign key.

**3. Faculty and Course (1:N Relationship)**

- **Tables Required:** faculty, courses
- A faculty member can teach only one course, but a course can be taught by multiple faculty members.
- The faculty table includes course_id as a foreign key.

**4. Courses and Exams (1:N Relationship)**

- **Tables Required:** courses, exams
- A course can conduct multiple exams, but each exam is associated with only one course.
- The exam table includes course_id as a foreign key and exam_id as the primary key.
- A unique constraint is applied to the combination of course_id and type.

**5. Courses and Results (1:N Relationship)**

- **Tables Required:** courses, results
- A course can have multiple results published, but each result belongs to only one course.
- The results table includes course_id as a foreign key.

**6. Students and Exams (N:M Relationship)**

- **Tables Required:** students, exams, takes_exams
- A student can take multiple exams, and an exam can be given by multiple students enrolled in the same course.
- The takes_exams table contains a composite primary key of student_id and exam_id.

**7. Students and Results (1:N Relationship)**

- **Tables Required:** students, results

- A student can receive multiple results, but each result is associated with only one student.
- The results table includes student_id as a foreign key.

**8. Students and Fees (1:N Relationship)**

- **Tables Required:** students, fees
- A student can pay multiple fees, but each fee is linked to only one student.
- The fees table includes student_id as a foreign key.

**9. Courses and Fees (1:1 Relationship)**

- **Tables Required:** courses, fees
- Each course charges only one fee.
- The fees table includes course_id as a foreign key.

**10. Exams and Fees (1:1 Relationship)**

- **Tables Required:** exams, fees
- Each exam charges only one fee.
- The fees table includes exam_id as a foreign key.

**11. Admin Management (1:1 Relationship)**

- **Tables Required:** admin
- The admin manages all entities. Since all entities are managed by the admin, no relationships need to be explicitly established.

# PARTICIPATION ANALYSIS

The participation of entities in various relationships within the University Management System is described below:

1. **Students and Courses:**
    - **Partial Participation (Both)** A student may not be enrolled in any course, and some courses may have no students enrolled. Therefore, both entities participate partially in the relationship.
2. **Faculty and Department:**
    - **Total Participation (Faculty) & Partial Participation (Department):** Every faculty member must belong to a department, but a department can exist without any faculty members. Faculty participation is total, while department participation is partial.
3. **Faculty and Courses:**
    - **Total Participation (Faculty) & Partial Participation (Course):** Every faculty member must teach at least one course. However, some courses may not be assigned to any faculty. Therefore, faculty participation is total, while course participation is partial.
4. **Departments and Courses:**
    - **Total Participation (Both):** Every department must offer at least one course, and every course must be offered by a department. Thus, both department and course participation are total.
5. **Courses and Exams:**
    - **Total Participation (Exam) & Partial Participation (Course):** Every exam must be associated with a course, but a course may not necessarily conduct exams. Therefore, exams participate totally, while courses participate partially.
6. **Students and Exams:**
    - **Total Participation (Exam) & Partial Participation (Student):** Every exam is taken by one or more students, but some students may not have participated in any exams. Hence, exams participate totally, while students participate partially.
7. **Courses, Exams, and Fees:**
    - **Total Participation (Course & Exam) & Partial Participation (Fee):** Every course and exam have associated fees, but some fees may not be related to any specific course or exam (e.g., registration fees). Therefore, fees participate partially.
8. **Students and Fees:**

- **Total Participation (Fee) & Partial Participation (Student):** Every fee must be associated with a student, but a student may not have paid any fees. Thus, fees participate totally, while students participate partially.

9. **Admin Management:**

    - **Total Participation (Both):** Every entity in the system is managed by an admin, and an admin cannot exist without management authority. Therefore, both participate totally.

10. **Students and Results:**

    - **Total Participation (Result) & Partial Participation (Student):** Every result is linked to a student, but not all students may have results published. Thus, results participate totally, while students participate partially.

11. **Courses and Results:**

    - **Total Participation (Result) & Partial Participation (Course):** Every result is associated with a course, but some courses may not have any results published. Therefore, results participate totally, while courses participate partially.

# TABLES

1. Student

   Students(Student_ID, First_Name, Middle_Name, Last_Name, street, district, state, country, Gender, Date_of_Birth, mail, College_Mail,Password, Enrollment_Year, Graduation_Year, Status)

2. Student Phone No

   PhoneNumbers(Student_ID, Phone)

3. Courses

   Courses(Course_ID, Course_Name, Semester, Credits, Price)

4. Enrollment

   Enrollment(Student_ID, Course_ID, Enrollment_On)

5. Fees

   Fees(Fee_ID, Student_ID,Exam_ID,Course_ID, Amount, Issued_Date, Payment_Date ,Type, Status,payment_ID)

6. Exams

   Exams(Exam_ID,Course_ID, Exam_Date, Exam_Duration, Exam_Type, Venue,Status)

7. Takes Exam

   Takes_Exam(Student_ID, Exam_ID,Status)

8.. Results

   Results(Result_ID, Exam_ID,Student_ID, Course_ID, Marks_Obtained,Grade,Status)

9. Department

   Department(Department_ID, Department_Name, Head_Of_Department)

10. Faculty

    Faculty(Faculty_ID, First_Name, Middle_Name, Last_Name, Date_Of_Joining, Designation, Mail,Official_Mail,Password,Status,Course_ID,Department_ID)

11. Faculty Phone No

    FacultyPhone(Faculty_ID, Phone)

12. Admin

    Admin(Admin_ID, User_Name, Password, Email,Password)