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 enrollment, 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.

PROBLEM STATEMENT

Managing academic and administrative tasks manually in universities is **inefficient, time-consuming, and error-prone**. Traditional record-keeping methods using paper or scattered digital files often lead to **data redundancy, loss of information, and difficulty in accessing student records, course details, and financial transactions**.

This project aims to develop a **University Management System** to address these challenges by providing a **centralized, automated, and secure platform** for managing various university operations.

Key problems addressed:

- **Student and Course Management:**
  Manual course registration and student enrollment can lead to scheduling conflicts, data loss, and inefficiencies. The system will allow students to **enroll in courses, view their results, and pay fees seamlessly**.
- **Faculty and Department Management:**
  Universities require efficient faculty assignment and department coordination. The system will provide a structured way for **faculty to manage courses, conduct exams, and belong to specific departments**.

- **Administrative Efficiency:**
  Without a **centralized** system, handling **student records, fee payments, exam scheduling, and result processing** can be overwhelming. The system will allow **admin users to manage all operations smoothly**.

By implementing a **database-driven University Management System**, this project ensures **efficiency, accuracy, and security**, reducing human errors and improving the overall **academic experience**.

OBJECTIVES

The following objectives were kept in mind while making this project:

- **To develop a centralized system** that efficiently manages student, faculty, course, and administrative data.
- **To automate university processes** such as course enrollment, fee payment, exam management, and result processing.
- **To enhance accessibility and security** by providing role-based access to students, faculty, and administrators.
- **To minimize manual errors and redundancy** by implementing a structured database-driven approach.
- **To improve decision-making** by providing quick access to student records, exam results, and financial transactions.
- **To create a scalable and reliable system** that can handle an increasing number of students, courses, and faculty members over time.
- **To provide an intuitive and user-friendly interface** for efficient interaction with the system.

Requirement Analysis

1. Functional Requirements

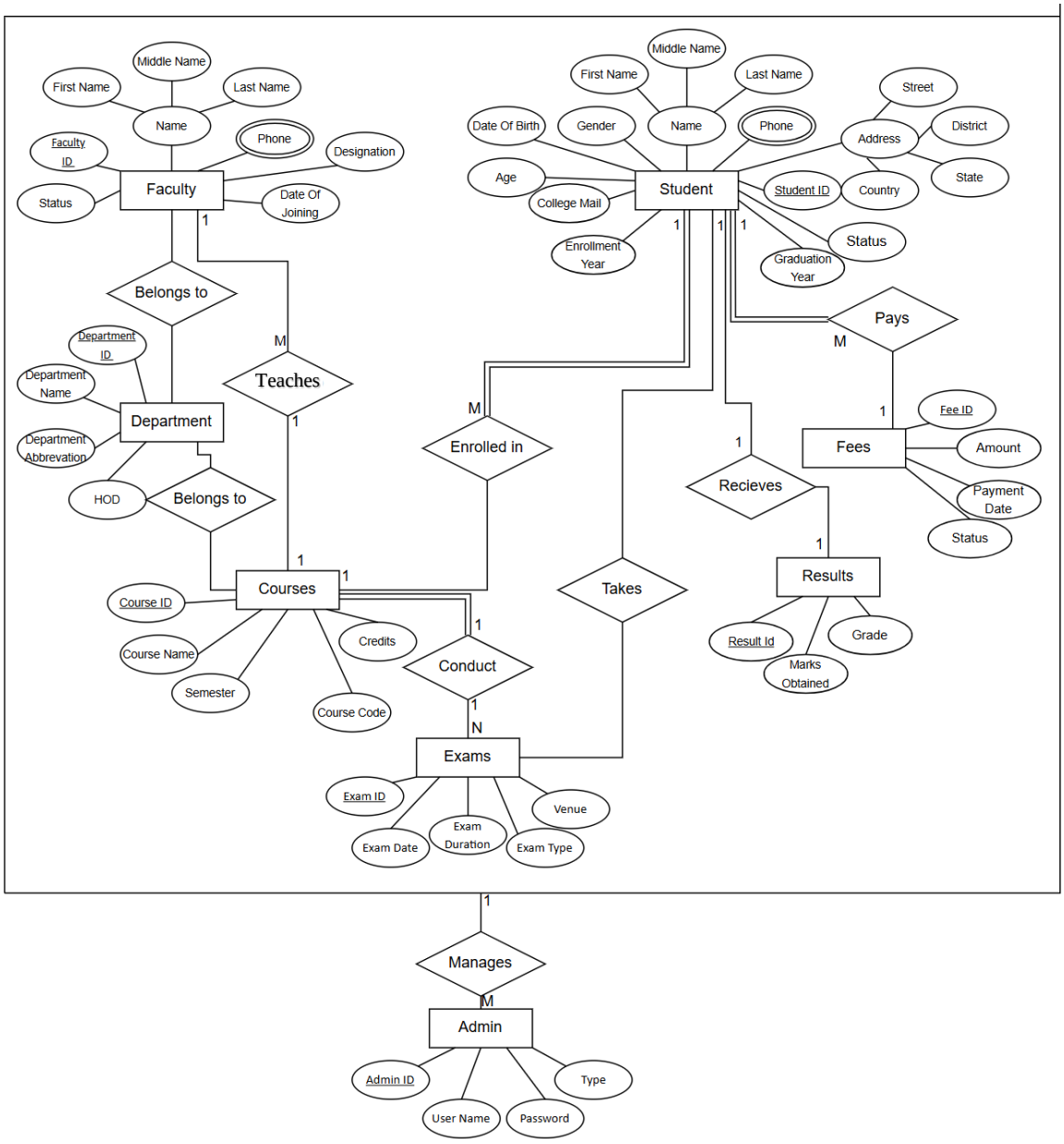
The system should provide the following functionalities:

- **Student Management** – Maintain student personal and academic records, enrollment details, and course selections.
- **Faculty Management** – Store faculty details, including personal information, designations, and assigned courses.
- **Course Management** – Manage courses, including course codes, credit hours, and department affiliations.
- **Enrollment System** – Allow students to enroll in courses and maintain their enrollment history.
- **Examination & Results** – Schedule exams, store exam details, and record student grades.
- **Fee Management** – Track student fee payments and statuses (paid, pending).
- **Admin Management** – Allow administrators to manage university operations efficiently.

2. Non-Functional Requirements

- **Scalability** – The system should handle a growing number of students, faculty, and courses.
- **Security** – Implement authentication and role-based access for students, faculty, and administrators.
- **Performance** – Ensure quick data retrieval and minimal response time for database queries.
- **Reliability** – Prevent data loss and ensure accurate record-keeping.
- **User-Friendly Interface** – Provide a simple and intuitive interface for users.

ER Diagram


Middle Name
First Name
Last Name
Name
Faculty ID
Phone
Designation
Status
Faculty
Date Of Joining
Belongs to
Department ID
Department Name
Department
Department Abbrevation
HOD
Belongs to
Teaches
Courses
Course ID
Course Name
Semester
Credits
Course Code
Conduct
Exams
Exam ID
Exam Date
Exam Duration
Exam Type
Venue
Middle Name
First Name
Last Name
Name
Date Of Birth
Gender
Phone
Age
College Mail
Student
Enrollment Year
Graduation Year
Student ID
Country
Status
Street
Address
District
State
Enrolled in
Takes
Pays
Fees
Fee ID
Amount
Payment Date
Status
Recieves
Results
Result Id
Marks Obtained
Grade
Manages
Admin
Admin ID
User Name
Password
Type

ER Diagram to Table

Relation 'Enrolled In'

Students: student_id (PK), first_name, middle_name, last_name, street, district, state, country, gender, dob, age, phone, college_mail, enrollment_year, graduation_year, status
Courses: course_id (PK), course_name, course_code, credits, semester
Enrollment: enrollment_id (PK), student_id (FK), course_id (FK), semester, grade, enrollment_date

---

Relation 'Pays'

Students: student_id (PK), first_name, middle_name, last_name, … (Other attributes)
Fees: fee_id (PK), student_id (FK), amount, payment_date, status

---

Relation 'Takes'

Students: student_id (PK), first_name, middle_name, last_name, … (Other attributes)
Exams: exam_id (PK), exam_date, exam_duration, exam_type, venue
Takes: student_id (FK), exam_id (FK)

Relation 'Receives'

Students: student_id (PK), first_name, middle_name, last_name, … (Other attributes)
Results: result_id (PK), student_id (FK), exam_id (FK), marks_obtained, grade

---

Relation 'Teaches'

Faculty: faculty_id (PK), first_name, middle_name, last_name, phone, date_of_joining, status, designation
Courses: course_id (PK), course_name, course_code, credits, semester
Teaches: faculty_id (FK), course_id (FK)

Relation 'Belongs to'

Faculty: faculty_id (PK), first_name, middle_name, last_name, … (Other attributes)
Department: department_id (PK), department_name, department_abbreviation, head_of_department
BelongsTo: faculty_id (FK), department_id (FK)

Relation 'Conducts'

Courses: course_id (PK), course_name, course_code, credits, semester
Exams: exam_id (PK), exam_date, exam_duration, exam_type, venue
Conducts: course_id (FK), exam_id (FK)

Relation 'Belong to'

Courses: course_id (PK), course_name, course_code, credits, semester
Department: department_id (PK), department_name, department_abbreviation,

head_of_department
BelongTo: course_id (FK), department_id (FK)

Relation ‘Manages’

Admin: admin_id (PK), username, password, type
Manages: admin_id (FK), student_id (FK), faculty_id (FK), course_id (FK), fee_id (FK)