# **EduTrack – Complete Educational Management Solution**

*Project Report with ER Model Reference*

## **1. Introduction**

Educational institutions face significant challenges in managing academic, administrative, and financial data for students, faculty, and the organization. Traditional record-keeping is inefficient, error-prone, and limits scalability. EduTrack provides a centralized, digital platform that empowers students, faculty, and institutional administrators to manage and access educational data efficiently, securely, and reliably. Its modular design blends modern database practices, robust analytics, and secure data management to transform educational administration into a seamless experience.

## **2. Problem Overview**

Without a well-structured management system, educational institutions typically struggle with:

* Delays in accessing and updating records, leading to inefficiencies.
* Data redundancy and inconsistency.
* Limited visibility into student progress and institutional operations.
* Security risks due to unprotected data flows or unregulated access.
* Challenges in reporting and decision-making due to fragmented data storage.

EduTrack eliminates these problems by providing a centralized relational database and role-based access, securely integrating all educational entities and operations.

## **3. Scope of the Project**

EduTrack aims to:

* Integrate student, faculty, and course details in a unified system.
* Record and manage academic progress, attendance, fees, exam schedules, communications, and more.
* Model relationships such as student-course enrollment, faculty assignments, and fee payments with optimal database structures.
* Support easy retrieval and updating of records via structured queries.
* Enable scalable growth and support for new features and modules.
* Ensure robust data security and controlled data access.

## **4. Objectives**

## **1. Centralized Data Management**

All records (students, faculty, courses, exams, payments) are maintained centrally, eliminating duplication and fragments.

## **2. Efficient Data Retrieval**

Optimized relational design supports fast and accurate queries—for grades, attendance, fees, and scheduling information.

## **3. Relationship Mapping**

Proper handling of entity relationships, such as:

* Students enrolling in multiple courses (many-to-many)
* Faculty teaching multiple courses (many-to-many)
* Student payments linked to enrollments (one-to-many)
* Notifications sent to many users (one-to-many)

## **4. Data Integrity and Consistency**

Database enforces primary and foreign key constraints, data validation, and referential integrity.

## **5. Security and Access Control**

Role-based access securely partitions student, faculty, and admin privileges.

## **6. Scalability and Maintainability**

Easily accommodates increased data volume (more students, courses) and allows regular updates.

## **7. Support for Reporting and Analytics**

Dashboards and reports offer insights for students, faculty, and administrators (e.g., student performance, fee collection, course popularity).

## **8. Minimizing Redundancy**

Normalization techniques reduce storage requirements and prevent data anomalies.

## **5. Significance of the Project**

Implementation of EduTrack brings:

* Faster operations: Automated updates and retrievals.
* Improved accuracy: Centralized, validated data records.
* Enhanced satisfaction: Responsive system for all users.
* Better security: Role-based protection of sensitive information.
* Informed decision-making: Visual analytics for institutional leadership.

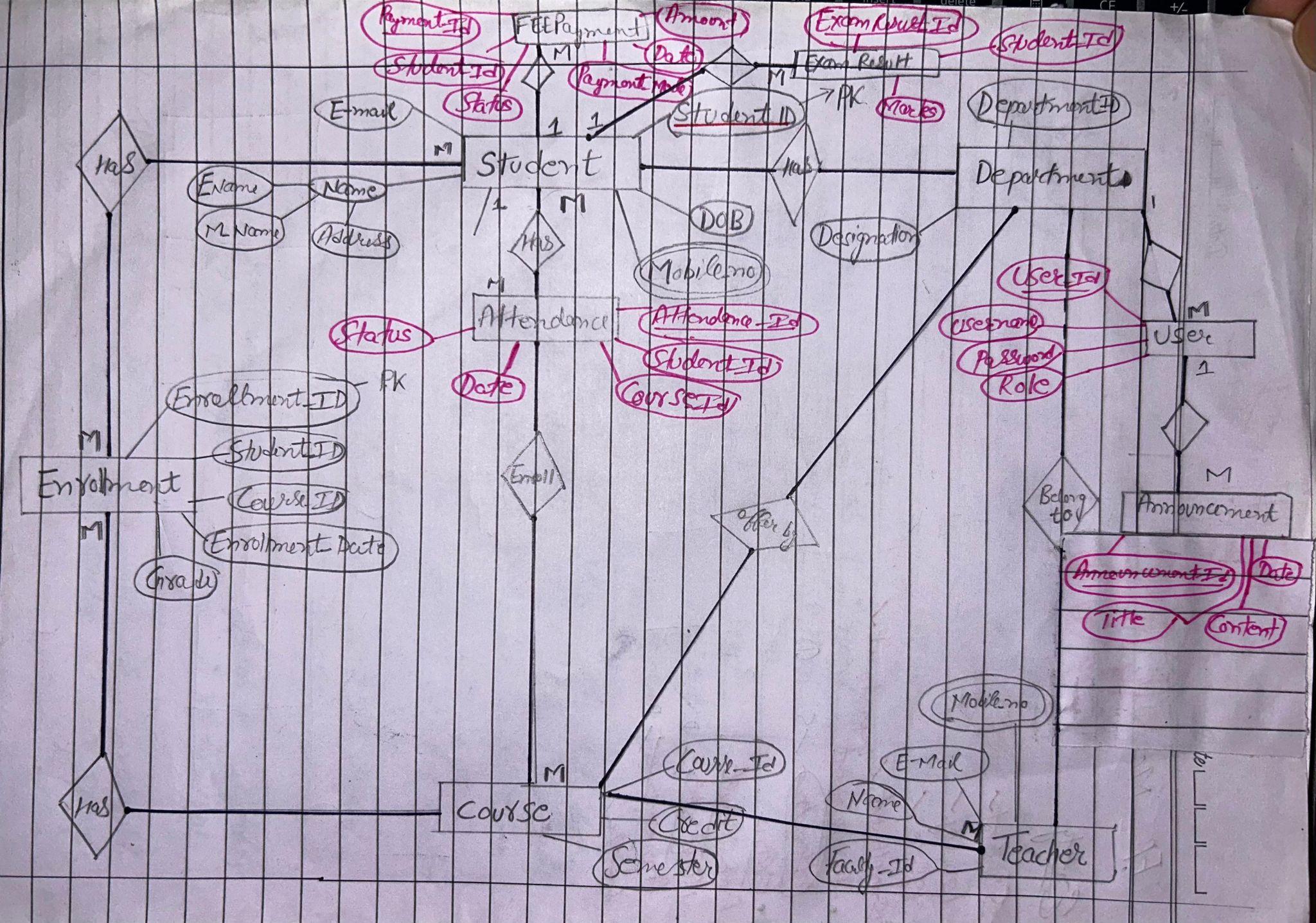
**6. Main Entities and Attributes**

|  |  |
| --- | --- |
| **Entity** | **Attributes** |
| Student | StudentID (PK), Name, Email, DOB, Phone |
| Faculty | FacultyID (PK), Name, Email, Phone |
| Course | CourseID (PK), Credit, Semester |
| Enrollment | EnrollmentID (PK), StudentID (FK), CourseID (FK), Enrollment\_Date, Grade |
| FeePayment | PaymentID (PK), StudentID (FK), Amount, Date, PaymentMode, Status |
| Attendance | AttendanceID (PK), StudentID (FK), CourseID (FK), Date, Status |
| Exam | ExamID (PK), Date, Time |
| ExamResult | ResultID (PK), StudentID (FK), Marks, Grade |
| User | UserID (PK), Username, Password, Role (Student/Faculty/Admin) |
| Announcement | AnnouncementID (PK),Content, Date |
|  |  |

**Relationships**:

* Student–Course: Many-to-Many
* Course–Faculty: Many-to-Many
* Student–FeePayment: One-to-Many
* Student–Attendance: One-to-Many
* Student–ExamResult: One-to-Many
* Student–Department: Many-to-One
* Faculty-Department: Many-to-One
* Announcement–User: One-to-Many

7. ER MODEL :



## 

**ER to Relational Model (Schema):**

Student(Student\_ID PK, F\_NAME, M\_NAME, Email, DOB, Mobile\_no,)

Faculty(Faculty\_ID PK, Name, Email, Mobile\_no)

Department(Department\_ID PK, Designation)

Course(Course\_ID PK, Credit, Semester)

Enrollment(Enrollment\_ID PK, Student\_ID FK, Course\_ID FK, Enrollment\_date, Grade)

Attendance(Attendance\_ID PK, Student\_ID FK, Course\_ID FK, Date, Status)

Fee\_Payment(Payment\_ID PK, Student\_ID FK, Amount, Date, Payment\_Mode, Status)

Exam\_Result(Result\_ID PK, Student\_ID FK, Marks)

User(User\_ID PK, Username, Password, Role)

Announcement(Announcement\_ID PK, Title, Content, Date, User\_ID FK)

**8. Relational Algebra (RA) Queries for EduTrack**

**Q1.** Get names of students who have paid their fees.

πName(σStatus=′Paid′(Student⨝Fee))π\_{Name}(σ\_{Status = 'Paid'}(Student ⨝ Fee))πName​(σStatus=′Paid′​(Student⨝Fee))

**Q2.** Find all attendance records of Student\_ID = 101.

σStudent\_ID=101(Attendance)σ\_{Student\\_ID = 101}(Attendance)σStudent\_ID=101​(Attendance

**Q3.** List names of students who scored grade “A” in any exam.

πName(σGrade=′A′(Result⨝Student))π\_{Name}(σ\_{Grade = 'A'}(Result ⨝ Student))πName​(σGrade=′A′​(Result⨝Student))

**Q4.** Get details of faculty teaching “Database Systems” course.

πName,Department(σCourse\_Name=′DatabaseSystems′(Course⨝Faculty))π\_{Name, Department}(σ\_{Course\\_Name = 'Database Systems'}(Course ⨝ Faculty))πName,Department​(σCourse\_Name=′DatabaseSystems′​(Course⨝Faculty))

**Q5.** Find courses offered by Institution\_ID = 1.

σInstitution\_ID=1(Course)σ\_{Institution\\_ID = 1}(Course)σInstitution\_ID=1​(Course)

**Q6.** Get names of students with unpaid fees.

πName(σStatus=′Unpaid′(Fee⨝Student))π\_{Name}(σ\_{Status = 'Unpaid'}(Fee ⨝ Student))πName​(σStatus=′Unpaid′​(Fee⨝Student))

**Q7.** Find all exams scheduled for Course\_ID = 301.

σCourse\_ID=301(Exam)σ\_{Course\\_ID = 301}(Exam)σCourse\_ID=301​(Exam)

**Q8.** List students enrolled in “Computer Networks” course.

πName(σCourse\_Name=′ComputerNetworks′(Student⨝Course⨝Result))π\_{Name}(σ\_{Course\\_Name = 'Computer Networks'}(Student ⨝ Course ⨝ Result))πName​(σCourse\_Name=′ComputerNetworks′​(Student⨝Course⨝Result))

**Q9.** Get all notifications sent to faculty.

σRole=′Faculty′(Notification)σ\_{Role = 'Faculty'}(Notification)σRole=′Faculty′​(Notification)

**Q10.** Retrieve students who attended class on “2025-01-10”.

πName(σDate=′2025−01−10′∧Status=′Present′(Attendance⨝Student))π\_{Name}(σ\_{Date = '2025-01-10' ∧ Status = 'Present'}(Attendance ⨝ Student))πName​(σDate=′2025−01−10′∧Status=′Present′​(Attendance⨝Student))

**Q11.** Find all faculty members working in the “CSE” department.

σDepartment=′CSE′(Faculty)σ\_{Department = 'CSE'}(Faculty)σDepartment=′CSE′​(Faculty)

**Q12.** Get list of students along with their GPA (assuming stored in Result).

πName,Grade(Student⨝Result)π\_{Name, Grade}(Student ⨝ Result)πName,Grade​(Student⨝Result)

**Q13.** Find all exams with total marks greater than 100.

σTotal\_Marks>100(Exam)σ\_{Total\\_Marks > 100}(Exam)σTotal\_Marks>100​(Exam)

**Q14.** Get students who paid fees after due date.

πName(σStatus=′Paid′∧Payment\_Date>Due\_Date(Fee⨝Student))π\_{Name}(σ\_{Status = 'Paid' ∧ Payment\\_Date > Due\\_Date}(Fee ⨝ Student))πName​(σStatus=′Paid′∧Payment\_Date>Due\_Date​(Fee⨝Student))

**Q15.** List all courses taught by Faculty\_ID = 202.

σFaculty\_ID=202(Course)σ\_{Faculty\\_ID = 202}(Course)σFaculty\_ID=202​(Course)

**Q16.** Find students who did not attend any classes.

πName(Student)−πName(Attendance⨝Student)π\_{Name}(Student) - π\_{Name}(Attendance ⨝ Student)πName​(Student)−πName​(Attendance⨝Student)

**Q17.** Get list of notifications after “2025-01-01”.

σDate>′2025−01−01′(Notification)σ\_{Date > '2025-01-01'}(Notification)σDate>′2025−01−01′​(Notification)

**Q18.** Retrieve results of Exam\_ID = 501.

σExam\_ID=501(Result)σ\_{Exam\\_ID = 501}(Result)σExam\_ID=501​(Result)

**Q19.** Find all faculty teaching more than one course.

πFaculty\_ID(Course)⨝GroupBy(Faculty\_ID,COUNT(Course\_ID)>1)π\_{Faculty\\_ID}(Course) ⨝ GroupBy(Faculty\\_ID, COUNT(Course\\_ID) > 1)πFaculty\_ID​(Course)⨝GroupBy(Faculty\_ID,COUNT(Course\_ID)>1)

**Q20.** Get names of students who scored below 40 marks.

πName(σMarks\_Obtained<40(Result⨝Student))π\_{Name}(σ\_{Marks\\_Obtained < 40}(Result ⨝ Student))πName​(σMarks\_Obtained<40​(Result⨝Student))

**Q21.** Find students who have at least one pending fee.

πName(σStatus=′Unpaid′(Fee⨝Student))π\_{Name}(σ\_{Status = 'Unpaid'}(Fee ⨝ Student))πName​(σStatus=′Unpaid′​(Fee⨝Student))

**Q22.** Get faculty emails who are associated with Institution\_ID = 1.

πEmail(σInstitution\_ID=1(Faculty))π\_{Email}(σ\_{Institution\\_ID = 1}(Faculty))πEmail​(σInstitution\_ID=1​(Faculty))

**Q23.** List courses having more than 3 credits.

σCredits>3(Course)σ\_{Credits > 3}(Course)σCredits>3​(Course)

**Q24.** Find students who received notifications.

πName(Notification⨝Student)π\_{Name}(Notification ⨝ Student)πName​(Notification⨝Student)

**Q25.** Get students who appeared in all exams of Course\_ID = 301.

πName(Student⨝Exam⨝Result) where Course\_ID = 301π\_{Name}(Student ⨝ Exam ⨝ Result) \ \text{where Course\\_ID = 301}πName​(Student⨝Exam⨝Result) where Course\_ID = 301