

**Step 1: Database Project Proposal/Choice Summary**

I have chosen the **Student Database Management System** for my project. The system is very important in keeping the records of students for the good management of universities, as well as it is easy to access data regarding general information, the academic performance of students, and financial information. Its ease of structure plus wide practical applicability makes it just perfect for database design.

**Step 2: Research on Similar Applications & Requirements****Primary Actors**

- Admin: manages student records, course assignments, and faculty data.
- Students: View your individual academic records, attendance, and fees.
- Professors/Instructors View and update grades and attendance for students.
- Finance Department: keeps records of student fee payments and scholarships.
- Library Staff - manages student-issued books and due dates.

**Geographic & User Demographics**

- Intended for educational institutions-schools, colleges, and universities.
- Used by students, faculty members, and administrative staff.
- Available on campus, and remotely through web-based applications.

**20+ System Requirements**

- Admin should be able to add, update, and delete student records.
- Students should be able to view their profile and update contact details.
- Professors should be able to view assigned courses and enrolled students.
- Students should be able to enroll in courses through an online portal.
- Admin should manage the courses and subjects offered per semester.
- Faculty should be able to mark student attendance.
- Students should have access to attendance records.
- Professors should update and manage student grades.
- Admin should handle student fee transactions and pending payments.
- System should generate fee receipts and payment history.
- Scholarship records should be maintained and linked to student profiles.
- Students should have access to course schedules.
- The library system should maintain book-issue records per student.
- Faculty should be able to send notices and announcements.
- Students should be able to request transcripts.
- System should handle data encryption and user authentication.
- It generates reports on student performance analytics.

- Integrate it with email/SMS notifications about deadlines and alerts.
- Database to handle multiple users concurrently efficiently.
- The system will be web-based.

### Step 3: Entities, Relationships, and Constraints

#### Entities & Attributes

1. **Student** (StudentID, Name, DOB, Contact, Address, AdmissionYear, Course, FeesPaid, Scholarship)
2. **Professor** (ProfessorID, Name, Department, Contact, Email)
3. **Course** (CourseID, CourseName, Credits, Department, Semester)
4. **Enrollment** (StudentID, CourseID, Semester, Year, Grade) [*Many-to-Many: Student-Course*]
5. **Attendance** (StudentID, CourseID, Date, Status)
6. **FeeTransaction** (TransactionID, StudentID, Amount, Date, PaymentMethod)
7. **Library** (BookID, Title, Author, ISBN, StudentID, IssueDate, DueDate)

#### Relationships

- A **Student** enrolls in **Courses** (*Many-to-Many*).
- A **Professor** teaches multiple **Courses** (*One-to-Many*).
- **Students** have **Attendance** records for each course (*Many-to-Many*).
- A **Student** makes **Fee Transactions** (*One-to-Many*).
- A **Student** borrows **Library Books** (*One-to-Many*).

#### Constraints

- StudentID and ProfessorID should be unique.
- Each course must be assigned to at least one professor.
- Students cannot enroll in more than X courses per semester.
- Library books must be returned within a due date, else a fine is imposed.

### Step 4: Database Schema

```
CREATE TABLE Student (  
    StudentID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    DOB DATE,  
    ContactInfo VARCHAR(255),  
    Address TEXT,  
    AdmissionYear INT,  
    CourseEnrolled VARCHAR(255),  
    FeeStatus VARCHAR(50) );
```

```
CREATE TABLE Course (  
    CourseID INT PRIMARY KEY,  
    CourseName VARCHAR(255),  
    Credits INT,  
    Department VARCHAR(100),  
    InstructorID INT  
);
```

```
CREATE TABLE Instructor (  
    InstructorID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    Department VARCHAR(100),  
    Email VARCHAR(255),  
    ContactNo VARCHAR(50)  
);
```

```
CREATE TABLE Enrollment (  
    EnrollmentID INT PRIMARY KEY,  
    StudentID INT,  
    CourseID INT,  
    Semester VARCHAR(50),  
    Year INT,  
    Status VARCHAR(50),  
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);
```

```
CREATE TABLE Attendance (  
    AttendanceID INT PRIMARY KEY,  
    StudentID INT,  
    CourseID INT,  
    Date DATE,  
    Status VARCHAR(50),  
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);
```

```
CREATE TABLE Grade (  
    GradeID INT PRIMARY KEY,  
    StudentID INT,  
    CourseID INT,  
    Semester VARCHAR(50),  
    Year INT,  
    Grade VARCHAR(5),  
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID)  
);
```

```
CREATE TABLE Fee (  
    FeeID INT PRIMARY KEY,  
    StudentID INT,  
    Amount DECIMAL(10,2),  
    DueDate DATE,  
    PaymentStatus VARCHAR(50),  
    ScholarshipID INT,  
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
    FOREIGN KEY (ScholarshipID) REFERENCES Scholarship(ScholarshipID)  
);
```

```
CREATE TABLE Scholarship (  
    ScholarshipID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    EligibilityCriteria TEXT,  
    Amount DECIMAL(10,2)  
);
```

### Step 5: UI Prototype

The frontend UI for this project will be designed using **Figma or Proto.io**, featuring:

1. **Student Dashboard** - View courses, grades, attendance, and fee status
2. **Admin Panel** - Manage students, courses, instructors, and payments
3. **Instructor Dashboard** - Manage student grades and attendance
4. **Registration & Login** - Secure access for different user roles
5. **Fee Payment Interface** - View pending fees and process payments.

FRONTEND UI:

STUDENT  
MANAGMENT

DASHBOARD

ADMIN PANEL

LOGIN / SIGN UP

FEE PAYMENT


HOME

COURSES

GRADES

ATTENDANCE

FEES



COURSES

GRADES

ATTENDANCE

FEE STATUS

Course Name	Grade	Attendance %	Instructor Name	Credits
Introduction to AI	A	95%	Dr. Smith	3
Data Structures	B+	90%	Prof. Johnson	4
Computer Networks	A-	88%	Dr. Lee	3
Machine Learning	A	92%	Dr. Taylor	4
Database Systems	B	85%	Prof. Wilson	3