**Project : Student Database Management System**

1.Requirements:

Based on the SQL tables provided (`students`, `courses`, `grades`, and `enrollments`), here are some possible features and functionalities that could be incorporated into a student management system:

1. Student Registration: Allow administrators to register new students by entering their personal information (name, date of birth, address, contact number) into the `students` table.

2. Course Management: Provide functionality to manage courses, including creating new courses and updating course information (name, description, credits) in the `courses` table.

3. Enrollment Management: Enable administrators to enroll students in courses by adding records to the `enrollments` table, specifying the `student\_id`, `course\_id`, and `enrollment\_date`.

4. Grade Management: Allow teachers to enter grades for students in different courses by adding records to the `grades` table, specifying the `student\_id`, `course\_id`, and `grade`.

5. Attendance Tracking: Extend the system to include an attendance table, where attendance records can be stored with the `student\_id`, `course\_id`, and `attendance\_date`.

6. Transcripts and Academic Progress: Develop functionality to calculate GPA based on the grades in the `grades` table and generate student transcripts or academic progress reports.

7. Student Search and Reporting: Implement search functionality to retrieve student information from the `students` table based on specific criteria, such as student ID, name, or date of birth. Generate reports, such as student lists or course rosters, using data from the various tables.

8. User Roles and Access Control: Assign different roles to users, such as administrators, teachers, and students, and define their access permissions to restrict access to sensitive data or limit the functionalities available to each role.

9. Communication and Messaging: Incorporate messaging functionality to enable communication between administrators, teachers, and students, such as sending notifications, announcements, or reminders.

10. Data Import/Export: Provide the ability to import and export data to and from the system, allowing for integration with external systems or easy transfer of data.

2.Design Database Schema

1. \*\*students\*\* table:

- student\_id (Primary Key)

- name

- date\_of\_birth

- address

- contact\_number

2. \*\*courses\*\* table:

- course\_id (Primary Key

- name

- description

- credits

3. \*\*enrollments\*\* table:

- student\_id (Foreign Key referencing students.student\_id)

- course\_id (Foreign Key referencing courses.course\_id)

- enrollment\_date

4. \*\*grades\*\* table:

- student\_id (Foreign Key referencing students.student\_id)

- course\_id (Foreign Key referencing courses.course\_id)

- grade

3.Create the database

To create the database, you will need to use a Database Management System (DBMS) such as MySQL, PostgreSQL, or SQLite. Here's an example of how you can create the database using MySQL:

* Install MySQL: If you haven't already, install MySQL on your system following the instructions provided by the official MySQL documentation.
* Connect to MySQL: Open a command-line interface or a MySQL client tool and connect to your MySQL server using the appropriate credentials.
* Create the Database: Use the following SQL statement to create the database:

CREATE DATABASE student\_management;

4.Implentinting the tables

CREATE TABLE students (

student\_id INT PRIMARY KEY,

name VARCHAR(50),

date\_of\_birth DATE,

address VARCHAR(100),

contact\_number VARCHAR(15));

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

name VARCHAR(50),

description VARCHAR(200),

credits INT

);

CREATE TABLE enrollments (

student\_id INT,

course\_id INT,

enrollment\_date DATE,

PRIMARY KEY (student\_id, course\_id),

FOREIGN KEY (student\_id) REFERENCES students (student\_id),

FOREIGN KEY (course\_id) REFERENCES courses (course\_id)

);

CREATE TABLE grades (

student\_id INT,

course\_id INT,

grade FLOAT,

PRIMARY KEY (student\_id, course\_id),

FOREIGN KEY (student\_id) REFERENCES students (student\_id),

FOREIGN KEY (course\_id) REFERENCES courses (course\_id)

);

5.Insert Sample Data

INSERT INTO students (student\_id, name, date\_of\_birth, address, contact\_number)

VALUES (1, 'John Doe', '2000-05-10', '123 Main St, City', '1234567890');

INSERT INTO students (student\_id, name, date\_of\_birth, address, contact\_number)

VALUES (2, 'Jane Smith', '1998-08-15', '456 Elm St, Town', '9876543210');

INSERT INTO courses (course\_id, name, description, credits)

VALUES (101, 'Mathematics', 'Introduction to Calculus', 3);

INSERT INTO courses (course\_id, name, description, credits)

VALUES (102, 'Physics', 'Mechanics and Motion', 4);

INSERT INTO enrollments (student\_id, course\_id, enrollment\_date)

VALUES (1, 101, '2022-09-01');

INSERT INTO enrollments (student\_id, course\_id, enrollment\_date)

VALUES (2, 102, '2022-09-01');

INSERT INTO enrollments (student\_id, course\_id, enrollment\_date)

VALUES (1, 101, '2022-09-01');

INSERT INTO enrollments (student\_id, course\_id, enrollment\_date)

VALUES (2, 102, '2022-09-01');

INSERT INTO grades (student\_id, course\_id, grade)

VALUES (1, 101, 85.5);

INSERT INTO grades (student\_id, course\_id, grade)

VALUES (2, 102, 92.0);

6. Implement CRUD Operations

INSERT INTO students (student\_id, name, date\_of\_birth, address, contact\_number)

VALUES (3, 'Alice Johnson', '1999-03-20', '789 Oak St, Village', '4567891230');

SELECT \* FROM students;

SELECT \* FROM students WHERE student\_id = 2;

UPDATE students

SET name = 'Bob Anderson', address = '321 Maple St, Town'

WHERE student\_id = 1;

DELETE FROM students WHERE student\_id = 3;