MAKERERE UNIVERSITY BUSINESS SCHOOL

COURSE WORK TWO EXAM FOR BACHELOR OF BUSINESS COMPUTING

OF MAKERERE UNIVERSITY, ACADEMIC YEAR 2024/2025

COURSE NAME: **ADVANCED WEB APPLICATION DEVELOPMENT**

COURSE CODE: BUC3131 **SEMESTER:** ONE **YEAR OF STUDY**: III

NAME	**REG NO**	**STUDENT NUMBER**
ODUKOI EMMANUEL JOSHUA	22/U/6771	2200706771
RUBANGAKENE BRIAN DAVID	22/U/6849	2200706849
AKOL SHARON	22/U/5723	2200705723
NASAKA SUZAN	21/U/12809/Ps	21007112809

Question:

Design and implement a **Student Management System** in **Laravel**. The application should allow administrators to manage student information, view student records, manage courses, and generate reports. The project will be broken into phases, with specific deliverables at each stage

b) Gather and document the application requirements:

1. Identify the key features of the Student Management System, such as student records management, course management, and report generation.

Key Features

1. **User Authentication (Admin Login)**:
   - Admins should be able to log in securely using email and password.
   - Passwords should be hashed for security.
   - Admin roles should be defined to restrict access to certain functionalities.
2. **Student Management (CRUD Operations)**:
   - **Create**: Admins can add new student records with details such as name, email, phone number, date of birth, and enrolled courses.
   - **Read**: Admins can view a list of all students, including search and filter options.
   - **Update**: Admins can edit existing student records.
   - **Delete**: Admins can remove student records om the system.
3. **Course Management (CRUD Operations)**:
   - **Create**: Admins can create new courses with attributes like course name, description, and credit hours.
   - **Read**: Admins can view a list of all courses with details.
   - **Update**: Admins can edit course information.
   - **Delete**: Admins can remove courses om the system.
4. **Enrollment Management Linking Students to Courses**:
   - Admins can enroll students in courses.
   - Admins can view which students are enrolled in each course.
   - Admins can manage enrollments (add or remove students om courses).
5. **Report Generation for Student and Course Statistics**:
   - Generate reports on the number of students enrolled in each course.
   - Generate reports on student details (e.g., total number of students, average age).
   - Export reports in formats such as PDF or CSV.

2. Create a simple requirements document detailing the functional and non-functional requirements.

Functional Requirements

1. **User Authentication**:

- The system shall allow admins to register an account.
- The system shall allow admins to log in using email and password.
- The system shall allow admins to reset their password if forgotten.

2. **Student Management**:
   - The system shall allow admins to add new students with required fields (name, email, phone number, date of birth).
   - The system shall allow admins to view a list of all students.
   - The system shall allow admins to edit student information.
   - The system shall allow admins to delete student records.
3. **Course Management**:
   - The system shall allow admins to create new courses with required fields (course name, description, credit hours).
   - The system shall allow admins to view a list of all courses.
   - The system shall allow admins to edit course information.
   - The system shall allow admins to delete courses.
4. **Enrollment Management**:
   - The system shall allow admins to enroll students in courses.
   - The system shall allow admins to view students enrolled in each course.
   - The system shall allow admins to remove students from courses.
5. **Report Generation**:
   - The system shall generate reports on the number of students enrolled per course.
   - The system shall generate reports on total student statistics (e.g., total number of students).

Non-Functional Requirements

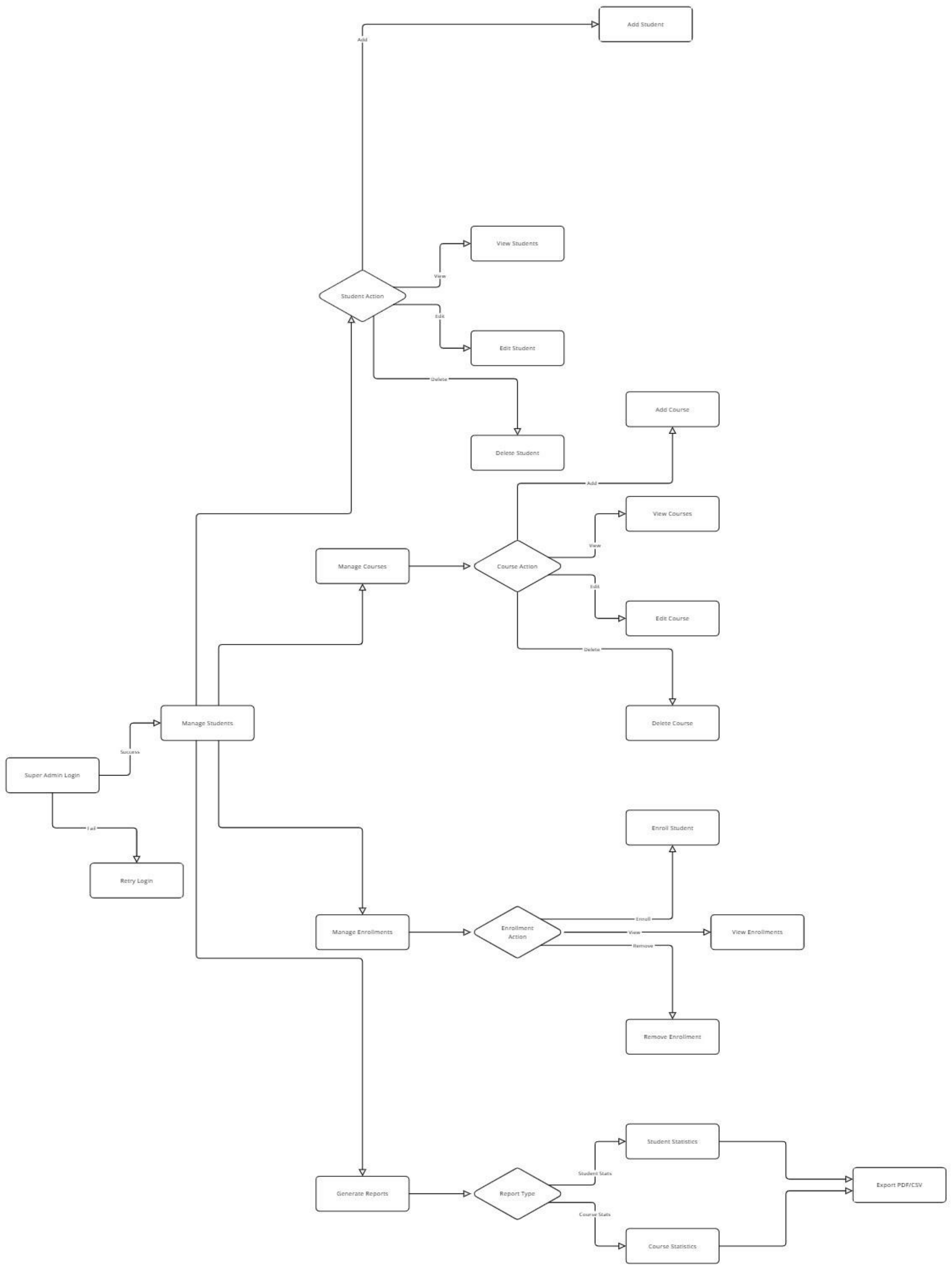
1. **Performance**:

- The system should respond to user requests within 2 seconds under normal load conditions.

2. **Scalability**:
    - The application should be able to handle up to 1000 concurrent users without performance degradation.
3. **Security**:
    - User passwords must be stored securely using hashing algorithms (e.g., bcrypt).
    - Sensitive data must be encrypted during transmission (e.g., using HTTPS).
4. **Usability**:
    - The user interface should be intuitive and easy to navigate for administrators.
5. **Responsiveness**:
    - The application should be responsive and accessible on various devices (desktop, tablet, mobile).
6. **Maintainability**:
    - Code should follow best practices for maintainability and readability.

System Architecture Diagram

The following is a textual representation of the Entity-Relationship Diagram (ERD) for the Student Management System:



Explanation of the ERD Components

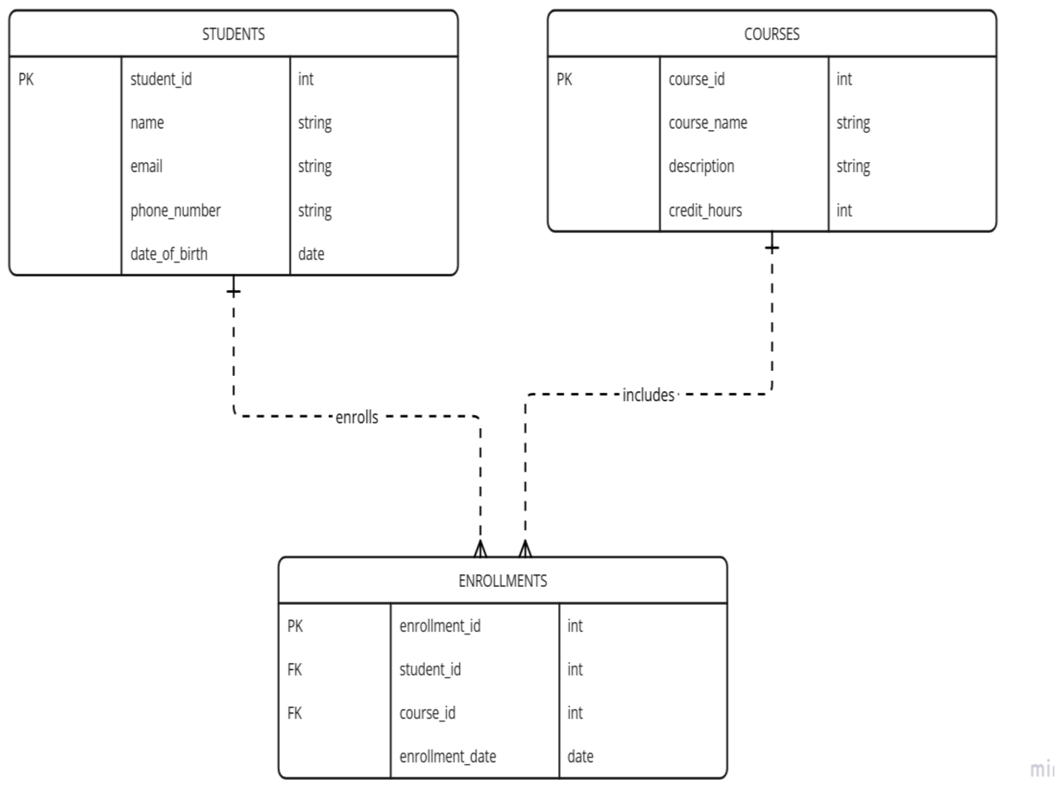
- **Students Table**:
    - **student_id (PK)**: Primary key, uniquely identifies each student.
    - **name**: The full name of the student.
    - **email**: The email address of the student.
    - **phone_number**: The contact number of the student.
    - **date_of_birth**: The birth date of the student.
- **Courses Table**:
    - **course_id (PK)**: Primary key, uniquely identifies each course.
    - **course_name**: The name of the course.
    - **description**: A brief description of what the course entails.
    - **credit_hours**: The number of credit hours assigned to the course.
- **Enrollments Table**:
    - **enrollment_id (PK)**: Primary key, uniquely identifies each enrollment record.
    - **student_id (FK)**: Foreign key that references student_id in the Students table, linking students to their enrollments.
    - **course_id (FK)**: Foreign key that references course_id in the Courses table, linking courses to their enrolled students.
    - **enrollment_date**: The date when the student was enrolled in the course.
- **Admins Table**:
    - **admin_id (PK)**: Primary key, uniquely identifies each admin user.
    - **name**: The full name of the admin.
    - **email**: The email address used for login and communication.
    - **password**: Hashed password for secure authentication.

Relationships in the Student Management System

1. Many-to-Many Relationship: Students and Courses

- Description:
    - Each student can enroll in multiple courses.
    - Each course can have multiple students enrolled.

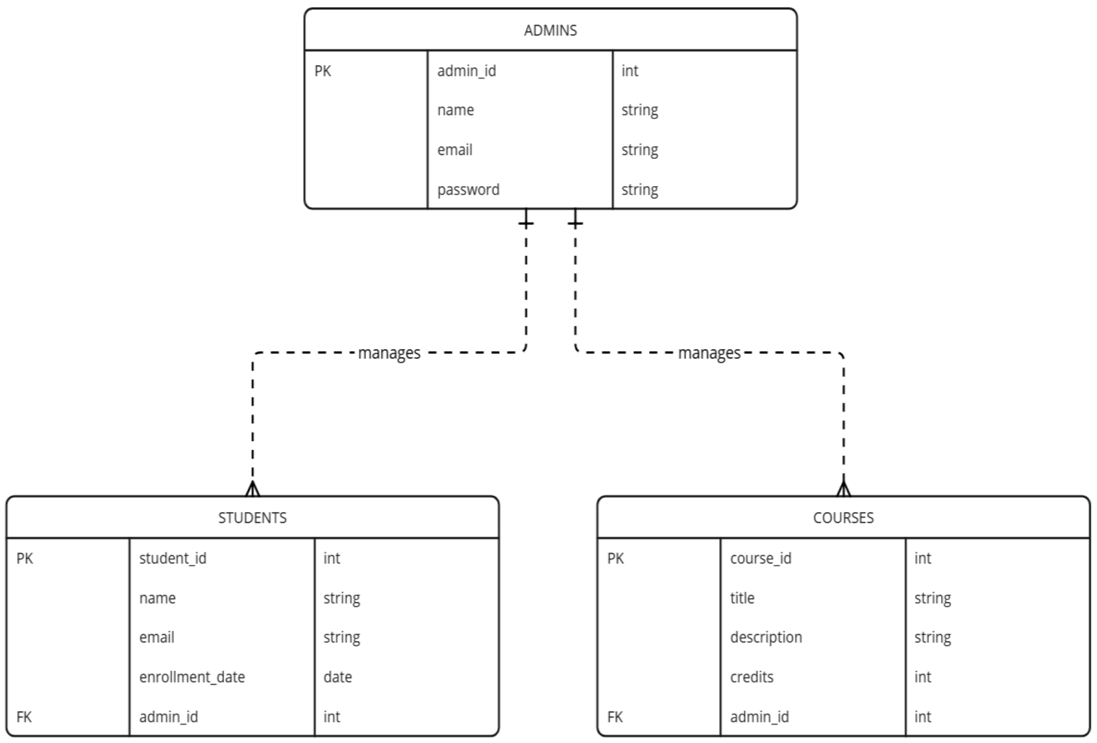
- Implementation:
    - This relationship is managed through a junction table called Enrollments. The Enrollments table contains foreign keys that reference both the Students and Courses tables.
- Entities Involved:
    - Students Table: Holds student information.
    - Courses Table: Holds course information.
    - Enrollments Table: Links students to courses.
- **ERD Representation**:



- **Foreign Keys in Enrollments Table**:
    - student_id: References student_id in the Students table.
    - course_id: References course_id in the Courses table.

2. One-to-Many Relationship: Admins and Both Students and Courses

- **Description**:
    - Each admin can manage multiple students and multiple courses, but each student or course is managed by one admin at a time.
- **Implementation**:
    - The Admins table holds information about administrators who have the authority to manage students and courses.
- **Entities Involved**:
    - **Admins Table**: Contains admin user information.
    - **Students Table**: Managed by admins.
    - **Courses Table**: Managed by admins.
- **ERD Representation**:



Summary of Relationships

1. **Students and Courses**:
    - Many-to-Many relationship facilitated by the Enrollments table.
    - Allows for flexible enrollment where students can take multiple courses and courses can have many students.
2. **Admins and Students/Courses**:
    - One-to-Many relationship where each admin can manage multiple students and courses.
    - Ensures that there is a structured way to control access and management of records within the system.

Benefits of This Structure

- **Data Integrity**: The use of foreign keys ensures that relationships between tables are maintained, preventing orphan records.
- **Scalability**: The many-to-many relationship allows for easy addition of new courses or students without altering existing records significantly.
- **Manageability**: Admins have a clear structure for managing both students and courses, making it easier to implement features like reporting, enrollment tracking, and user management.

2. Database Design and Implementation

a) Design the Database

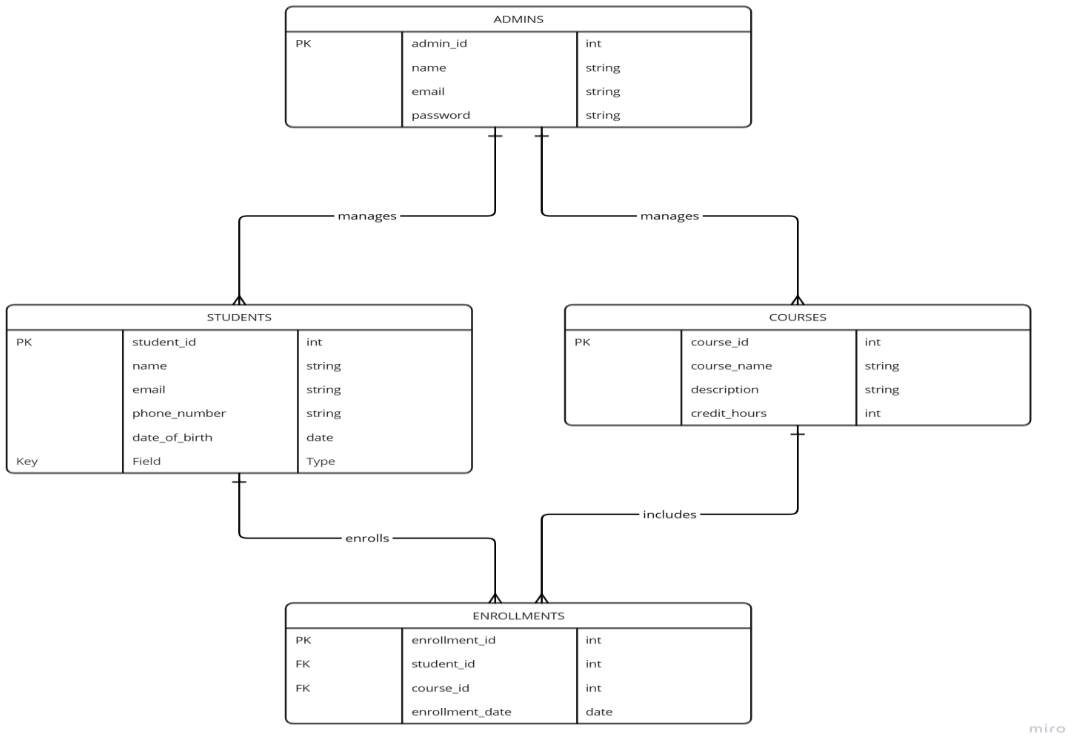
1. Identify Entities

The primary entities in the Student Management System are:

1. **Students**
   - Represents individual students enrolled in courses.
2. **Courses**
   - Represents the courses offered by the institution.
3. **Admins**
   - Represents administrators who manage students and courses.
4. **Enrollments**
   - A junction table that links students to courses, representing the many-to-many relationship.

2. Entity-Relationship Diagram (ERD)

Detailed representation of the ERD for the SMS



3. Attributes of Each Entity

1. Students Table

- **student_id (PK)**: Unique identifier for each student.
- **name**: Full name of the student.
- **email**: Email address of the student (should be unique).
- **phone_number**: Contact number of the student.
- **date_of_birth**: Birth date of the student.

2. Courses Table

- **course_id (PK)**: Unique identifier for each course.
- **course_name**: Name of the course.
- **description**: Brief description of what the course covers.
- **credit_hours**: Number of credit hours assigned to the course.

3. Enrollments Table

- **enrollment_id (PK)**: Unique identifier for each enrollment record.
- **student_id (FK)**: Foreign key referencing student_id in Students table.
- **course_id (FK)**: Foreign key referencing course_id in Courses table.
- **enrollment_date**: Date when the student was enrolled in the course.

4. Admins Table

- **admin_id (PK)**: Unique identifier for each admin user.
- **name**: Full name of the admin.
- **email**: Email address used for login (should be unique).
- **password**: Hashed password for secure authentication.

4. Relationships Between Entities

1. **Students and Courses**:
    - Relationship Type: Many-to-Many
    - Description: A single student can enroll in multiple courses, and a single course can have multiple students enrolled in it.
    - Implementation: This relationship is represented by the Enrollments table, which contains foreign keys referencing both Students and Courses.

2. **Admins and Students/Courses**:
   - Relationship Type: One-to-Many
   - Description: Each admin can manage multiple students and multiple courses, but each student or course is managed by one admin at a time.
   - Implementation: While not explicitly represented in this ERD, you could add an admin_id foreign key to both Students and Courses tables if you want to track which admin manages which records.

5. SQL Schema

```sql
CREATE TABLE admins (
  admin_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  email VARCHAR (100) NOT NULL UNIQUE,
  password VARCHAR (255) NOT NULL);

CREATE TABLE students (
  student_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR (100) NOT NULL,
  email VARCHAR (100) NOT NULL UNIQUE,
  phone_number VARCHAR (15),
  date_of_birth DATE NOT NULL);

CREATE TABLE courses (
  course_id INT AUTO_INCREMENT PRIMARY KEY,
  course_name VARCHAR (100) NOT NULL,
  description TEXT,
  credit_hours INT NOT NULL);

CREATE TABLE enrollments (
  enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
  student_id INT NOT NULL,
  course_id INT NOT NULL,
  enrollment_date DATE NOT NULL,
  FOREIGN KEY (student_id) REFERENCES students(student_id),
  FOREIGN KEY (course_id) REFERENCES courses(course_id));
```