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

GROUP COURSEWORK TWO

GROUP MEMBERS

NO.	NAME	REGISTRATION NUMBER	STUDENT NUMBER
1.	JJUUKO RONALD	22/U/5997	2200705997
2.	MUSINGUZI SIMON PETER	22/U/20505/EVE	2200720505

Project Overview:

The Elite Student Management System is a web-based application built with the Laravel framework. It is designed to help administrators efficiently manage student records, courses, enrollments, and generate reports. This project supports CRUD operations (Create, Read, Update, Delete) for students, courses, and enrollments. It is built with user-friendly features to streamline administrative tasks as well as automated assignment of some details on creation for example, course_id, student Email, Registration Number among others.

2. Requirements Gathering

a) Key Features of the Student Management System

1. **Student Management:**
    - Admin can add, edit, and delete student records.
    - Admin can view a list of students with search, filter, and pagination options.
    - Detailed student profiles, including name, level, email, and age, are accessible.
2. **Course Management:**
    - Admin can create, edit, and delete courses.
    - A comprehensive course list is available, including course details like name, description, and credit hours.
3. **Enrollment Management:**

    - Admin can enroll students in courses.
    - Admin can view enrollment details to see which students are enrolled in each course.

4. **Report Generation:**
    - Admin can generate detailed reports, such as:
        - Student profile summaries
        - Total students by level
        - Registrations Per month
5. **Authentication and Security:**
    - Admin login is required to manage the system.
    - Unauthorized users are redirected to a login page, ensuring restricted access to student and course data.

b) Functional and Non-Functional Requirements

- **Functional Requirements:**
    - CRUD operations for managing student and course records.
    - Report generation for students and course enrollments.
    - Secure authentication for the admin interface.
- **Non-Functional Requirements:**
    - **Security:** Implementation of role-based access controls to limit access to administrative functions.
    - **Scalability:** Ensuring the system can support a growing number of students, courses, and enrollments.
    - **Responsiveness:** Designing a user interface that works seamlessly on both mobile and desktop devices.

c) System Architecture Diagram

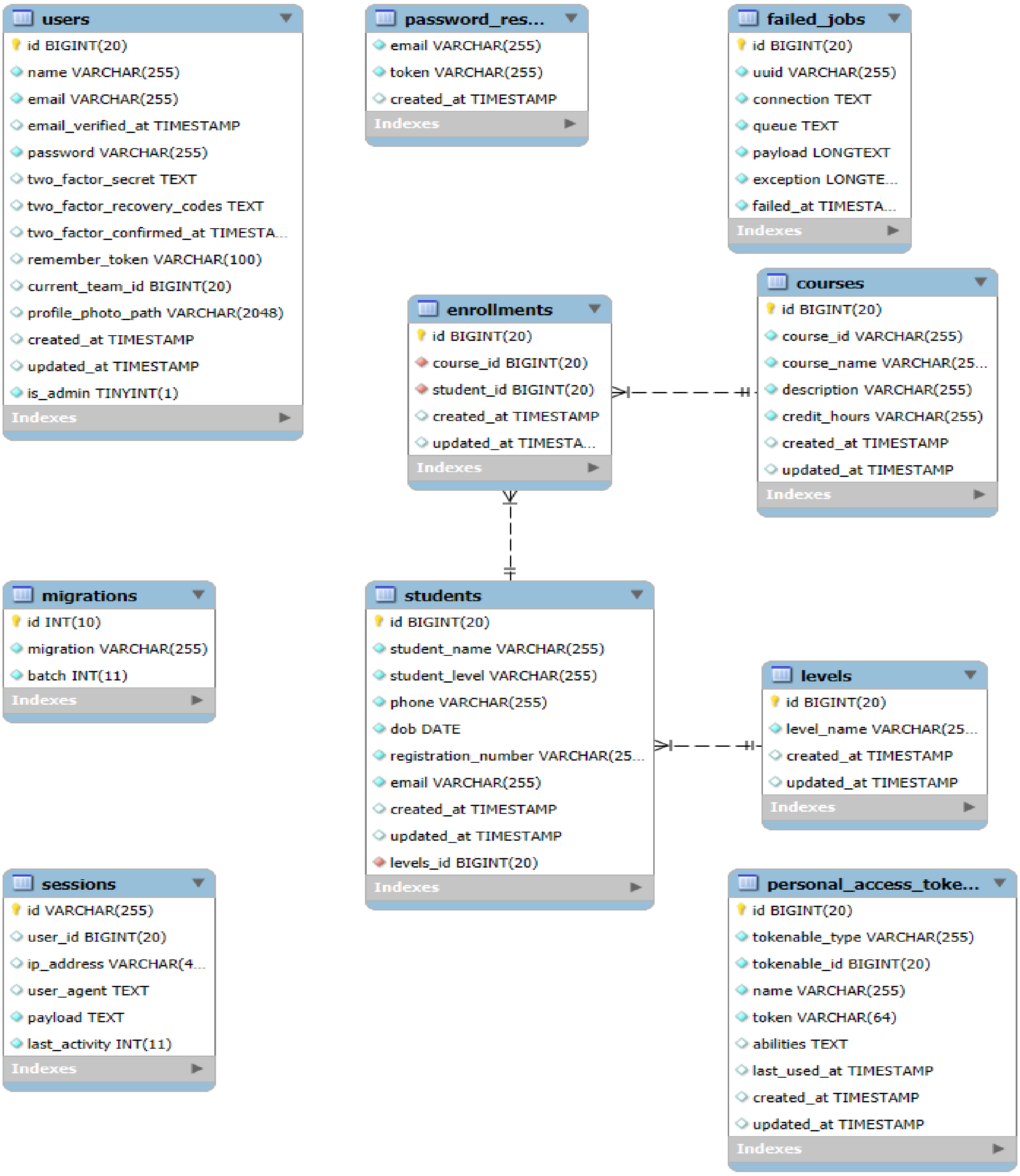
The Student Management System architecture consists of these main components:

- **Students:** Represents individual student records.
- **Courses:** Contains data about courses offered.
- **Enrollments:** Manages the many-to-many relationship between students and courses.
- **Admin Dashboard:** The admin interface for performing CRUD operations, viewing enrollments, and generating reports.

2. Database Design and Implementation

a) Database Schema Design

An Entity-Relationship (ER) diagram illustratrating the connections between key tables: Students, Courses, Enrollments, and Admins.(Generated with MySQL Workbench)



users
id BIGINT(20)
name VARCHAR(255)
email VARCHAR(255)
email_verified_at TIMESTAMP
password VARCHAR(255)
two_factor_secret TEXT
two_factor_recovery_codes TEXT
two_factor_confirmed_at TIMESTA...
remember_token VARCHAR(100)
current_team_id BIGINT(20)
profile_photo_path VARCHAR(2048)
created_at TIMESTAMP
updated_at TIMESTAMP
is_admin TINYINT(1)
Indexes

password_res...
email VARCHAR(255)
token VARCHAR(255)
created_at TIMESTAMP
Indexes

failed_jobs
id BIGINT(20)
uuid VARCHAR(255)
connection TEXT
queue TEXT
payload LONGTEXT
exception LONGTE...
failed_at TIMESTA...
Indexes

enrollments
id BIGINT(20)
course_id BIGINT(20)
student_id BIGINT(20)
created_at TIMESTAMP
updated_at TIMESTA...
Indexes

courses
id BIGINT(20)
course_id VARCHAR(255)
course_name VARCHAR(25...
description VARCHAR(255)
credit_hours VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes

migrations
id INT(10)
migration VARCHAR(255)
batch INT(11)
Indexes

students
id BIGINT(20)
student_name VARCHAR(255)
student_level VARCHAR(255)
phone VARCHAR(255)
dob DATE
registration_number VARCHAR(25...
email VARCHAR(255)
created_at TIMESTAMP
updated_at TIMESTAMP
levels_id BIGINT(20)
Indexes

levels
id BIGINT(20)
level_name VARCHAR(25...
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes

sessions
id VARCHAR(255)
user_id BIGINT(20)
ip_address VARCHAR(4...
user_agent TEXT
payload TEXT
last_activity INT(11)
Indexes

personal_access_toke...
id BIGINT(20)
tokenable_type VARCHAR(255)
tokenable_id BIGINT(20)
name VARCHAR(255)
token VARCHAR(64)
abilities TEXT
last_used_at TIMESTAMP
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes

Explanation of Tables and Relationships:

1. **Students Table:**
   - **id (PK):** Unique identifier for each student.

- **student_name:** Name of the student.
- **Student_level:** Educational level or grade of the student.
- **registration_number.** Automatically assigned by the system
- **email:** Automatically assigned by the system
- **date_of_birth:** Student's date of birth.

2. **Courses Table:**
    - **id (PK):** Unique identifier for each course.
    - **course_id:** Automatically assigned by the system
    - **name:** Course title.
    - **description:** description of the course content.
    - **credits:** Number of credits awarded for the course.
3. **Enrollments Table:**
    - **id (PK):** Unique identifier for each enrollment record.
    - **student_id (FK):** Foreign key referencing the Students table.
    - **course_id (FK):** Foreign key referencing the Courses table.
4. **Admins Table:**
    - **id (PK):** Unique identifier for each admin.
    - **username:** Admin username for login.
    - **password:** Encrypted admin password.
    - **email:** Admin's contact email.
    - **created_at, updated_at:** Timestamps for record creation and updates.

Relationships:

- **Students and Courses (Many-to-Many Relationship):**
    - A student can enroll in multiple courses, and each course can have many students enrolled. The Enrollments table connects students with courses, supporting this many-to-many relationship.
- **Level and Students (One-to-Many Relationship):**
    - Each level may have multiple students, represented by the student_level field in the Students table.
    - **Admins and Data Management:**

- Admins interact with student, course, and enrollment data through the Admin Dashboard, which provides a controlled interface for performing CRUD operations without direct database interactions.