PROJECT REPORT

UNIVERSITY ERP SYSTEM

Abstract

The **University Enterprise Resource Planning (ERP) System** is a comprehensive software solution designed to streamline and automate the administrative and academic processes of a university. Built using **Java (Swing)** and **MySQL**, the system provides a centralized platform for Students, Instructors, and Administrators to interact efficiently.

In modern educational institutions, managing course registrations, grading, and user authentication manually is prone to errors and inefficiency. This project addresses these challenges by offering a role-based access control system. Administrators can manage the infrastructure (users, courses, sections); Instructors can manage their classes and submit grades; and Students can view catalogs, register for courses, and download transcripts.

Key features include a secure login system using **BCrypt** password hashing, a maintenance mode for administrative safety, and cross-database queries to separate authentication data from operational data. The system ensures data integrity, security, and a user-friendly interface for all stakeholders.

---

1. Introduction

The University ERP System is a desktop-based application aimed at digitalizing the core functions of university management. It serves as a bridge between the university administration, faculty, and the student body. The project utilizes Object-Oriented Programming (OOP) principles to create a modular and scalable architecture.

1.1 Problem Statement

Traditional methods of university management often involve disjointed systems or manual paperwork for tasks such as course registration, grade submission, and user management. This leads to:

- Redundant data entry.
- Lack of real-time information for students regarding grades and enrollments.
- Security risks associated with storing plain-text passwords.
- Difficulty in managing concurrent academic sessions.

1.2 Objectives

The primary objectives of this project are:

- To develop a secure authentication system that differentiates between Admins, Instructors, and Students.
- To provide a graphical interface (GUI) for easy navigation and operation.
- To enable dynamic course and section management.
- To facilitate real-time course registration and grade computation.
- To implement secure data storage using relational databases.

1.3 Scope

The scope of the ERP System covers:

- **Authentication:** Secure login with lockout mechanisms and role detection.
- **Administration:** User creation, course catalog management, and system maintenance.
- **Academic Management:** Instructor assignment, grade calculation (A-F scale), and statistical analysis of class performance.
- **Student Services:** Self-service registration, dropping courses, and CSV transcript generation.

---

2. System Analysis

2.1 Feasibility Study

- **Technical Feasibility:** The project uses Java (JDK 11+) and MySQL, which are open-source and widely supported. The use of JDBC ensures robust database connectivity.
- **Operational Feasibility:** The UI is built with Java Swing, mimicking modern dashboard aesthetics (Sidebar navigation, Cards), ensuring ease of use for non-technical staff.

2.2 Software Requirements

- **Operating System:** Windows, Linux, or macOS.
- **Language:** Java (JDK 11 or higher).
- **Database:** MySQL Server 8.0+.
- **IDE:** IntelliJ IDEA
- **Libraries:**
    - `mysql-connector-j-9.5.0.jar` (JDBC Driver)
    - `jbcrypt-0.4.jar` (Password Security)

3. System Architecture

The project follows a **Layered Architecture**:

1. **Presentation Layer (UI):**
    - Built using `javax.swing`.
    - Contains classes like `LoginWindow`, `StudentDashboard`, `AdminDashboard`, etc.
    - Handles user input and visualization.
2. **Service Layer (Business Logic):**
    - Contains `AdminService`, `StudentService`, `InstructorService`, `AuthService`.
    - Implements rules like "Only students can register" or "Maintenance Mode blocks changes".
3. **Data Access Layer (DAO):**
    - Contains `UserDAO`, `CourseDAO`, `SectionDAO`, `EnrollmentDAO`.
    - Executes direct SQL queries via JDBC.
4. **Database Layer:**
    - **University Auth DB:** Stores credentials (`users_auth`).
    - **University ERP DB:** Stores academic data (`students`, `courses`, `grades`).

---

4. Modules Description

The system is divided into three primary functional modules based on user roles.

4.1 Administrative Module

The Admin module allows full control over the system configuration.

- **Manage Users:** Create new Student or Instructor profiles. The system automatically links credentials in the Auth DB with profiles in the ERP DB.
- **Course & Section Management:** Create new courses (e.g., "CS101") and schedule sections (Time, Room, Capacity).
- **Assign Instructors:** Map specific instructors to course sections.
- **Maintenance Mode:** A toggle switch that locks the database for write operations (Registration/Dropping) to prevent data inconsistency during updates.
- **System Logs:** View a simulated audit log of system events.

4.2 Instructor Module

Designed for faculty to manage their academic responsibilities.

- **My Sections:** View a list of courses currently taught by the logged-in instructor.
- **Roster Management:** View the list of students enrolled in specific sections.
- **Grading:** Enter raw scores (Quiz, Midterm, Final) to compute a final letter grade (A, B, C, D, F) or manually submit grades.

- **Statistics:** View grade distribution summaries (e.g., "How many students got an A?").

4.3 Student Module

Allows students to manage their academic journey.

- **Course Catalog:** View available sections, including schedule and capacity.
- **Registration:** Enroll in open sections (checks for duplicate enrollment and system maintenance status).
- **My Courses:** View current enrollments and grades.
- **Drop Course:** Withdraw from a section.
- **Transcript:** Export a CSV file containing the academic history and grades.

---

5. Database Schema

The system uses two separate databases to enforce separation of concerns: `university_auth_db` for security and `university_erp_db` for operations.w

5.1 Database: `university_auth_db`

Table: `users_auth` Stores login credentials. | Column | Type | Description | | :--- | :--- | :--- | | `user_id` | INT (PK) | Auto-increment unique ID | | `username` | VARCHAR | Unique login name | | `password_hash` | VARCHAR | BCrypt encrypted password | | `role` | VARCHAR | 'Admin', 'Student', or 'Instructor' | | `failed_attempts`| INT | Counter for brute-force protection | | `lock_time` | TIMESTAMP | Time until account unlock |

5.2 Database: `university_erp_db`

Table: `students` | Column | Type | Description | | :--- | :--- | :--- | | `user_id` | INT (PK, FK) | Links to `users_auth.user_id` | | `roll_no` | VARCHAR | Unique Roll Number (e.g., S101) | | `program` | VARCHAR | Degree program | | `year` | INT | Admission year |

Table: `instructors` | Column | Type | Description | | :--- | :--- | :--- | | `user_id` | INT (PK, FK) | Links to `users_auth.user_id` | | `department` | VARCHAR | Faculty Department |

Table: `courses` | Column | Type | Description | | :--- | :--- | :--- | | `course_id` | INT (PK) | Auto-increment ID | | `code` | VARCHAR | Course Code (e.g., CS101) | | `title` | VARCHAR | Course Name | | `credits` | DOUBLE | Credit weight |

Table: `sections` | Column | Type | Description | | :--- | :--- | :--- | | `section_id` | INT (PK) | Auto-increment ID | | `course_id` | INT (FK) | Links to `courses` | | `instructor_id`| INT

(FK) | Links to `instructors` | | `day`, `time`, `room` | VARCHAR | Scheduling info | | `capacity` | INT | Max students allowed |

Table: `enrollments` | Column | Type | Description | | :--- | :--- | :--- | | `enrollment_id` | INT (PK) | Auto-increment ID | | `student_id` | INT (FK) | Links to `students` | | `section_id` | INT (FK) | Links to `sections` | | `grade` | VARCHAR | Final Grade (A, B, C...) | | `status` | VARCHAR | 'Registered', 'Dropped' |

Table: `settings` | Column | Type | Description | | :--- | :--- | :--- | | `key_name` | VARCHAR (PK) | Config key (e.g., 'maintenance_on') | | `value` | VARCHAR | Config value ('true'/'false') |

---

6. Implementation Details

6.1 Security Mechanisms

- **BCrypt Hashing:** The project uses `jbcrypt` to salt and hash passwords. Plain text passwords are never stored in the database.
- **Account Lockout:** The `AuthService` tracks failed login attempts. If a user fails 5 consecutive attempts, the account is temporarily locked for 30 seconds.
- **Session Management:** A singleton `UserSession` class manages the currently logged-in user context, ensuring that data (like "My Grades") is only fetched for the authenticated user.

6.2 Key Classes

- **`Main.java`**: The entry point that launches the `LoginWindow`.
- **`DBConnection.java`**: A utility class managing connections to both databases. It allows cross-database joins (e.g., fetching a username from `auth_db` while querying `enrollments` in `erp_db`).
- **`ThemeManager.java`**: Handles the UI look and feel, allowing for a consistent design and Dark/Light mode toggling.
- **`AccessService.java`**: A middleware-style service that checks permissions (e.g., is the system in maintenance mode?) before allowing transactions.

6.3 Code Structure

Plaintext

```
src/
├── edu/univ/erp/
│   ├── auth/
│   ├── access/
│   ├── data/
```

```
│   ├── domain/
│   ├── service/
│   └── ui/
└── Main.java
```

---

7. User Interface & Features

The User Interface is designed using Java Swing components tailored with custom painting for a modern look.

7.1 Login Screen

A secure entry point featuring a background image of the university. It includes username/password fields and a "Dark Mode" toggle button. It handles validation errors and account lockouts visually.

7.2 Dashboards

- **Admin Dashboard:** Features a "Maintenance Mode" toggle switch in the header. The sidebar provides navigation to management dialogs. The main area displays status banners.
- **Student Dashboard:** Uses a `CardLayout` to switch between the "Course Catalog" and "My Grades" views. Tables are styled with custom headers and row heights for readability.
- **Instructor Dashboard:** Provides a clear view of assigned sections. Specialized dialogs (`ComputeGradesDialog`) allow instructors to input component scores (Quiz, Midterm, Final) and automatically calculate the letter grade.

7.3 Data Export

The Student module includes a **CSV Export** feature. This uses Java file I/O to generate a `transcript.csv` file containing the student's academic history, suitable for opening in Excel.

---

8. How to Run the Project

Follow these exact steps to set up and run the University ERP System.

Step 1: Database Setup

1. Open **MySQL Workbench** or your MySQL Command Line.

2. Execute the following SQL commands to create the required databases and tables.
   Note: The code expects these exact database names.

SQL

```
-- 1. Create Auth Database
CREATE DATABASE university_auth_db;
USE university_auth_db;

CREATE TABLE users_auth (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role ENUM('Admin', 'Student', 'Instructor') NOT NULL,
    failed_attempts INT DEFAULT 0,
    lock_time TIMESTAMP NULL
);

-- 2. Create ERP Database
CREATE DATABASE university_erp_db;
USE university_erp_db;

CREATE TABLE students (
    user_id INT PRIMARY KEY,
    roll_no VARCHAR(20),
    program VARCHAR(100),
    year INT
);

CREATE TABLE instructors (
    user_id INT PRIMARY KEY,
    department VARCHAR(100)
);

CREATE TABLE courses (
    course_id INT AUTO_INCREMENT PRIMARY KEY,
    code VARCHAR(20),
    title VARCHAR(100),
    credits DOUBLE
);

CREATE TABLE sections (
    section_id INT AUTO_INCREMENT PRIMARY KEY,
    course_id INT,
    instructor_id INT,
    day VARCHAR(20),
    time VARCHAR(50),
    room VARCHAR(20),
    capacity INT,
```

```
    semester VARCHAR(20),
    year INT
);

CREATE TABLE enrollments (
    enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
    student_id INT,
    section_id INT,
    grade VARCHAR(5),
    status VARCHAR(20)
);

CREATE TABLE settings (
    key_name VARCHAR(50) PRIMARY KEY,
    value VARCHAR(50)
);

-- 3. Insert Initial Settings
INSERT INTO settings VALUES ('maintenance_on', 'false');
```

Step 2: Configure Application

1. Open `src/edu/univ/erp/data/DBConnection.java`.

Update the `USER` and `PASSWORD` constants to match your local MySQL credentials:
Java
```
private static final String USER = "root";
private static final String PASSWORD = "your_mysql_password";
```

2.

Step 3: Seed Initial Users

1. Run `src/edu/univ/erp/auth/HashGenerator.java`. It will print hashed passwords to the console.

Copy the output and use it to insert an Admin user into the database manually:
SQL
```
USE university_auth_db;
INSERT INTO users_auth (username, password_hash, role)
VALUES ('admin1', 'HASH_FROM_CONSOLE', 'Admin');
```

2.

Step 4: Run the Application

1. Ensure `lib/mysql-connector-j-9.5.0.jar` and `lib/jbcrypt-0.4.jar` are added to your project's **Classpath/Build Path**.
2. Run `src/Main.java`.
3. Login with `admin1` (password: `MyAdmin1Pass` or whatever you hashed).
4. Use the Admin Dashboard to create Student and Instructor users via the GUI.

---

9. Conclusion

The University ERP System successfully meets its design objectives by providing a secure, robust, and user-friendly platform for university management. By integrating role-based security with a layered architecture, the system ensures that sensitive administrative functions are protected while providing students and faculty with the transparency they need.

The use of distinct databases for authentication and operations enhances security best practices. The application's ability to handle course registrations, grade computations, and maintenance lockdowns demonstrates its readiness for real-world academic scenarios.

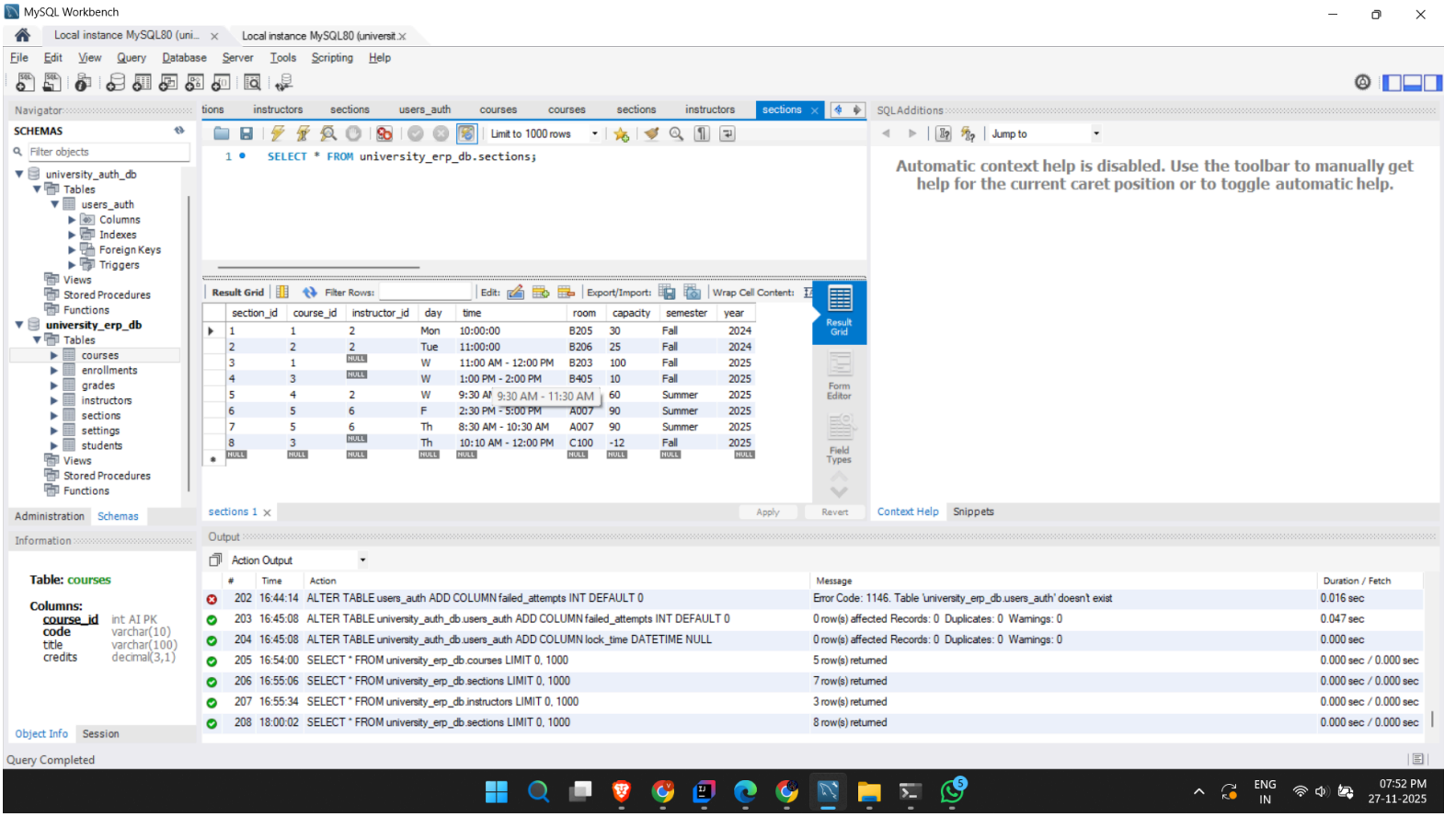
10. Future Scope

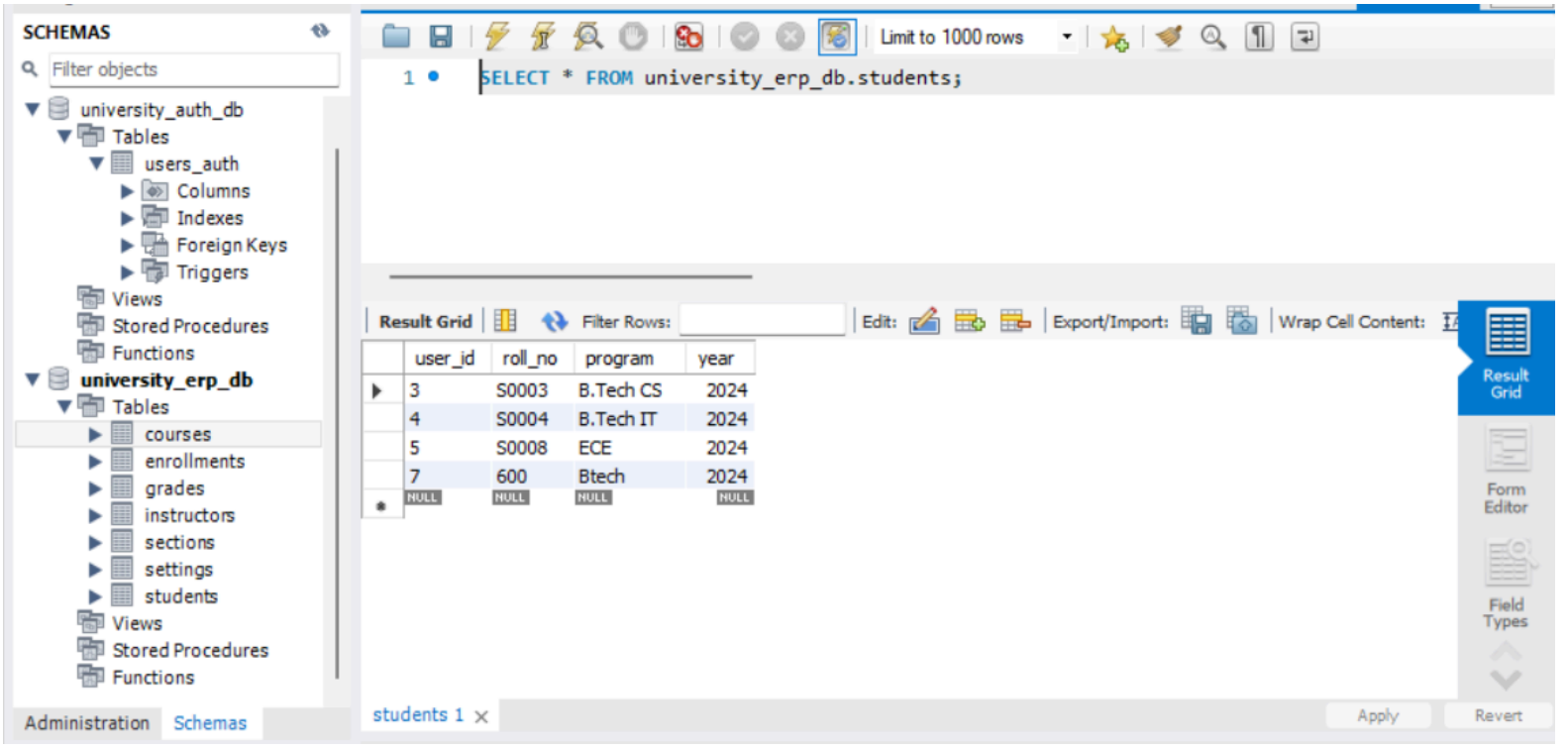
While the current system covers the core academic lifecycle, potential future enhancements include:

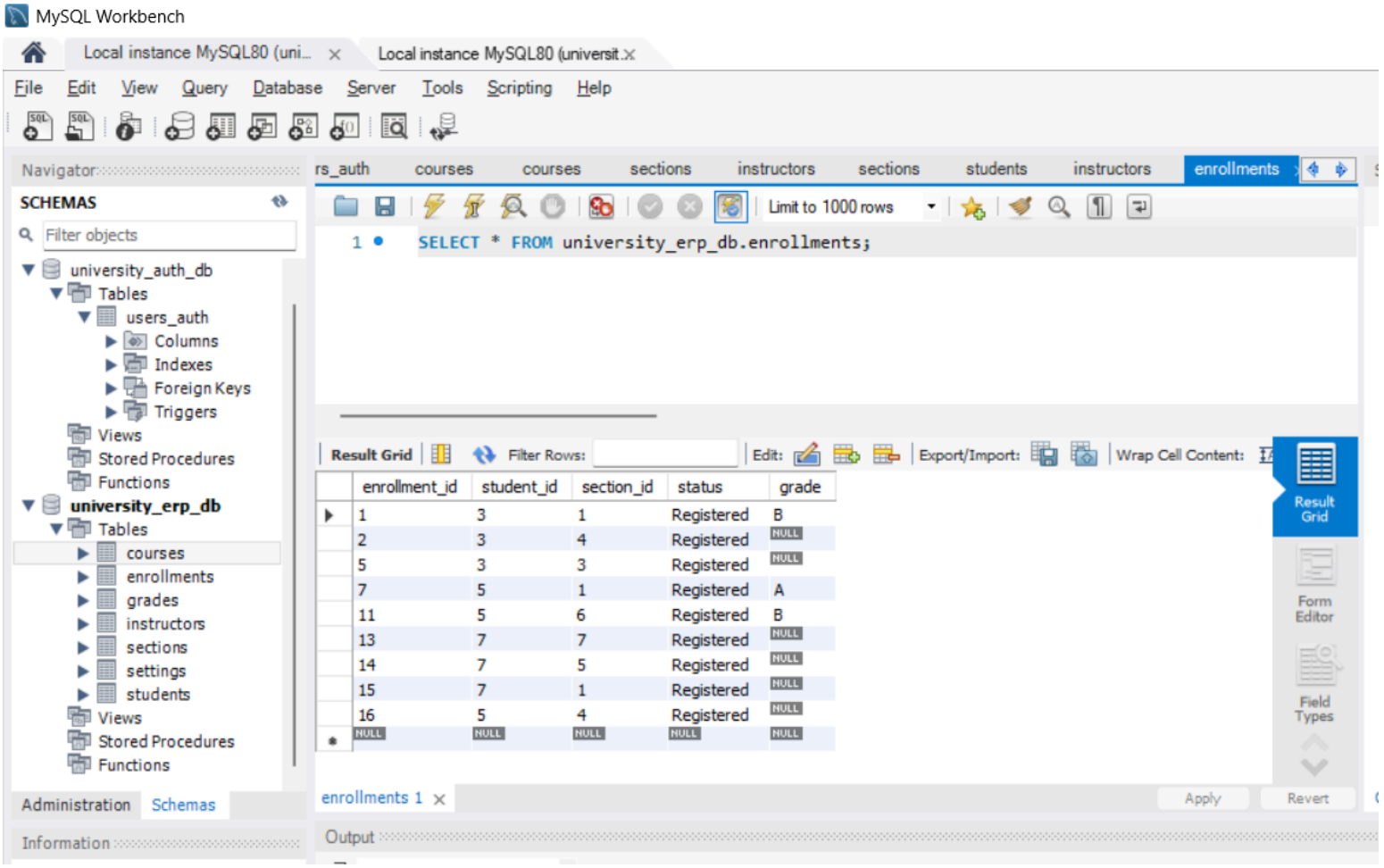
- **Payment Gateway Integration:** To allow students to pay tuition fees directly via the dashboard.
- **Attendance Tracking:** A module for instructors to record daily attendance.
- **Email Notifications:** Integrating JavaMail API to send automatic email alerts for grade postings or registration confirmations.
- **Mobile App:** Developing a companion Android/iOS application using REST APIs exposed by the service layer.

11. References

- MySQL 8.0 Reference Manual.
- BCrypt Library Documentation.


MySQL Workbench
Local instance MySQL80 (uni...
Local instance MySQL80 (universit...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
university_auth_db
Tables
users_auth
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
university_erp_db
Tables
courses
enrollments
grades
instructors
sections
settings
students
Views
Stored Procedures
Functions
Administration Schemas
Information
Table: courses
Columns:
course_id int AI PK
code varchar(10)
title varchar(100)
credits decimal(3,1)
Object Info Session
tions instructors sections users_auth courses courses sections instructors sections
Limit to 1000 rows
1 • SELECT * FROM university_erp_db.sections;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
section_id course_id instructor_id day time room capacity semester year
1 1 2 Mon 10:00:00 B205 30 Fall 2024
2 2 2 Tue 11:00:00 B206 25 Fall 2024
3 1 NULL W 11:00 AM - 12:00 PM B203 100 Fall 2025
4 3 NULL W 1:00 PM - 2:00 PM B405 10 Fall 2025
5 4 2 W 9:30 AM - 11:30 AM 60 Summer 2025
6 5 6 F 2:30 PM - 5:00 PM A007 90 Summer 2025
7 5 6 Th 8:30 AM - 10:30 AM A007 90 Summer 2025
8 3 NULL Th 10:10 AM - 12:00 PM C100 -12 Fall 2025
Result Grid
Form Editor
Field Types
sections 1
Apply Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
202 16:44:14 ALTER TABLE users_auth ADD COLUMN failed_attempts INT DEFAULT 0 Error Code: 1146. Table 'university_erp_db.users_auth' doesn't exist 0.016 sec
203 16:45:08 ALTER TABLE university_auth_db.users_auth ADD COLUMN failed_attempts INT DEFAULT 0 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.047 sec
204 16:45:08 ALTER TABLE university_auth_db.users_auth ADD COLUMN lock_time DATETIME NULL 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.000 sec
205 16:54:00 SELECT * FROM university_erp_db.courses LIMIT 0, 1000 5 row(s) returned 0.000 sec / 0.000 sec
206 16:55:06 SELECT * FROM university_erp_db.sections LIMIT 0, 1000 7 row(s) returned 0.000 sec / 0.000 sec
207 16:55:34 SELECT * FROM university_erp_db.instructors LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
208 18:00:02 SELECT * FROM university_erp_db.sections LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 sec
Query Completed
ENG IN
07:52 PM 27-11-2025


SCHEMAS
Filter objects
university_auth_db
Tables
users_auth
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
university_erp_db
Tables
courses
enrollments
grades
instructors
sections
settings
students
Views
Stored Procedures
Functions
Administration Schemas
Limit to 1000 rows
1 • SELECT * FROM university_erp_db.students;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
user_id roll_no program year
3 S0003 B.Tech CS 2024
4 S0004 B.Tech IT 2024
5 S0008 ECE 2024
7 600 Btech 2024
NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
students 1
Apply Revert


MySQL Workbench
Local instance MySQL80 (uni...
Local instance MySQL80 (universit...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
university_auth_db
Tables
users_auth
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
university_erp_db
Tables
courses
enrollments
grades
instructors
sections
settings
students
Views
Stored Procedures
Functions
Administration Schemas
Information
rs_auth courses courses sections instructors sections students instructors enrollments
Limit to 1000 rows
1 SELECT * FROM university_erp_db.enrollments;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
enrollment_id student_id section_id status grade
1 3 1 Registered B
2 3 4 Registered NULL
5 3 3 Registered NULL
7 5 1 Registered A
11 5 6 Registered B
13 7 7 Registered NULL
14 7 5 Registered NULL
15 7 1 Registered NULL
16 5 4 Registered NULL
NULL NULL NULL NULL NULL
Result Grid
Form Editor
Field Types
enrollments 1
Apply
Revert
Output