Academic ERP System Report

Project Name: AP_Project (University ERP) **Course:** Advanced Programming **Date:** November 2025

1. System Overview

The Academic ERP System is a comprehensive Java-based application designed to manage the academic lifecycle of a university. It facilitates interaction between three distinct user roles: **Students, Instructors, and Administrators**. The system is built using a **Swing GUI** with FlatLaf styling for the frontend, **Java JDBC** with **HikariCP** for high-performance database interactions, and **MariaDB** as the backend storage.

The architecture separates concerns into Service, Domain, Data, and UI layers, ensuring modularity and testability (verified via JUnit 5).

2. Database Schema & Tables

The system utilizes a dual-database architecture to separate authentication credentials from operational data for enhanced security.

2.1. Authentication Database (auth_db)

This database handles user login credentials and role distribution. It ensures that sensitive password hashes are kept separate from general user data.

Table Name	Description	Key Columns
users	Stores login credentials for all actors.	User name, password_hash, role

2.2. ERP Database (erp_db)

This database stores the core business logic data, including academic records, course catalogs, and system settings.

Table Name	Description	Key Columns
students	Stores student personal and academic info.	roll_number , program,batch
instructors	Stores faculty details.	instructor_id ,dept
courses	The catalog of available courses.	course_code , title, section , credits
sections	Specific instances of courses for a semester.	course_code, section ,instructor_id, room, capacity, semester ,year
enrollments	Links students to sections.	student_id, section, course code , status
grades	Stores marks and final grades.	Student id,course code ,section , midsem, endsem, assignment, grade
settings	Global flags for system control.	setting_key , setting_value

notifications System alerts for users. id , user id, message

3. Role Enforcement & Maintenance Mode

3.1. Role-Based Access Control (RBAC)

Security is enforced at the entry point of the application.

1. **Authentication:** When a user logs in via loginPage.java, the LoginService hashes the input password using **BCrypt** and compares it against the auth_db.
2. **Role Retrieval:** Upon success, the database returns the user's role string ("admin", "instructor", or "student").
3. **Dashboard Routing:** The UI logic strictly routes the user to their specific dashboard:
    - **Student:** studentDashboard.java (Cannot access Admin/Instructor features).
    - **Instructor:** InstructorDashboard.java.
    - **Admin:** adminDashboard.java.

3.2. Maintenance Mode Enforcement

The system includes a **Maintenance Mode** to prevent data inconsistency during updates (e.g., when an Admin is modifying course catalogs).

- **Activation:** An Admin toggles this mode via the maintainMode.java UI. This updates the system_settings table in erp_db (setting maintenance_mode = '1').
- **Enforcement:**
    - Critical operations in StudentService.java (like registerCourse or dropCourse) call the isSystemActive() helper method.
    - If isSystemActive() returns false, the operation is blocked immediately, and an exception is thrown/displayed to the user: *"System is currently under maintenance."*
    - This logic is verified in StudentServiceTest.java.

4. Final Grade Weighting Rule

The grading logic is encapsulated within InstructorService.java and utilized in the ComputeFinalFrame.java UI. The system automatically calculates a numerical score based on weighted components and assigns a letter grade.

4.1. The Formula

The final score is calculated using the following strict weighting:

*FinalScore = (MidSem *times 0.30) + (EndSem *times 0.50) + (Assignment *times 0.20)*

```java
public double computeStats(String courseCode, String section) throws SQLException {
    String[] grades = ErpCommandRunner.instructorStatsHelper(courseCode, section);
    if (grades == null || grades.length == 0) { return -1.0; }
    int totalPoints = 0;
    for (String grade : grades) {
        switch (grade.toUpperCase()) {
            case "A": totalPoints += 3; break;
            case "B": totalPoints += 2; break;
            case "C": totalPoints += 1; break;
            case "F": totalPoints += 0; break;
            default: break;
        }
    }
    return (double) totalPoints / grades.length;
}
```

4.2. Implementation Details

- **Input Validation:** The system checks that input marks do not exceed the maximum allowed for that component (e.g., Midsem max is 100).
- **Grade Mapping:** The calculated numerical score is mapped to a letter grade:

```java
public String computeAndAssignGrade(String insRollno, String stdRollno, String courseCode, String sec
    try {
        int quiz = Integer.parseInt(quizMarks);
        int mid = Integer.parseInt(midMarks);
        int end = Integer.parseInt(endMarks);

        // Marks Validation
        if (quiz > 10 || mid > 10 || end > 10) {
            return "Error: Marks cannot exceed 10.";
        }

        // Calculate Grade
        int total = (int) (0.2 * quiz + 0.3 * mid + 0.5 * end);
        String finalGrade;
        if (total >= 8) finalGrade = "A";
        else if (total >= 6) finalGrade = "B";
        else if (total >= 4) finalGrade = "C";
        else finalGrade = "F";
        // ------------------------
```

  - **A:** 10>=Score >= 9
  - **B:** 9> Score >= 8
  - **C:** 8> Score >= 7
  - **D:** 7> Score >= 6
  - **F:** Score <6

5. Extras & Advanced Features Added

Beyond the standard requirements, the following "Extra" features were implemented to enhance usability, security, and performance.

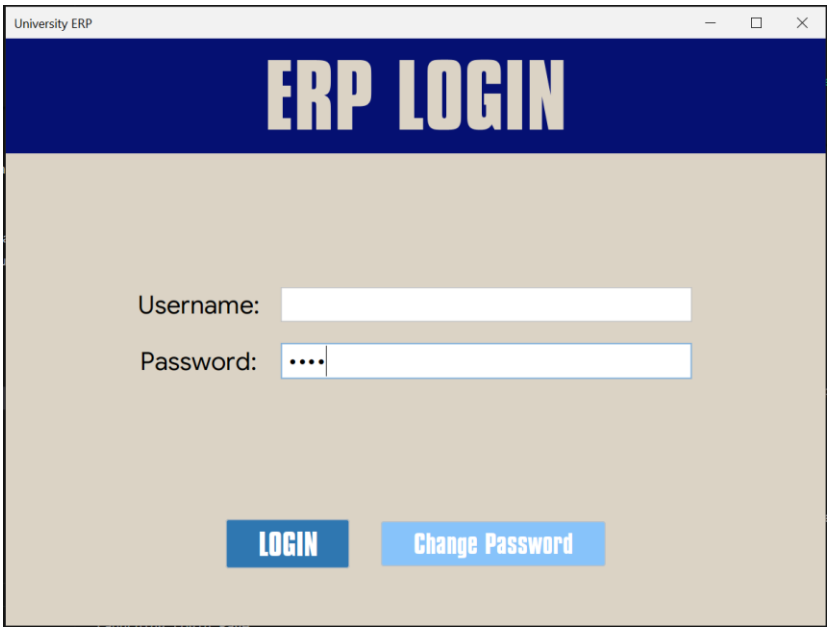
1. **HikariCP Connection Pooling:**
   - Instead of opening a new JDBC connection for every query (which is slow), the system uses HikariDataSource in DatabaseConnector.java. This maintains a pool of ready-to-use connections, significantly improving login and dashboard loading times.

2. **Secure Password Hashing (BCrypt):**
    - Passwords are never stored in plain text. The HashGenerator class uses the **BCrypt** algorithm (via jbcrypt library) to salt and hash passwords, ensuring industry-standard security.
3. **Modern UI (FlatLaf & Custom Fonts):**
    - The application uses **FlatLaf** (Flat Light Look and Feel) for a modern, clean aesthetic, replacing the outdated default Java Swing look.
    - A custom font loader (BREATHEFONT.java) integrates distinct typography ("Compacta" and "Google Sans") for branding.
4. **Backup and Restore Utility:**
    - An admin interface (BackupRestorePage.java) allows administrators to:
        - **Backup:** Generate a SQL dump of the current database state.
        - **Restore:** Revert the database to a previous state from a SQL file.
    - *Note: This relies on mariadb-dump and mariadb CLI tools being present in the system PATH.*
5. **Notifications System:**
    - A custom NotificationDialog and database table allow the system to provide asynchronous feedback to users (e.g., confirming a course drop or grade update).

6. Application Screenshots

6.1. Login Screen

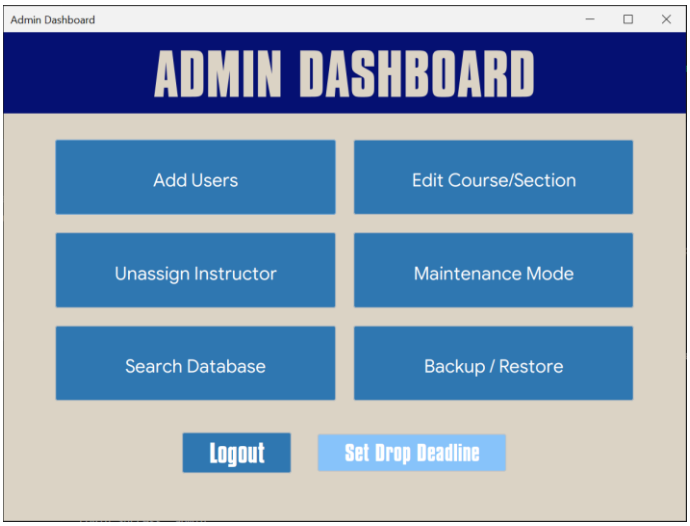
Shows the custom font "ERP LOGIN" and FlatLaf styling

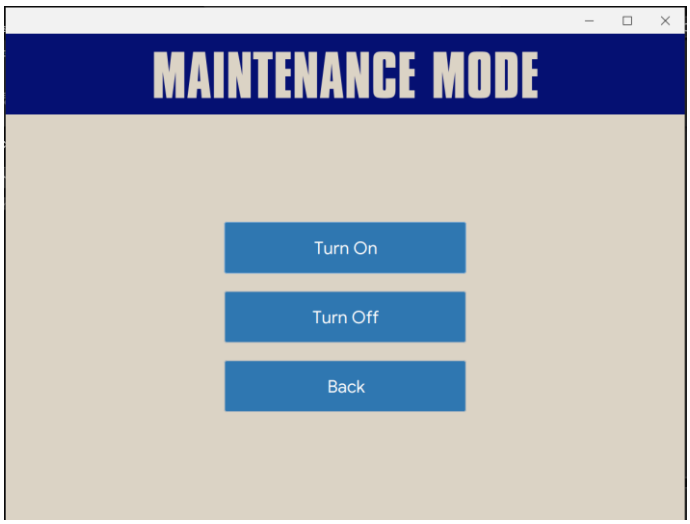


The secure login portal supporting Student, Instructor, and Admin credentials.

6.2. Admin Dashboard & Maintenance Mode

Shows the grid layout of admin functions and the Maintenance Mode toggle.

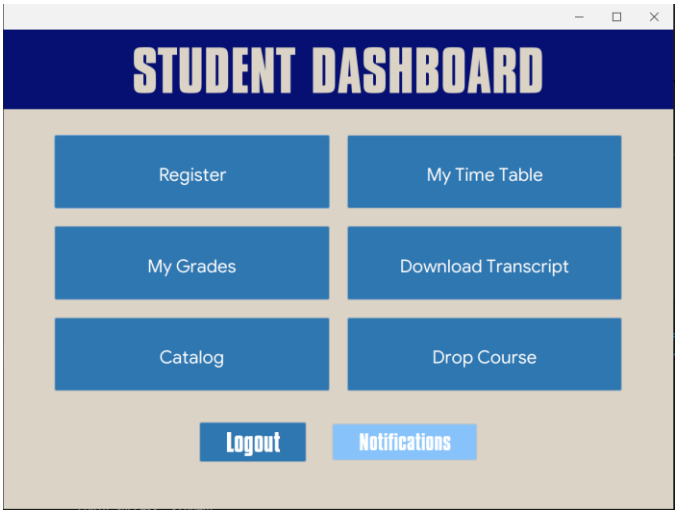


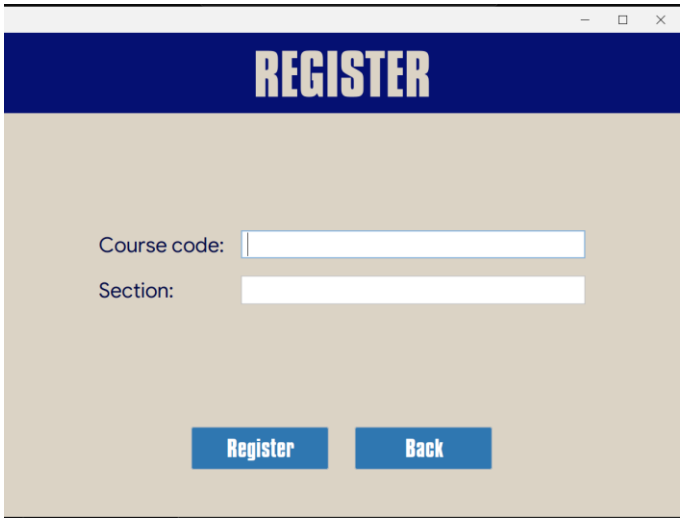


Admin Dashboard showing course creation, user management, and maintenance controls.

6.3. Student Dashboard & Registration

Shows the student's personal details and the course registration interface

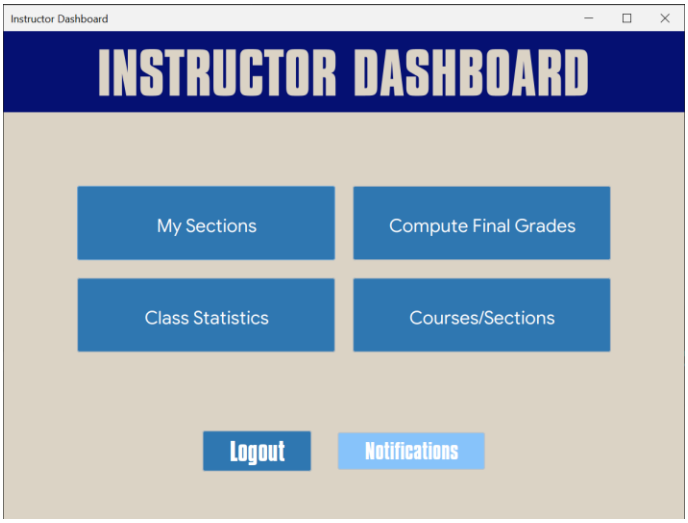


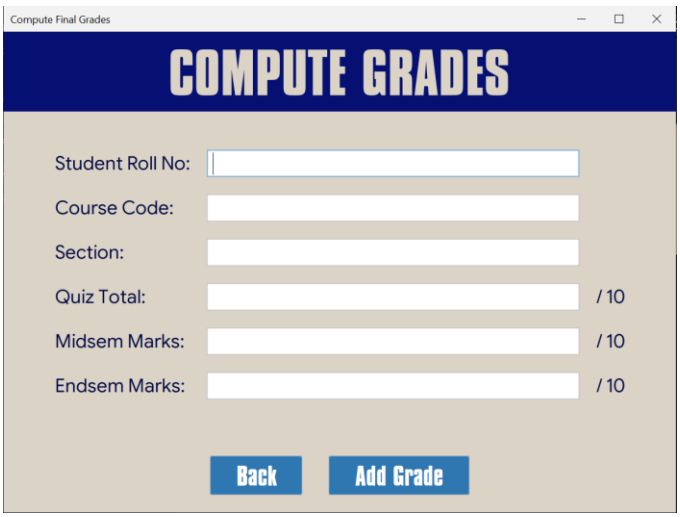


Student interface for viewing transcripts and registering for courses.

6.4. Instructor Grading Interface

Shows the input fields for Midsem, Endsem, and Assignment marks, demonstrating the weighting rule UI

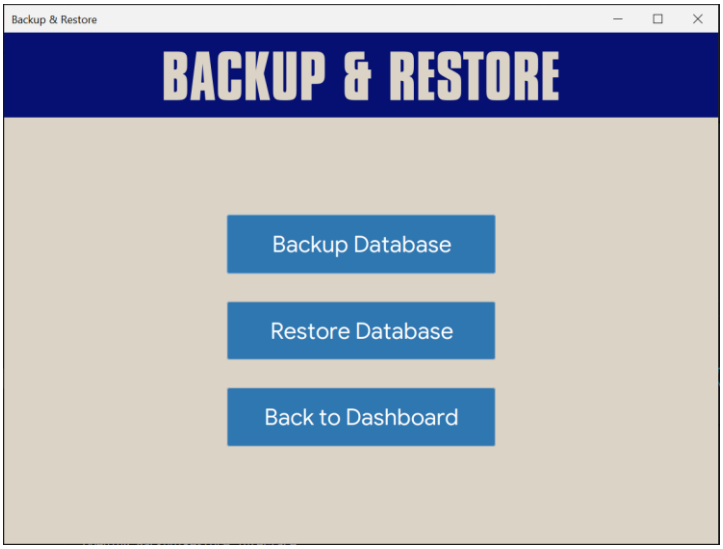




Instructor interface for computing final grades based on the 30-50-20 weighting logic.

6.5. Backup & Restore Page

Shows the "Backup Data" and "Restore Data" buttons.



Database administration tools for disaster recovery.

7. Conclusion

The AP ERP Project successfully implements a robust, three-tier academic management system. By enforcing strict database constraints, implementing secure authentication, and wrapping the business logic in a responsive Swing GUI, the system meets the complex needs of academic administration while ensuring data integrity and user security.