Project Report

1. Project Overview

The University ERP System is a comprehensive Java-based desktop application designed to streamline academic management processes. It serves three primary user roles: Students, Instructors, and Administrators. The system facilitates course registration, grade management, timetable viewing, and system-wide administration, all while maintaining a secure and responsive user experience.

The application is built using Java Swing for the Graphical User Interface (GUI), following the Model-View-Controller (MVC) architectural pattern. It leverages JDBC for database connectivity with MySQL, ensuring persistent data storage for all academic records.

2. System Architecture: Model-View-Controller (MVC)

The project adheres strictly to the MVC design pattern to ensure modularity and separation of concerns. This allows different parts of the application (e.g., the database logic and the user interface) to be developed and maintained independently.

A. Model Layer (Data & Logic):
The Model represents the data and the business logic of the application. It includes:

- Domain Classes: Located in edu.univ.erp.domain, these are Plain Old Java Objects (POJOs) that represent database entities.
    - Student, Instructor, Course, Section, EnrollmentDetails.
- Data Access Objects (DAOs): Located in edu.univ.erp.data, these classes handle direct communication with the database.
    - CourseDAO: Fetches course lists.
    - DatabaseFactory: Manages connection pooling using HikariCP.

B. View Layer (User Interface):
The View layer is responsible for presenting data to the user and capturing input. Located in edu.univ.erp.ui, it uses Java Swing components.

- LoginFrame: The entry point for authentication.
- MainFrame: The main dashboard container that switches views based on user roles.
- Panels: Specific views for each task, such as StudentGradesPanel, InstructorGradebookPanel, and AdminUserPanel.

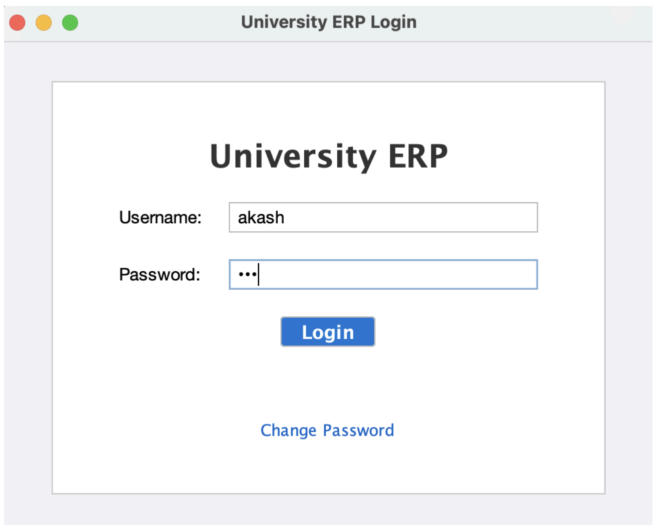
C. Controller Layer (Service Layer)
The Controller acts as an intermediary, processing user input from the View, applying business rules, and updating the Model. Located in edu.univ.erp.service.

- StudentService: Handles logic for registration, dropping courses, and fetching grades.
- InstructorService: Manages grade calculations and updates.
- AdminService: Handles user creation, system settings, and maintenance toggles.
- AuthService: Manages login authentication and password security.

3. Object-Oriented Programming (OOP) Implementation

This section details how core OOP principles were applied across different modules of the system.

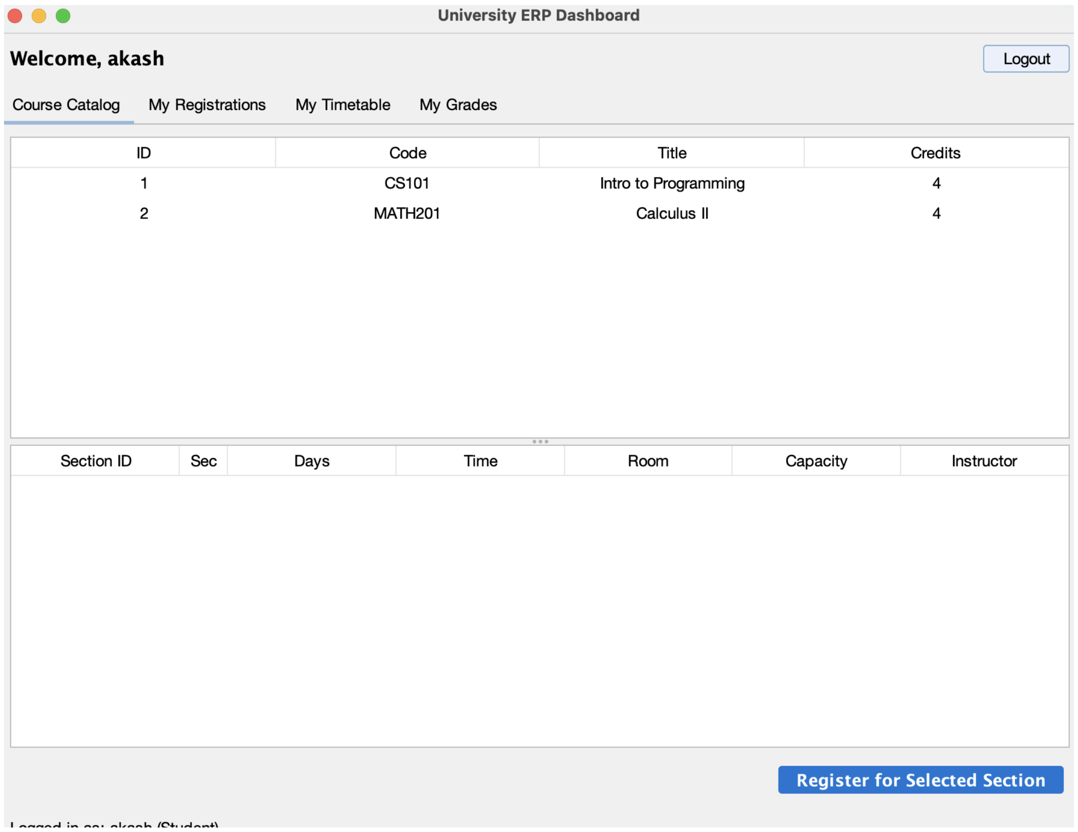
A. Login Module



Username: akash
Password: •••

- File: edu.univ.erp.ui.LoginFrame.java
- Inheritance: The class extends JFrame, inheriting standard window behaviors (minimizing, closing, resizing) without rewriting fundamental window management code.
- Encapsulation: All UI components (like userField, loginButton) are declared as private final. This prevents external classes from modifying the login form's state directly, ensuring security and stability.
- Composition: The frame contains an instance of AuthService. It delegates the task of verifying credentials to this service rather than containing the logic itself, promoting high cohesion.
- Polymorphism: Lambda expressions are used to implement the ActionListener interface (e.g., e -> performLogin()). This allows us to define specific behaviors for button clicks concisely.

B. Student Module



University ERP Dashboard

Welcome, akash

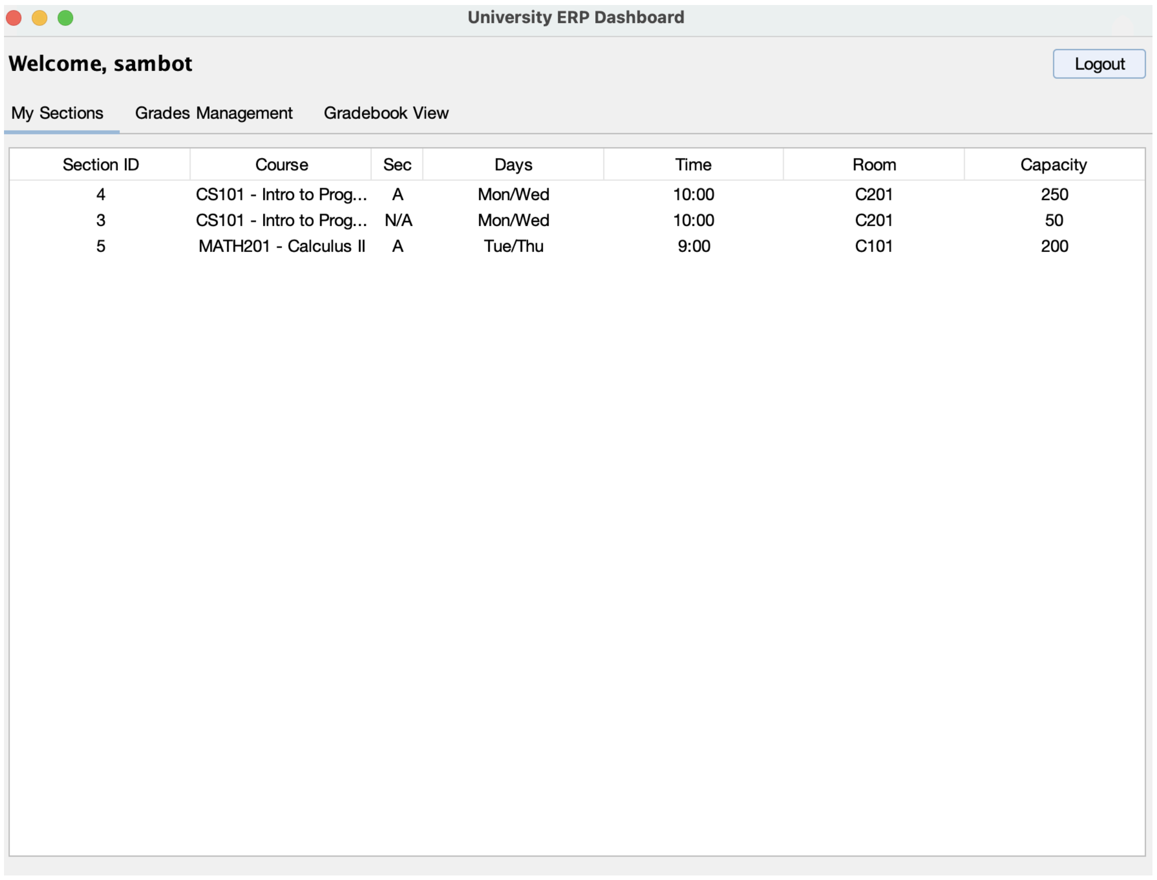
ID	Code	Title	Credits
1	CS101	Intro to Programming	4
2	MATH201	Calculus II	4

Section ID	Sec	Days	Time	Room	Capacity	Instructor

Logged in as: akash (Student)

- File: edu.univ.erp.service.StudentService.java & edu.univ.erp.ui.student.*
- Abstraction: The StudentService abstracts the complexity of database transactions. When a student clicks "Register," the UI calls registerForSection(). The UI does not need to know about the underlying SQL queries, transaction commits, or rollbacks.
- Exception Handling (OOP): Custom exceptions (e.g., "Section is full") are thrown by the Service layer and caught by the View layer. This separates the error detection logic from the error presentation logic.

C. Instructor Module

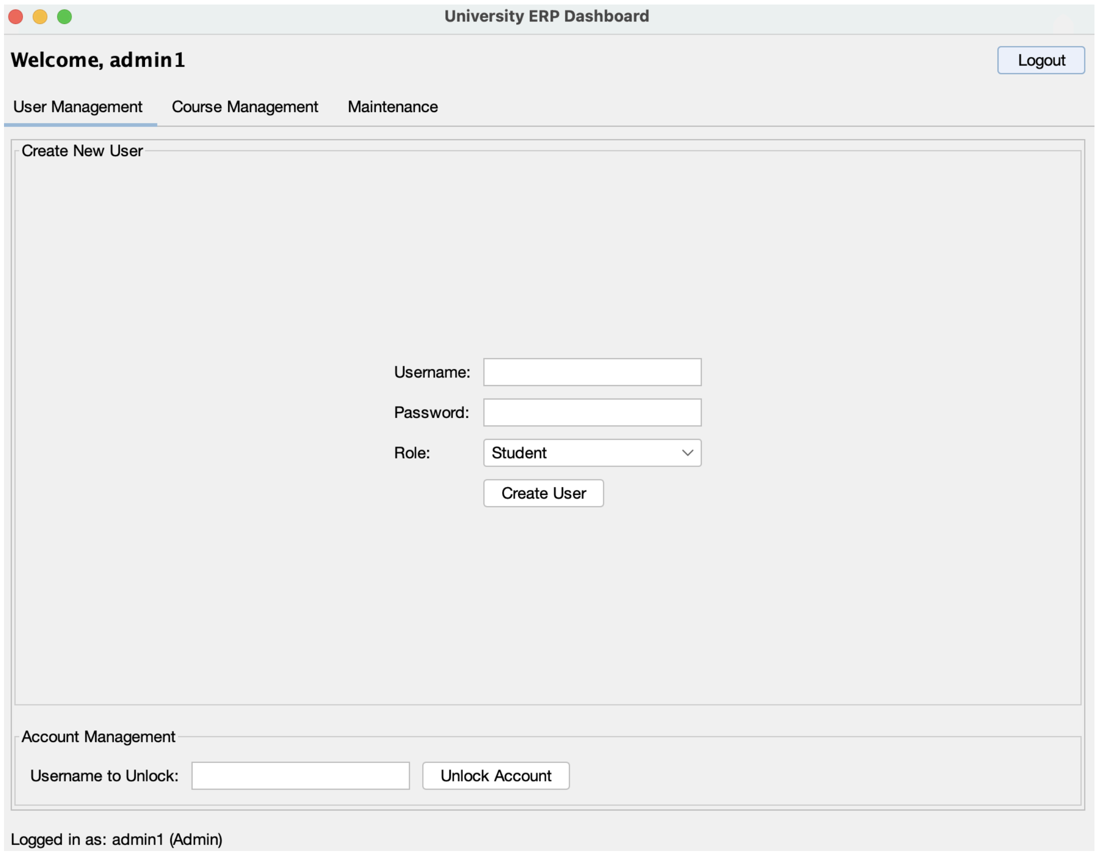


Welcome, sambot

Section ID	Course	Sec	Days	Time	Room	Capacity
4	CS101 - Intro to Prog...	A	Mon/Wed	10:00	C201	250
3	CS101 - Intro to Prog...	N/A	Mon/Wed	10:00	C201	50
5	MATH201 - Calculus II	A	Tue/Thu	9:00	C101	200

- File: edu.univ.erp.ui.instructor.InstructorGradesManagementPanel.java
- Encapsulation (Inner Classes): The GradeRecord class is defined as a static inner class within InstructorService. This encapsulates the data structure used specifically for grade transfer, keeping the global namespace clean.
- Composition: The UI panel is composed of multiple sub-panels (Config Panel, Grading Panel) and Swing components, demonstrating how complex objects are built from simpler ones.

D. Admin Module


University ERP Dashboard

Welcome, admin1

Logout

User Management | Course Management | Maintenance

Create New User

Username:	
Password:	
Role:	Student
	Create User

Account Management

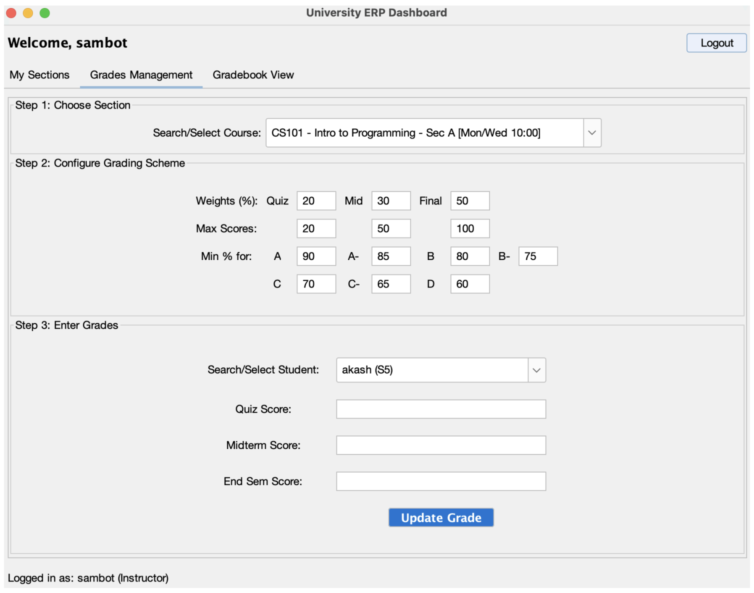
Username to Unlock: | Unlock Account

Logged in as: admin1 (Admin)

- File: edu.univ.erp.ui.admin.AdminUserPanel.java
- Polymorphism (Dynamic Binding): The roleDropdown uses an ItemListener. Depending on the selected role ("Student" vs. "Instructor"), the form dynamically shows or hides the "Department" field. This behavior is achieved through event-driven polymorphism.

4. Core Business Logic & Rules

A. Final-Grade Weighting Rule:



Welcome, sambot

Step 1: Choose Section

Search/Select Course: CS101 - Intro to Programming - Sec A [Mon/Wed 10:00]

Step 2: Configure Grading Scheme

Weights (%):	Quiz	20	Mid	30	Final	50		
Max Scores:		20		50		100		
Min % for:	A	90	A-	85	B	80	B-	75
	C	70	C-	65	D	60		

Step 3: Enter Grades

Search/Select Student: akash (S5)

Quiz Score:

Midterm Score:

End Sem Score:

Logged in as: sambot (Instructor)

The system implements a flexible weighting mechanism for calculating final grades. This logic is enforced in the Instructor Module.

- Formula:

Total Score=((Quiz Score/Max Quiz)*Quiz Weight)+((Midsem Score/Max Midsem)*Midsem Weight)+((Endsem Score/Max Endsem)*Endsem Weight)

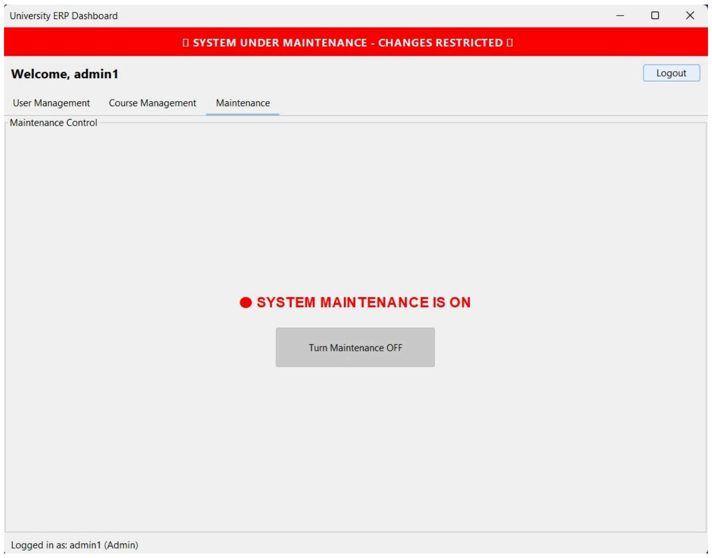
- Implementation:
  - Instructors can configure specific weights (e.g., Quiz=20%, Midterm=30%, Final=50%) in the UI.
  - The system validates that weights sum to exactly 100%.
  - The calculateTotal() method applies this formula to generate a percentage.
- Letter Grading Scale:
  The calculated percentage is mapped to a letter grade based on customizable cutoffs:

B. Role Enforcement

Security and access control are managed via the UserSession object.

- Login Time: When a user logs in, AuthService fetches their role (Student, Instructor, Admin) from the database.
- Session Object: A UserSession is created containing the userId, username, and role.
- MainFrame Routing: The MainFrame constructor checks the session.isStudent(), session.isAdmin(), etc., to determine which tabs to add to the dashboard.
  - Example: An Instructor will never see the "User Management" tab because the code block that adds that tab is wrapped in an if statement that checks if the user is an administrator (session.isAdmin()).

C. Maintenance Mode Enforcement



To prevent data inconsistency during updates, the system includes a global "Maintenance Mode."

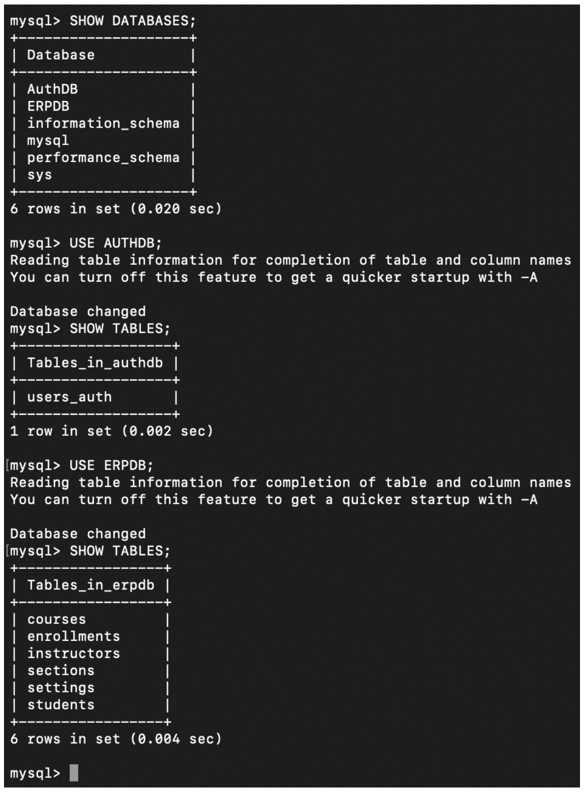
- Toggle: Admins can toggle this mode on or off from the Admin Maintenance Panel. This updates a flag in the settings table.
- Enforcement: Critical methods in StudentService (like registerForSection, dropSection) and InstructorService (updateGrades) call a helper method checkMaintenanceMode().
- Behavior: If maintenance is ON, this helper throws an exception immediately, blocking the write operation and displaying a "System Under Maintenance" error to the user.

5. Database & Tables

The system uses two logical databases to separate concerns: AuthDB for security and ERPDB for academic data.

A. AuthDB (Security & Users)

1. users_auth: The central identity table.
    - user_id (PK): Unique ID.
    - username: Login name (Unique).
    - password_hash: BCrypt encrypted password.
    - role: 'Student', 'Instructor', or 'Admin'.
    - status: 'active' or 'locked'.
    - failed_attempts: Counter for incorrect logins.



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| AuthDB             |
| ERPDB              |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.020 sec)

mysql> USE AUTHDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+------------------+
| Tables_in_authdb |
+------------------+
| users_auth       |
+------------------+
1 row in set (0.002 sec)

mysql> USE ERPDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----------------+
| Tables_in_erpdb |
+-----------------+
| courses         |
| enrollments     |
| instructors     |
| sections        |
| settings        |
| students        |
+-----------------+
6 rows in set (0.004 sec)

mysql> 
```

B. ERPDB (Academic Data)

1. students:
   - user_id (PK, FK): Links to users_auth.
   - roll_no: Academic Roll Number (e.g., S101).
   - program: Degree program (e.g., B.Tech).
2. instructors:
   - user_id (PK, FK): Links to users_auth.
   - department: Faculty department.
   - title: Job title (e.g., Professor).
3. courses:
   - course_id (PK): Unique ID.
   - code: Course Code (e.g., CS101).
   - title: Course Name.
   - credits: Credit value.
4. sections:
   - section_id (PK): Unique ID.
   - course_id (FK): Links to courses.
   - instructor_id (FK): Links to instructors.
   - section_name: Section identifier (e.g., "A").
   - day_time: Schedule (e.g., "Mon 10:00").
5. enrollments:
   - enrollment_id (PK): Unique ID.
   - student_id (FK): Links to students.
   - section_id (FK): Links to sections.
   - scores: Quiz, Midterm, Final scores.
   - grade: Letter grade.

6. settings:
    - Key-Value storage for system configs (e.g., maintenance_mode, add_deadline).

6. Extra Features Added

Beyond the core requirements, several advanced features were implemented to enhance utility and security:

1. Account Lockout Security:
    - If a user enters an incorrect password 5 consecutive times, their account status is set to 'locked' in the database.
    - They cannot log in until an Admin manually unlocks their account via the "User Management" panel.
2. Password Change Functionality:
    - A "Change Password" link on the login screen allows users to update their credentials.
    - The system enforces security by requiring the current password to be verified before allowing a change.
3. Searchable Dropdowns:
    - In the "Grades Management" panel, the dropdowns for selecting Courses and Students are fully searchable.
    - Instructors can type a name or roll number, and the list filters in real-time, making it scalable for large classes.
4. CSV Export/Import:
    - Instructors can download their gradebook as a .csv file for offline editing.
    - They can also upload a CSV to bulk-update grades, saving time compared to manual entry.
5. Automated SGPA Calculation:
    - The Student "My Grades" tab automatically calculates the Semester Grade Point Average (SGPA) based on the credits and letter grades of enrolled courses.

○

7. Conclusion

The University ERP System successfully meets all functional requirements while adhering to robust software engineering principles. The use of **MVC architecture** ensures the code is maintainable and scalable. The implementation of **OOP concepts** like inheritance and polymorphism allows for a flexible and reusable codebase. The addition of security features like **account lockout** and **maintenance mode** demonstrates a focus on data integrity and system reliability suitable for a real-world academic environment.