University ERP System - Project Documentation

Advanced Programming (Java + Swing)

1. Executive Summary

This desktop application implements a University ERP system using Java and Swing, managing course registration, enrollment, and grading across three user roles: Student, Instructor, and Admin.

Dual-Database Architecture:

- • Auth DB (authdb): User authentication with BCrypt password hashing
- • ERP DB (erpdb): Application data (students, courses, sections, enrollments, grades)

Key Features Implemented:

- • Role-based access control with secure authentication
- • Course registration with capacity management and deadline enforcement
- • Gradebook with CSV import/export functionality
- • Maintenance mode to freeze write operations system-wide
- • Duplicate course enrollment prevention across different sections

2. System Architecture & Package Structure

The application follows a modular MVC-like architecture:

- • auth/ - Authentication logic, password hashing
- • dao/ - Data Access Objects for database operations
- • domain/ - Entity classes (Student, Course, Section, etc.)
- • service/ - Business logic layer
- • ui/ - Swing UI components (admin, instructor, student panels)
- • data/ - Database connection management

Design Principle: UI components never directly access the database. All operations go through the service layer.

3. Authentication & Security

Login Process:

1. 1. User credentials validated against authdb
2. 2. Password verification using BCrypt.checkpw()
3. 3. On success: Update last_login, create UserSession
4. 4. On failure: Increment attempt counter; 5 failures = 30-second lockout

Password Security:

- • Passwords never stored in plaintext - only BCrypt hashes in authdb
- • ERP database contains NO password information

- • Change Password feature available in all dashboards

4. Role-Based Access Control

Admin: Manage users, courses, sections; set deadlines; toggle maintenance mode; backup/restore database

Instructor: View assigned sections; add assessments; enter grades; CSV import/export; calculate final grades; view statistics

Student: Browse courses; register/drop sections; view timetable and grades; download transcript

Access Enforcement: Every write operation checks: (1) User role authorization, (2) Maintenance mode OFF, (3) Resource-specific permission (e.g., instructor can only grade their sections)

5. Core Features Implementation

5.1 Maintenance Mode

System-wide freeze for administrative tasks.

• When ON: All users can VIEW data; NO write operations allowed
• Red banner displays "MAINTENANCE MODE: ON" in all dashboards
• Boolean flag stored in erpdb.settings table
• Toggle available only to Admin

5.2 Course Registration with Duplicate Prevention

Registration Validation:

✓ Maintenance mode OFF
✓ Deadline not passed
✓ Section not full
✓ Not enrolled in this section
✓ Not enrolled in ANY section of this course (prevents duplicates)
✓ Credit limit not exceeded

Duplicate Prevention Implementation:

When attempting registration, system checks all student enrollments to see if any belong to the same course (different section). If found, shows:

"You are already enrolled in CS101 (Introduction to Programming)\nExisting enrollment: Section taught by Prof. Smith\n\nYou cannot register for the same course with a different professor."

5.3 Gradebook Management

Assessment Component System:

• Instructors define: Name, Max Score, Weightage (0.0-1.0)
• Weightage validation: Sum must equal exactly 1.0 (100%)
• System blocks final grade calculation if weightage sum ≠ 1.0

Final Grade Calculation Formula:

```
Final Grade = Σ(score_i / max_score_i × weightage_i × 100)
```

Example: Quiz (15/20, weight 0.2) + Midterm (40/50, weight 0.3) + Final (85/100, weight 0.5) = 81.5%

5.4 CSV Grade Import

Instructors can bulk-import grades from CSV files.

CSV Format:

```
Roll No,Quiz1,Midterm,Final\n2024316,15,40,85\n2024317,18,45,92
```

Important: First column must be "Roll No"; subsequent columns must exactly match assessment names (case-sensitive, no extra formatting)

Import Process:

1. File chooser opens for .csv selection
2. Validate header (Roll No + assessment names)
3. For each row: lookup student, enrollment, assessment; insert/update grade
4. Show results: "Successfully imported: 47 grades" + error details Uses OpenCSV library for robust parsing (handles quotes, commas in values, etc.)

5.5 Delete Student with Cascade

Admin can delete students with automatic cascade deletion of all related data:

- Student profile (erpdb.students)
- All enrollments (erpdb.enrollments)
- All grades (erpdb.grades)
- User authentication record (authdb.users_auth)

Confirmation dialog warns about permanent deletion and lists all data to be removed.

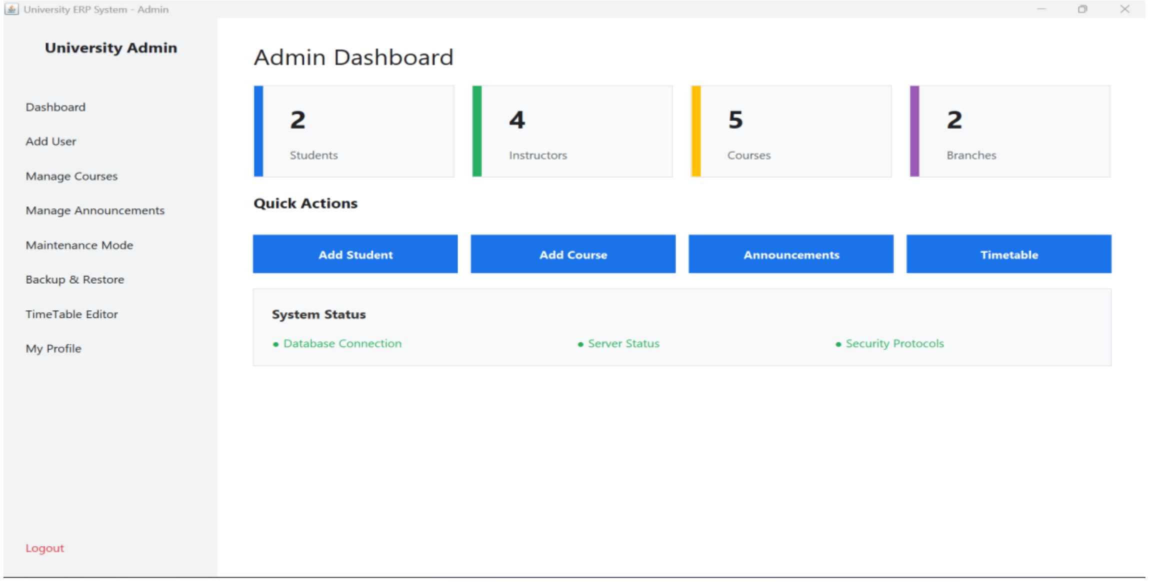
5.6 Registration Deadline Management

Admin sets deadline (YYYY-MM-DD format) in settings table. Students cannot register/drop after this date.

Enforcement: SettingsService.isAfterRegistrationDeadline() checks current date vs. stored deadline before any registration action.

6. User Interface Screenshots

Some Admin Panels



Admin Dashboard

2 Students

4 Instructors

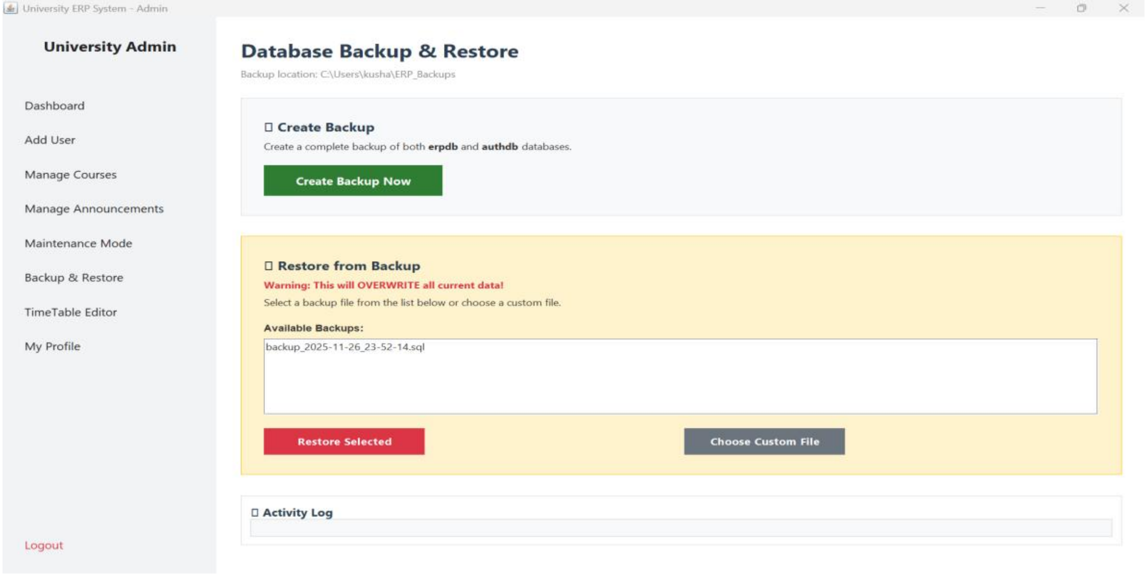
5 Courses

2 Branches

Quick Actions

System Status

● Database Connection ● Server Status ● Security Protocols



Database Backup & Restore

Backup location: C:\Users\kusha\ERP_Backups

□ Create Backup

Create a complete backup of both **erpdb** and **authdb** databases.

□ Restore from Backup

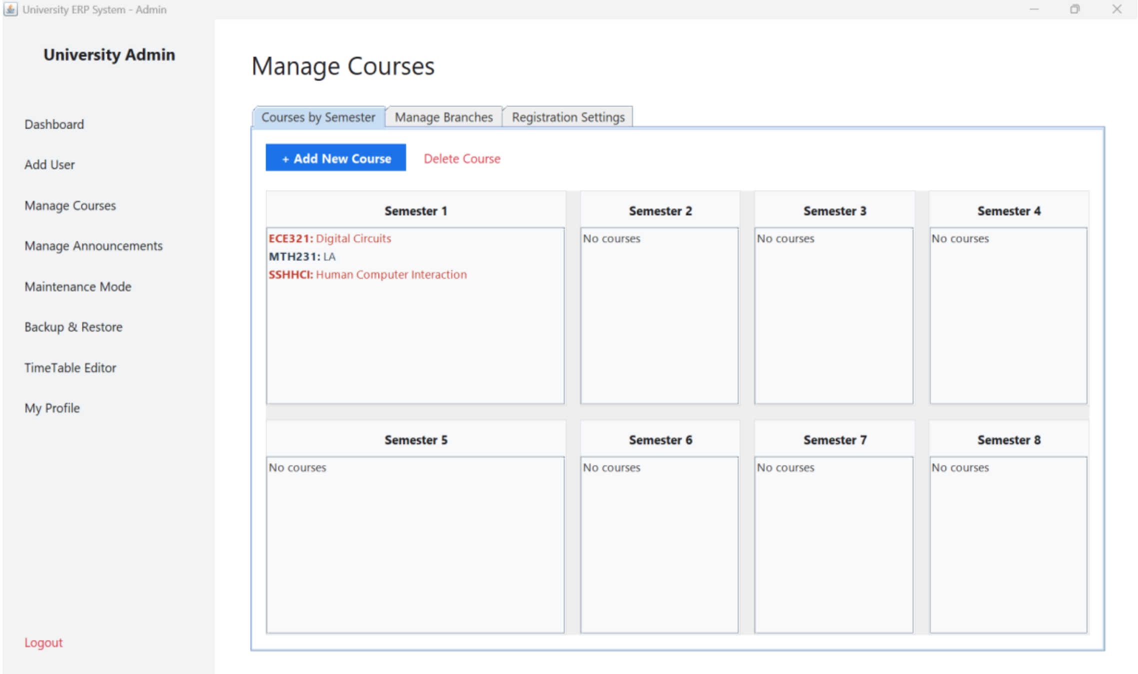
Warning: This will OVERWRITE all current data!

Select a backup file from the list below or choose a custom file.

Available Backups:

backup_2025-11-26_23-52-14.sql

□ Activity Log

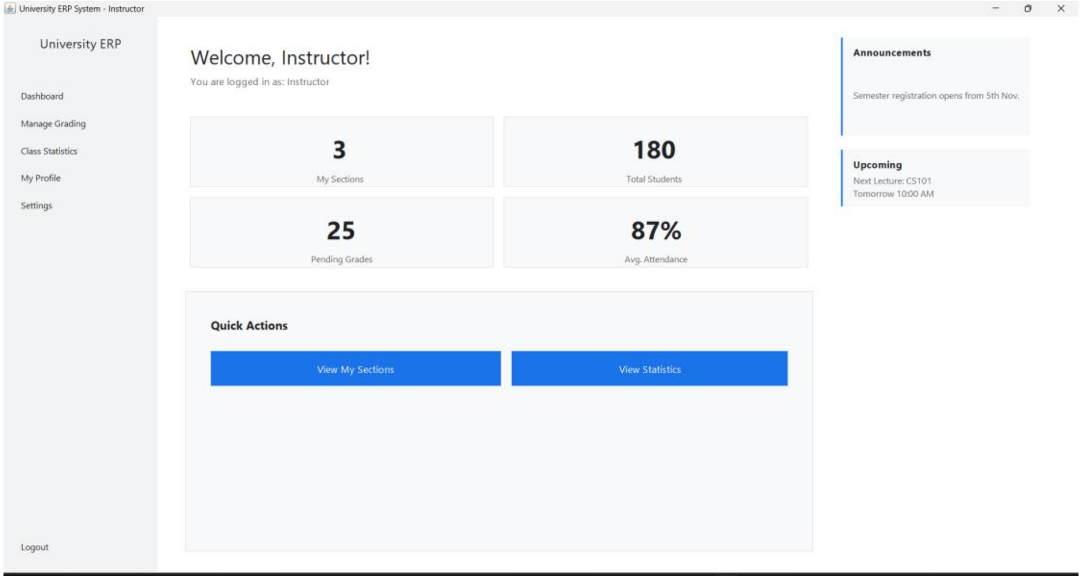


Manage Courses

Semester 1	Semester 2	Semester 3	Semester 4
ECE321: Digital Circuits **MTH231:** LA **SSHHCI:** Human Computer Interaction	No courses	No courses	No courses

Semester 5	Semester 6	Semester 7	Semester 8
No courses	No courses	No courses	No courses

Some Instructor Panels



Welcome, Instructor!

You are logged in as: Instructor

3 My Sections

180 Total Students

25 Pending Grades

87% Avg. Attendance

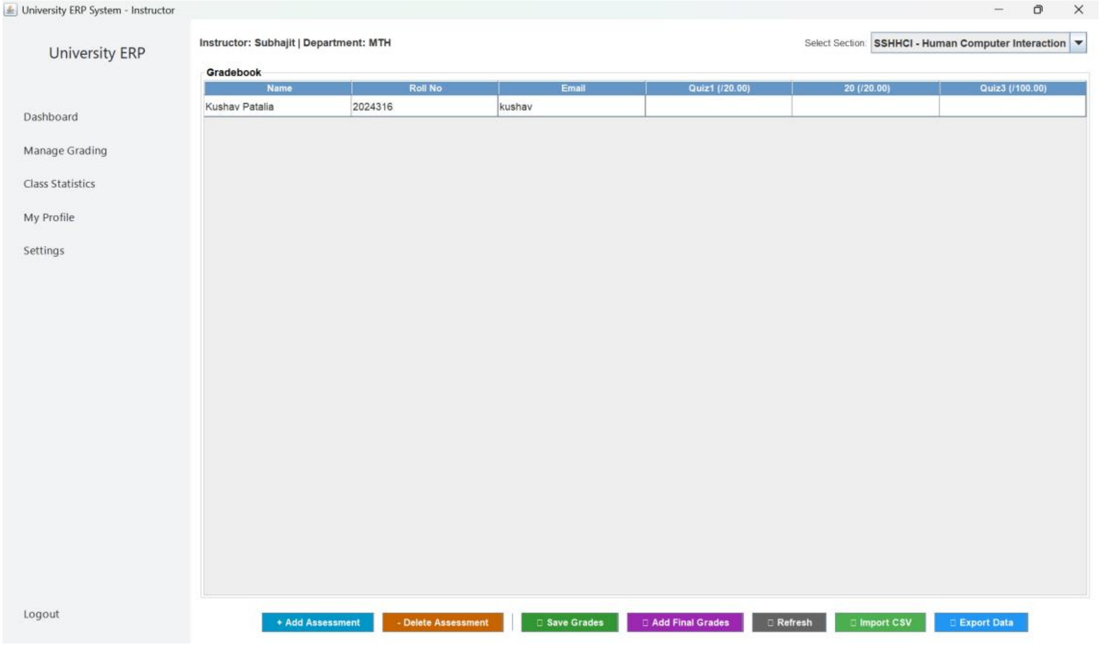
Quick Actions

Announcements

Semester registration opens from 5th Nov.

Upcoming

Next Lecture: CS101
Tomorrow 10:00 AM

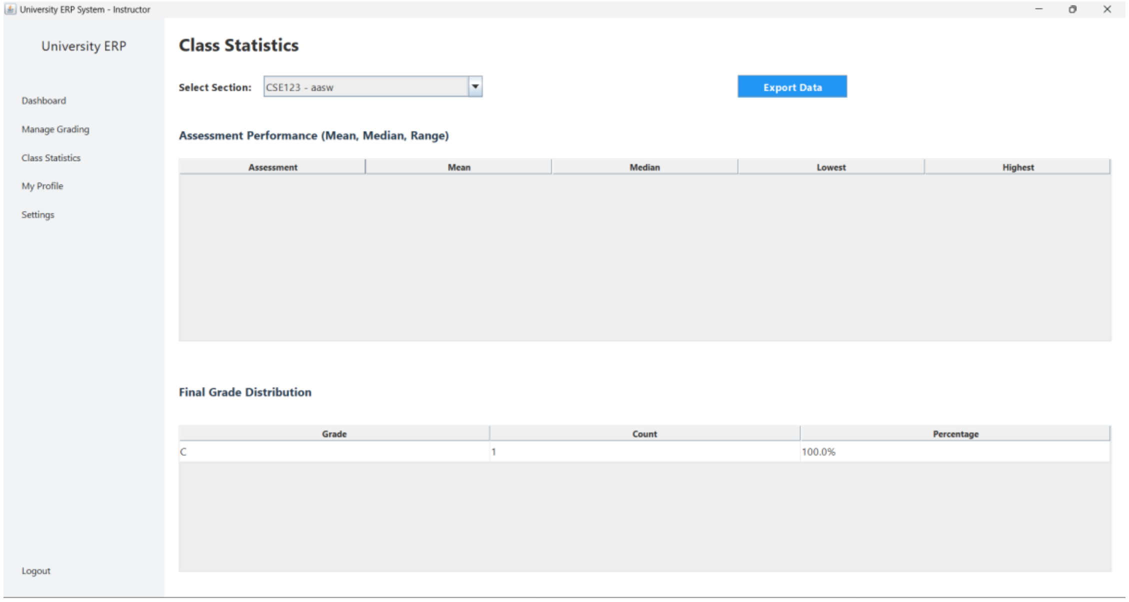


Instructor: Subhajit | Department: MTH

Select Section: SSHHCI - Human Computer Interaction

Gradebook

Name	Roll No	Email	Quiz1 (/20.00)	20 (/20.00)	Quiz3 (/100.00)
Kushav Patalia	2024316	kushav			



Class Statistics

Select Section: CSE123 - aasw

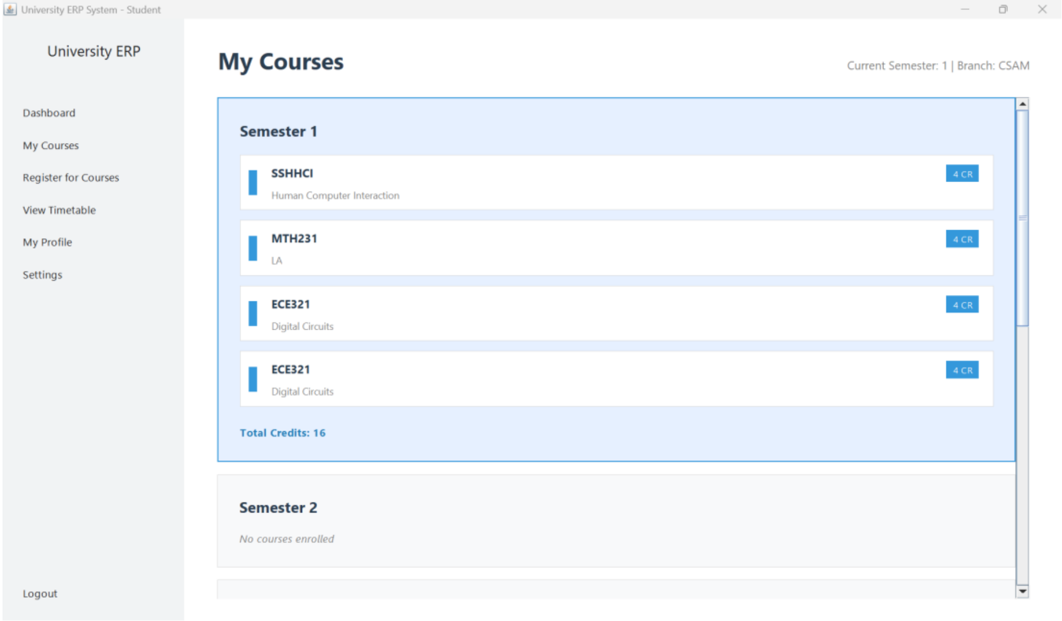
Assessment Performance (Mean, Median, Range)

Assessment	Mean	Median	Lowest	Highest

Final Grade Distribution

Grade	Count	Percentage
C	1	100.0%

Some Student Panels



My Courses

Current Semester: 1 | Branch: CSAM

Semester 1

SSHHCI 4 CR
Human Computer Interaction

MTH231 4 CR
LA

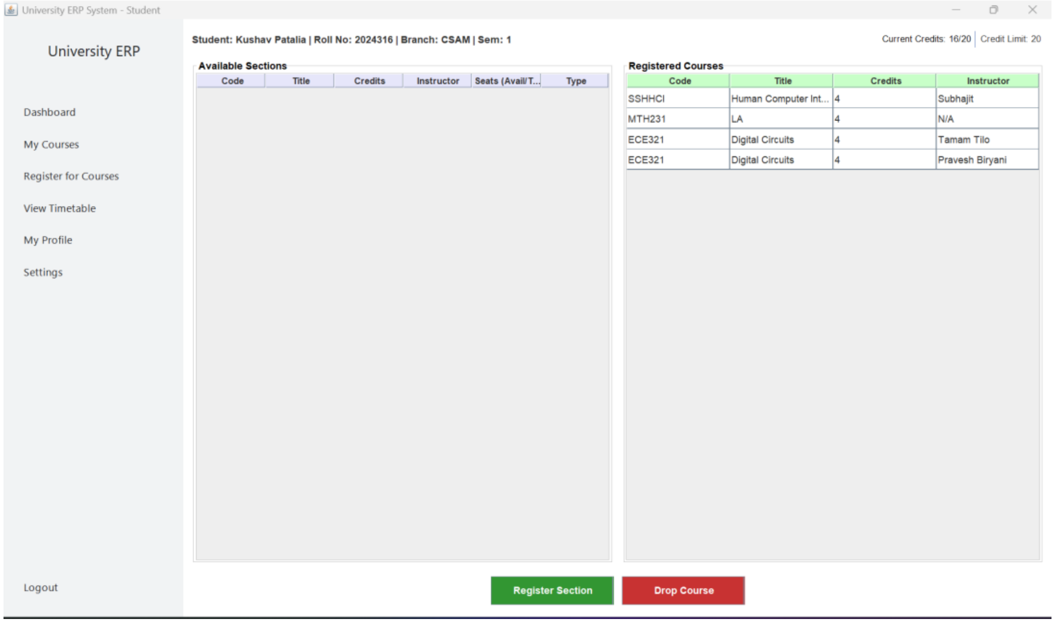
ECE321 4 CR
Digital Circuits

ECE321 4 CR
Digital Circuits

Total Credits: 16

Semester 2

No courses enrolled



Student: Kushav Patalia | Roll No: 2024316 | Branch: CSAM | Sem: 1

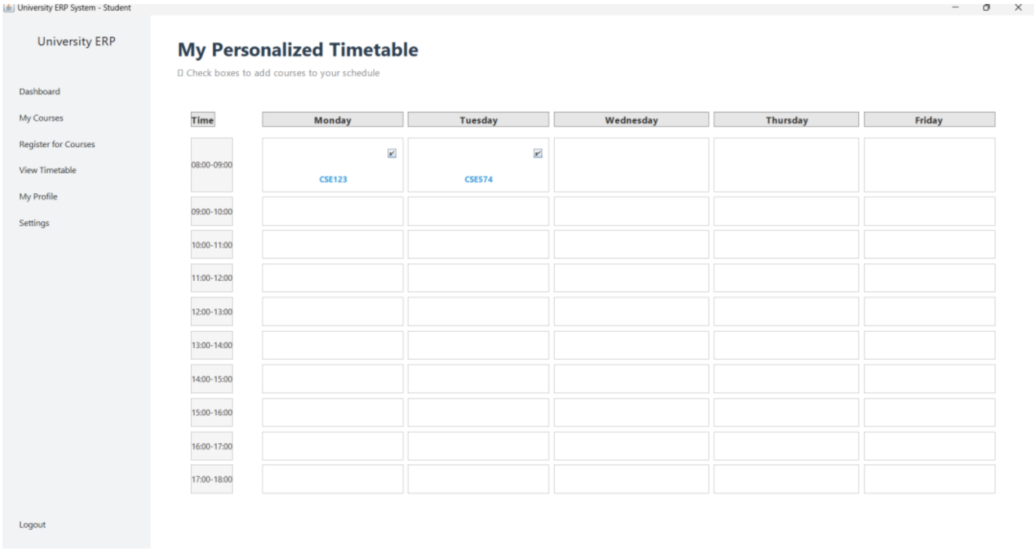
Current Credits: 16/20 | Credit Limit: 20

Available Sections

Code	Title	Credits	Instructor	Seats (Avail/T...	Type

Registered Courses

Code	Title	Credits	Instructor
SSHHCI	Human Computer Int...	4	Subhajit
MTH231	LA	4	N/A
ECE321	Digital Circuits	4	Tamam Tilo
ECE321	Digital Circuits	4	Pravesh Biryani



My Personalized Timetable

□ Check boxes to add courses to your schedule

Time	Monday	Tuesday	Wednesday	Thursday	Friday
08:00-09:00	☑ CSE123	☑ CSE574			
09:00-10:00					
10:00-11:00					
11:00-12:00					
12:00-13:00					
13:00-14:00					
14:00-15:00					
15:00-16:00					
16:00-17:00					
17:00-18:00					

7. Key Bug Fixes Implemented

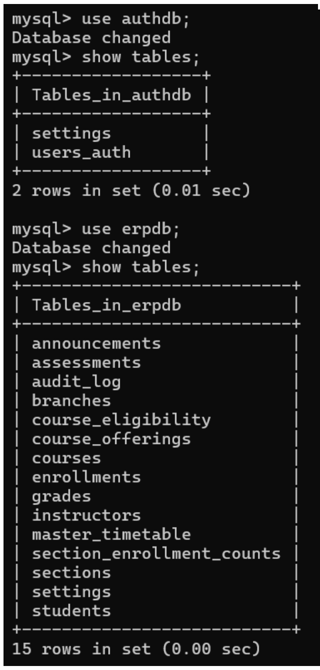
NullPointerException when section.getInstructorId() is null - Added null checks

CSV import "Student Not Found" - Clarified format, improved validation messages

Duplicate course enrollment - Added isStudentEnrolledInCourse() check

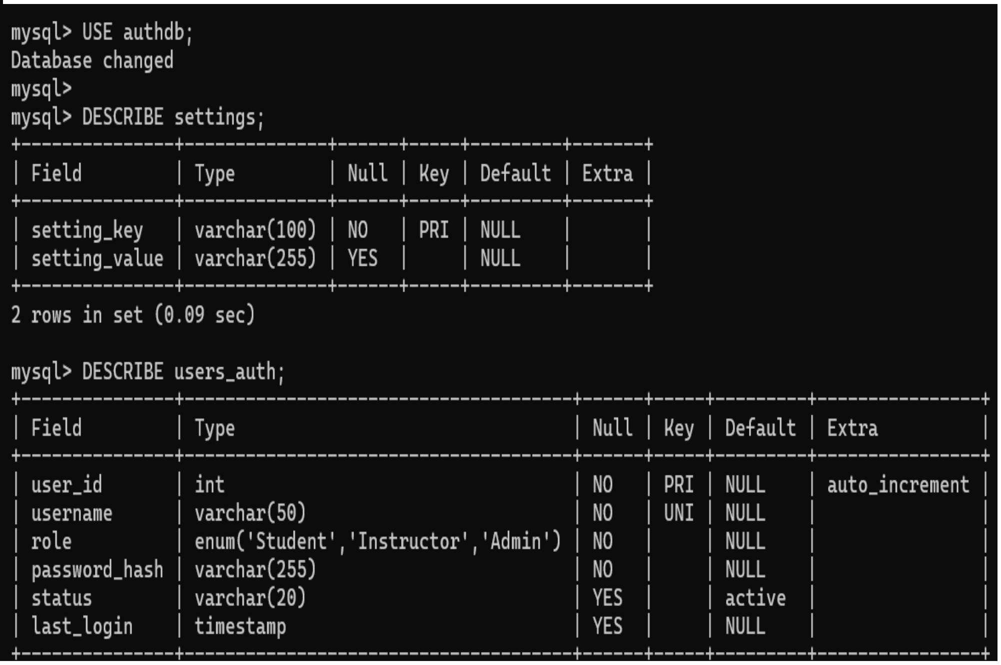
Maintenance mode not blocking all actions - Added check to every write operation

8. Databases



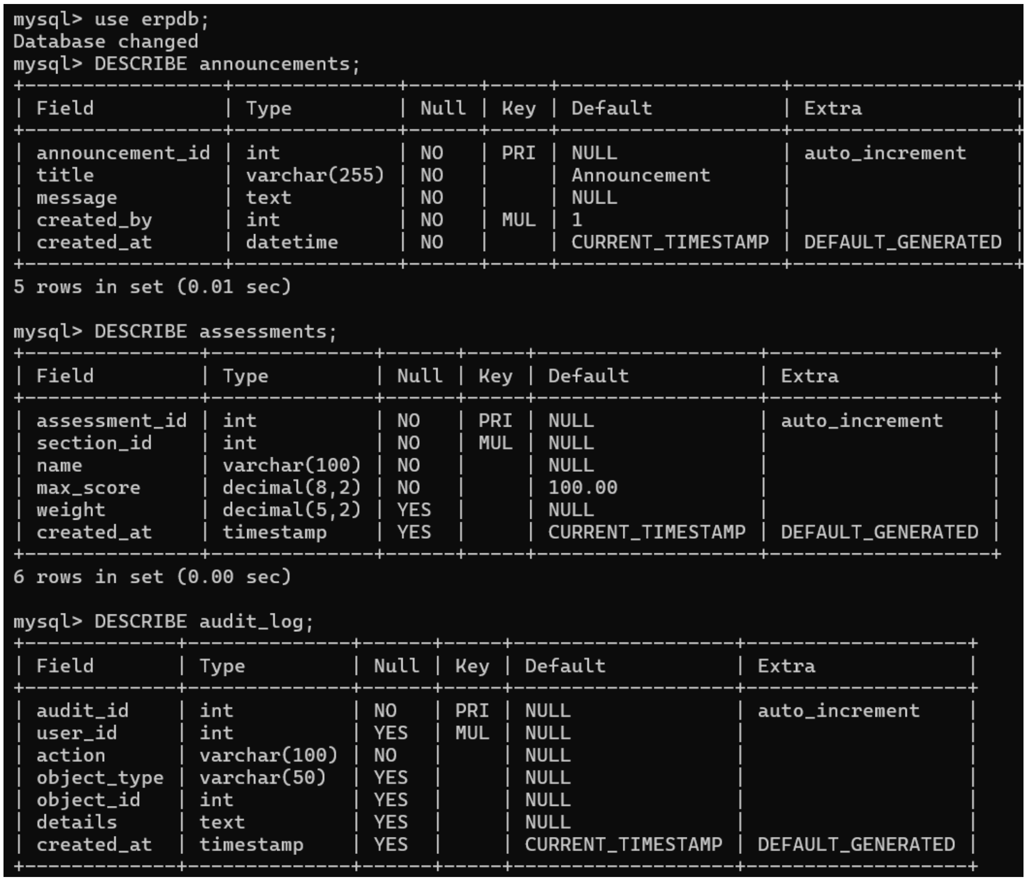
```
mysql> use authdb;
Database changed
mysql> show tables;
+------------------+
| Tables_in_authdb |
+------------------+
| settings         |
| users_auth       |
+------------------+
2 rows in set (0.01 sec)

mysql> use erpdb;
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_erpdb         |
+-------------------------+
| announcements           |
| assessments             |
| audit_log               |
| branches                |
| course_eligibility      |
| course_offerings        |
| courses                 |
| enrollments             |
| grades                  |
| instructors             |
| master_timetable        |
| section_enrollment_counts |
| sections                |
| settings                |
| students                |
+-------------------------+
15 rows in set (0.00 sec)
```



```
mysql> USE authdb;
Database changed
mysql>
mysql> DESCRIBE settings;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| setting_key   | varchar(100) | NO   | PRI | NULL    |       |
| setting_value | varchar(255) | YES  |     | NULL    |       |
+---------------+--------------+------+-----+---------+-------+
2 rows in set (0.09 sec)

mysql> DESCRIBE users_auth;
+---------------+--------------------------------------+------+-----+---------+----------------+
| Field         | Type                                 | Null | Key | Default | Extra          |
+---------------+--------------------------------------+------+-----+---------+----------------+
| user_id       | int                                  | NO   | PRI | NULL    | auto_increment |
| username      | varchar(50)                          | NO   | UNI | NULL    |                |
| role          | enum('Student','Instructor','Admin') | NO   |     | NULL    |                |
| password_hash | varchar(255)                         | NO   |     | NULL    |                |
| status        | varchar(20)                          | YES  |     | active  |                |
| last_login    | timestamp                            | YES  |     | NULL    |                |
+---------------+--------------------------------------+------+-----+---------+----------------+
```



```
mysql> use erpdb;
Database changed
mysql> DESCRIBE announcements;
+-----------------+--------------+------+-----+-------------------+-------------------+
| Field           | Type         | Null | Key | Default           | Extra             |
+-----------------+--------------+------+-----+-------------------+-------------------+
| announcement_id | int          | NO   | PRI | NULL              | auto_increment    |
| title           | varchar(255) | NO   |     | Announcement      |                   |
| message         | text         | NO   |     | NULL              |                   |
| created_by      | int          | NO   | MUL | 1                 |                   |
| created_at      | datetime     | NO   |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+--------------+------+-----+-------------------+-------------------+
5 rows in set (0.01 sec)

mysql> DESCRIBE assessments;
+---------------+--------------+------+-----+-------------------+-------------------+
| Field         | Type         | Null | Key | Default           | Extra             |
+---------------+--------------+------+-----+-------------------+-------------------+
| assessment_id | int          | NO   | PRI | NULL              | auto_increment    |
| section_id    | int          | NO   | MUL | NULL              |                   |
| name          | varchar(100) | NO   |     | NULL              |                   |
| max_score     | decimal(8,2) | NO   |     | 100.00            |                   |
| weight        | decimal(5,2) | YES  |     | NULL              |                   |
| created_at    | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+---------------+--------------+------+-----+-------------------+-------------------+
6 rows in set (0.00 sec)

mysql> DESCRIBE audit_log;
+-------------+--------------+------+-----+-------------------+-------------------+
| Field       | Type         | Null | Key | Default           | Extra             |
+-------------+--------------+------+-----+-------------------+-------------------+
| audit_id    | int          | NO   | PRI | NULL              | auto_increment    |
| user_id     | int          | YES  | MUL | NULL              |                   |
| action      | varchar(100) | NO   |     | NULL              |                   |
| object_type | varchar(50)  | YES  |     | NULL              |                   |
| object_id   | int          | YES  |     | NULL              |                   |
| details     | text         | YES  |     | NULL              |                   |
| created_at  | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------+--------------+------+-----+-------------------+-------------------+
```

8. Testing Summary

Sample Accounts:

Admin: "admin" / "password123"\nInstructor: "subhajit" / "password123"\nStudents: "kushav", "rahul" / "password123"

Tests Passed:

✓ Authentication (login, lockout, password hashing)
✓ Admin functions (create users/courses, maintenance toggle)
✓ Student registration (capacity, deadline, duplicate checks)
✓ Instructor grading (add assessments, enter grades, CSV import)
✓ Edge cases (null instructor, invalid CSV, weightage validation)

9. How to Run

Prerequisites

- Java: JDK 17 or later

- MySQL: Version 8.0 or higher (server must be running)

- Libraries: All required JARs are included in the lib/ folder

Step 1: Configure Database Password

1. Navigate to src/resources/database.properties

2. Update the MySQL password to match your system:

```
jdbc.password=YOUR_MYSQL_PASSWORD
```

3. Save the file

Note: Keep the username as root unless you use a different MySQL user.

Step 2: Compile and Run the Application

Open terminal/command prompt in the project directory and run these commands:

For Windows:

```
dir /s /b src\*.java > sources.txt
javac -cp "lib\*;." -d out @sources.txt
java -cp "out;lib\*" edu.univ.erp.auth.LoginFrame
```

For macOS/Linux:

```
find src -name "*.java" > sources.txt
javac -cp "lib/*:." -d out @sources.txt
java -cp "out:lib/*" edu.univ.erp.auth.LoginFrame
```

First Run Behavior

On first run, the application will automatically:

- Check if databases exist
- Create authdb and erpdb databases if missing
- Create all required tables
- Insert sample data (users, courses, sections, enrollments)
- Open the login screen

You should see console output confirming: *"Database initialization complete!"* and *"Auth DB connection: OK" / "ERP DB connection: OK"*

On subsequent runs, the app detects existing databases and skips initialization, directly opening the login screen.

Default Login Credentials

Role	**Username**	**Password**
Admin	admin	admin123
Student	ankit	ankit123
Instructor	sonal	sonal

Troubleshooting

"Access denied for user 'root'" - Your MySQL password in database.properties is incorrect. Update it and restart the application.

"DATABASE_SCHEMA.sql not found" - Ensure DATABASE_SCHEMA.sql is in src/resources/ folder. Without it, databases cannot be auto-created.

Login fails after first run - Check if users were created by running SELECT * FROM authdb.users_auth; in MySQL. If empty, the INSERT statements in DATABASE_SCHEMA.sql may have errors.

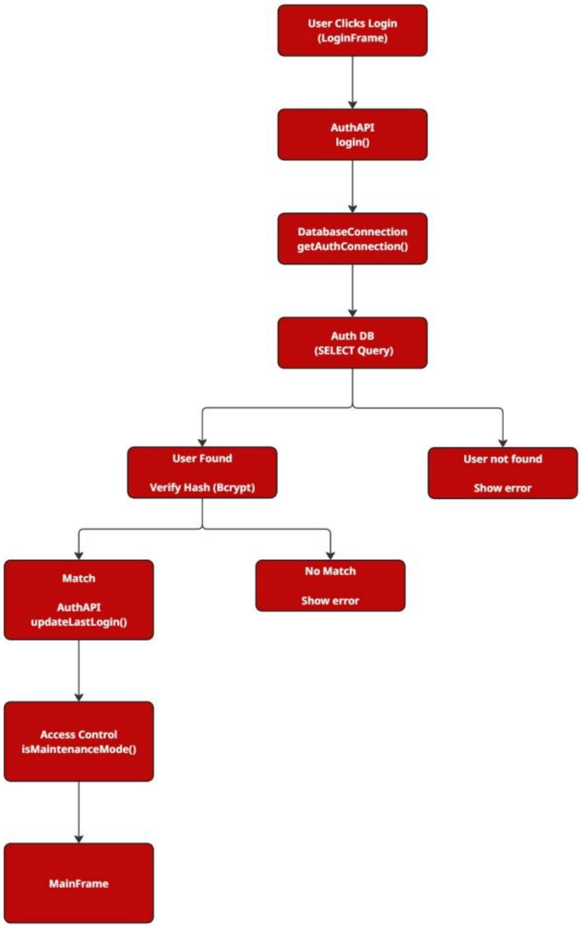
Manual Database Setup (If Auto-initialization Fails)

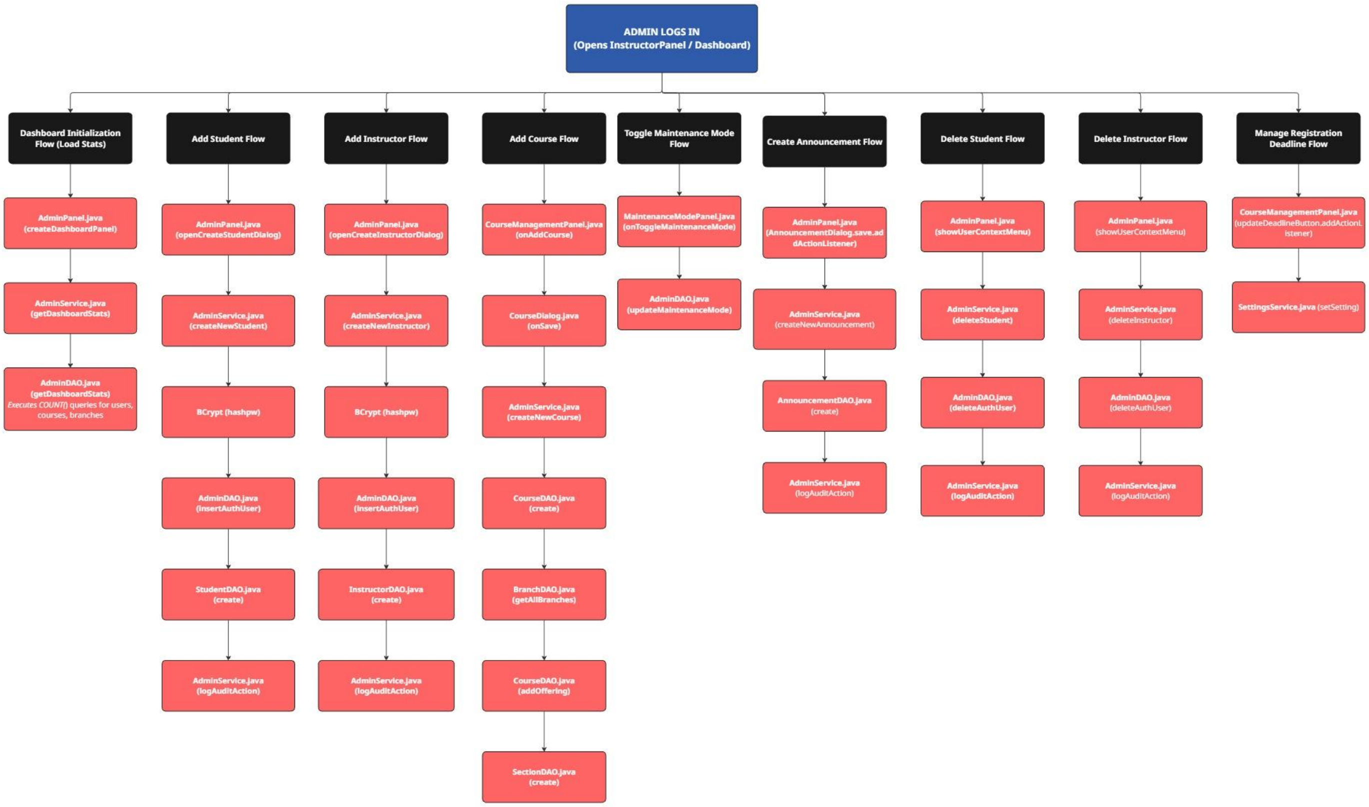
If automatic setup doesn't work, you can manually create databases by running:

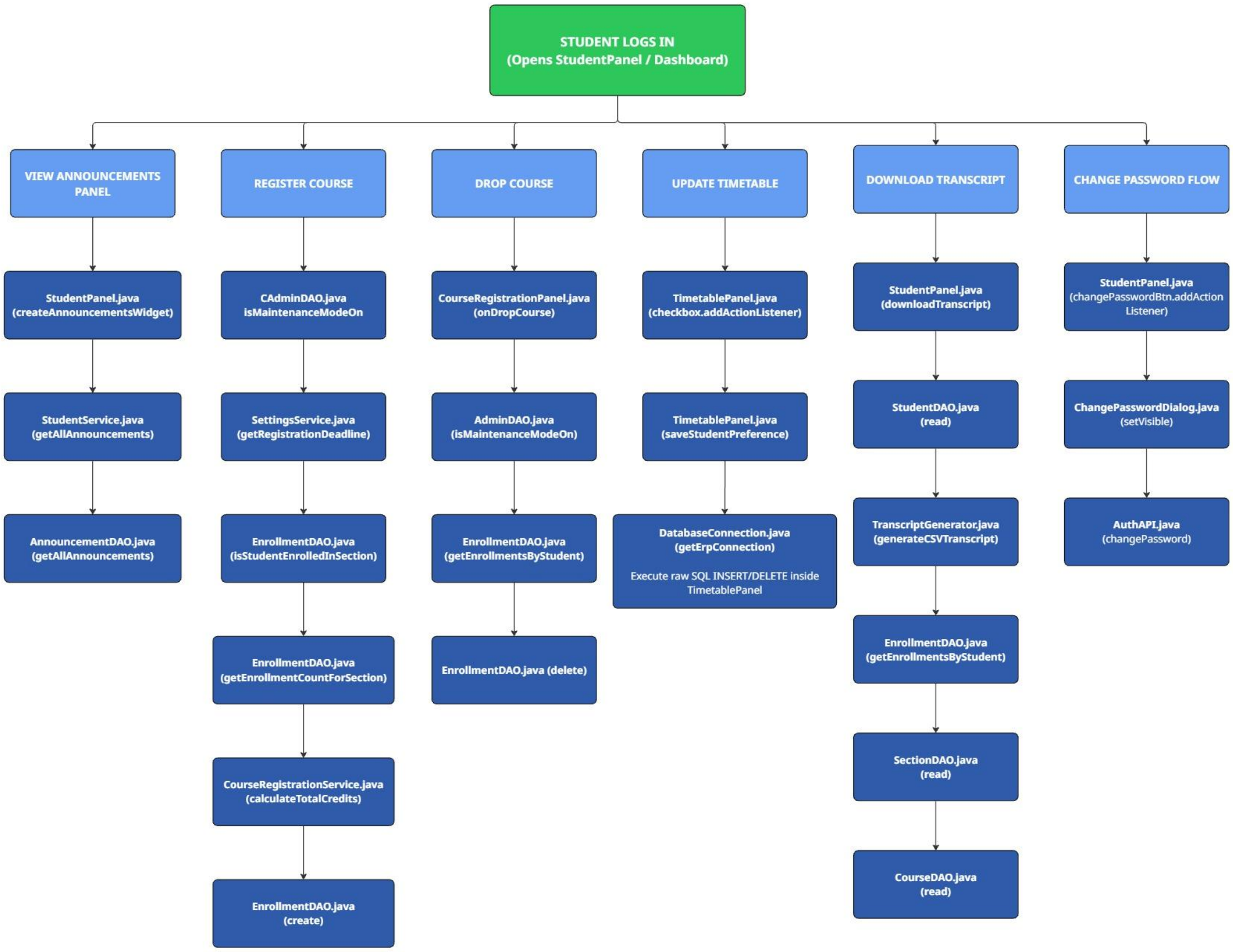
```
mysql -u root -p < src/resources/DATABASE_SCHEMA.sql
```

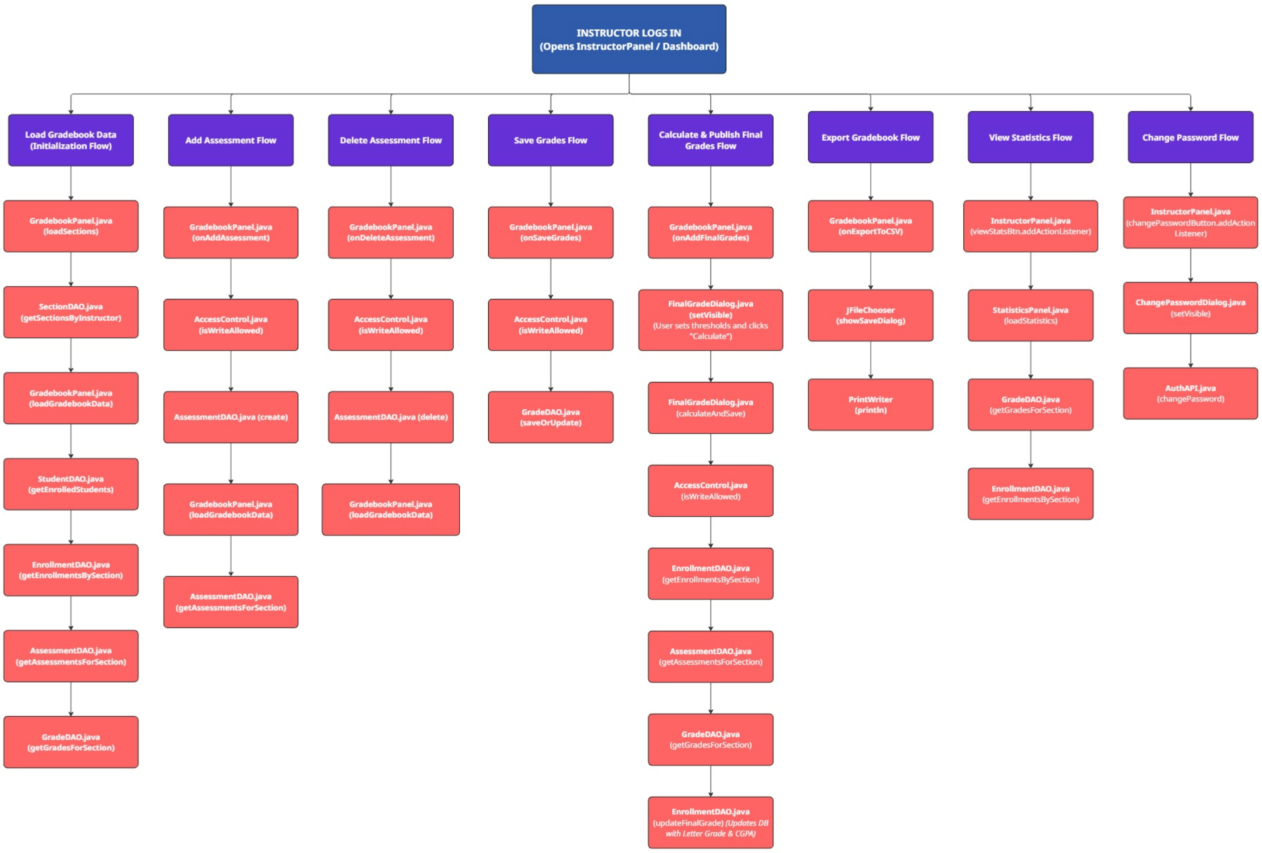
in your terminal, then run the application normally.

10. User Flow Diagrams


User Clicks Login (LoginFrame)
AuthAPI login()
DatabaseConnection getAuthConnection()
Auth DB (SELECT Query)
User Found
Verify Hash (Bcrypt)
User not found
Show error
Match
AuthAPI updateLastLogin()
No Match
Show error
Access Control isMaintenanceMode()
MainFrame


ADMIN LOGS IN
(Opens InstructorPanel / Dashboard)
Dashboard Initialization Flow (Load Stats)
AdminPanel.java (createDashboardPanel)
AdminService.java (getDashboardStats)
AdminDAO.java (getDashboardStats) Executes COUNT() queries for users, courses, branches
Add Student Flow
AdminPanel.java (openCreateStudentDialog)
AdminService.java (createNewStudent)
BCrypt (hashpw)
AdminDAO.java (insertAuthUser)
StudentDAO.java (create)
AdminService.java (logAuditAction)
Add Instructor Flow
AdminPanel.java (openCreateInstructorDialog)
AdminService.java (createNewInstructor)
BCrypt (hashpw)
AdminDAO.java (insertAuthUser)
InstructorDAO.java (create)
AdminService.java (logAuditAction)
Add Course Flow
CourseManagementPanel.java (onAddCourse)
CourseDialog.java (onSave)
AdminService.java (createNewCourse)
CourseDAO.java (create)
BranchDAO.java (getAllBranches)
CourseDAO.java (addOffering)
SectionDAO.java (create)
Toggle Maintenance Mode Flow
MaintenanceModePanel.java (onToggleMaintenanceMode)
AdminDAO.java (updateMaintenanceMode)
Create Announcement Flow
AdminPanel.java (AnnouncementDialog.save.addActionListener)
AdminService.java (createNewAnnouncement)
AnnouncementDAO.java (create)
AdminService.java (logAuditAction)
Delete Student Flow
AdminPanel.java (showUserContextMenu)
AdminService.java (deleteStudent)
AdminDAO.java (deleteAuthUser)
AdminService.java (logAuditAction)
Delete Instructor Flow
AdminPanel.java (showUserContextMenu)
AdminService.java (deleteInstructor)
AdminDAO.java (deleteAuthUser)
AdminService.java (logAuditAction)
Manage Registration Deadline Flow
CourseManagementPanel.java (updateDeadlineButton.addActionListener)
SettingsService.java (setSetting)


STUDENT LOGS IN
(Opens StudentPanel / Dashboard)
VIEW ANNOUNCEMENTS PANEL
StudentPanel.java (createAnnouncementsWidget)
StudentService.java (getAllAnnouncements)
AnnouncementDAO.java (getAllAnnouncements)
REGISTER COURSE
CAdminDAO.java isMaintenanceModeOn
SettingsService.java (getRegistrationDeadline)
EnrollmentDAO.java (isStudentEnrolledInSection)
EnrollmentDAO.java (getEnrollmentCountForSection)
CourseRegistrationService.java (calculateTotalCredits)
EnrollmentDAO.java (create)
DROP COURSE
CourseRegistrationPanel.java (onDropCourse)
AdminDAO.java (isMaintenanceModeOn)
EnrollmentDAO.java (getEnrollmentsByStudent)
EnrollmentDAO.java (delete)
UPDATE TIMETABLE
TimetablePanel.java (checkbox.addActionListener)
TimetablePanel.java (saveStudentPreference)
DatabaseConnection.java (getErpConnection)
Execute raw SQL INSERT/DELETE inside TimetablePanel
DOWNLOAD TRANSCRIPT
StudentPanel.java (downloadTranscript)
StudentDAO.java (read)
TranscriptGenerator.java (generateCSVTranscript)
EnrollmentDAO.java (getEnrollmentsByStudent)
SectionDAO.java (read)
CourseDAO.java (read)
CHANGE PASSWORD FLOW
StudentPanel.java (changePasswordBtn.addAction Listener)
ChangePasswordDialog.java (setVisible)
AuthAPI.java (changePassword)


INSTRUCTOR LOGS IN
(Opens InstructorPanel / Dashboard)
Load Gradebook Data (Initialization Flow)
GradebookPanel.java (loadSections)
SectionDAO.java (getSectionsByInstructor)
GradebookPanel.java (loadGradebookData)
StudentDAO.java (getEnrolledStudents)
EnrollmentDAO.java (getEnrollmentsBySection)
AssessmentDAO.java (getAssessmentsForSection)
GradeDAO.java (getGradesForSection)
Add Assessment Flow
GradebookPanel.java (onAddAssessment)
AccessControl.java (isWriteAllowed)
AssessmentDAO.java (create)
GradebookPanel.java (loadGradebookData)
AssessmentDAO.java (getAssessmentsForSection)
Delete Assessment Flow
GradebookPanel.java (onDeleteAssessment)
AccessControl.java (isWriteAllowed)
AssessmentDAO.java (delete)
GradebookPanel.java (loadGradebookData)
Save Grades Flow
GradebookPanel.java (onSaveGrades)
AccessControl.java (isWriteAllowed)
GradeDAO.java (saveOrUpdate)
Calculate & Publish Final Grades Flow
GradebookPanel.java (onAddFinalGrades)
FinalGradeDialog.java (setVisible) (User sets thresholds and clicks "Calculate")
FinalGradeDialog.java (calculateAndSave)
AccessControl.java (isWriteAllowed)
EnrollmentDAO.java (getEnrollmentsBySection)
AssessmentDAO.java (getAssessmentsForSection)
GradeDAO.java (getGradesForSection)
EnrollmentDAO.java (updateFinalGrade) (Updates DB with Letter Grade & CGPA)
Export Gradebook Flow
GradebookPanel.java (onExportToCSV)
JFileChooser (showSaveDialog)
PrintWriter (println)
View Statistics Flow
InstructorPanel.java (viewStatsBtn.addActionListener)
StatisticsPanel.java (loadStatistics)
GradeDAO.java (getGradesForSection)
EnrollmentDAO.java (getEnrollmentsBySection)
Change Password Flow
InstructorPanel.java (changePasswordButton.addActionListener)
ChangePasswordDialog.java (setVisible)
AuthAPI.java (changePassword)

11. Conclusion

This University ERP system successfully implements comprehensive course management with secure authentication, role-based access control, and robust validation. Key achievements include BCrypt password security, duplicate course prevention, CSV grade import, maintenance mode, and cascade deletion.

The modular architecture (UI → Service → DAO) ensures maintainability and allows easy future enhancements such as prerequisite checking, GPA calculation, and waiting lists.

Project Team:

Kushav Nipul Patalia

Ankit Agarwal

Course: Advanced Programming