University Management System – Software Design Specification Document

Project Objective:

The University Management System (UMS) is a desktop-based application built using Java Swing, offering role-based access for Admins, Students, and Teachers. It supports core academic and administrative tasks. A unique feature is the integration of an AI assistant named Spark AI, available upon user login for help and queries.

A. Requirements Specification

1. User Stories in JIRA (Features & Epics)

Epic 1: Admin Features

- **Story 1:** Add/Remove Courses
- **Story 2:** View Time Table (Section Wise, Day Wise, Room Wise, Teacher Wise)
- **Story 3:** Add/Remove Students
- **Story 4:** Add/Remove Rooms
- **Story 5:** Add Departments
- **Story 6:** Search Student Availability
- **Story 7:** Generate and Track Challans

Epic 2: Student Features

- **Story 1:** Check Time Table
- **Story 2:** View Announcements
- **Story 3:** Register Courses
- **Story 4:** Check Scores
- **Story 5:** View Attendance
- **Story 6:** Submit/View Assignments
- **Story 7:** View Challans

Epic 3: Teacher Features

- **Story 1:** Mark Attendance

- **Story 2:** View Time Table

- **Story 3:** Make Announcements

- **Story 4:** Enter Student Marks

- **Story 5:** View Student Profiles

- **Story 6:** Upload Assignments

- **Story 7:** View Submissions

Epic 4: Spark AI Integration

- **Story 1:** Login/Signup for Spark AI

- **Story 2:** Access Help from Spark AI

Non-Functional Requirements:

- Secure, role-based login.

- All operations should complete under 3 seconds.

- The system should support at least 100 concurrent users.

- Should maintain 99.9% uptime and data persistence.

GITHUB LINK : https://github.com/aizazahmed001/SDA-PROJECT

B. Design Specification

1. System Architecture and Design Patterns

Architecture Style: Model-View-Controller (MVC)
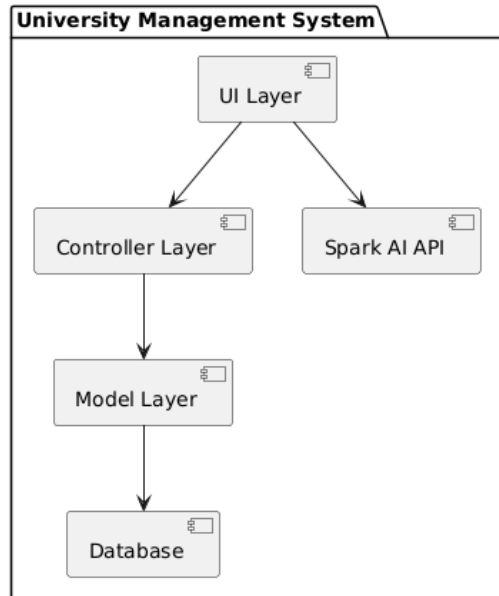
Justification:

- Promotes separation of concerns.

- Enables easy modifications in UI or logic.

- Scalable and testable.

Design Patterns Used:

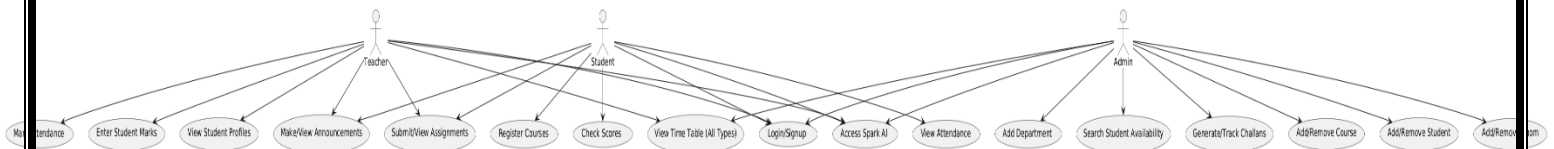- **Singleton:** For database connection management.

- **Factory:** For user-role object creation.

- **Observer:** For announcements and notifications.

Architecture Diagram (PlantUML)

University Management System

UI Layer

Controller Layer   Spark AI API

Model Layer

Database

---

2. Detailed Design

A. Use Case Diagram:

Teacher   Student   Admin

Make Attendance   Enter Student Marks   View Student Profiles   Make/View Announcements   Submit/View Assigments   Register Courses   Check Scores   View Time Table (All Types)   Login/Signup   Access Spark AI   View Attendance   Add Department   Search Student Availability   Generate/Track Challans   Add/Remove Course   Add/Remove Student   Add/Remove Room
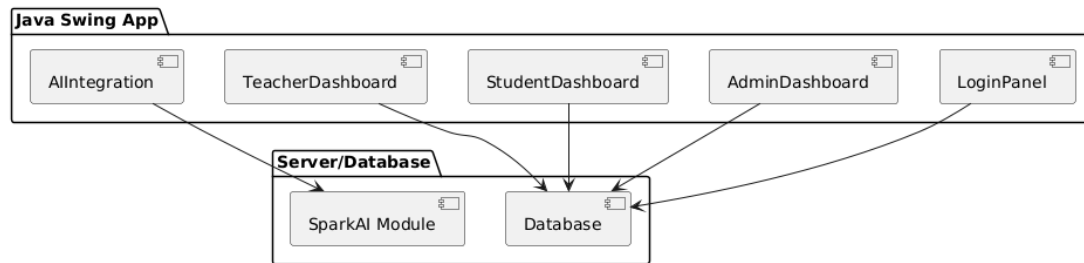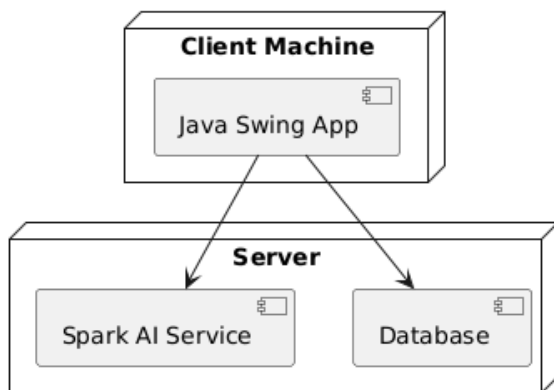
---

b. UI Design

- **Admin Dashboard:** Tabbed panes for student/course/room/dept management, time tables, challan module.

- **Student Dashboard:** Assignment viewer, course registration, scores, attendance.

- **Teacher Dashboard:** Assignment uploader, attendance sheet, mark entry, student viewer.

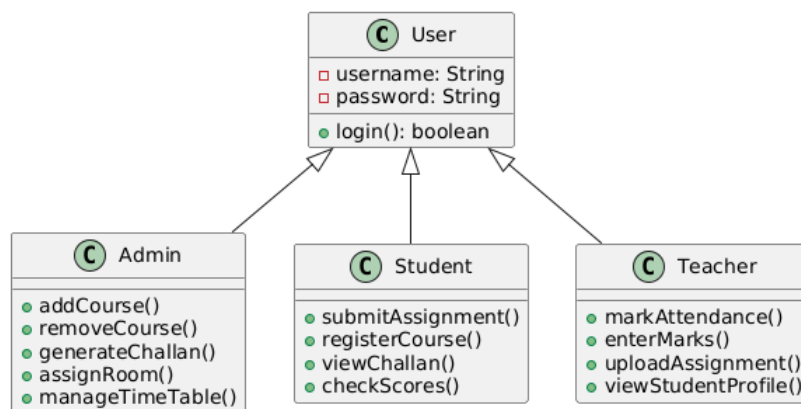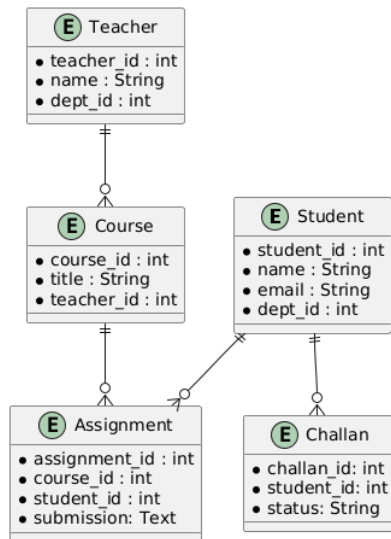- Common Login Form, Spark AI Chat Interface

c. Component Diagram

Java Swing App

| AIIntegration | TeacherDashboard | StudentDashboard | AdminDashboard | LoginPanel |

Server/Database

| SparkAI Module | Database |

d. Deployment Diagram

Client Machine

Java Swing App

Server

Spark AI Service   Database

e. Class Diagram

C User
- username: String
- password: String
- login(): boolean

C Admin
- addCourse()
- removeCourse()
- generateChallan()
- assignRoom()
- manageTimeTable()

C Student
- submitAssignment()
- registerCourse()
- viewChallan()
- checkScores()

C Teacher
- markAttendance()
- enterMarks()
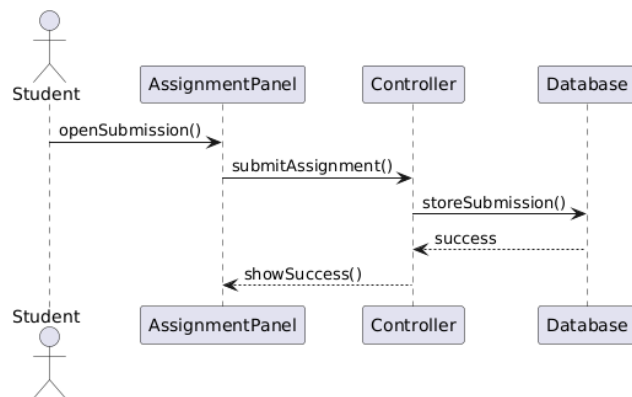- uploadAssignment()
- viewStudentProfile()
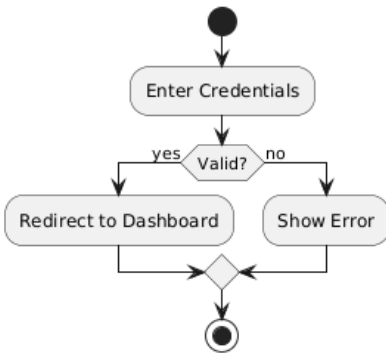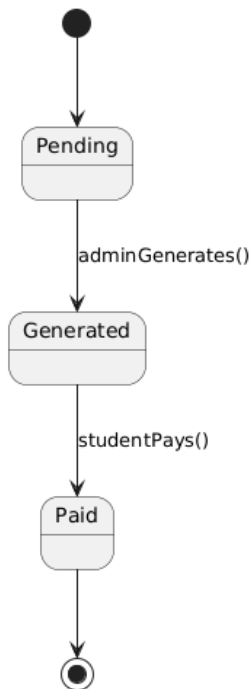
f. Data Model

g. Dynamic Views

1. Sequence Diagram - Submit Assignment:

2. Activity Diagram - Login Flow:

3. State Diagram - Challan Status

Constraints Handled

1. **Authentication:** Only authenticated users can access features.

2. **Authorization:** Role-based access controls actions.

3. **Separation of Concerns:** MVC and design patterns enforce modularity.

4. **Data Access:** All data via centralized system.

5. **Extensibility:** Easily extendable features and components.