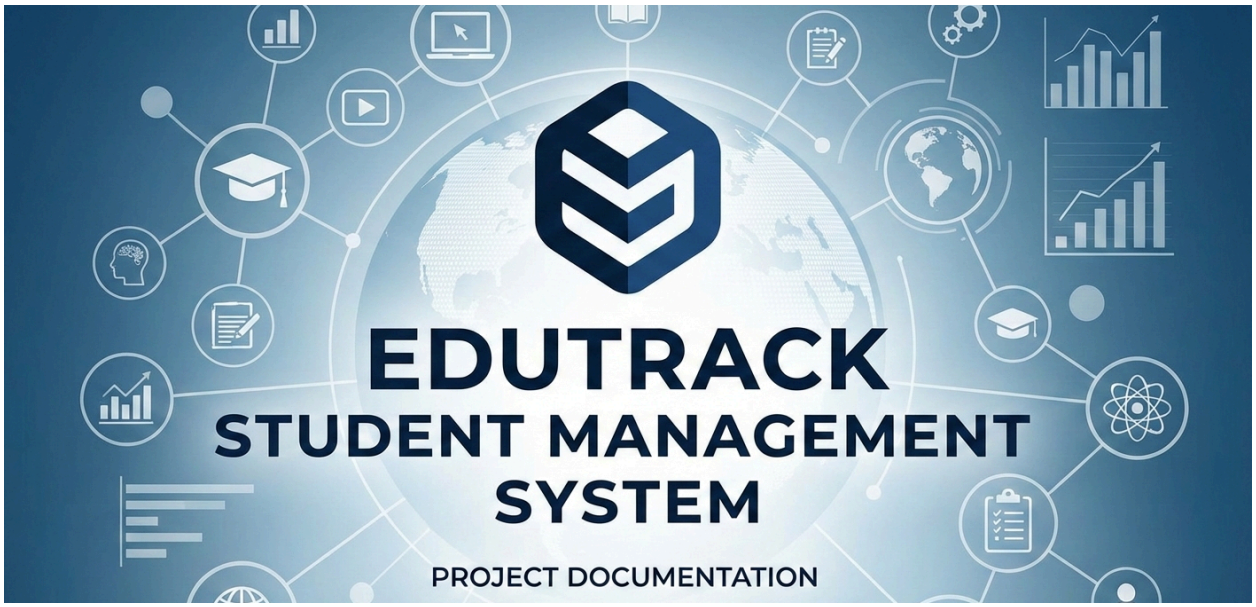
Case Study on

EduTrack Student Management System



1. INTRODUCTION

The **EduTrack Student Management System** is a sophisticated digital ecosystem designed to transform how educational institutions manage their daily academic and administrative operations. In an era where data-driven decisions are paramount, EduTrack serves as a single source of truth for student records, faculty management, and performance analytics.

The system addresses the fundamental flaws of manual academic tracking:

- **Data Redundancy:** Traditional paper-based systems often duplicate data across departments.
- **Lack of Real-time Insight:** Manual records make it nearly impossible to get an instant view of institution-wide attendance or performance.
- **Security Risks:** Physical records are prone to damage, loss, and unauthorized access.

EduTrack provides a role-based, secure, and transparent interface for three primary stakeholders:

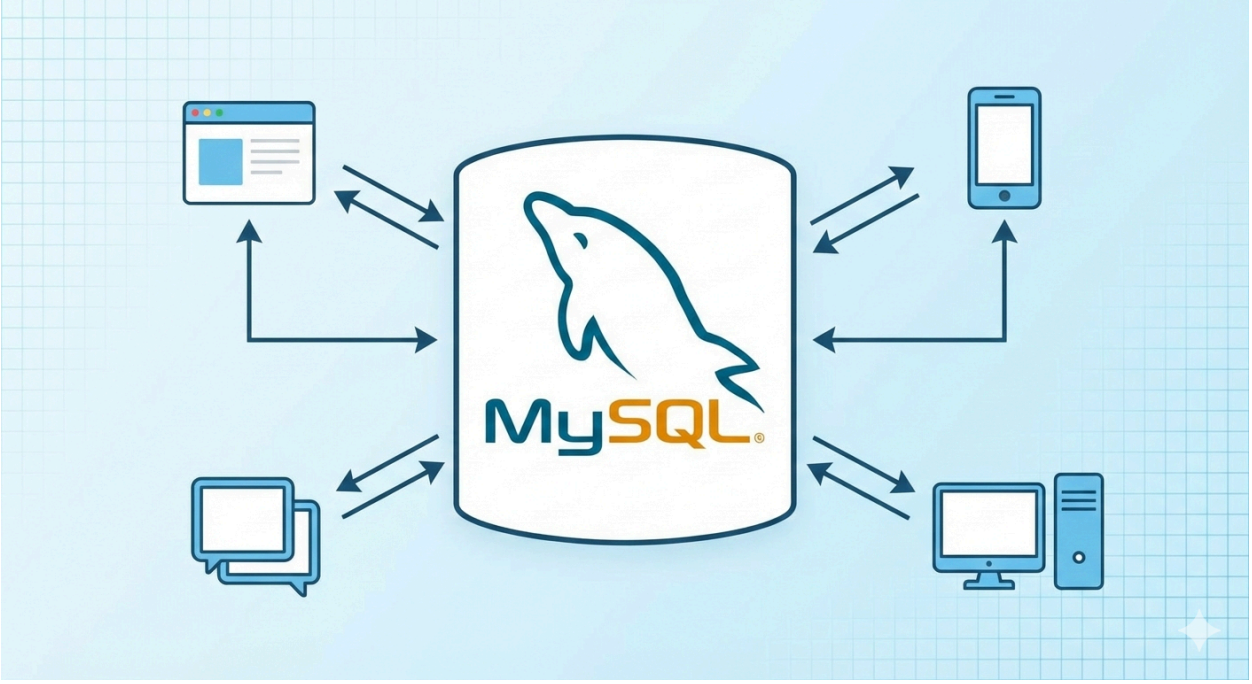
- **Administrators:** To maintain high-level oversight of institutional data.
- **Teachers (Faculty):** To automate the tedious tasks of attendance marking and grading.
- **Students:** To engage with their academic progress through personalized dashboards.

By integrating a Spring Boot backend with a responsive web frontend, EduTrack ensures that institutional management is efficient, accurate, and scalable for future growth

2. ABSTRACT

The **EduTrack Student Management System** is a comprehensive academic administration application developed to replace outdated manual attendance and record systems with a reliable digital platform. The primary objective is to streamline the student lifecycle—from enrollment to graduation—by automating key tasks such as schedule management, attendance tracking, and academic reporting.

The system is architected using a **Client-Server model**, utilizing **Spring Boot** for robust backend services and **MySQL** for persistent relational data storage. Security is a cornerstone of the application, implemented through role-based access control (RBAC) to ensure that sensitive data like student grades and faculty schedules are protected.



Key Features include:

- **Real-time Attendance:** Automatic calculation of attendance percentages.
- **Role-Based Portals:** Dedicated views for Admin, Faculty, and Students.
- **API-First Design:** Leveraging RESTful architecture for seamless data exchange.
- **Quality Assurance:** Extensive unit testing using JUnit 5 and API verification via Postman.

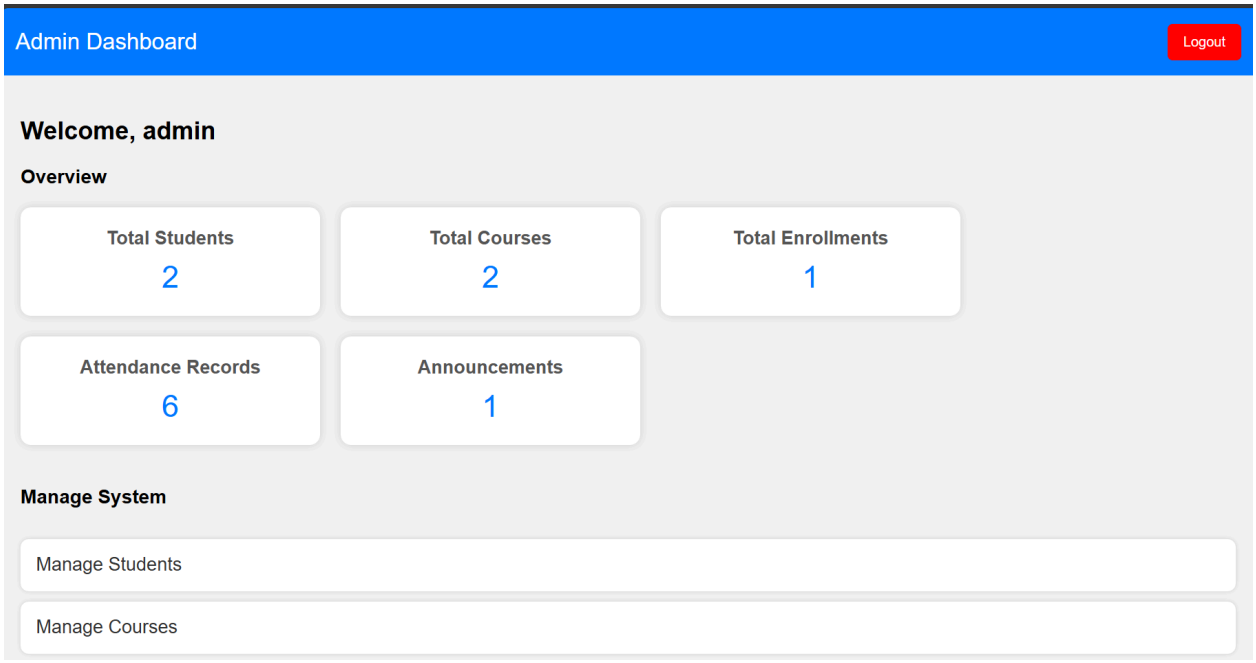
This project serves as a practical demonstration of modern software engineering principles, including Layered Architecture, Database Normalization, and Secure API Development.

3. CLIENT REQUIREMENTS

The client requires a robust and scalable educational management system that centralizes academic data while maintaining the highest standards of security and performance.

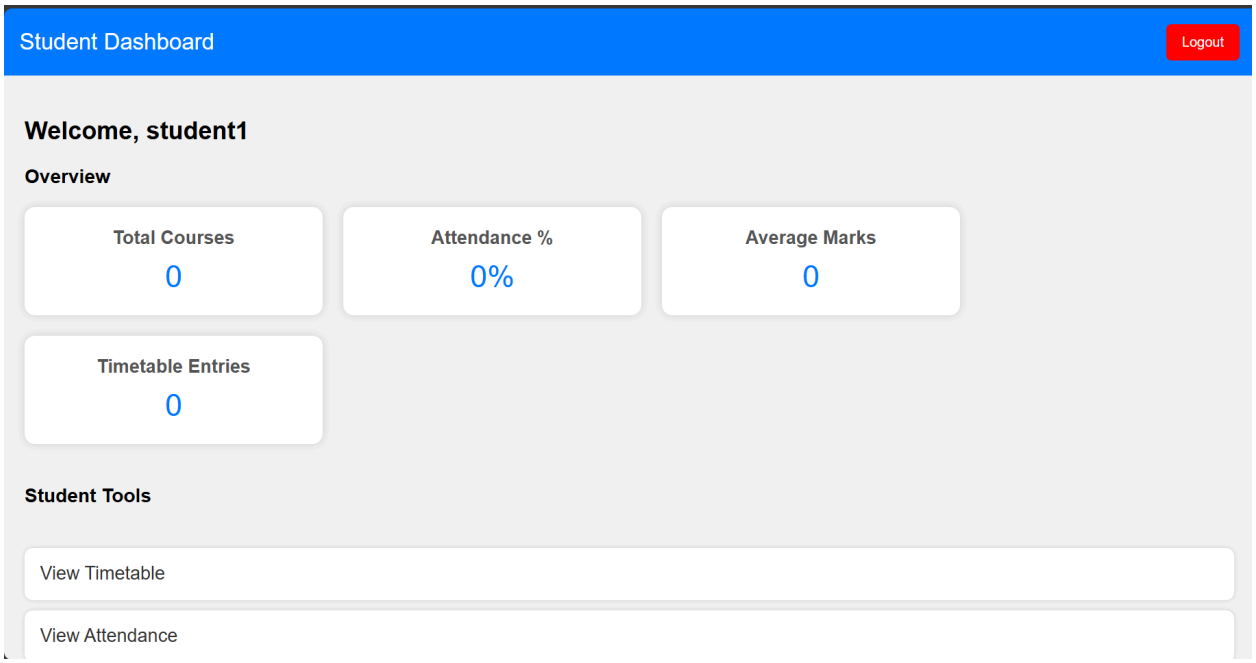
3.1 Functional Requirements

- **Administrator Portal:** Ability to perform CRUD (Create, Read, Update, Delete) operations on student, faculty, and course records.



- **Faculty Portal:** Interface to mark daily attendance, input academic marks, and review class performance.

- **Student Portal:** Access to view personal attendance trends, exam results, and course timetables.



- **Automation:** The system must automatically calculate cumulative attendance and average marks.
- **Announcement System:** A way for administrators to broadcast important notices across the platform.

3.2 Non-Functional Requirements

- **High Performance:** API response times must stay under 200ms for standard operations.
- **Security:** Use of secure login mechanisms and session management (JWT suggested).
- **Usability:** A clean, intuitive interface that requires minimal training for users.
- **Scalability:** The architecture must allow for the addition of new modules (e.g., fee management) in the future.
- **Reliability:** Data integrity must be maintained through relational constraints and transaction management.

4. WHAT WE ARE GOING TO BUILD

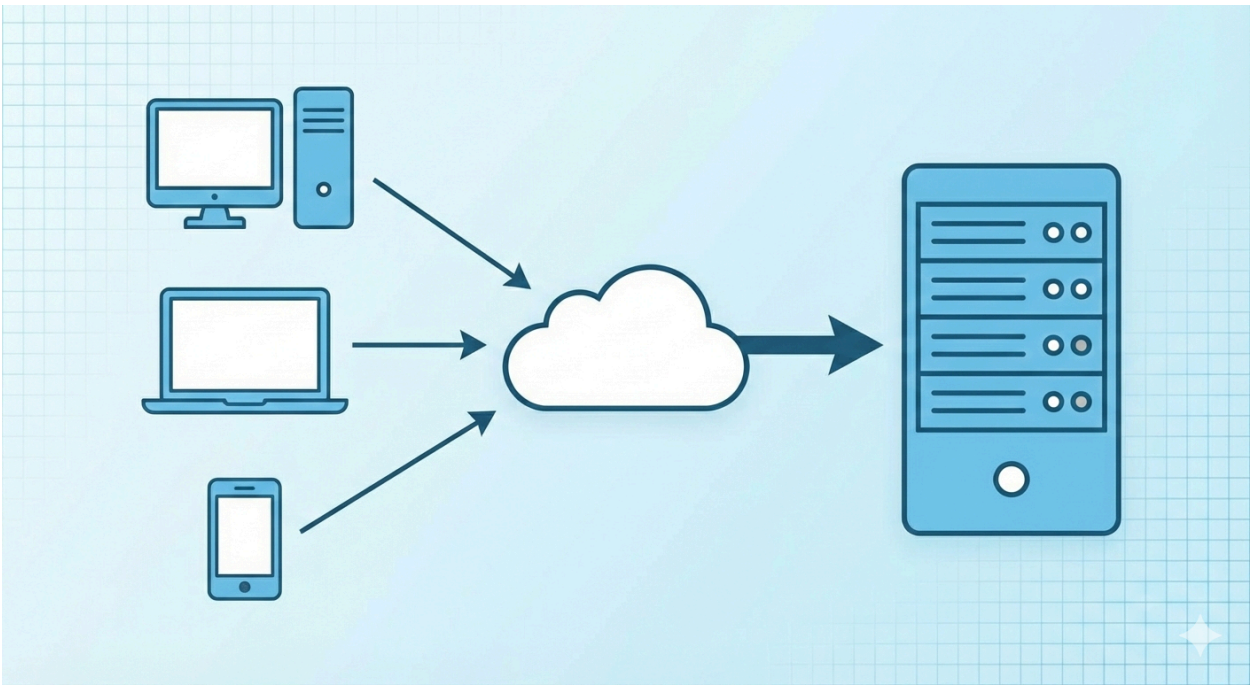
EduTrack is a modular application where each component is designed to handle a specific domain of institutional management while communicating through a standardized API layer.

4.1 System Components

- **Authentication Engine:** Secure login/logout functionality based on user roles.
- **Management Hub:** Core modules for Students, Courses, and Timetables.
- **Interaction Layer:** Modules for marking attendance and broadcasting announcements.
- **Dashboard Engine:** Real-time data visualization of institutional KPIs (Key Performance Indicators).

4.2 Key Features

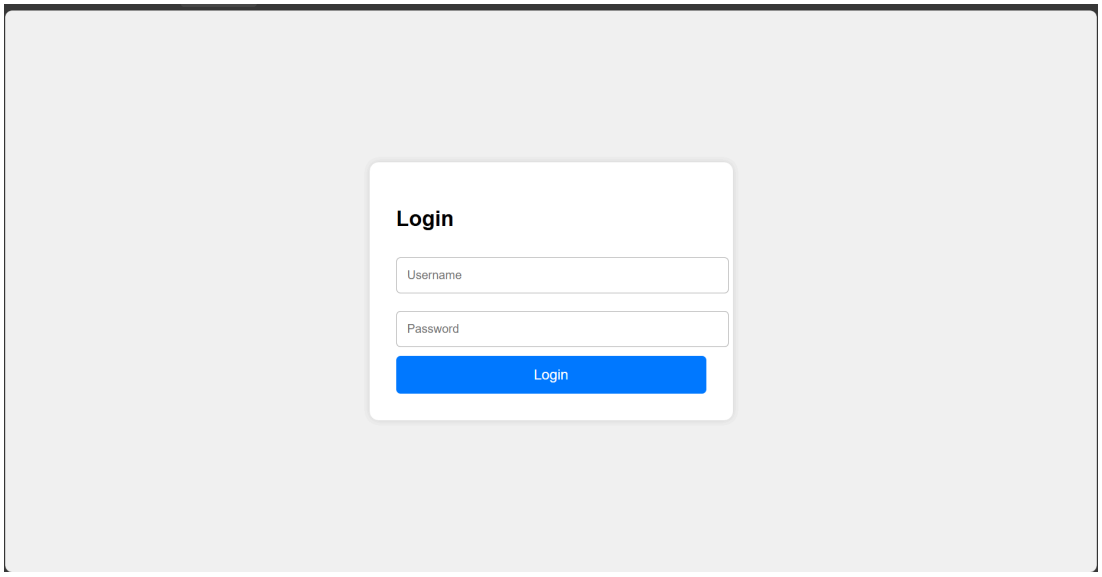
- **Centralized Database:** A single MySQL instance for all academic records.
- **Automated Scheduling:** Conflict-free timetable management.
- **Performance Analytics:** Calculating and displaying student averages and attendance percentages on dashboards.



5. TECHNICAL FEATURES

To meet the high standards of modern software, EduTrack incorporates several advanced technical features:

- **Role-Based Authentication:** Ensuring that a Student cannot access Admin tools.
- **RESTful Architecture:** Using standard HTTP methods (GET, POST, PUT, DELETE) for predictable API behavior.
- **Input Validation:** Strict server-side validation using Spring Validation annotations to ensure data quality.
- **Global Exception Handling:** A centralized mechanism to catch errors and return user-friendly messages instead of raw stack traces.
- **Responsive Web UI:** Using Tailwind-inspired styling to ensure the dashboard works across different screen sizes.
- **CORS Management:** Configuring Cross-Origin Resource Sharing to allow the frontend to safely communicate with the backend.



6. TECHNOLOGIES AND TOOLS USED

The project utilizes a modern tech stack chosen for its reliability, community support, and performance.

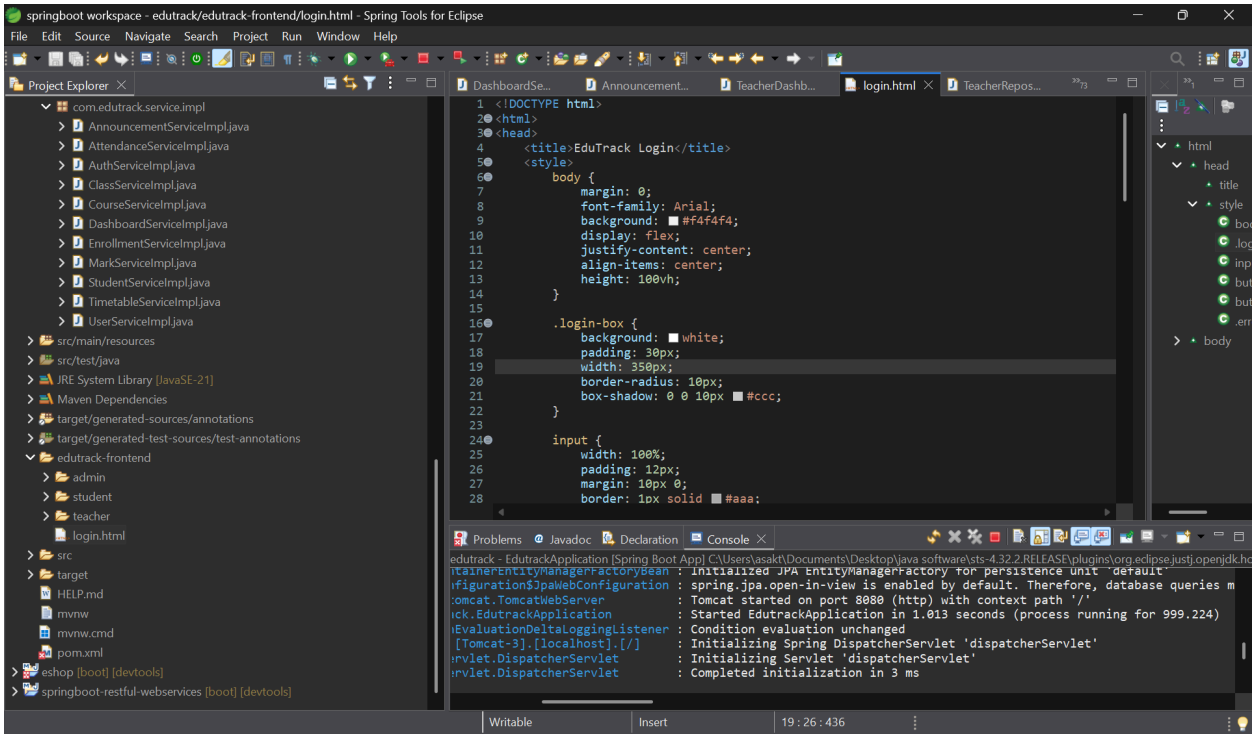
Tool / Technology	Purpose
Java 8+	Primary programming language for backend logic.
Spring Boot	Framework for creating standalone, production-grade applications.
Spring Data JPA	For simplified database interactions and Object-Relational Mapping.
Hibernate	The underlying engine for JPA to handle database schemas.
MySQL	Relational database for storing user, course, and attendance data.
Spring Security	Framework for securing the application and managing user roles.
Maven	Build tool for dependency management and project lifecycle.
Spring Tool Suite (STS)	Optimized IDE for Spring application development.
Postman	Comprehensive platform for API development and testing.
Swagger / OpenAPI	For generating interactive API documentation.

7. SYSTEM REQUIREMENTS

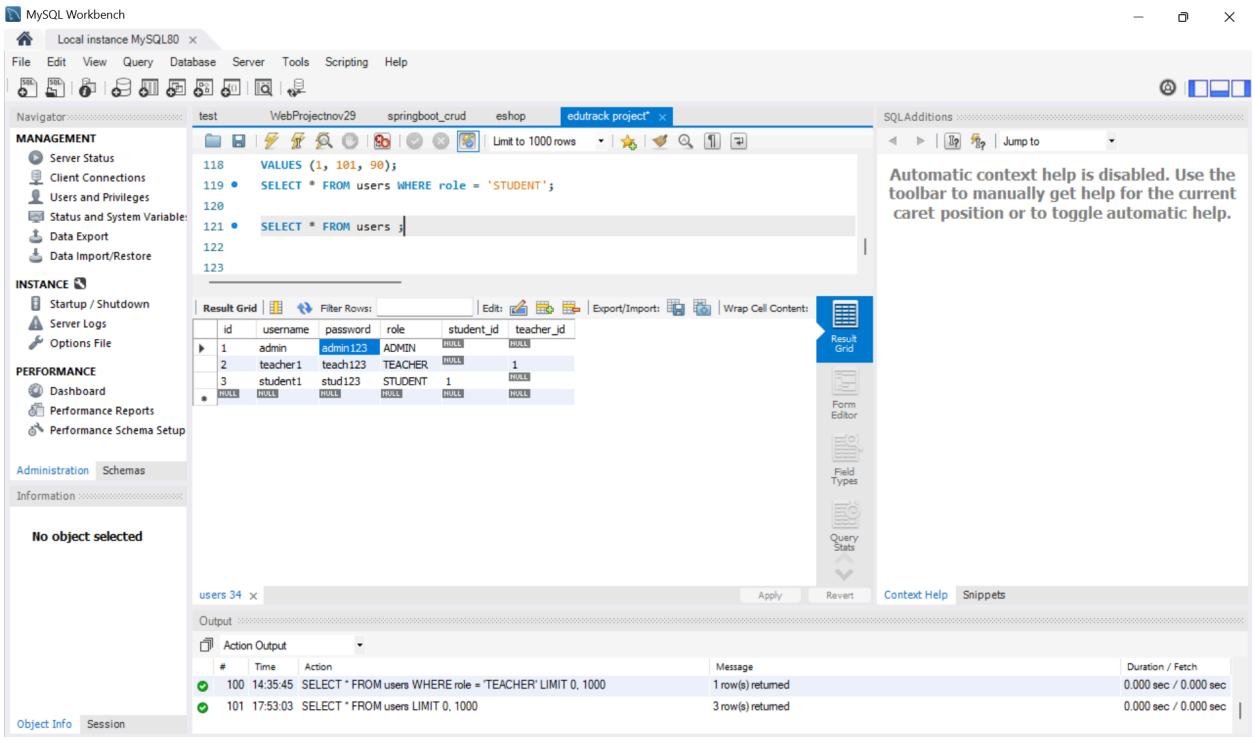
7.1 Software Requirements

- **Operating System:** Windows 10/11, macOS, or Linux.
- **Java Development Kit:** JDK 11 or higher.

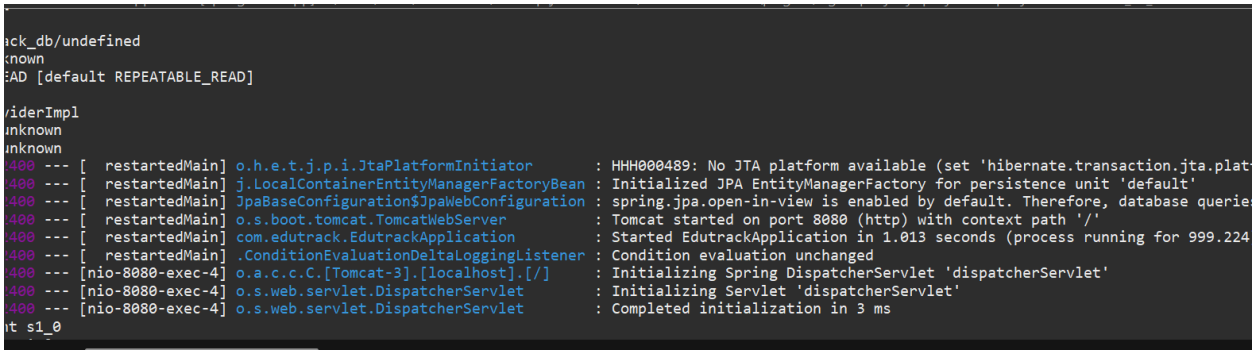
- **Build Tool:** Apache Maven 3.8+., springboot



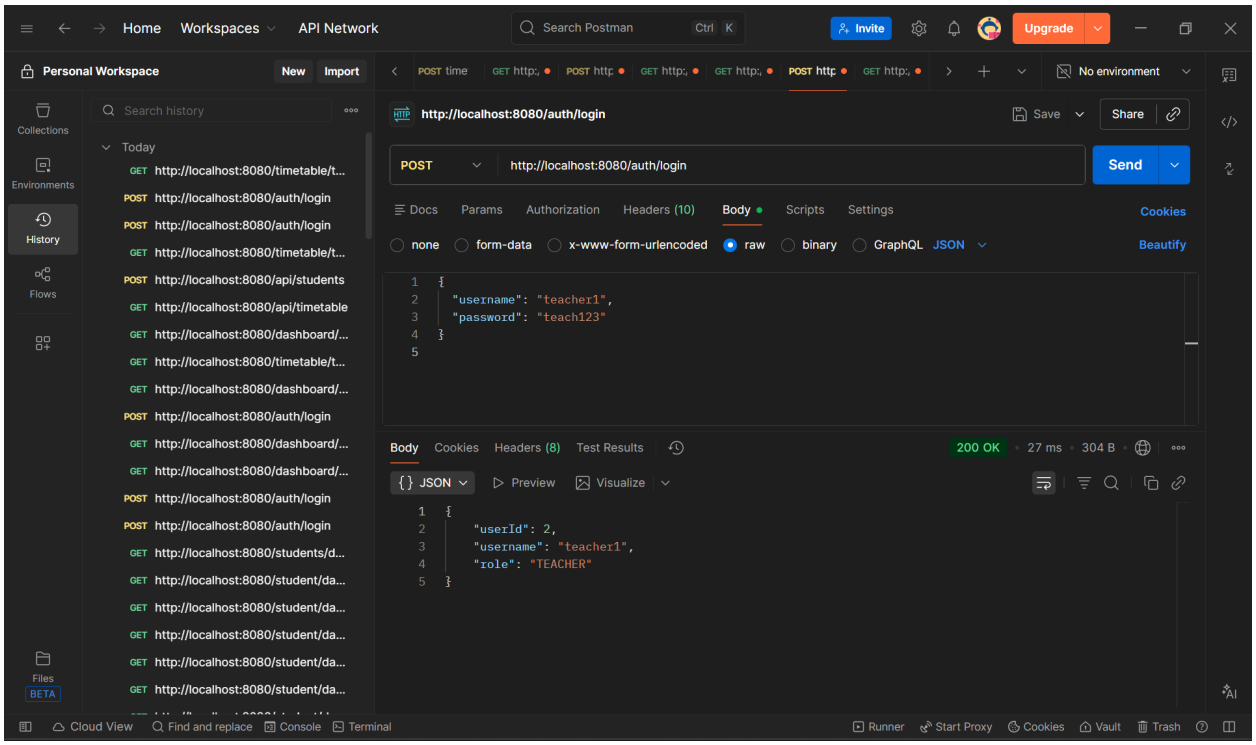
- **Database:** MySQL Server 8.0.



- **Web Server:** Embedded Apache Tomcat 9.0.



- **API Client:** Postman v10.0+.



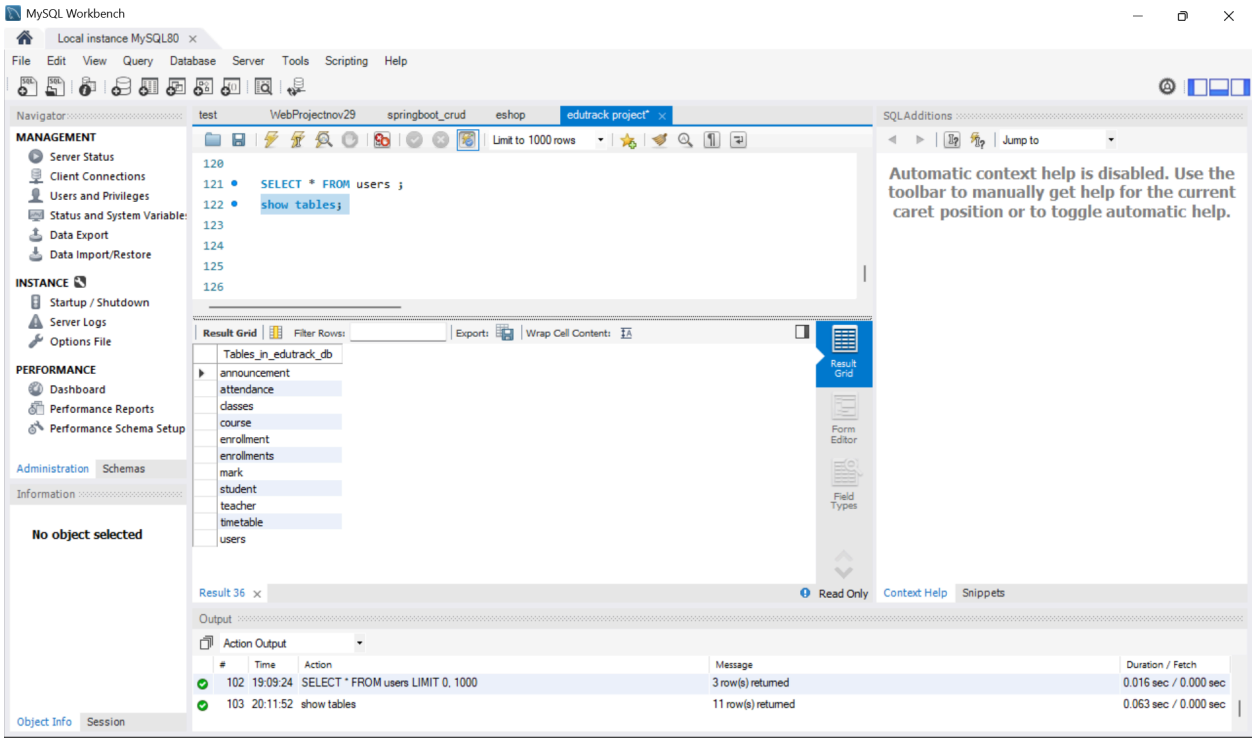
7.2DATABASE DESIGN

The database is designed using normalization principles to reduce redundancy and maintain data integrity. MySQL is used as the relational database management system.

Database Tables

- **Admin** – Stores admin credentials and details
- **Student** – Stores student information
- **Faculty** – Stores faculty details
- **Attendance** – Stores attendance records
- **Course** – Stores course information

Each table is related through primary and foreign keys to maintain relational integrity.



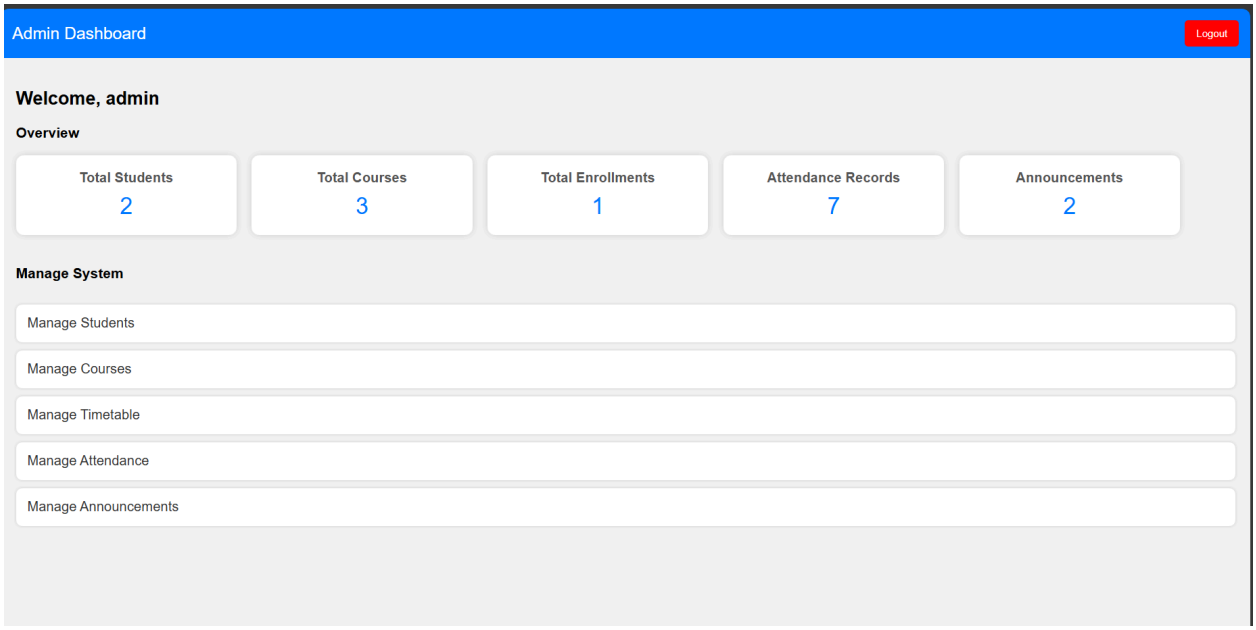
8. PROJECT MODULES

The system is divided into five logical modules to ensure separation of concerns and maintainability.

8.1 Admin Module

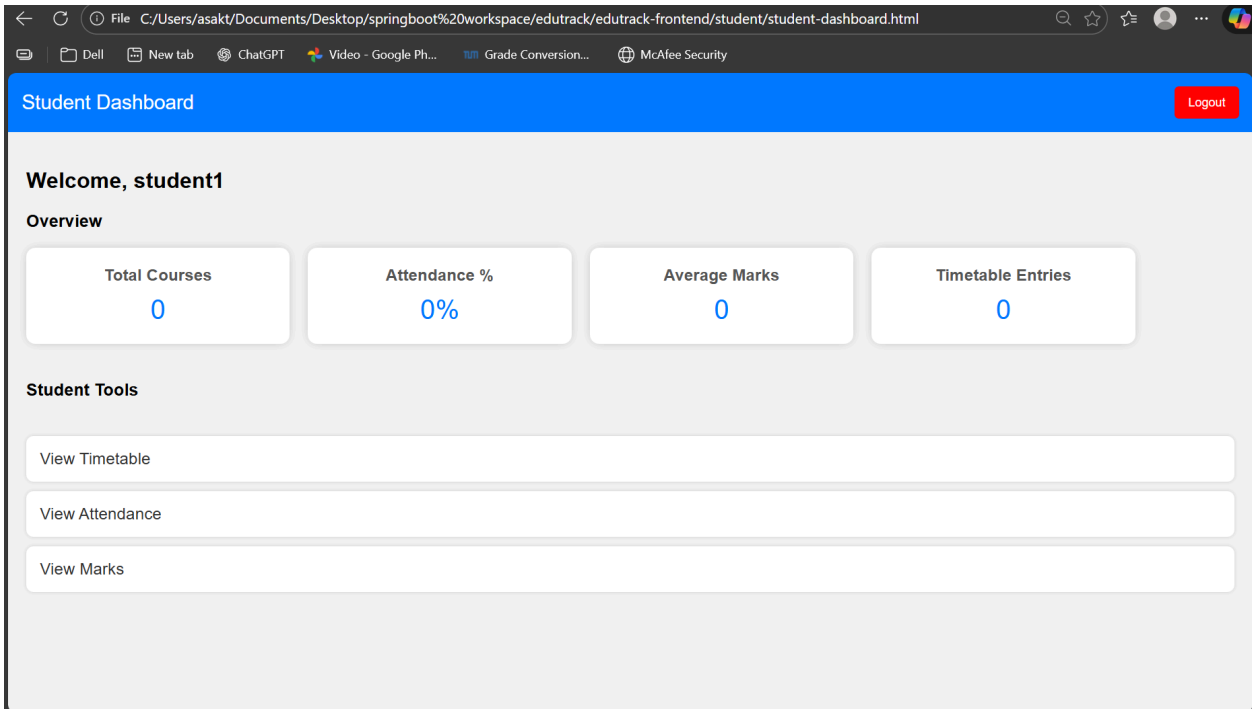
The administrative backbone. Responsible for registering users, creating courses, and setting up the academic timetable.

- Admin login and authentication
- Manage student and faculty records
- View attendance reports
- Perform CRUD operations using REST APIs



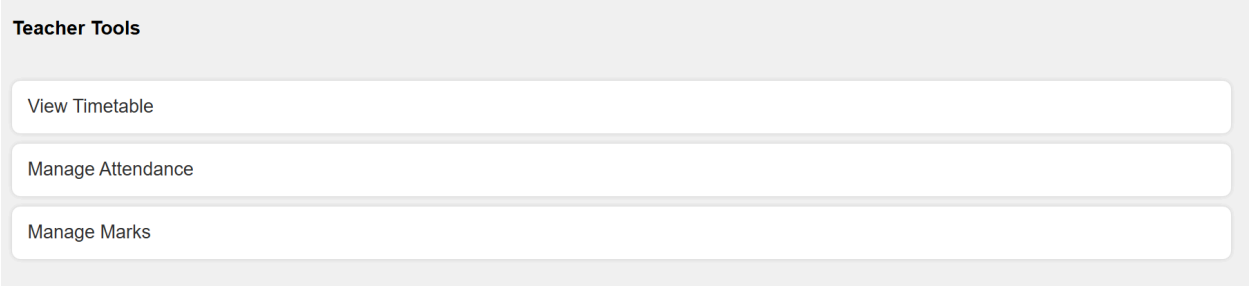
8.2 Student Module

Provides students with their academic records. Includes the student dashboard which shows attendance percentage and average marks.



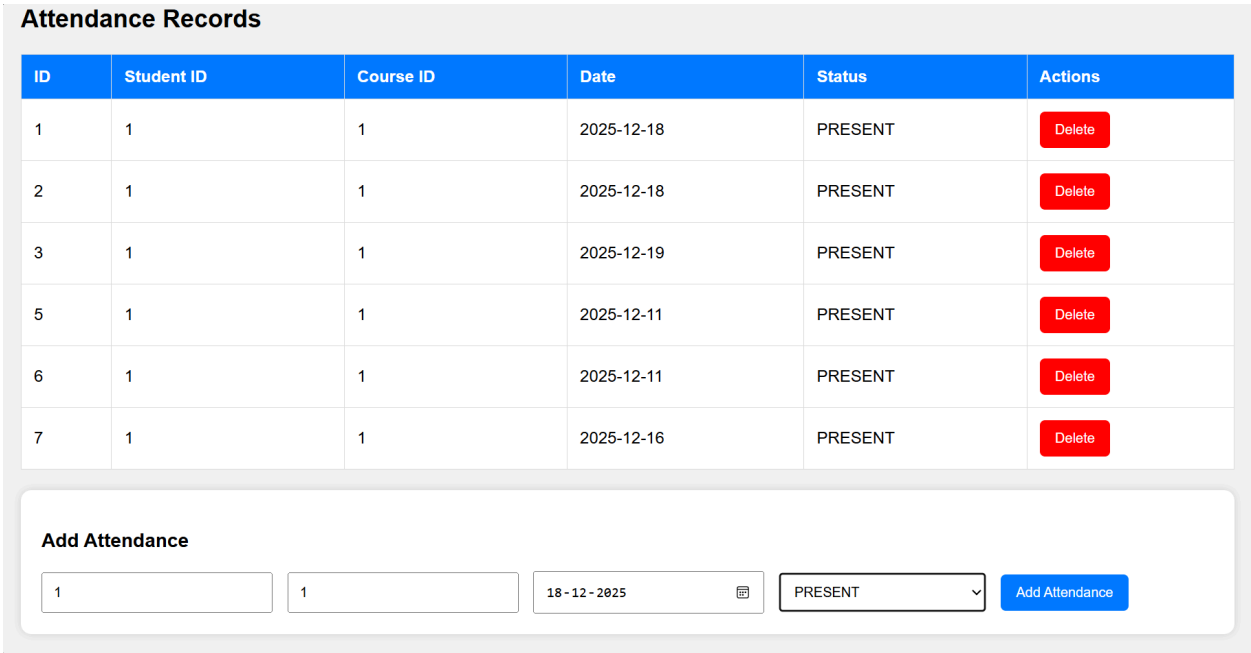
8.3 Teacher Module

The primary module for data entry. Teachers use this to mark daily attendance and input marks for various assessments.



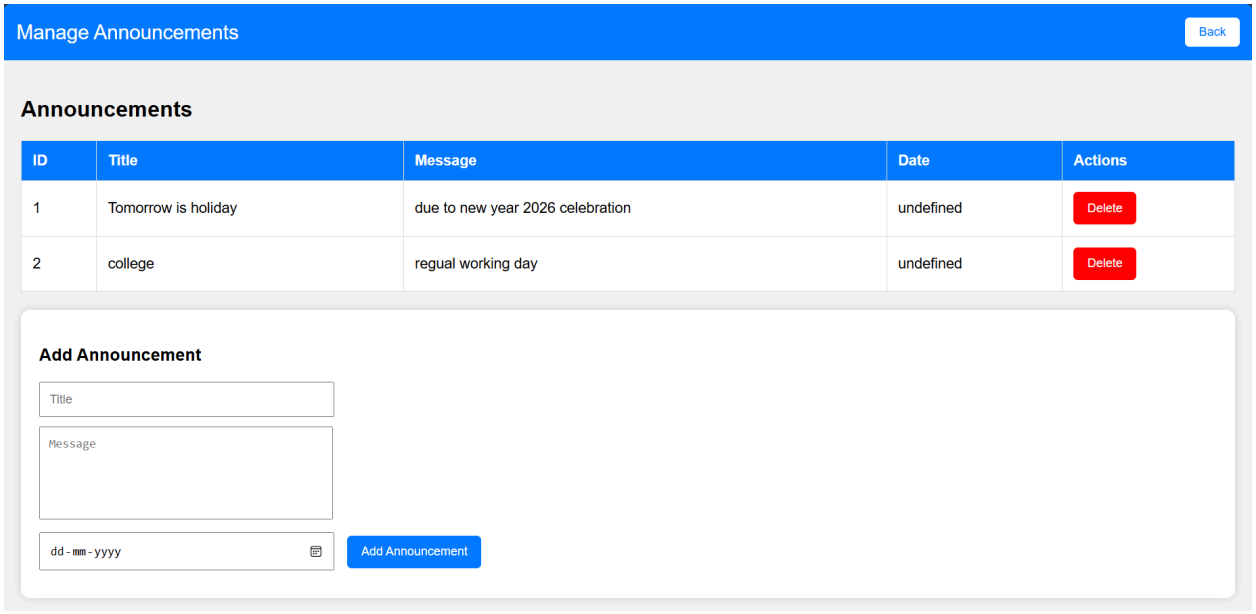
8.4 Attendance Module

The core tracking engine. It records daily presence/absence and provides the data needed for cumulative attendance reports.



8.5 Report & Announcement Module

Handles the broadcasting of information. Admins can create announcements that appear on both Student and Teacher dashboards.

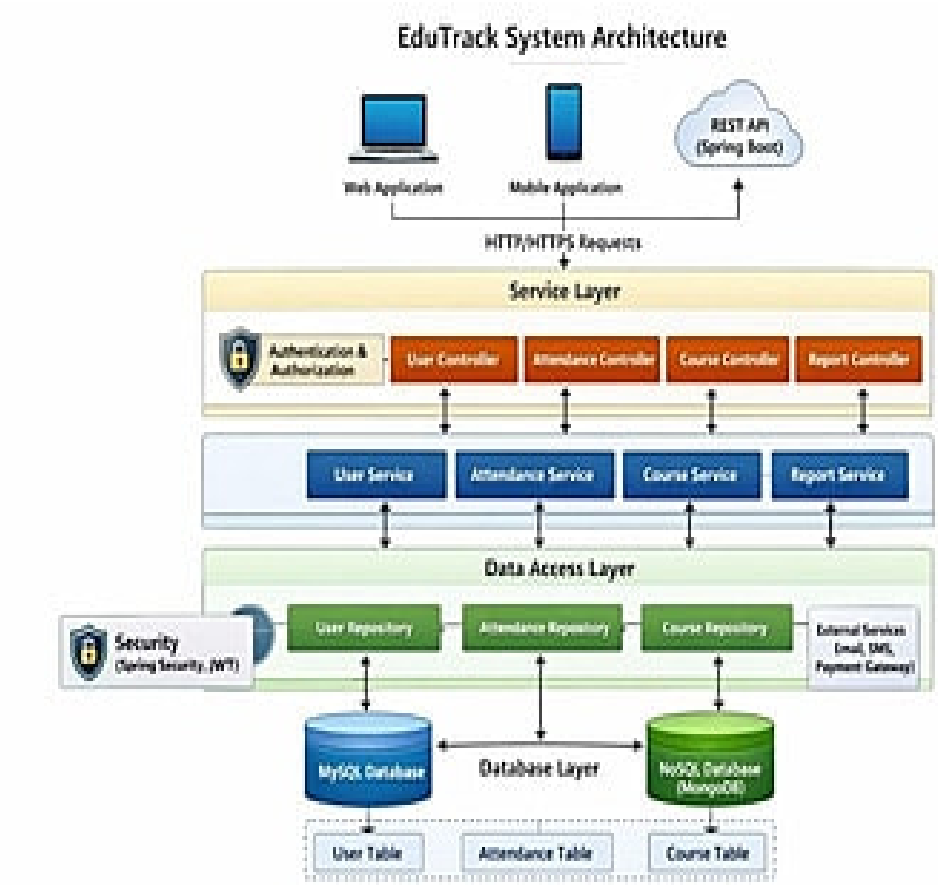


9. SYSTEM ARCHITECTURE

EduTrack utilizes a **Layered Architecture** within a **Client-Server model** to ensure that logic, data access, and presentation are decoupled.

9.1 The Layered Approach

1. **Presentation Layer (Frontend):** HTML, CSS, and JS files that handle user interactions and fetch data from the API.
2. **Controller Layer (API):** Spring RestControllers that handle HTTP requests and map them to service methods.
3. **Service Layer (Business Logic):** This is where calculations happen (e.g., calculating the average marks for a student).
4. **Repository Layer (Data Access):** Interfaces that interact with the MySQL database via Spring Data JPA.
5. **Database Layer (Storage):** The physical MySQL tables where data is persisted.



9.2 Data Flow Process

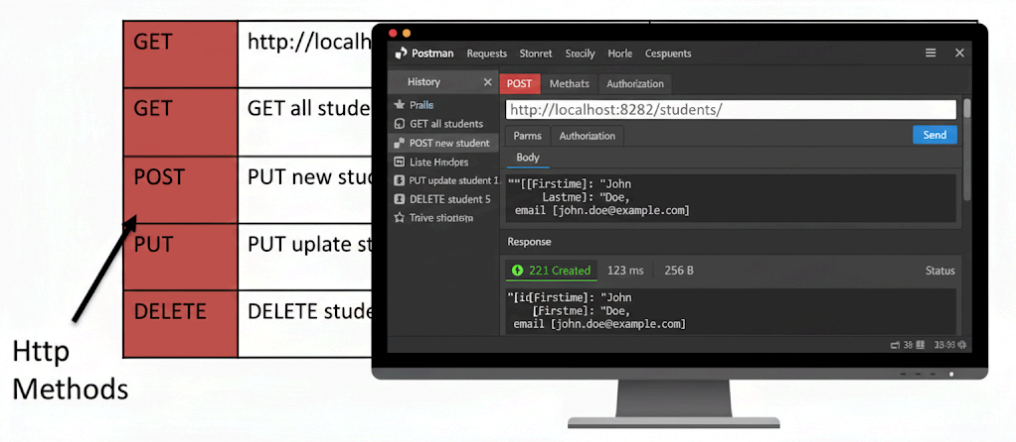
1. User fills out a form in the browser (e.g., Add Student).
2. JavaScript sends a POST request with a JSON payload to the backend.
3. The Controller validates the JSON and passes it to the Service.
4. The Service checks if the student already exists and then calls the Repository.
5. The Repository saves the student to MySQL.
6. A success response (201 Created) is sent back to the browser.

10. HTTP REQUEST METHODS & API DOCUMENTATION

The system follows REST principles using the standard set of HTTP methods.

10.1 Mapping Table

Method	Endpoint	Description
POST	/auth/login	Authenticates user and returns Role/ID.
GET	/students	Retrieves a list of all students (Admin Only).
POST	/students	Creates a new student record.
GET	/dashboard/admin	Fetches aggregate stats for the admin dashboard.
GET	/attendance	Retrieves all attendance records.
POST	/attendance	Records daily presence for a student in a course.
GET	/marks/student/{id}	Fetches academic grades for a specific student.
POST	/announcements	Publishes a new system-wide notice.

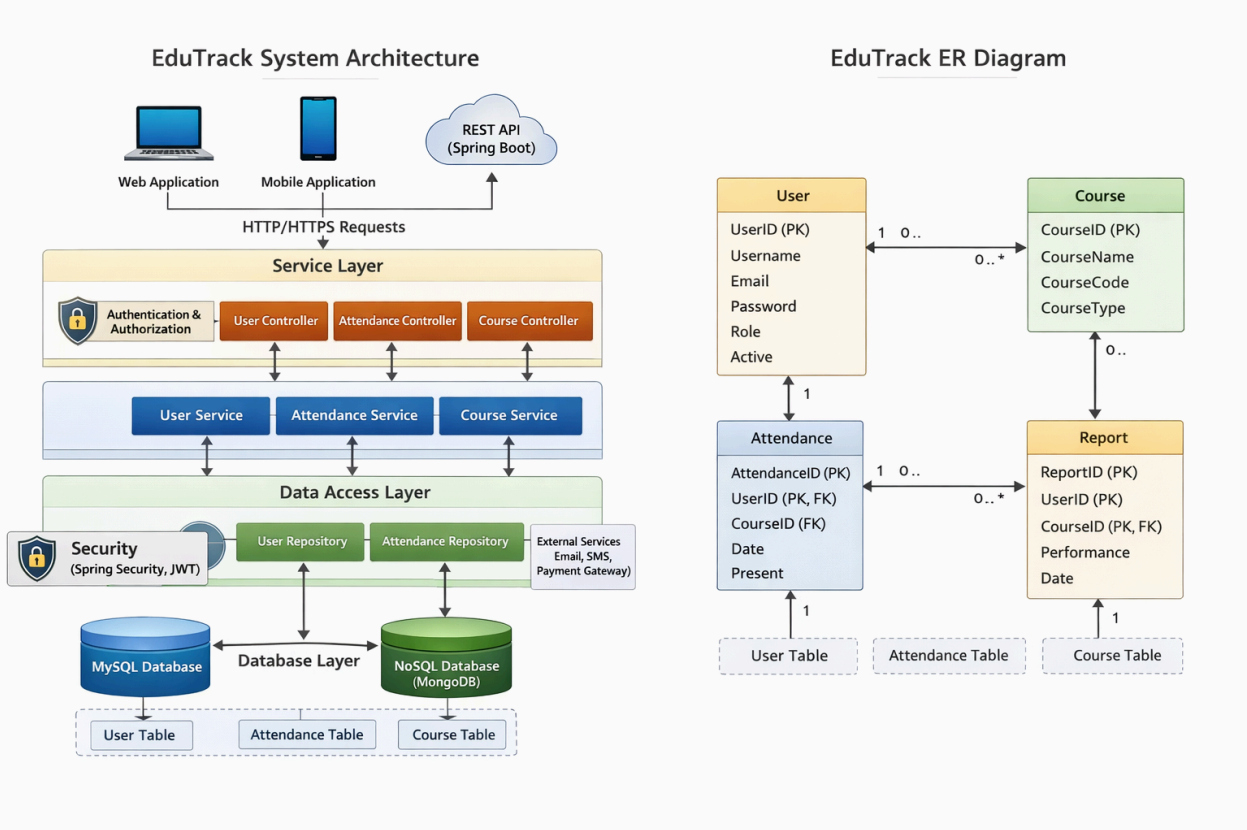


11. DATABASE DESIGN

The database is designed with high normalization (3NF) to minimize redundancy and prevent data anomalies.

11.1 ER Diagram Summary

- **User (Table):** Base table for admin, teacher, and student credentials.
- **Student (Table):** Contains profile details (name, email, department).
- **Course (Table):** Stores course codes and department information.
- **Timetable (Table):** Join table relating Student, Teacher, and Course with time slots.
- **Attendance (Table):** Tracks date, status (Present/Absent) per Student and Course.
- **Marks (Table):** Records marks value for a Student in a Course.



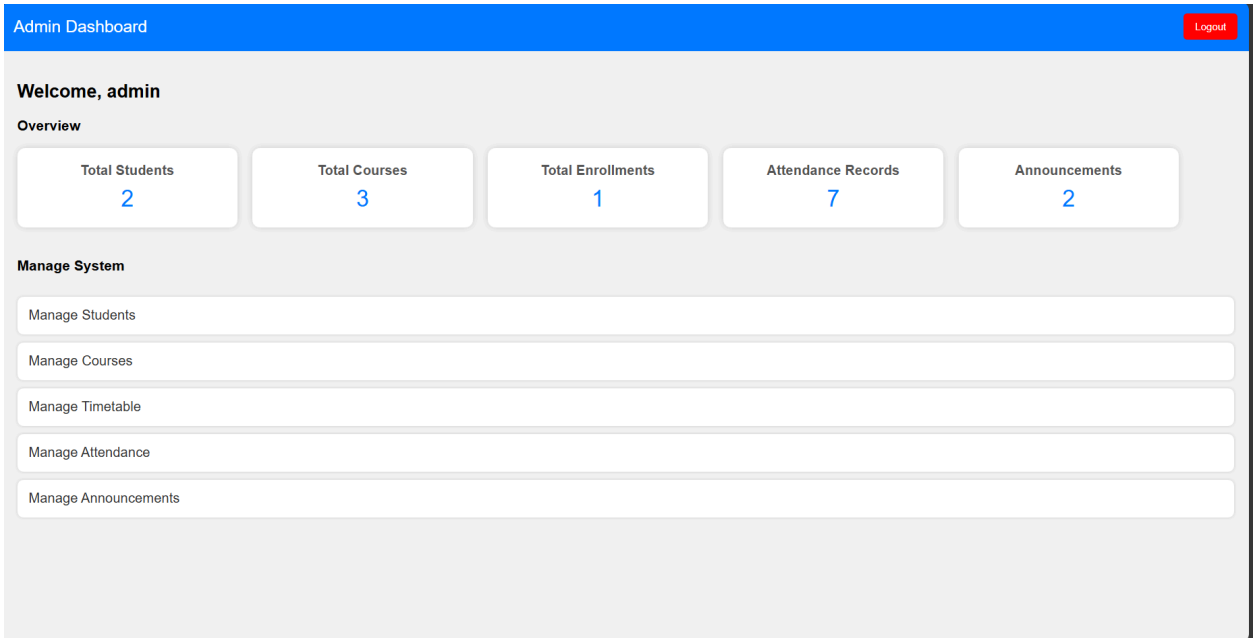
11.2 Key Data Definitions

- **Primary Keys:** Every table uses an auto-incrementing id column.
- **Foreign Keys:** attendance.student_id references student.id.
- **Unique Constraints:** student.email must be unique to prevent duplicate registrations.

12. ADMIN MODULE DETAILS

The Admin has the authority to manage the system's foundational data.

- **Manage Students:** Through students.html, Admins can view the full student table and add new entries.
- **Manage Courses:** Through courses.html, courses are defined for different departments.
- **Timetable Setup:** The timetable.html interface allows Admins to schedule classes, assigning a teacher and student to a specific course time.
- **Announcement Hub:** Admins post notices like holiday alerts or exam schedules via announcements.html.

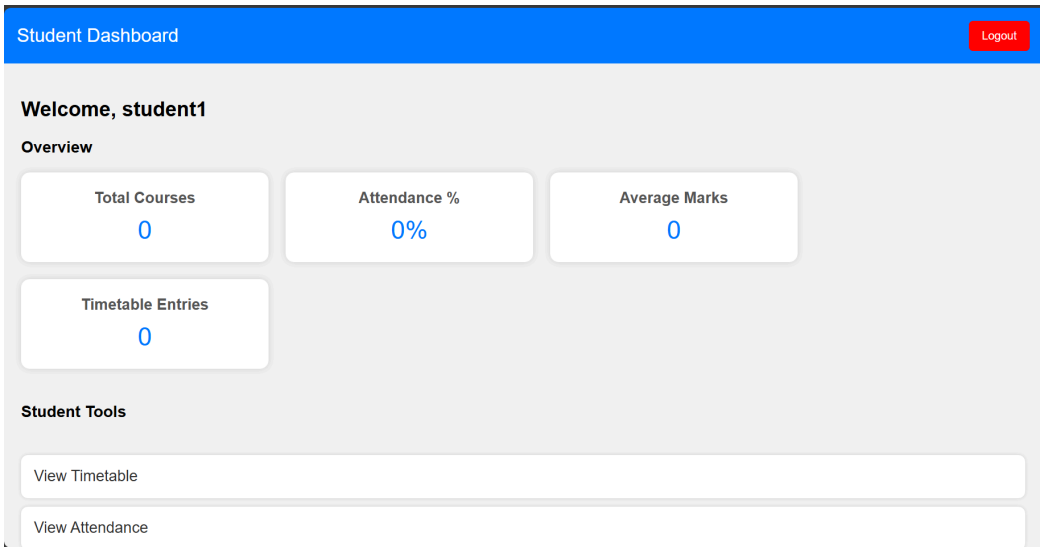


13. STUDENT & FACULTY PORTALS

13.1 Student Portal (student-dashboard.html)

The dashboard fetches real-time data using the endpoint /dashboard/student/{id}.

- **Total Courses:** Number of subjects enrolled.
- **Attendance %:** Automatically calculated as (Present Sessions / Total Sessions) * 100.
- **Average Marks:** Real-time GPA/Average tracker.



13.2 Faculty Portal

- **Marking Attendance:** Teachers select a course and date to mark students as Present or Absent.
- **Grading:** Access to input assignment, mid-term, and final exam scores

.

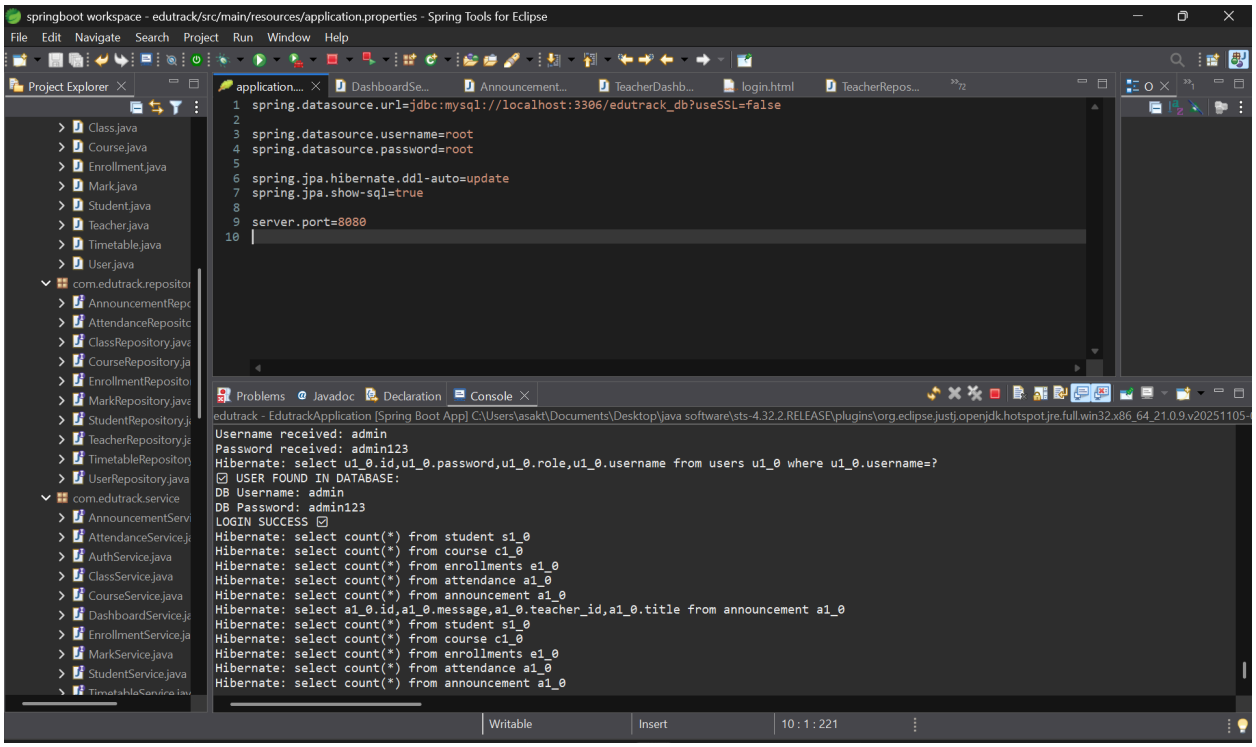
14. TESTING

Testing ensures that the application is reliable and ready for deployment.

14.1 Unit Testing (JUnit 5)

Tests are written for the Repository layer to ensure that database queries work as expected.

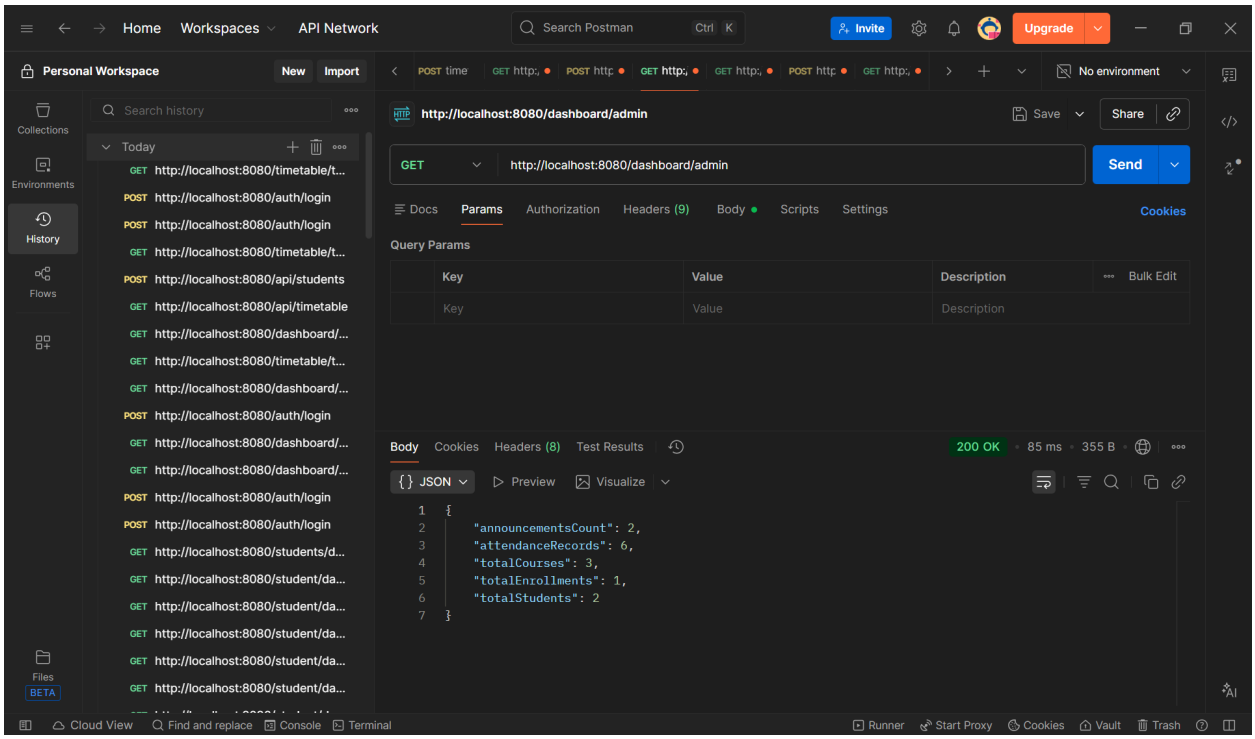
- saveStudentTest(): Verifies that a student record is correctly written to the DB.
- getAttendanceByDate(): Verifies that the repository correctly filters records by date.



14.2 API Testing (Postman)

Every REST endpoint is tested for:

- **Response Code:** Ensuring 200 OK or 201 Created.
- **Data Integrity:** Ensuring the JSON returned matches the database values.
- **Security:** Verifying that unauthorized users (e.g., a student trying to access /admin endpoints) get a 403 Forbidden error.



15. CONCLUSION & FUTURE ENHANCEMENTS

15.1 Conclusion

The **EduTrack Student Management System** provides a robust, professional framework for academic administration. By automating attendance, grading, and scheduling, it eliminates the risks associated with manual data entry. Its layered architecture ensures it is easy to maintain and extend, while its role-based security protects sensitive institutional information.

15.2 Future Enhancements

- **Mobile App:** Developing a Flutter or React Native app for students to check attendance on the go.
- **Automatic Notifications:** Integration with SMS/Email gateways to alert parents of student absences.
- **Analytics Dashboard:** Using charts (Chart.js) to visualize attendance trends over a semester.
- **Cloud Integration:** Deploying the MySQL database and Spring Boot app to AWS for global accessibility.

PROJECT VIDEO LINK :

https://drive.google.com/drive/folders/1izP6ATPEtk_WrmppnUcEpdcU2l-wjqg2?usp=drive_link