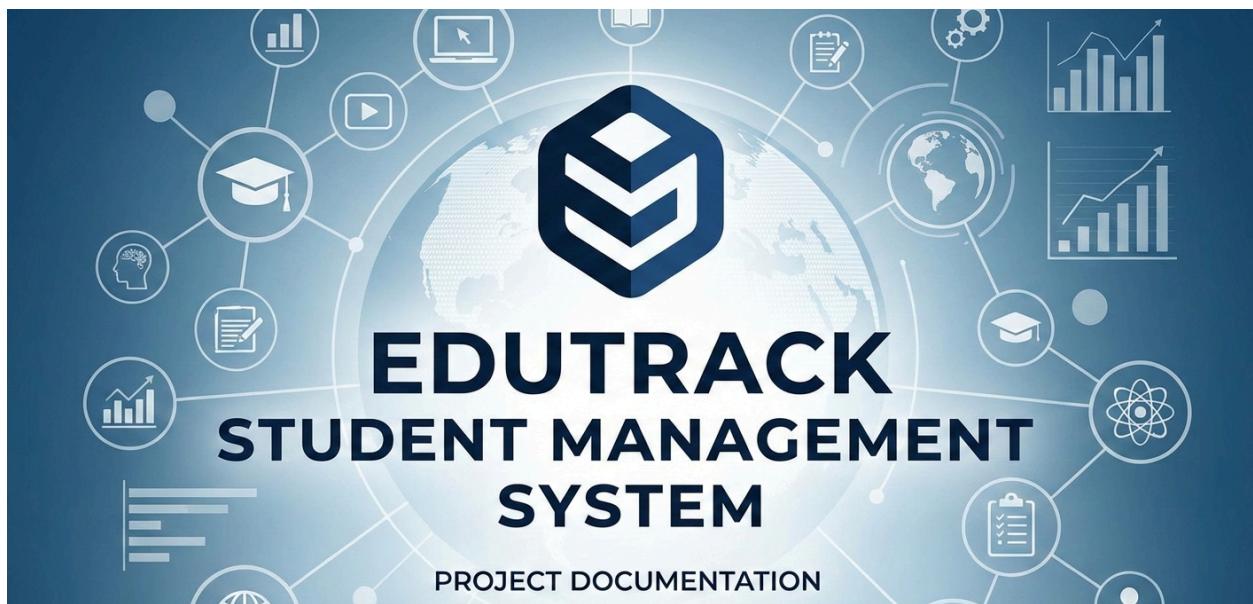


## Case Study on

# 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:

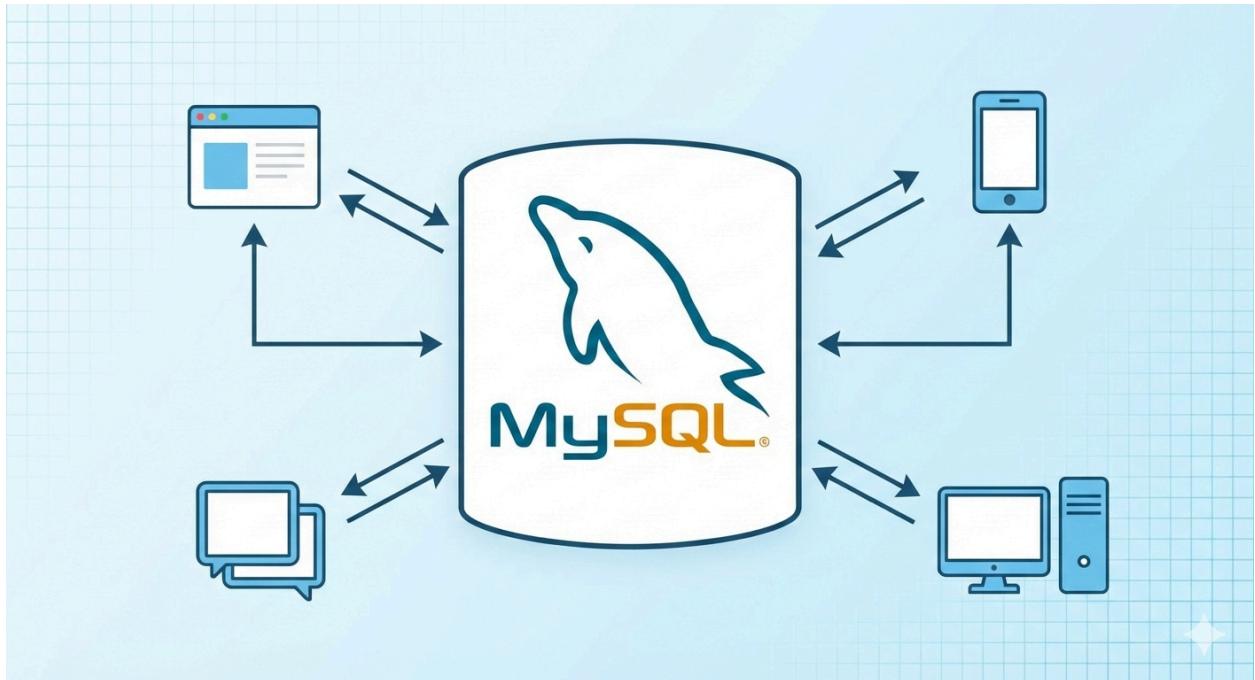
- **Administrators:** To maintain high-level oversight of institutional data.
- **Teachers (Faculty):** To automate the tedious tasks of attendance marking and grading.
- **Students:** To engage with their academic progress through personalized dashboards.

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The **EduTrack Student Management System** is a comprehensive academic administration application developed to replace outdated manual attendance and record systems with a reliable digital platform.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.

**Welcome, admin****Overview**

Total Students

2

Total Courses

2

Total Enrollments

1

Attendance Records

6

Announcements

1

**Manage System**[Manage Students](#)[Manage Courses](#)

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

**Teacher Tools**[View Timetable](#)[Manage Attendance](#)[Manage Marks](#)

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

Student Dashboard

Welcome, student1

**Overview**

Total Courses 0	Attendance % 0%	Average Marks 0
Timetable Entries 0		

**Student Tools**

[View Timetable](#)

[View Attendance](#)

- **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

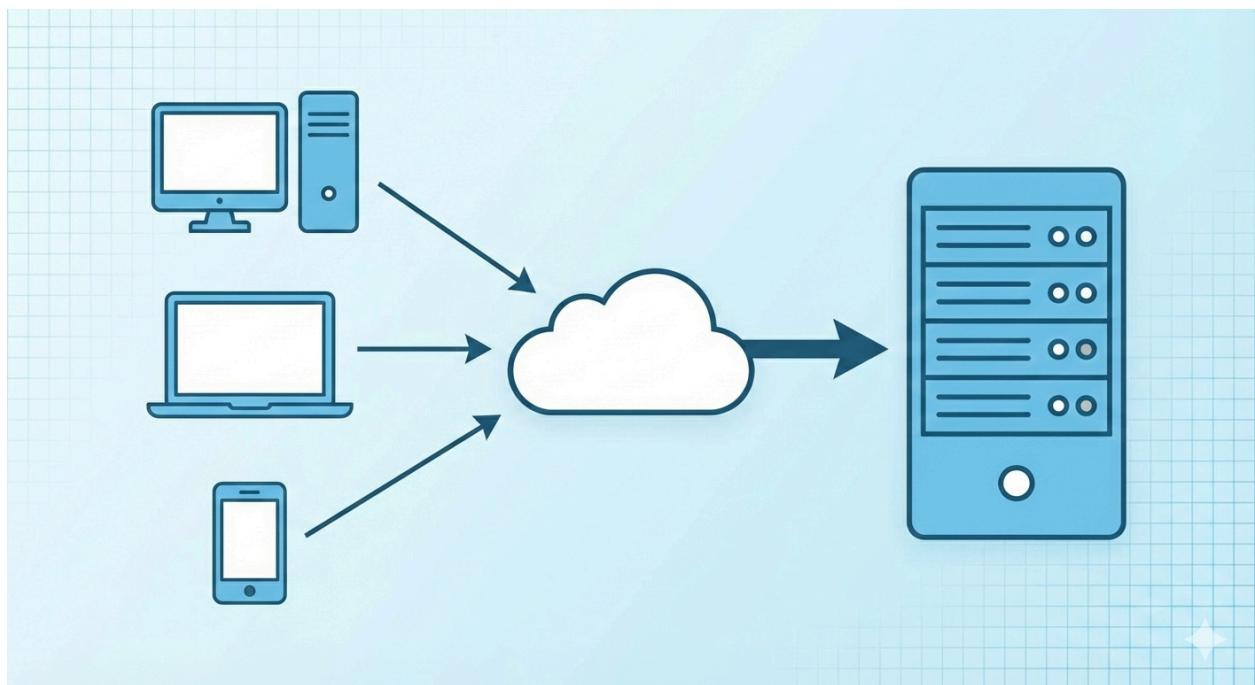
EduTrack is a modular application where each component is designed to handle a specific domain of institutional management while communicating through a standardized API layer.

### 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

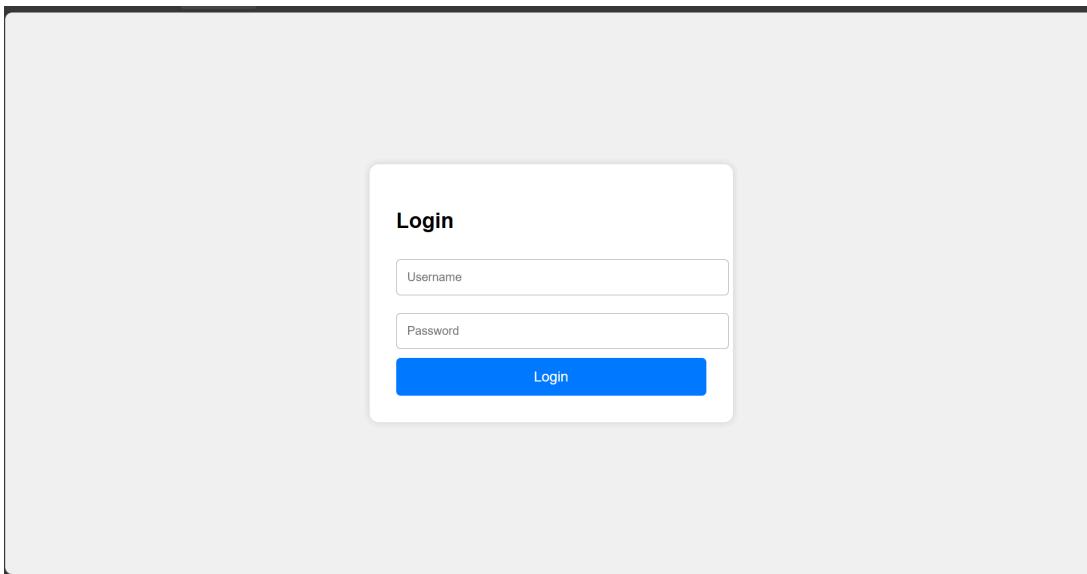
- **Centralized Database:** A single MySQL instance for all academic records.
- **Automated Scheduling:** Conflict-free timetable management.
- **Performance Analytics:** Calculating and displaying student averages and attendance percentages on dashboards.



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

The screenshot shows the Eclipse IDE interface with the following components visible:

- Project Explorer:** On the left, it displays the project structure. It includes a main package named "com.edutrack.service.impl" containing various service implementation classes like AnnouncementServiceImpl.java, AttendanceServiceImpl.java, AuthServiceImpl.java, ClassServiceImpl.java, CourseServiceImpl.java, DashboardServiceImpl.java, EnrollmentServiceImpl.java, MarkServiceImpl.java, StudentServiceImpl.java, TimetableServiceImpl.java, and UserServiceImpl.java. It also shows resources, test Java files, JRE System Library, Maven Dependencies, generated sources, and annotations.
- Editor:** The central area shows the content of a file named "login.html". The code is a simple HTML form for user login, featuring a title, a style block for styling the body and login box, and an input field for the password.
- Console:** At the bottom, the "Console" tab is active, displaying logs from the application's startup. The logs indicate the initialization of the EntityManagerFactory, configuration of the OpenWebConfiguration, the start of the Tomcat web server, the startup of the EdutrackApplication, and the initialization of the DispatcherServlet.

- **Database:** MySQL Server 8.0.

The screenshot shows the MySQL Workbench interface. The left sidebar contains navigation links for MANAGEMENT, INSTANCE, PERFORMANCE, and Administration. The central area has tabs for test, WebProjectnov29, springboot\_crud, eshop, and edutrack project. A query editor window is open with the following content:

```

118     VALUES (1, 101, 99);
119 •  SELECT * FROM users WHERE role = 'STUDENT';
120
121 •  SELECT * FROM users ;
122
123

```

The Result Grid shows the following data:

	id	username	password	role	student_id	teacher_id
▶	1	admin	admin123	ADMIN	NULL	NULL
▶	2	teacher1	teach123	TEACHER	NULL	1
▶	3	student1	stud123	STUDENT	1	NULL
▶	4	NULL	NULL	NULL	NULL	NULL

The Output panel shows the following log entries:

#	Time	Action	Message	Duration / Fetch
100	14:35:45	SELECT * FROM users WHERE role = 'TEACHER' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
101	17:53:03	SELECT * FROM users LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec

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

```

ack_db/undefined
known
READ [default REPEATABLE_READ]

viderImpl
unknown
unknown
[ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform')
[ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
[ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries
[ restartedMain] o.s.boot.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path ''
[ restartedMain] com.edutrack.EdutrackApplication : Started EdutrackApplication in 1.013 seconds (process running for 999.224
[ restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
[nio-8080-exec-4] o.a.c.c.C.[Tomcat-3].[localhost].[] : Initializing Spring DispatcherServlet 'dispatcherServlet'
[nio-8080-exec-4] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
[nio-8080-exec-4] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
it s1_0

```

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

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Personal Workspace' containing collections, environments, history, flows, and files. The main area shows a list of recent requests. A specific POST request to `http://localhost:8080/auth/login` is selected. The 'Body' tab shows a JSON payload:

```

1 {
2   "username": "teacher1",
3   "password": "teach123"
4 }
5
    
```

The response pane shows a 200 OK status with a response time of 27 ms and a size of 304 B. The response body is also a JSON object:

```

1 {
2   "userId": 2,
3   "username": "teacher1",
4   "role": "TEACHER"
5 }
    
```

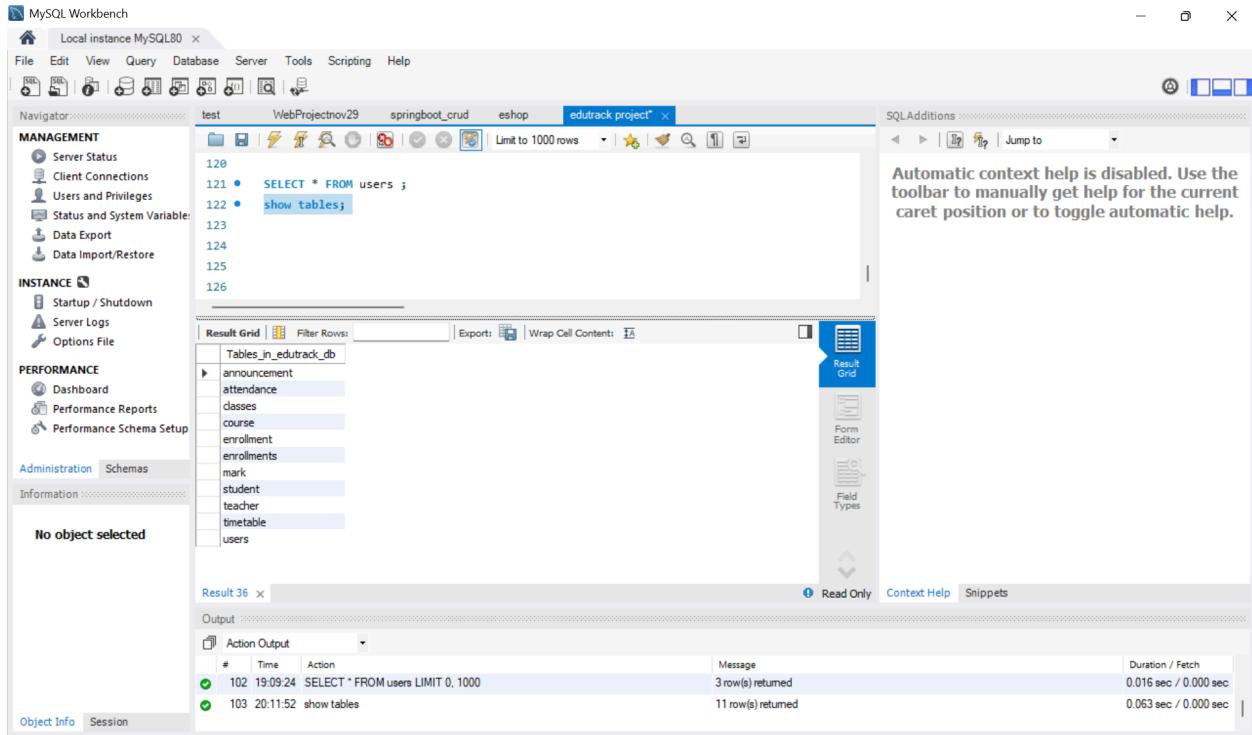
## 7.2 DATABASE DESIGN

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

Admin Dashboard

Welcome, admin

Logout

Overview

Total Students 2

Total Courses 3

Total Enrollments 1

Attendance Records 7

Announcements 2

Manage System

Manage Students

Manage Courses

Manage Timetable

Manage Attendance

Manage Announcements

## 8.2 Student Module

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

C:\Users\asakt\Documents\Desktop\springboot%20workspace\edutrack\edutrack-frontend\student\student-dashboard.html

File ChatGPT Video - Google Ph... Grade Conversion... McAfee Security

Student Dashboard

Welcome, student1

Logout

Overview

Total Courses 0

Attendance % 0%

Average Marks 0

Timetable Entries 0

Student Tools

View Timetable

View Attendance

View Marks

## 8.3 Teacher Module

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

**Teacher Tools**

[View Timetable](#)

[Manage Attendance](#)

[Manage Marks](#)

## 8.4 Attendance Module

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

**Attendance Records**

ID	Student ID	Course ID	Date	Status	Actions
1	1	1	2025-12-18	PRESENT	<button>Delete</button>
2	1	1	2025-12-18	PRESENT	<button>Delete</button>
3	1	1	2025-12-19	PRESENT	<button>Delete</button>
5	1	1	2025-12-11	PRESENT	<button>Delete</button>
6	1	1	2025-12-11	PRESENT	<button>Delete</button>
7	1	1	2025-12-16	PRESENT	<button>Delete</button>

**Add Attendance**

1      1      18 - 12 - 2025      PRESENT      Add Attendance

## 8.5 Report & Announcement Module

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

The screenshot shows a web-based application titled "Manage Announcements". At the top, there is a blue header bar with the title and a "Back" button. Below the header, the page is titled "Announcements". A table lists two existing announcements:

ID	Title	Message	Date	Actions
1	Tomorrow is holiday	due to new year 2026 celebration	undefined	<button>Delete</button>
2	college	regual working day	undefined	<button>Delete</button>

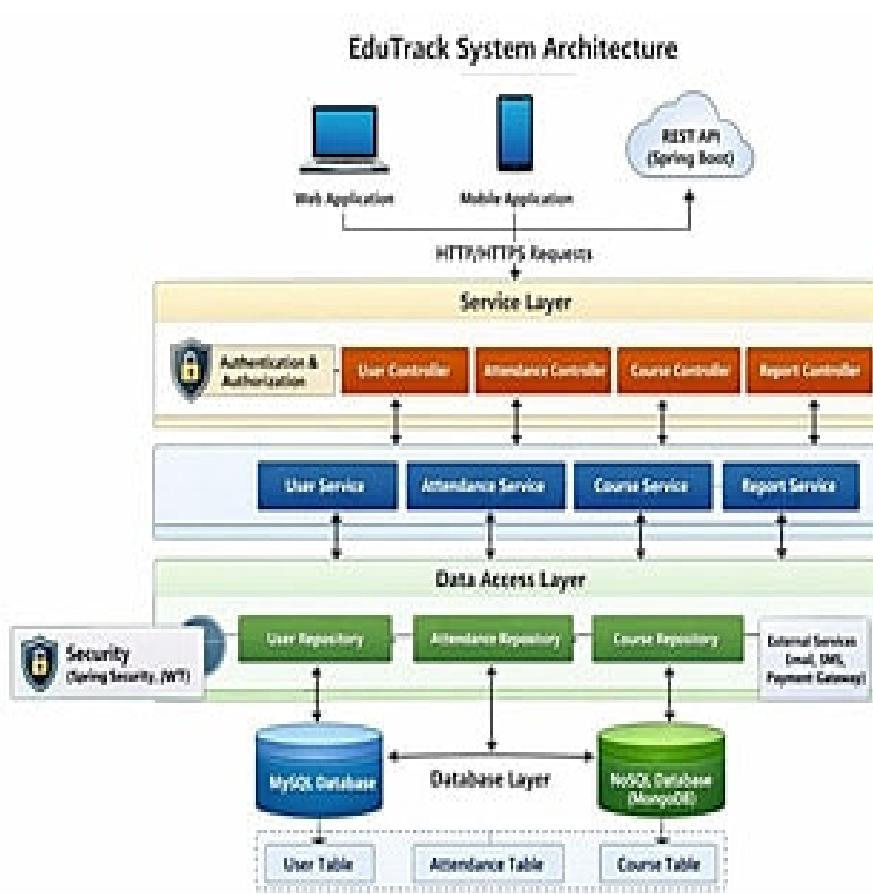
Below the table, there is a form titled "Add Announcement" with fields for "Title" (input type="text"), "Message" (input type="text"), and "Date" (input type="date" placeholder="dd-mm-yyyy"). There is also a "Add Announcement" button.

## 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

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



## 9.2 Data Flow Process

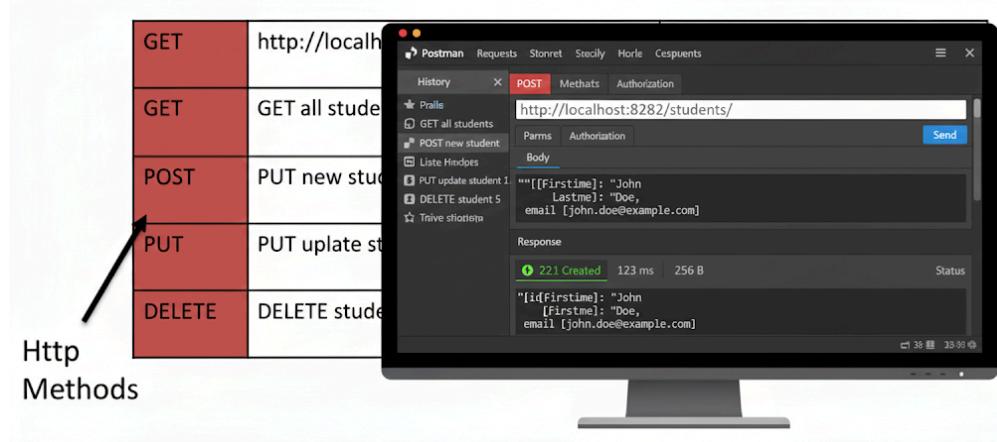
1. User fills out a form in the browser (e.g., Add Student).
2. JavaScript sends a POST request with a JSON payload to the backend.
3. The Controller validates the JSON and passes it to the Service.
4. The Service checks if the student already exists and then calls the Repository.
5. The Repository saves 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
<b>POST</b>	/auth/login	Authenticates user and returns Role/ID.
<b>GET</b>	/students	Retrieves a list of all students (Admin Only).
<b>POST</b>	/students	Creates a new student record.
<b>GET</b>	/dashboard/admin	Fetches aggregate stats for the admin dashboard.
<b>GET</b>	/attendance	Retrieves all attendance records.
<b>POST</b>	/attendance	Records daily presence for a student in a course.
<b>GET</b>	/marks/student/{id}	Fetches academic grades for a specific student.
<b>POST</b>	/announcements	Publishes a new system-wide notice.

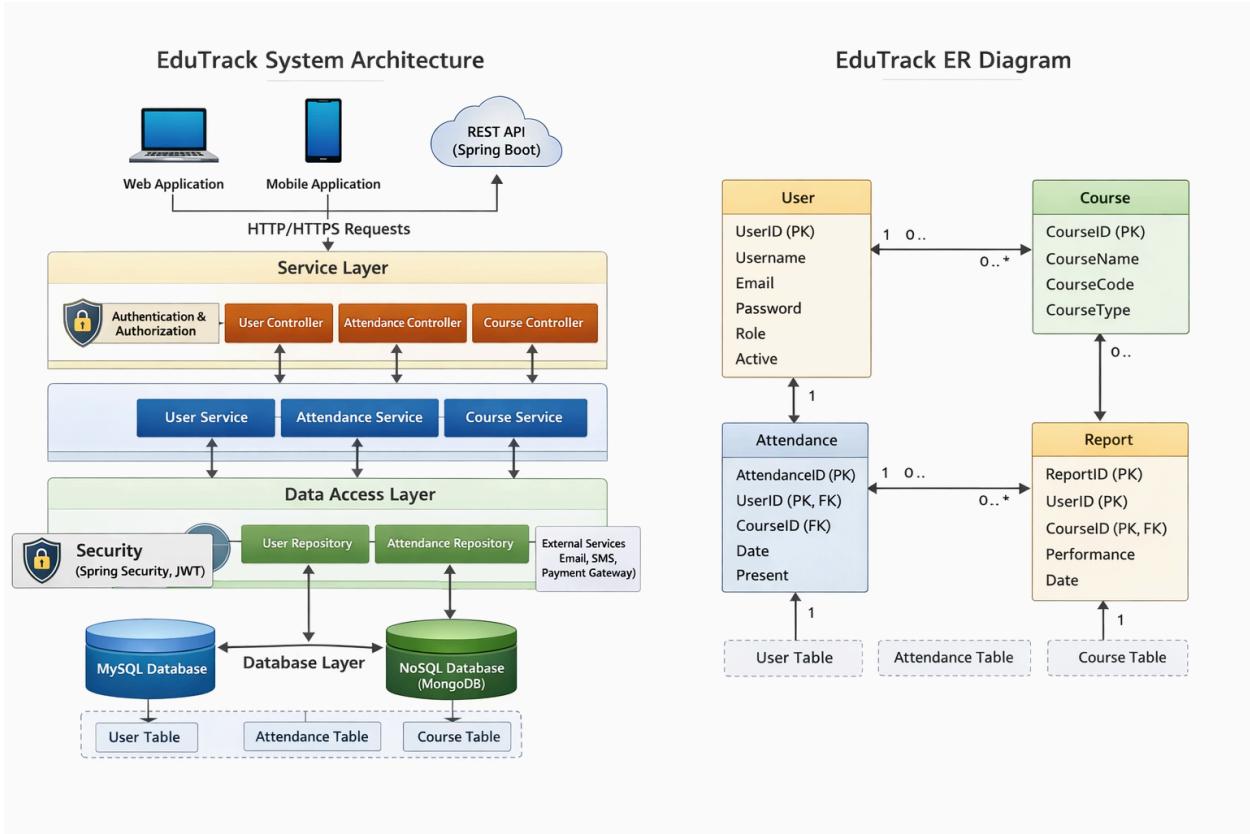


## 11. DATABASE DESIGN

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.

The screenshot shows the Admin Dashboard interface. At the top, a blue header bar displays "Admin Dashboard" on the left and a red "Logout" button on the right. Below the header, a welcome message "Welcome, admin" is shown. A section titled "Overview" contains five cards with summary data:

Total Students	Total Courses	Total Enrollments	Attendance Records	Announcements
2	3	1	7	2

Below the overview, a section titled "Manage System" lists several administrative tasks:

- Manage Students
- Manage Courses
- Manage Timetable
- Manage Attendance
- Manage Announcements

## 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.

The screenshot shows the Student Dashboard interface. At the top, a blue header bar displays "Student Dashboard" on the left and a red "Logout" button on the right. Below the header, a welcome message "Welcome, student1" is shown. A section titled "Overview" contains four cards with summary data:

Total Courses	Attendance %	Average Marks
0	0%	0

Below the overview, a section titled "Student Tools" lists two options:

- View Timetable
- View Attendance

## 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.

The screenshot shows the Spring Tools for Eclipse interface. The Project Explorer view on the left lists various Java files and repositories. The application.properties file is open in the main editor area, containing configuration for a MySQL database and a Spring Boot application. The Console tab at the bottom displays the output of a Java application. It shows a successful login attempt for the user 'admin' with password 'admin123'. The logs from Hibernate show the execution of several SQL queries to find the user in the database and to count records in tables like student, course, enrollment, attendance, and announcement.

```
spring.datasource.url=jdbc:mysql://localhost:3306/edutrack_db?useSSL=false
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
server.port=8080

application.properties

Username received: admin
Password received: admin123
Hibernate: select u1_0.id,u1_0.password,u1_0.role,u1_0.username from users u1_0 where u1_0.username=?
[User FOUND IN DATABASE]
DB Username: admin
DB Password: admin123
LOGIN SUCCESS [User FOUND IN DATABASE]
Hibernate: select count(*) from student s1_0
Hibernate: select count(*) from course c1_0
Hibernate: select count(*) from enrollments e1_0
Hibernate: select count(*) from attendance a1_0
Hibernate: select count(*) from announcement a1_0
Hibernate: select a1_0.id,a1_0.message,a1_0.teacher_id,a1_0.title from announcement a1_0
Hibernate: select count(*) from student s1_0
Hibernate: select count(*) from course c1_0
Hibernate: select count(*) from enrollments e1_0
Hibernate: select count(*) from attendance a1_0
Hibernate: select count(*) from announcement a1_0
```

## 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.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Personal Workspace' containing 'Collections', 'Environments', 'History', 'Flows', and 'Files (BETA)'. The main area shows a list of recent requests. A specific request to 'http://localhost:8080/dashboard/admin' is selected, showing a 'GET' method. The 'Params' tab is active, displaying an empty table for query parameters. Below the table, the 'Body' tab is selected, showing a JSON response with the following content:

```
1 {
2   "announcementsCount": 2,
3   "attendanceRecords": 6,
4   "totalCourses": 3,
5   "totalEnrollments": 1,
6   "totalStudents": 2
7 }
```

The status bar at the bottom indicates a '200 OK' response with '85 ms' and '355 B'.

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