Student Management System

Student Management System using Spring Boot, Spring Data JPA, MySQL, and Thymeleaf.

📋 Project Overview

Tech Stack:

- Backend: Spring Boot 3.x
- Database: MySQL 8.x
- ORM: Spring Data JPA (Hibernate)
- Frontend: Thymeleaf + Bootstrap 5
- Security: Spring Security
- Build Tool: Maven

🗂 Phase 1: Project Setup & Database Design

Step 1.1: Initialize Spring Boot Project

1. Go to start.spring.io
2. Select dependencies:
   - Spring Web
   - Spring Data JPA
   - MySQL Driver
   - Thymeleaf
   - Spring Security
   - Spring Boot DevTools
   - Validation
   - Lombok (optional, for cleaner code)

Step 1.2: Database Schema Design

```sql
-- Create database
CREATE DATABASE student_management_system;

-- Tables structure (relationships):
User (id, username, password, role_id, created_at)
Role (id, role_name) -- Admin, Student
Student (id, user_id, first_name, last_name, email, phone, address, dob, enrollment_date)
Course (id, course_name, course_code, description, duration, fees, start_date, end_date)
Enrollment (id, student_id, course_id, enrollment_date, status)
Attendance (id, enrollment_id, date, status) -- Present, Absent, Late
Payment (id, student_id, enrollment_id, amount, payment_date, payment_method, status)
Notification (id, title, message, created_date, target_role, is_read)
```

Step 1.3: Configure `application.properties`

```
spring.datasource.url=jdbc:mysql://localhost:3306/student_management_system
spring.datasource.username=root
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
server.port=8080
```

🏗 Phase 2: Entity Classes

Entity classes with proper relationships:

Entity Relationships:

1. **User** ↔ **Role**: ManyToOne
2. **User** ↔ **Student**: OneToOne
3. **Student** ↔ **Enrollment**: OneToMany
4. **Course** ↔ **Enrollment**: OneToMany
5. **Enrollment** ↔ **Attendance**: OneToMany
6. **Student** ↔ **Payment**: OneToMany
7. **Enrollment** ↔ **Payment**: OneToMany

Order of Creation:

1. `Role.java` (no dependencies)
2. `User.java` (depends on Role)
3. `Course.java` (no dependencies)
4. `Student.java` (depends on User)
5. `Enrollment.java` (depends on Student, Course)
6. `Attendance.java` (depends on Enrollment)
7. `Payment.java` (depends on Student, Enrollment)
8. `Notification.java` (no dependencies)

🔧 Phase 3: Repository Layer

Repository interfaces for each entity:

```
repositories/
├── RoleRepository.java
├── UserRepository.java
├── StudentRepository.java
├── CourseRepository.java
├── EnrollmentRepository.java
├── AttendanceRepository.java
├── PaymentRepository.java
└── NotificationRepository.java
```

Custom query methods (e.g., `findByUsername`, `findByStudentId`, etc.)

🎯 Phase 4: Service Layer

Service classes with business logic:

```
services/
├── RoleService.java
├── UserService.java
├── StudentService.java
├── CourseService.java
├── EnrollmentService.java
├── AttendanceService.java
├── PaymentService.java
└── NotificationService.java
```

CRUD operations for each entity:

- `enrollStudentInCourse()`
- `markAttendance()`
- `processPayment()`
- `sendNotification()`
- `getStudentDashboard()`

🔐 Phase 5: Security Configuration

Step 5.1: Spring Security

- Configure authentication and authorization
- Create custom `UserDetailsService`
- Password encoding with BCrypt
- Role-based access control (ADMIN, STUDENT)

Step 5.2: Access Control Rules

- **/admin/**: Only ADMIN
- **/student/**: Only STUDENT
- **/login**, **/**: Public

💻 Phase 6: Controller Layer

Controllers for different functionalities:

```
controllers/
├── HomeController.java          // Login, Home page
├── AdminController.java         // Admin dashboard
├── StudentController.java       // Student dashboard
├── CourseController.java        // Course management
├── EnrollmentController.java    // Enrollment management
├── AttendanceController.java    // Attendance tracking
├── PaymentController.java       // Payment processing
└── NotificationController.java  // Notification management
```

🎨 Phase 7: Frontend with Thymeleaf

Step 7.1: Layout Structure

```
templates/
├── layout/
│   ├── header.html
│   ├── footer.html
│   └── sidebar.html
├── login.html
├── admin/
│   ├── dashboard.html
│   ├── students.html
│   ├── courses.html
│   ├── enrollments.html
│   ├── attendance.html
│   ├── payments.html
│   └── notifications.html
└── student/
    ├── dashboard.html
    ├── my-courses.html
    ├── my-attendance.html
    ├── my-payments.html
    └── notifications.html
```

Step 7.2: Bootstrap 5

- Responsive design
- Forms, tables, cards
- Navigation bars
- Modals for add/edit operations

📝 Phase 8: Feature Implementation

Priority Order:

1. ☑ User authentication (login/logout)
2. ☑ Admin dashboard with statistics
3. ☑ Student CRUD operations
4. ☑ Course CRUD operations
5. ☑ Student enrollment in courses
6. ☑ View enrolled students per course
7. ☑ Student dashboard showing enrolled courses
8. ☑ Attendance marking by admin
9. ☑ Attendance viewing by student
10. ☑ Attendance reports
11. ☑ Payment recording
12. ☑ Payment history
13. ☑ Fee status tracking
14. ☑ Notification system
15. ☑ Mark notifications as read
16. ☑ Targeted notifications (by role)

🧪 Phase 9: Testing & Validation

1. **Validation**: Validation annotations
2. **Error handling**: Custom error pages
3. **Testing**: Manual testing of all features
4. **Data integrity**: Test cascade operations
5. **Security testing**: Verify access controls

🚀 Phase 10: Deployment Preparation

1. Initial data (roles, admin user)
2. Production database configuration
3. Package as JAR file
4. Documentation (README, API docs)

📊 Development Checklist

Must-Have Features:

- [ ] User login/logout
- [ ] Role-based access (Admin/Student)
- [ ] Student management (Add/Edit/Delete/View)
- [ ] Course management (Add/Edit/Delete/View)
- [ ] Course enrollment
- [ ] Attendance tracking
- [ ] Payment recording
- [ ] Dashboard for both roles

- [ ] Notifications

Nice-to-Have (Future Enhancements):

- [ ] Email notifications
- [ ] PDF report generation
- [ ] Bulk attendance upload (CSV)
- [ ] Payment gateway integration
- [ ] Student profile picture upload
- [ ] Advanced search and filters
- [ ] Analytics and charts
