Student Course Management System - Complete Workflow

🎯 System Overview

This system manages students, courses, enrollments, and academic analytics with three user roles:

- **ADMIN**: Full system access
- **TEACHER**: Manage students, courses, grades
- **STUDENT**: View their own data

📋 WORKFLOW BREAKDOWN

Phase 1: SYSTEM SETUP (One-Time)

Step 1: Register Admin User

```
POST /api/auth/register
{
  "username": "admin",
  "password": "admin123",
  "email": "admin@school.edu",
  "role": "ADMIN"
}

Response:
{
  "token": "eyJhbGc...",
  "username": "admin",
  "role": "ADMIN",
  "message": "User registered successfully"
}
```

What happens:

- System creates admin account
- Password is encrypted with BCrypt
- JWT token is generated
- Admin can now access all features

Phase 2: CREATE BASIC DATA (Admin/Teacher)

Step 2: Login as Admin

```
POST /api/auth/login
{
  "username": "admin",
  "password": "admin123"
}

Response: Token to use in all future requests
```

Step 3: Create Students

```
POST /students
Authorization: Bearer {admin_token}
{
  "name": "Alice Johnson",
  "email": "alice@school.edu"
}

Response: Student with ID 1
```

Repeat for multiple students

Step 4: Create Courses

```
POST /courses
Authorization: Bearer {admin_token}
{
  "courseName": "Data Structures",
  "credits": 4
}

Response: Course with ID 1
```

Repeat for multiple courses

Step 5: Create Teacher Accounts

```
POST /api/auth/register
{
  "username": "teacher1",
  "password": "teacher123",
  "email": "teacher@school.edu",
  "role": "TEACHER"
}
```

Step 6: Create Student User Accounts

```
POST /api/auth/register
{
  "username": "alice_user",
  "password": "password123",
  "email": "alice@school.edu",
  "role": "STUDENT"
}
```

Now the system is ready for operation!

Phase 3: ENROLLMENT PROCESS (Teacher/Admin)

Step 7: Enroll Students in Courses

```
POST /enrollments
Authorization: Bearer {admin_token}
{
  "student": {"id": 1},
  "course": {"courseId": 1}
}

Response: Enrollment created with status "ACTIVE"
```

What happens:

- System checks student exists
- System checks course exists
- System checks for duplicate enrollment
- Creates enrollment with current date
- Sets status to ACTIVE

Phase 4: TEACHING & GRADING (Teacher)

Step 8: View Course Enrollments

```
GET /enrollments/course/1
Authorization: Bearer {teacher_token}

Response: List of all students enrolled in course 1
```

Step 9: Update Student Grade

```
PUT /enrollments/1/grade?grade=A
Authorization: Bearer {teacher_token}

Response: Enrollment updated with grade A and status COMPLETED
```

Valid grades:

- A_PLUS, A, A_MINUS
- B_PLUS, B, B_MINUS
- C_PLUS, C, C_MINUS
- D_PLUS, D
- F

What happens:

- Grade is recorded
- Status changes to COMPLETED
- GPA automatically recalculates

Phase 5: ANALYTICS & REPORTS (Admin/Teacher)

Step 10: View Dashboard Statistics

```
GET /api/analytics/dashboard
Authorization: Bearer {admin_token}

Response:
{
  "totalStudents": 10,
  "totalCourses": 10,
  "totalEnrollments": 20,
  "activeEnrollments": 8,
  "averageGPA": 3.2,
  "studentsOnDeansList": 2,
  "studentsOnProbation": 1
}
```

Step 11: View All Students' GPAs

```
GET /api/analytics/students/gpa
Authorization: Bearer {admin_token}

Response: List of all students sorted by GPA (highest first)
```

Step 12: View Dean's List

```
GET /api/analytics/deans-list
Authorization: Bearer {admin_token}

Response: Students with GPA ≥ 3.5
```

Step 13: View Course Analytics

```
GET /api/analytics/courses/1/analytics
Authorization: Bearer {teacher_token}

Response:
{
  "courseId": 1,
  "courseName": "Data Structures",
  "totalEnrolled": 15,
  "averageGrade": 3.2,
  "gradeDistribution": {
    "A": 5,
    "B": 7,
    "C": 2,
    "D": 1
  },
  "passRate": 93,
  "dropRate": 0
}
```

Phase 6: STUDENT SELF-SERVICE (Student)

Step 14: Student Login

```
POST /api/auth/login
{
  "username": "alice_user",
  "password": "password123"
}
```

Step 15: View Own Enrollments

```
GET /enrollments/student/1
Authorization: Bearer {student_token}

Response: All courses student is enrolled in
```

Step 16: View Own GPA

```
GET /api/analytics/students/1/gpa
Authorization: Bearer {student_token}

Response:
{
  "studentId": 1,
  "studentName": "Alice Johnson",
  "gpa": 3.85,
  "totalCredits": 12,
  "completedCourses": 3,
  "academicStanding": "Dean's List (High Honors)"
}
```

Step 17: View Transcript

```
GET /api/analytics/students/1/transcript
Authorization: Bearer {student_token}

Response: Complete academic transcript with all courses and grades
```

Phase 7: ADMINISTRATIVE TASKS (Admin Only)

Step 18: View Probation List

```
GET /api/analytics/probation
Authorization: Bearer {admin_token}

Response: Students with GPA < 2.0 (need intervention)
```

Step 19: Update Student Information

```
PUT /students/1
Authorization: Bearer {admin_token}
{
  "name": "Alice Johnson-Smith",
  "email": "alice.smith@school.edu"
}
```

Step 20: Drop/Withdraw Enrollment

```
PUT /enrollments/5/status?status=WITHDRAWN
Authorization: Bearer {admin_token}

Response: Enrollment status updated to WITHDRAWN
```

🔄 TYPICAL SEMESTER WORKFLOW

Beginning of Semester:

1. Admin creates new courses
2. Admin/Teacher creates student accounts
3. Students enroll in courses (status: ACTIVE)

During Semester:

4. Teachers view their course enrollments
5. Students view their enrolled courses
6. Students may withdraw (status: WITHDRAWN)

End of Semester:

7. Teachers update grades for all enrollments
8. System automatically:
   - Calculates GPAs
   - Determines academic standing
   - Updates statistics
9. Admin generates reports:
   - Dean's List
   - Probation list
   - Course analytics

New Semester:

10. Repeat process with new courses

📊 DATA FLOW

```
1. AUTHENTICATION
   User Login → JWT Token → All Future Requests

2. STUDENT CREATION
   POST /students → Database → Student Record

3. COURSE CREATION
   POST /courses → Database → Course Record

4. ENROLLMENT
   POST /enrollments → Validate Student & Course → Create Enrollment

5. GRADING
   PUT /enrollments/{id}/grade → Update Grade → Trigger GPA Calculation

6. ANALYTICS
   GET /api/analytics/* → Query Database → Calculate Metrics → Return Results
```

🎓 REAL-WORLD EXAMPLE

Scenario: Alice's Academic Journey

1. **Day 1**: Admin creates Alice's student record
2. **Day 2**: Alice registers her user account (alice_user/password123)
3. **Week 1**: Teacher enrolls Alice in "Data Structures" course
4. **Week 1**: Alice logs in, sees she's enrolled in Data Structures (status: ACTIVE)
5. **Semester**: Alice attends classes (tracked outside system)
6. **Week 15**: Teacher grades Alice with an "A"
7. **Week 15**: System automatically:
   - Changes enrollment status to COMPLETED
   - Calculates Alice's GPA: 4.0
   - Sets academic standing: "Dean's List (High Honors)"
8. **Week 16**: Alice logs in, views transcript, sees GPA 4.0
9. **Week 16**: Admin generates Dean's List, Alice appears
10. **Next Semester**: Alice enrolls in new courses, process repeats

---

🔐 SECURITY & ACCESS CONTROL

Endpoint	ADMIN	TEACHER	STUDENT
`POST /students`	✅	✅	❌
`GET /students`	✅	✅	❌
`POST /courses`	✅	✅	❌
`POST /enrollments`	✅	✅	✅
`PUT /enrollments/{id}/grade`	✅	✅	❌
`GET /api/analytics/dashboard`	✅	✅	❌
`GET /api/analytics/students/{id}/gpa`	✅	✅	✅ (own)
`GET /api/analytics/probation`	✅	❌	❌

⚡ QUICK TEST WORKFLOW

bash

```
# 1. Register Admin
curl -X POST http://localhost:8080/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123","email":"admin@school.edu","role":"ADMIN"}'

# 2. Login (save token)
TOKEN=$(curl -s -X POST http://localhost:8080/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}' | jq -r '.token')

# 3. Create Student
curl -X POST http://localhost:8080/students \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"name":"Alice","email":"alice@school.edu"}'

# 4. Create Course
curl -X POST http://localhost:8080/courses \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"courseName":"Data Structures","credits":4}'

# 5. Enroll Student
curl -X POST http://localhost:8080/enrollments \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"student":{"id":1},"course":{"courseId":1}}'

# 6. Give Grade
curl -X PUT "http://localhost:8080/enrollments/1/grade?grade=A" \
  -H "Authorization: Bearer $TOKEN"

# 7. View Dashboard
curl http://localhost:8080/api/analytics/dashboard \
  -H "Authorization: Bearer $TOKEN"
```

🎯 KEY FEATURES IN ACTION

1. **Automatic GPA Calculation**: No manual calculation needed
2. **Academic Standing**: System determines Dean's List/Probation

3. **Duplicate Prevention**: Can't enroll same student in same course twice
4. **Role-Based Security**: Each role sees only what they should
5. **Real-time Analytics**: Dashboard updates automatically
6. **Grade Distribution**: Teachers see how class performs
7. **Transcript Generation**: Complete academic history on demand

This is your complete end-to-end workflow! 🎓