Learning Management System (LMS)

- SHOBHITHA BHAT

Minimum Project Requirements

1. **Java Development Kit (JDK)**: Version 17 or higher.
2. **Apache Maven**: For project management and build automation.
3. **Integrated Development Environment (IDE)**: IntelliJ IDEA, Eclipse, or any Java-supporting IDE.
4. **Database**: MySQL or PostgreSQL.

Project Structure & Key Modules

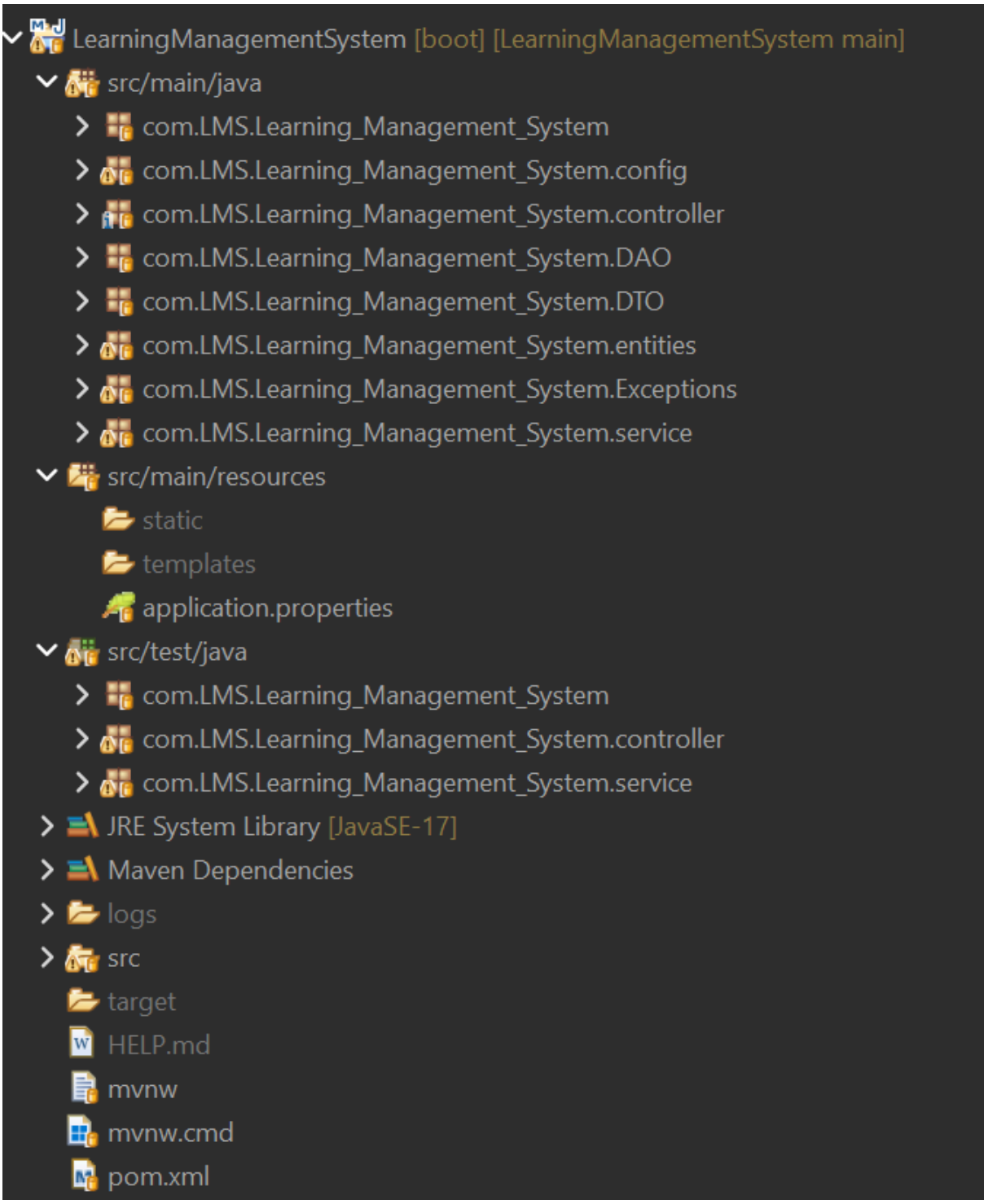
1. Core Packages

These are the main functional modules of your application:

- **`com.LMS.Learning_Management_System`**
  - Base package; contains the main application class to bootstrap the Spring Boot app.
- **`com.LMS.Learning_Management_System.config`**
  - Handles configuration settings
- **`com.LMS.Learning_Management_System.controller`**
  - Handles incoming HTTP requests and maps them to services.
- **`com.LMS.Learning_Management_System.service`**
  - Contains business logic.
- **`com.LMS.Learning_Management_System.DAO`**
  - Data Access Objects; interacts with the database.
- **`com.LMS.Learning_Management_System.DTO`**
  - Data Transfer Objects; for safely sending data between layers.
- **`com.LMS.Learning_Management_System.entities`**
  - Entity classes representing database tables.
- **`com.LMS.Learning_Management_System.Exceptions`**
  - Custom exceptions for error handling.

2. Resources

- **`src/main/resources/static`** – Contains static files like CSS, JS, images.
- **`src/main/resources/templates`** – Contains Thymeleaf or HTML templates for frontend views.
- **`application.properties`** – Central configuration file (DB, server port, logging, etc.).



3. Testing Modules

- `src/test/java` – Contains unit and integration tests.
  - **controller tests** – for endpoints.
  - **service tests** – for business logic.

4. Project Utilities

- **`pom.xml`** – Maven dependencies and build configurations.
- **`mvnw` & `mvnw.cmd`** – Maven wrapper for consistent build across environments.
- **`logs/`** – Folder for application logs.

Controller (`com.LMS.Learning_Management_System.controller`)

Handles incoming HTTP requests and forwards them to service classes.

- **CourseController.java** – CRUD operations and course-related endpoints.
- **EnrollmentController.java** – Handles student enrollments in courses.
- **InstructorController.java** – CRUD operations for instructors.
- **ReportController.java** – Provides reporting endpoints
- **StudentController.java** – CRUD operations for students.
- **ReportController.java** – Handles reporting functionality for the LMS. Provides aggregated and structured data about students, instructors, and courses, instead of just raw entity data.

DAO / Repository (`com.LMS.Learning_Management_System.DAO`)

Handles database operations using Spring Data JPA repositories.

- **CourseRepo.java** – Repository for `Course` entity.
- **EnrollmentRepo.java** – Repository for `Enrollment` entity.
- **InstructorRepo.java** – Repository for `Instructor` entity.
- **StudentRepo.java** – Repository for `Student` entity.

Entities (`com.LMS.Learning_Management_System.entities`)

Represents the database tables.

- **Course.java** – Course entity.
- **EnrollmentId.java** – Composite key for `Enrollments`.
- **Enrollments.java** – Enrollment entity linking students and courses.
- **Instructor.java** – Instructor entity.
- **Student.java** – Student entity.

Exceptions (`com.LMS.Learning_Management_System.Exceptions`)

Handles custom exceptions and global error handling.

- **CourseNotFoundException.java** – Thrown when a course is not found.
- **EnrollmentNotFoundException.java** – Thrown when enrollment is not found.
- **ErrorResponseEntity.java** – Standard error response structure.

- **GlobalExceptionHandler.java** – Handles exceptions globally using `@ControllerAdvice`.
- **InstructorNotFoundException.java** – Thrown when instructor is not found.
- **InvalidCourseException.java** – For invalid course input.
- **InvalidEnrollmentException.java** – For invalid enrollment input.
- **InvalidInstructorException.java** – For invalid instructor input.
- **InvalidStudentException.java** – For invalid student input.
- **StudentNotFoundException.java** – Thrown when student is not found.

Service (`com.LMS.Learning_Management_System.service`)

Contains business logic for each module.

- **CourseService.java** – Business logic for course management.
- **EnrollmentService.java** – Handles enrollment logic.
- **InstructorService.java** – Business logic for instructor operations.
- **ReportService.java** – Logic to generate reports.
- **StudentService.java** – Business logic for student operations.

Login & Authentication

Username: user
Password: lmsPortal
For the current prototype, the system uses a single username and password for all users. This was done to keep the implementation simple and allow focus on building the core LMS functionalities (course management, enrollments, instructors). However, this approach is not secure, as it does not distinguish between different roles. In future iterations, unique logins and role-based access will be introduced using Spring Security with encrypted passwords.

ENDPOINTS

STUDENTCONTROLLER

HTTP Method	Endpoint	Description
POST	`/students`	Add a new student

GET	`/students`	Get all students with their enrollments
GET	`/students/{usn}`	Get a single student by USN with enrollments
PUT	`/students/{usn}`	Update student details (use DTO `UpdateStudent`)
DELETE	`/students/{usn}`	Delete a student by USN

COURSECONTROLLER

HTTP Method	Endpoint	Description
POST	/addcourse	Add a new course
GET	/courses	Get all courses with their faculty
GET	/course/{c_id}	Get a course by ID with its instructors
GET	/coursefacultylist/{c_id}	Get all instructors of a specific course
PUT	/course/{c_id}/addInstructor/{f_id}	Assign a faculty to a course
DELETE	/deleteCourse/{cid}	Delete a course by ID

INSTRUCTORCONTROLLER

HTTP Method	Endpoint	Description
POST	/addinstructor	Add a new instructor
GET	/instructors	Get all instructors with their courses
GET	/instructor/{fid}	Get a specific instructor by ID with assigned courses
PUT	/instructor/{f_id}/addCourse/{c_id}	Assign a course to an instructor
DELETE	/deleteInstructor/{fid}	Delete an instructor by ID

ENROLLMENTCONTROLLER

HTTP Method	Endpoint	Description
GET	/getenrollments/{c_id}	Get all students enrolled in a course
POST	/addenrollment/{usn}/{cid}	Enroll a student in a course
POST	/addenrollments	Enroll students in courses
PUT	/updateEnrollment	Update student marks in a course
DELETE	/deleteEnrollment/{usn}/{c_id}	Remove a student's enrollment from a course

REPORTCONTROLLER

HTTP Method	Endpoint	Description

GET	/studentReport	Get a report of all students (with details such as their enrolled courses, marks, etc. – based on DTO `StudentReport`)
GET	/instructorReport	Get a report of all instructors (with details such as the courses they teach – based on DTO `InstructorReport`)
GET	/courseReport	Get a report of all courses (with details such as assigned instructors, enrolled students – based on DTO `CourseReport`)

Git & GitHub Usage - Version Control

- The project was version-controlled using Git, with the codebase hosted on GitHub for remote access and collaboration.
- I followed the practice of frequent commits, ensuring that every small change or feature addition was recorded with a meaningful commit message.
- This approach helped maintain a clear development history and made debugging easier by identifying changes step by step.
- After each significant update, the code was pushed to GitHub, making the project accessible remotely and ensuring a reliable backup.

GITHUB REPOSITORY LINK:
https://github.com/Shobhitha-Bhat/Learning_Management_System

TESTING

1. Unit Testing (JUnit)

- The business logic inside the `service` layer was tested using JUnit.
- These tests validate the correctness of the core functionalities independent of the database and controller layers.

2. API Testing (PoStman)

- All REST endpoints were tested using Postman.
- CRUD operations for Student, Course, Instructor, and Enrollment were validated.
- Edge cases were also tested (e.g., invalid USN, duplicate enrollment, deleting a non-existent course).

- Screenshots of successful and failed API calls have been included in the appendix section.

3. Results

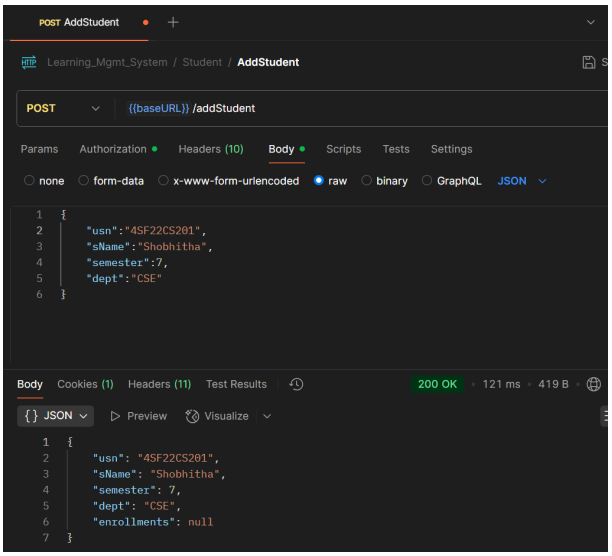
- Unit tests passed successfully .
- API tests confirmed that endpoints behave as expected, with proper error responses handled by `GlobalExceptionHandler`.

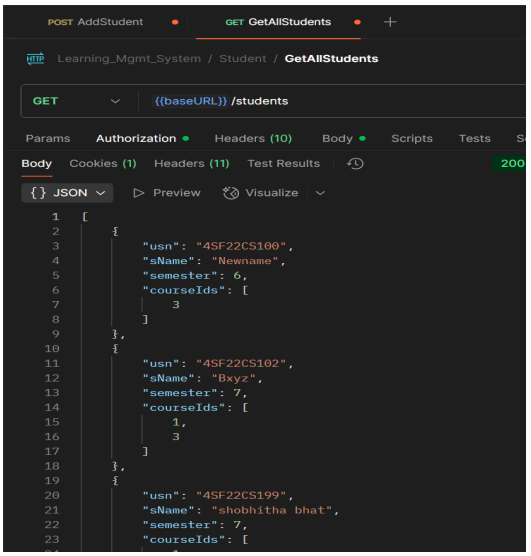
APPENDIX

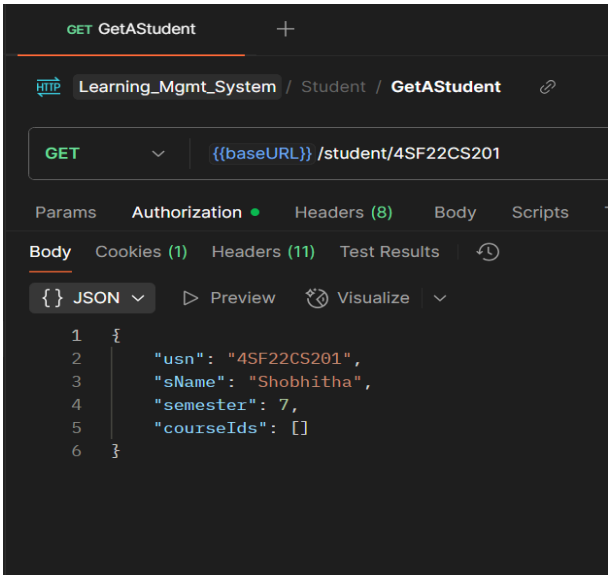
Postman API Testing Screenshots

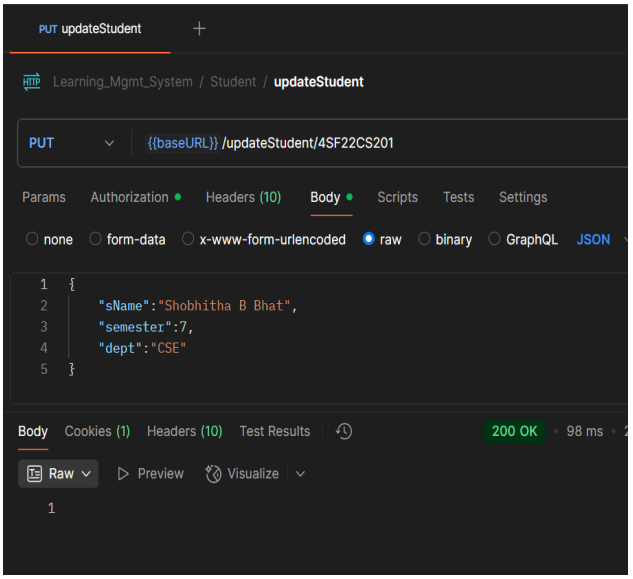
A . StudentController Endpoints

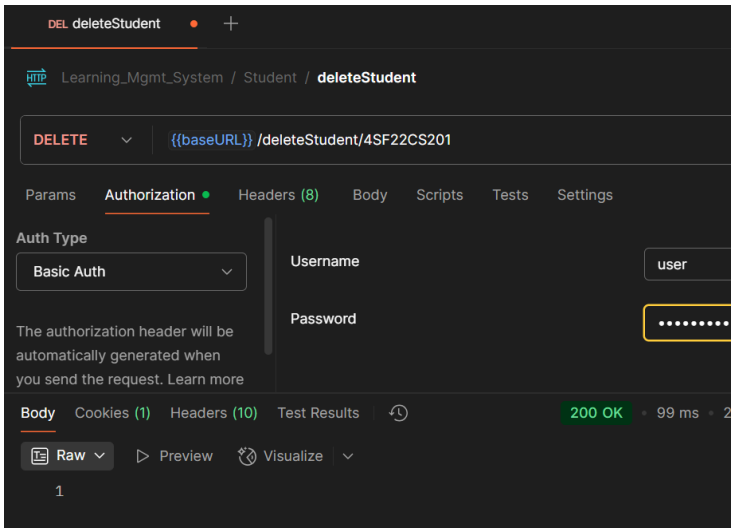
- POST `/students` → Add a new student - Figure 1
- GET `/students` → Fetch all students - Figure 2
- GET `/students/{usn}` → Fetch student by USN - Figure 3
- PUT `/students/{usn}` → Update student details – Figure 4
- DELETE `/students/{usn}` → Delete student - Figure 5

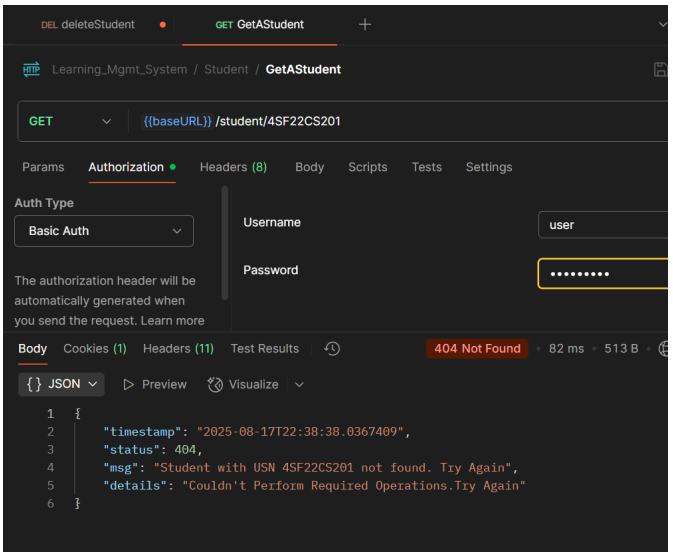






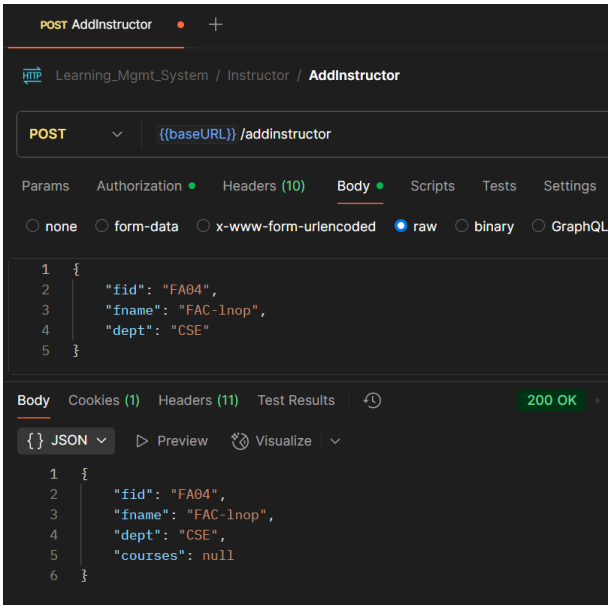


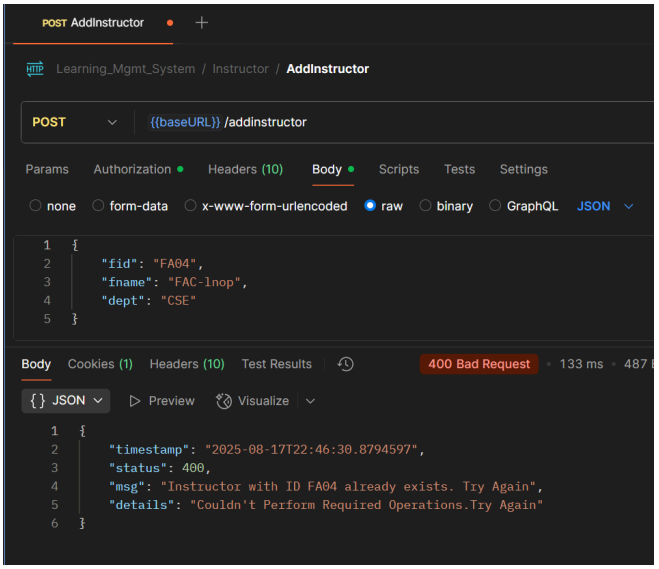




B . InstructorController Endpoints

- POST `/addinstructor` → Add a new instructor
- GET `/instructors` → Get all instructors
  GET `/instructor/{fid}` → Get instructor by ID
- PUT `/instructor/{f_id}/addCourse/{c_id}` → Assign course to instructor
- DELETE `/deleteInstructor/{fid}` → Delete instructor





POST AddInstructor | GET AllFaculties

Learning_Mgmt_System / Instructor / **AllFaculties**

GET {{baseURL}} /instructors

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings

Body | Cookies (1) | Headers (11) | Test Results | 200 OK

{} JSON | Preview | Visualize


```
[
    {
        "fId": "FA01",
        "fName": "FAC-abc",
        "dept": "CSE",
        "courses": [
            {
                "cId": 1,
                "cName": "Linux-C"
            }
        ]
    },
    {
        "fId": "FA02",
        "fName": "FAC-xyz",
        "dept": "CSE",
        "courses": [
            {
                "cId": 1,
                "cName": "Linux-C"
            },
            {
                "cId": 4,
```


POST AddInstructor | GET AllFaculties | GET GEtAFaculty | PUT AddFacultyCc | DEL deleteFacult

Learning_Mgmt_System / Instructor / **GEtAFaculty**

GET {{baseURL}} /instructor/FA04

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: user

Password: •••••••••

Body | Cookies (1) | Headers (11) | Test Results | 404 Not Found | 120 ms | 508

{} JSON | Preview | Visualize


```
{
    "timestamp": "2025-08-17T22:48:26.2394438",
    "status": 404,
    "msg": "Instructor with ID FA04 not found.Try Again",
    "details": "Couldn't Perform Required Operations.Try Again"
}
```


POST AddInstructor | GET AllFaculties | GET GEtAFaculty | PUT AddFacultyCourse

Learning_Mgmt_System / Instructor / **AddFacultyCourse**

PUT {{baseURL}} /instructor/FA04/addCourse/3

Params | Authorization | Headers (9) | Body | Scripts | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
Ctrl+Alt+P for Postbot
```

Body | Cookies (1) | Headers (10) | Test Results | 200 OK | 160 ms

Raw | Preview | Visualize

POST AddInstructor | GET AllFaculties | GET GEtAFaculty | PUT AddFacultyC

Learning_Mgmt_System / Instructor / **GEtAFaculty**

GET {{baseURL}} /instructor/FA04

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: us

Password: ••

Body | Cookies (1) | Headers (11) | Test Results | 200 OK | 133

{} JSON | Preview | Visualize


```
{
    "fId": "FA04",
    "fName": "FAC-1nop",
    "dept": "CSE",
    "courses": [
        {
            "cId": 3,
            "cName": "Java"
        }
    ]
}
```


POST AddInstructo | GET AllFaculties | GET GEtAFaculty | PUT AddFacultyCc | DEL deleteF

Learning_Mgmt_System / Instructor / **deleteFaculty**

DELETE {{baseURL}} /deleteInstructor/FA04

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: us

Password: ••

Body | Cookies (1) | Headers (10) | Test Results | 200 OK | 146

Raw | Preview | Visualize

```
1
```

POST AddInstructo | GET AllFaculties | GET GEtAFaculty | PUT AddFacultyCc | DEL deleteFacult

Learning_Mgmt_System / Instructor / **GEtAFaculty**

GET {{baseURL}} /instructor/FA04

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: user

Password: •••••••••

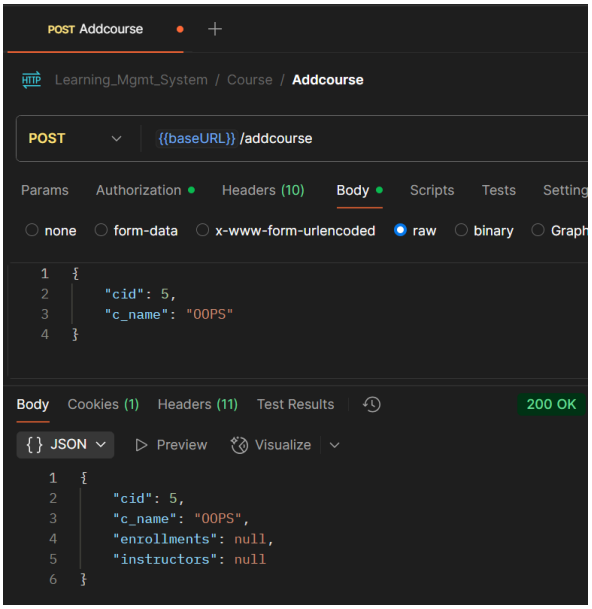
Body | Cookies (1) | Headers (11) | Test Results | 404 Not Found | 120 ms | 508

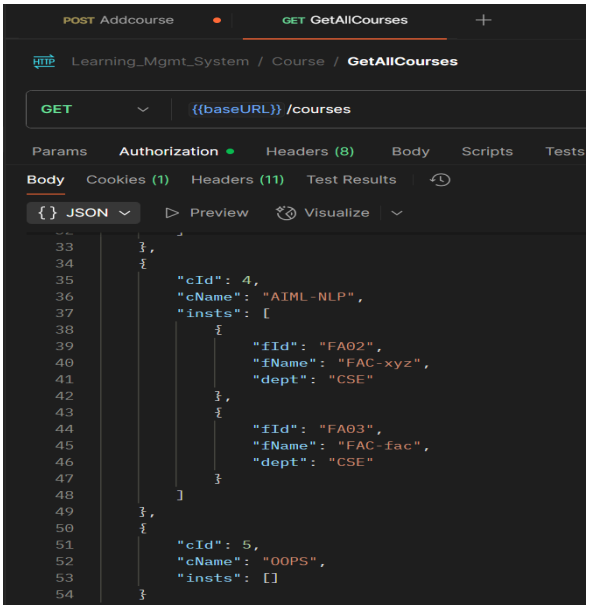
{} JSON | Preview | Visualize

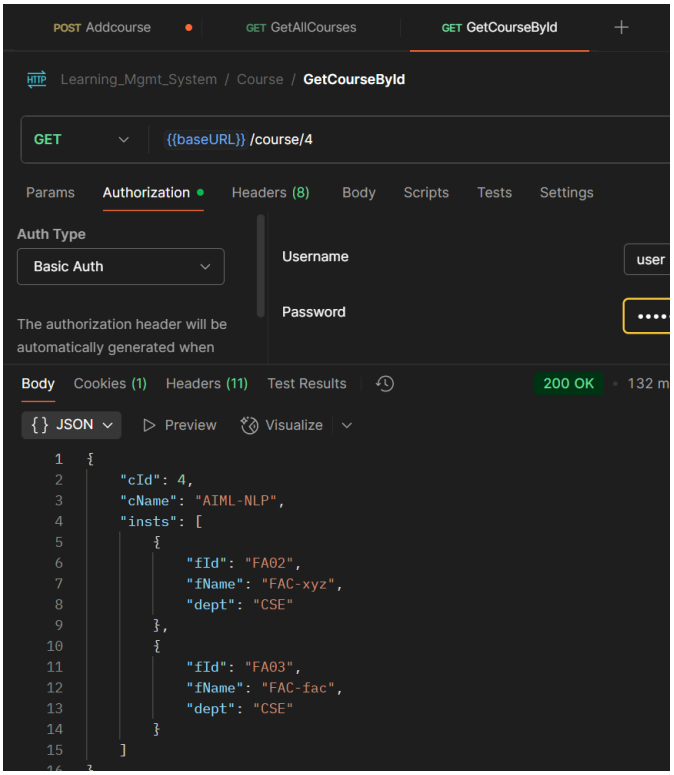

```
{
    "timestamp": "2025-08-17T22:48:26.2394438",
    "status": 404,
    "msg": "Instructor with ID FA04 not found.Try Again",
    "details": "Couldn't Perform Required Operations.Try Again"
}
```

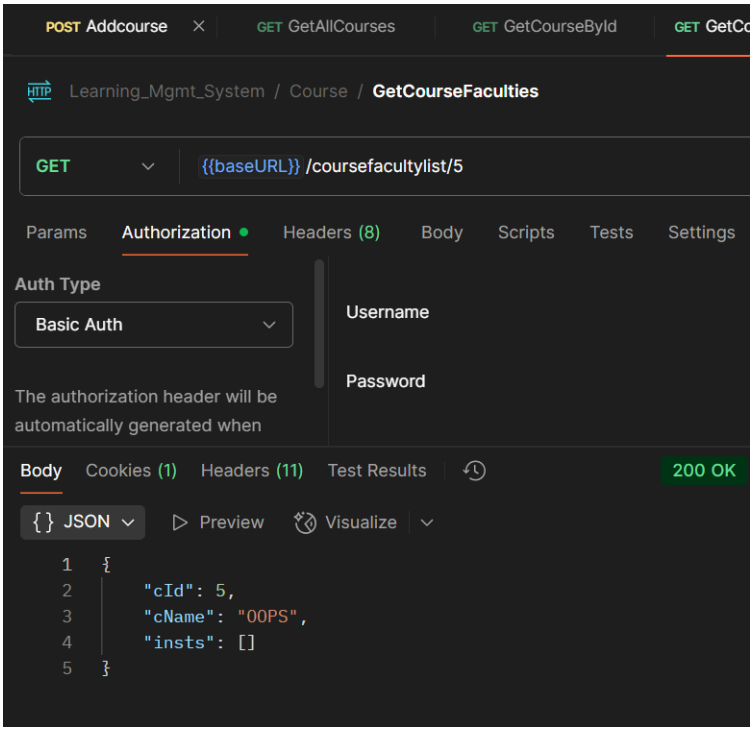
C . CourseController Endpoints

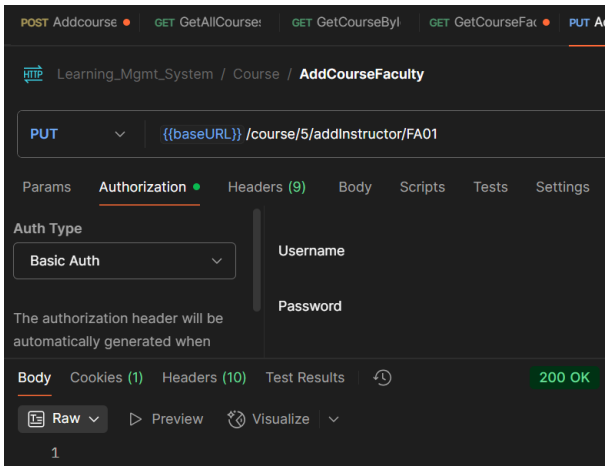
- POST `/addcourse` → Add a new course
- GET `/courses` → Get all courses
- GET `/course/{c_id}` → Get course by ID
- GET `/coursefacultylist/{c_id}` → Get instructors for a course
- PUT `/course/{c_id}/addInstructor/{f_id}` → Assign instructor to course
- DELETE `/deleteCourse/{cid}` → Delete course

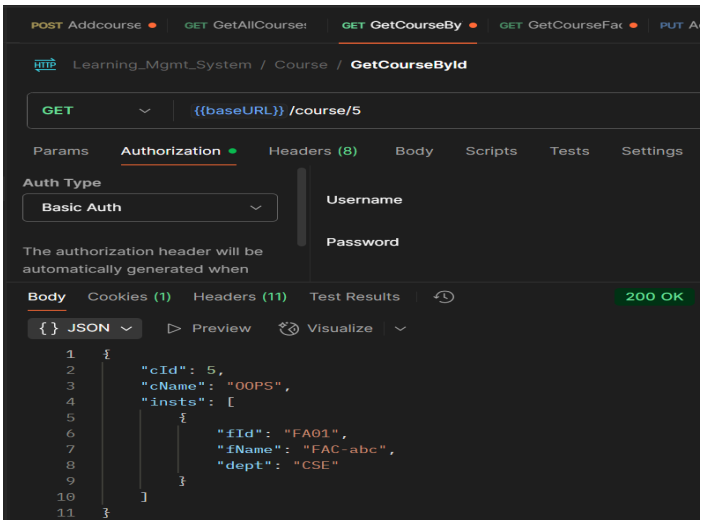


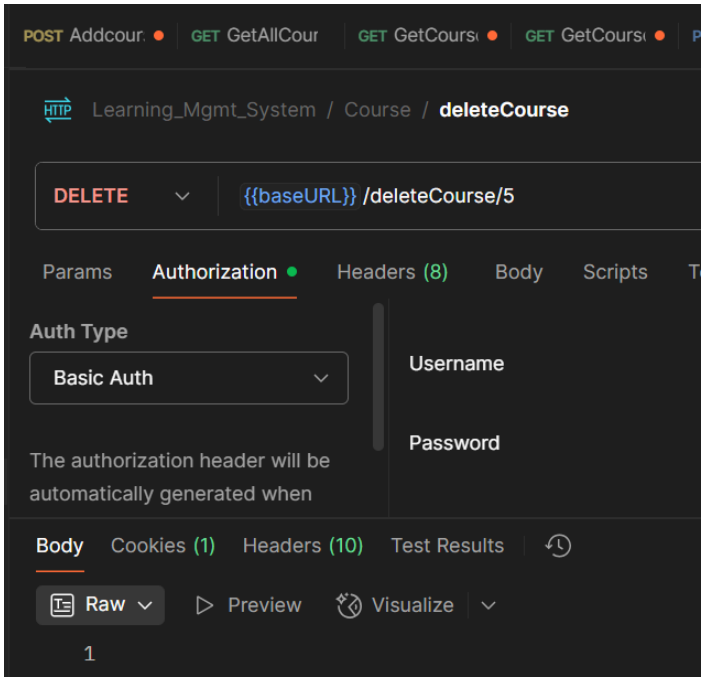


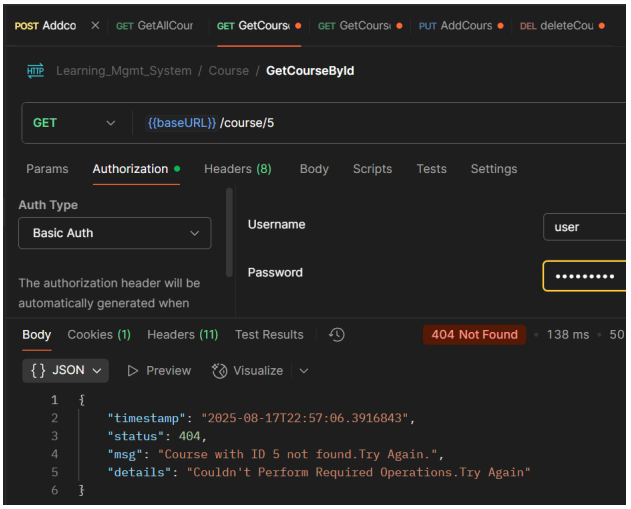












D . EnrollmentController Endpoints

- GET `/getenrollments/{c_id}` → Get all students in a course
- POST `/addenrollment/{usn}/{cid}` → Enroll a student
- POST `/addenrollments` → Enroll students to multiple courses
- PUT `/updateEnrollment` → Update student marks
  DELETE `/deleteEnrollment/{usn}/{c_id}` → Remove enrollment

GET GetEnrollmentsforaCou

Learning_Mgmt_System / Enrollments / **GetEnrollmentsforaCourse**

GET {{baseURL}} /getenrollments/3

Params Authorization Headers (8) Body Scripts Tests Se

Body Cookies (1) Headers (11) Test Results

{} JSON Preview Visualize


```
{
    "cId": 3,
    "stList": [
        {
            "usn": "4SF22CS100",
            "sName": "Newname",
            "courseMarks": 100
        },
        {
            "usn": "4SF22CS102",
            "sName": "Bxyz",
            "courseMarks": 100
        },
        {
            "usn": "4SF22CS199",
            "sName": "shobhitha bhat",
            "courseMarks": 100
        },
        {
```


GET GetEnrollmentsforaCou GET GetAllCourses POST AddEnrollments

Learning_Mgmt_System / Enrollments / **AddEnrollments**

POST {{baseURL}} /addenrollment/4SF22CS199/3

Params Authorization Headers (9) Body Scripts Tests Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: user

Password: •••••••••

Body Cookies (1) Headers (10) Test Results 400 Bad Request 80 ms 481 B

{} JSON Preview Visualize


```
{
    "timestamp": "2025-08-17T23:03:35.1792122",
    "status": 400,
    "msg": "Enrollment with Id already exists.Try Again",
    "details": "Couldn't Perform Required Operations.Try Again"
}
```


GET GetEnrollmentsfo GET GetAllCourses POST AddEnrollments POST AddEnrollmentsLi

Learning_Mgmt_System / Enrollments / **AddEnrollmentsList**

POST {{baseURL}} /addenrollments

Params Authorization Headers (10) Body Scripts Tests Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: user

Password: •••••••••

Body Cookies (1) Headers (11) Test Results 404 Not Found 83 ms 50

{} JSON Preview Visualize


```
{
    "timestamp": "2025-08-17T23:03:56.4199203",
    "status": 404,
    "msg": "Course with ID 5 not found.Try Again.",
    "details": "Couldn't Perform Required Operations.Try Again"
}
```


GET GetEnrollment GET GetAllCourse: POST AddEnrollme POST AddEnrollme DEL deleteEnro

Learning_Mgmt_System / Enrollments / **deleteEnrollment**

DELETE {{baseURL}} /deleteEnrollment/4SF22CS199/5

Params Authorization Headers (8) Body Scripts Tests Settings

Auth Type: Basic Auth

The authorization header will be automatically generated when

Username: use

Password: •••

Body Cookies (1) Headers (11) Test Results 404 Not Found 74

{} JSON Preview Visualize


```
{
    "timestamp": "2025-08-17T23:04:17.8808652",
    "status": 404,
    "msg": "Invalid Enrollment ID.Try Again",
    "details": "Couldn't Perform Required Operations.Try Again"
}
```


GET GetEnrolln GET GetAllCour POST AddEnro POST AddEnro DEL deleteEnr PUT updateEn

Learning_Mgmt_System / Enrollments / **updateEnrollment**

PUT {{baseURL}} /updateEnrollment

Params Authorization Headers (10) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON


```
{
    "stUsn":"4SF22CS199",
    "cId":3,
    "marks":100
```

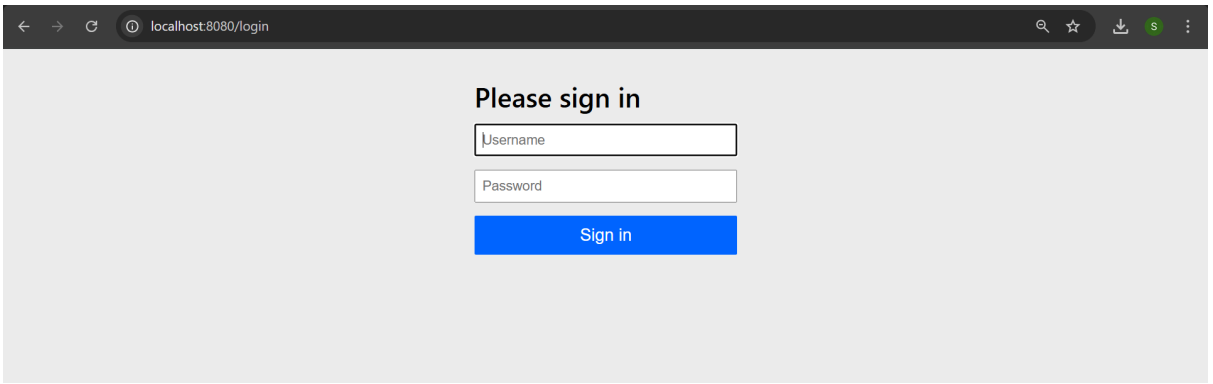

Body Cookies (1) Headers (10) Test Results 200 OK 73 ms

Raw Preview Visualize

```
1
```

E . ReportController Endpoints

- GET `/studentReport` → View student report
- GET `/instructorReport` → View instructor report
- GET `/courseReport` → View course report



localhost:8080/studentReport?continue

Pretty print


```
[
  {
    "usn": "4SF22CS100",
    "sname": "Newname",
    "dept": "ISE",
    "semester": 6,
    "average": 100,
    "screport": [
      {
        "cid": 3,
        "cname": "Java",
        "marks": 100
      }
    ]
  },
  {
    "usn": "4SF22CS102",
    "sname": "Bxyz",
    "dept": "CSE",
    "semester": 7,
    "average": 50,
    "screport": [
      {
        "cid": 1,
        "cname": "Linux-C",
        "marks": 0
      },
      {
        "cid": 3,
        "cname": "Java",
        "marks": 100
      }
    ]
  },
  {
    "usn": "4SF22CS199",
    "sname": "shobhitha bhat",
    "dept": "CSE",
    "semester": 7,
    "average": 50,
    "screport": [
      {
        "cid": 1,
        "cname": "Linux-C",
        "marks": 0
      },
      {
```

localhost:8080/courseReport

Pretty print ✔


```
[
  {
    "cid": 1,
    "cname": "Linux-C",
    "instructors": [
      {
        "fId": "FA02",
        "fName": "FAC-xyz",
        "dept": "CSE"
      },
      {
        "fId": "FA03",
        "fName": "FAC-fac",
        "dept": "CSE"
      },
      {
        "fId": "FA01",
        "fName": "FAC-abc",
        "dept": "CSE"
      }
    ],
    "stCount": 3,
    "averageMarks": 0
  },
  {
    "cid": 3,
    "cname": "Java",
    "instructors": [
      {
        "fId": "FA03",
        "fName": "FAC-fac",
        "dept": "CSE"
      }
    ],
    "stCount": 4,
    "averageMarks": 75
  },
  {
    "cid": 4,
    "cname": "AIML-NLP",
    "instructors": [
      {
        "fId": "FA02",
        "fName": "FAC-xyz",
        "dept": "CSE"
      },
      {
        "fId": "FA03"
```

localhost:8080/instructorReport

Pretty print ☑


```
[
  {
    "fid": "FA01",
    "fname": "FAC-abc",
    "dept": "CSE",
    "icreport": [
      {
        "cid": 1,
        "cname": "Linux-C",
        "avgMarks": 0,
        "studentcnt": 3
      }
    ]
  },
  {
    "fid": "FA02",
    "fname": "FAC-xyz",
    "dept": "CSE",
    "icreport": [
      {
        "cid": 1,
        "cname": "Linux-C",
        "avgMarks": 0,
        "studentcnt": 3
      },
      {
        "cid": 4,
        "cname": "AIML-NLP",
        "avgMarks": 0,
        "studentcnt": 1
      }
    ]
  },
  {
    "fid": "FA03",
    "fname": "FAC-fac",
    "dept": "CSE",
    "icreport": [
      {
        "cid": 1,
        "cname": "Linux-C",
        "avgMarks": 0,
        "studentcnt": 3
      },
      {
        "cid": 4,
        "cname": "AIML-NLP",
        "avgMarks": 0
```