Software Maintenance and Evolution

Name	ID	Group
Omar Ibrahim	20226066	S4
Mohab Khaled	20226106	S4
Shahd Kamal	20226051	S4
Zeyad Mohamed	20226158	S4

☞ GITHUB ☜

Project Description

The Learning Management System (LMS) is a web-based platform that enables instructors to create and manage courses, lessons, quizzes, and assignments, while allowing students to enroll, access materials, submit work, and receive feedback. Administrators oversee user accounts, system settings, and reporting.

System Purpose

Provide a centralized, secure environment for delivering educational content, automating enrollment and grading workflows, and sending timely notifications via in-app messages and email.

User Roles

ADMIN – full control over users, roles, courses, and system configuration
INSTRUCTOR – create and manage own courses, lessons, assessments, and view student performance
STUDENT – browse and enroll in courses, access lessons, submit assignments, take quizzes, and view grades

---

Backend Architecture

Built with Java Spring Boot exposing RESTful APIs. Data persisted in MySQL via Spring Data JPA. Security enforced with JWT-based authentication and role-based authorization filters. Application events drive notification services. Configuration (database credentials, JWT secrets) is externalized for different environments. Maven handles build lifecycle and dependencies.

Key Functional Features

- **User Management**: Sign-up, login, profile updates, password reset
- **Course & Lesson Management**: Instructors perform CRUD on courses and lessons; upload multimedia content

- **Enrollment:** Students enroll in courses; instructors view rosters
- **Attendance Tracking:** Generate and validate one-time OTP codes for session attendance
- **Assessments:**
    - Quizzes with question banks, timed attempts, automatic scoring
    - Assignments with file upload, submission tracking, and instructor feedback
- **Notifications:** Real-time in-app alerts and email notifications for deadlines, new content, grades

---

Security and Validation

Authentication via signed JWTs with one-day expiry. Endpoints protected by role checks. Model fields validated using Jakarta Validation annotations. Passwords hashed before storage. Sensitive settings kept outside source code.

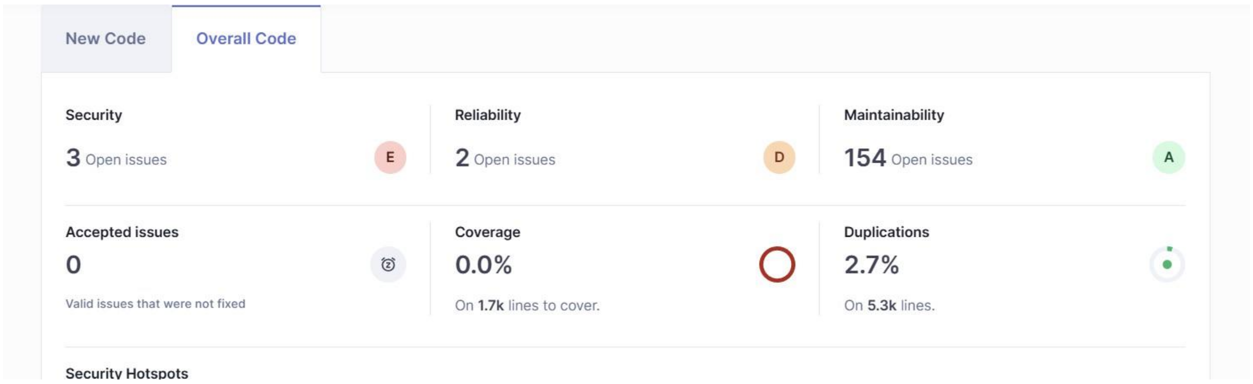
Entities Overview

User – username, email, hashed password, role
Course – title, description, associated instructor
Lesson – content items linked to a course
Enrollment – relation between student and course
Attendance – records of OTP-validated check-ins
Quiz & **Question** – definitions of quizzes and their question banks
QuizAttempt – student submissions and scores
Assignment & **Submission** – assignment definitions and student uploads
Notification – in-app/email messages triggered by events

Technology Stack

Java 17, Spring Boot, Spring Data JPA, MySQL 8, JWT security, Maven build, Gmail SMTP for email

Sonarqube analysis


New Code | Overall Code

Security
3 Open issues E

Reliability
2 Open issues D

Maintainability
154 Open issues A

Accepted issues
0
Valid issues that were not fixed

Coverage
0.0%
On 1.7k lines to cover.

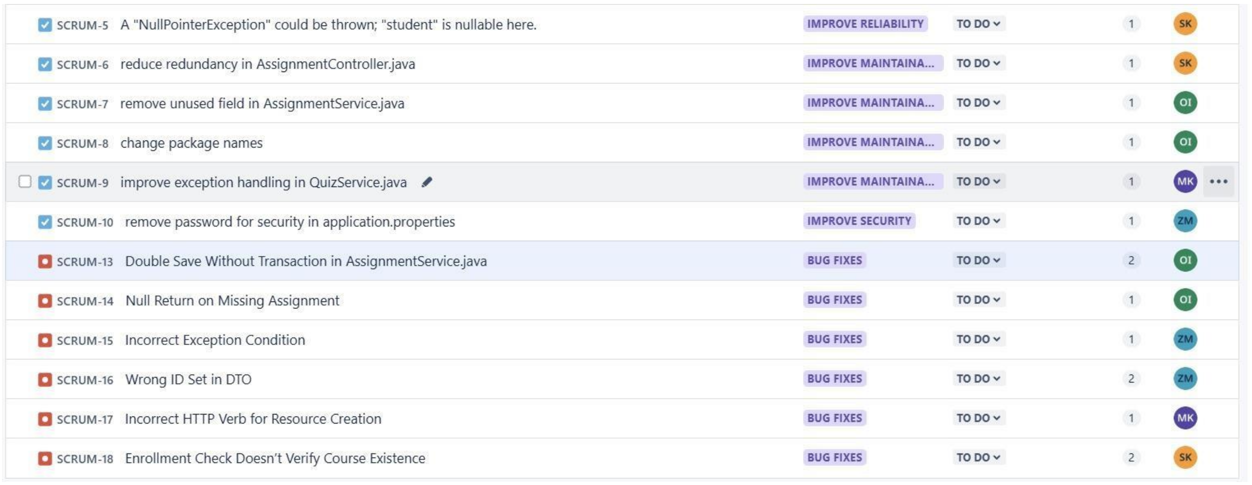
Duplications
2.7%
On 5.3k lines.

Security Hotspots

This dashboard snapshot shows the current state of our codebase as evaluated by static analysis tools. Key takeaways:

- **Security**: 3 open issues (rated E) requiring immediate attention.
- **Reliability**: 2 open issues (rated D) indicating potential runtime risks.
- **Maintainability**: 154 open issues (rated A), reflecting generally good code structure but room for cleanup.
- **Test Coverage**: 0.0% on 1.7 k lines—tests must be implemented to validate functionality.
- **Code Duplication**: 2.7% on 5.3 k lines, a low duplication rate, indicating decent reuse.

Jira tasks



SCRUM-5	A "NullPointerException" could be thrown; "student" is nullable here.	IMPROVE RELIABILITY	TO DO	1	SK
SCRUM-6	reduce redundancy in AssignmentController.java	IMPROVE MAINTAINA...	TO DO	1	SK
SCRUM-7	remove unused field in AssignmentService.java	IMPROVE MAINTAINA...	TO DO	1	OI
SCRUM-8	change package names	IMPROVE MAINTAINA...	TO DO	1	OI
SCRUM-9	improve exception handling in QuizService.java	IMPROVE MAINTAINA...	TO DO	1	MK
SCRUM-10	remove password for security in application.properties	IMPROVE SECURITY	TO DO	1	ZM
SCRUM-13	Double Save Without Transaction in AssignmentService.java	BUG FIXES	TO DO	2	OI
SCRUM-14	Null Return on Missing Assignment	BUG FIXES	TO DO	1	OI
SCRUM-15	Incorrect Exception Condition	BUG FIXES	TO DO	1	ZM
SCRUM-16	Wrong ID Set in DTO	BUG FIXES	TO DO	2	ZM
SCRUM-17	Incorrect HTTP Verb for Resource Creation	BUG FIXES	TO DO	1	MK
SCRUM-18	Enrollment Check Doesn't Verify Course Existence	BUG FIXES	TO DO	2	SK

Reliability

SCRUM-5

Summary: Potential NullPointerException due to nullable student

Description:

Audit all code paths where student is used without null checks. Add validation or wrap in Optional to prevent NPEs.

Maintainability

SCRUM-6

Summary: Eliminate redundancy in AssignmentController.java

Description:

Refactor AssignmentController by extracting repeated validation and response logic into shared helper methods or a common base controller to improve maintainability.

SCRUM-7

Summary: Remove unused field in AssignmentService.java

Description:

Locate and delete the obsolete field in AssignmentService, update any impacted tests or documentation, and confirm no residual side-effects remain.

SCRUM-8

Summary: Rename Java packages to match new conventions

Description:

Update package names project-wide (e.g. com.app.lms.course → com.app.lms.courseManagement), adjust all import statements, then rebuild and run tests to ensure everything compiles.

SCRUM-9

Summary: Strengthen exception handling in QuizService.java

Description:

Define and throw a dedicated exception instead of using a generic one, ensuring clearer error semantics and easier troubleshooting.

Security

SCRUM-10

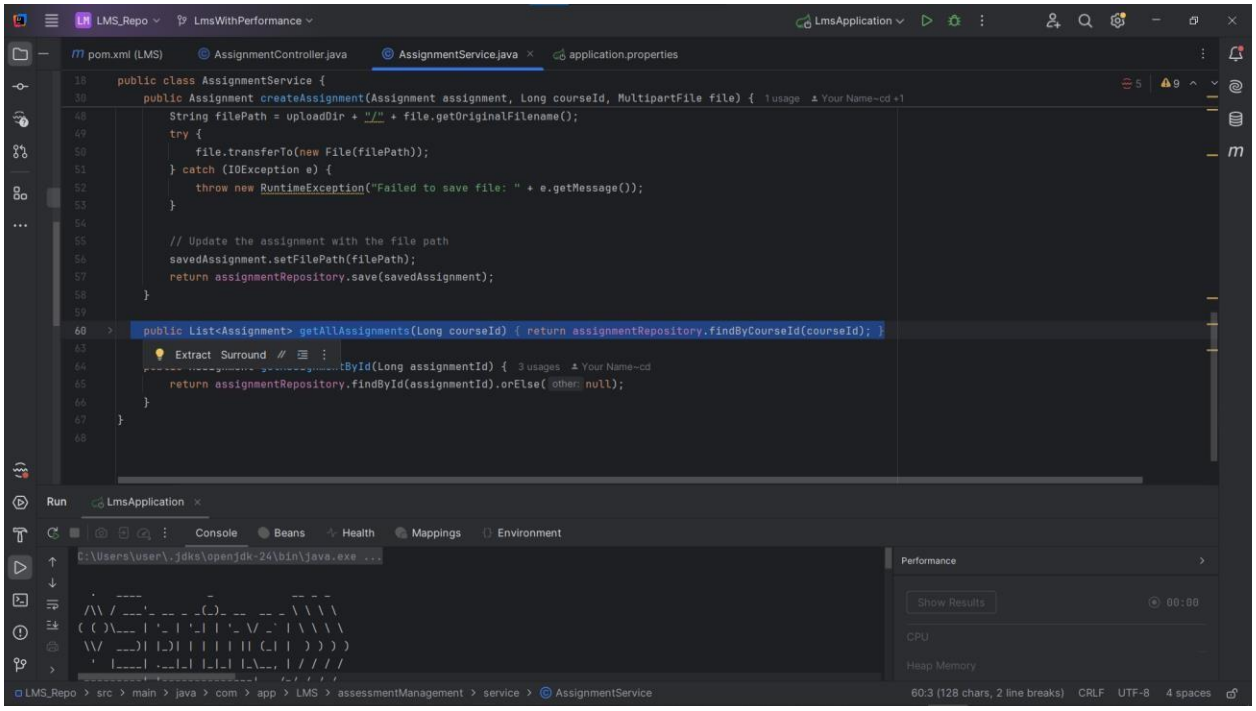
Summary: Externalize security password from application.properties

Description:

Remove hard-coded security.password from properties file, load it from an environment variable, and verify secure handling in all deployment environments.

Report: Identified Issues in AssignmentService.java

Issue 1: Double Save Without Transaction



```java
public class AssignmentService {
    public Assignment createAssignment(Assignment assignment, Long courseId, MultipartFile file) {
        String filePath = uploadDir + "/" + file.getOriginalFilename();
        try {
            file.transferTo(new File(filePath));
        } catch (IOException e) {
            throw new RuntimeException("Failed to save file: " + e.getMessage());
        }

        // Update the assignment with the file path
        savedAssignment.setFilePath(filePath);
        return assignmentRepository.save(savedAssignment);
    }

    public List<Assignment> getAllAssignments(Long courseId) { return assignmentRepository.findByCourseId(courseId); }

    public Assignment getAssignmentById(Long assignmentId) {
        return assignmentRepository.findById(assignmentId).orElse( other: null);
    }
}
```

```
C:\Users\user\.jdks\openjdk-24\bin\java.exe ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
```

Description:

The createAssignment method persists the Assignment twice:

1. First call to assignmentRepository.save(savedAssignment) to generate an ID.
2. After setting the filePath on the saved entity, a second call to assignmentRepository.save(savedAssignment).

Without a transactional boundary (@Transactional), if the second save or the file write fails, the database remains in an inconsistent state—either missing the file path or leaving an orphaned record.

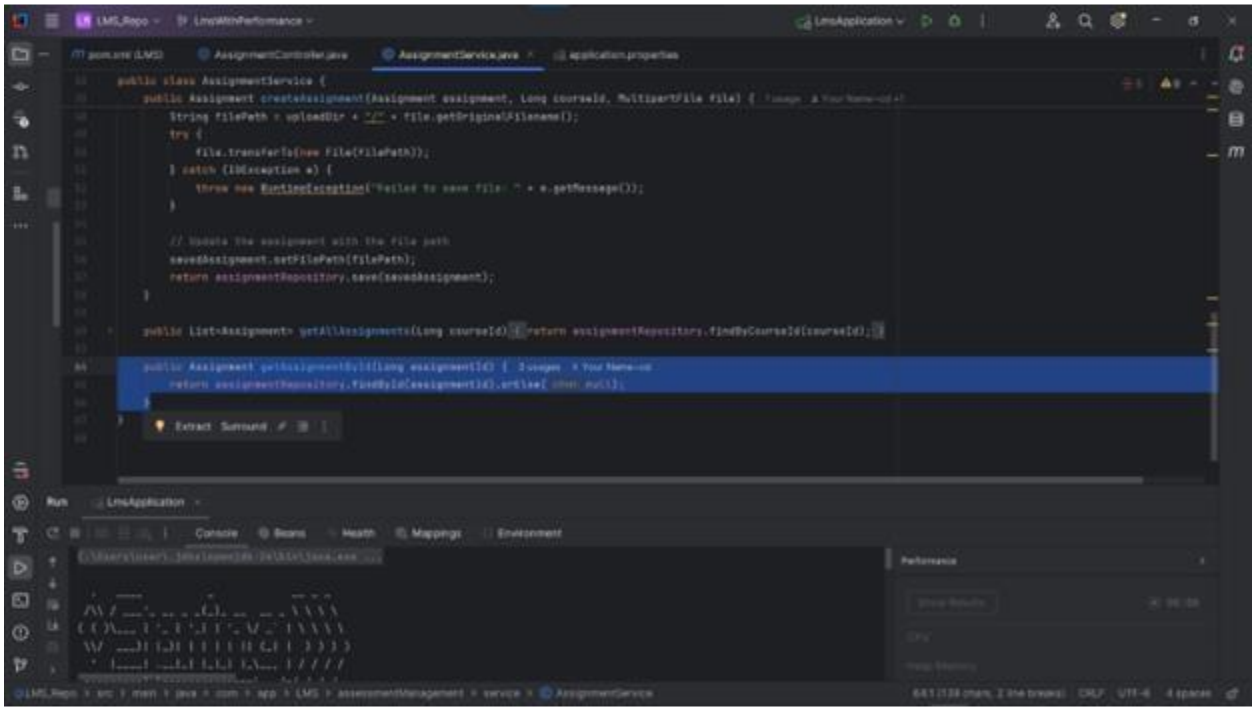
Impact:

- **Data inconsistency:** Partial updates may be committed.
- **Orphaned records:** Assignments without their associated file paths remain in the database.
- **Error recovery complexity:** Rolling back only the failed part isn't automatic.

Issue 2: Null Return on Missing Assignment

Description:

The method getAssignmentById(Long assignmentId) returns null if the repository lookup fails:



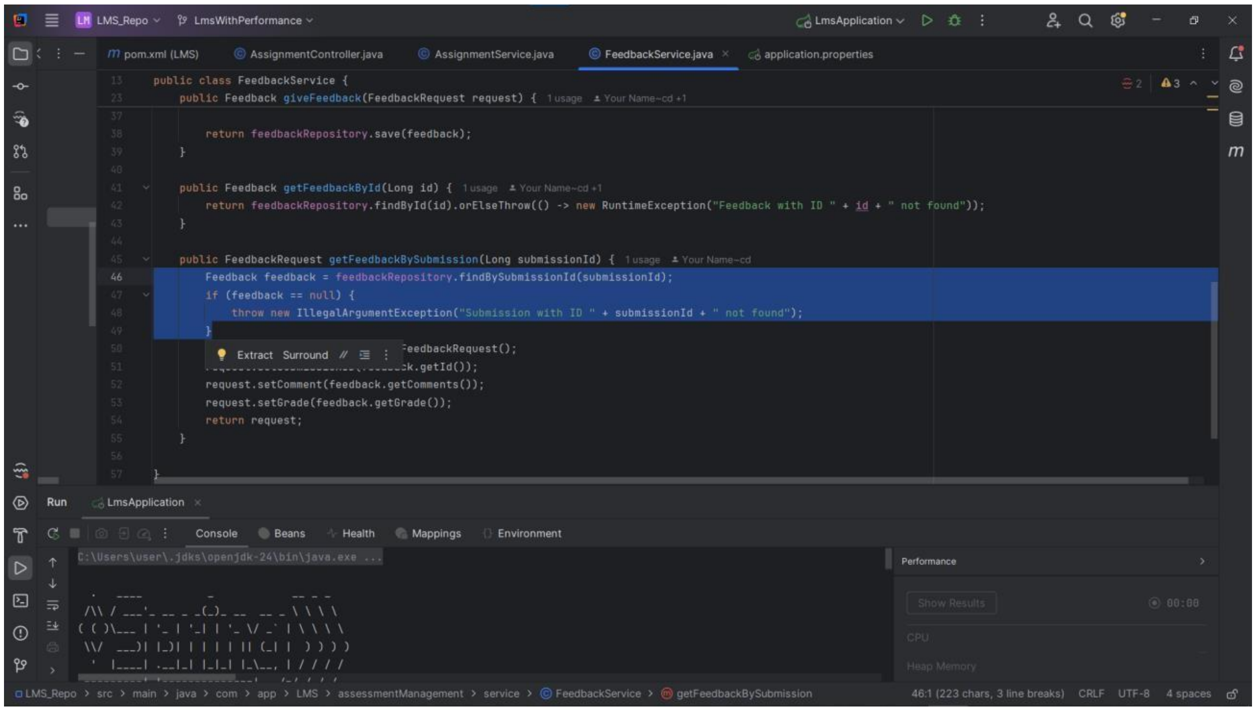
```java
public class AssignmentService {
    public Assignment createAssignment(Assignment assignment, Long courseId, MultipartFile file) {
        String filePath = uploadDir + "/" + file.getOriginalFilename();
        try {
            file.transferTo(new File(filePath));
        } catch (IOException e) {
            throw new RuntimeException("Failed to save file: " + e.getMessage());
        }

        // Update the assignment with the file path
        savedAssignment.setFilePath(filePath);
        return assignmentRepository.save(savedAssignment);
    }

    public List<Assignment> getAllAssignments(Long courseId) { return assignmentRepository.findByCourseId(courseId); }

    public Assignment getAssignmentById(Long assignmentId) {
        return assignmentRepository.findById(assignmentId).orElse( other: null);
    }
}
```

Impact:

- **Prone to NullPointerException:** Callers must remember to null-check every call.
- **Undescriptive failures:** A null return gives no context for why the assignment is missing.

Identified Issues in FeedbackService.java

Issue 3: Incorrect Exception Condition



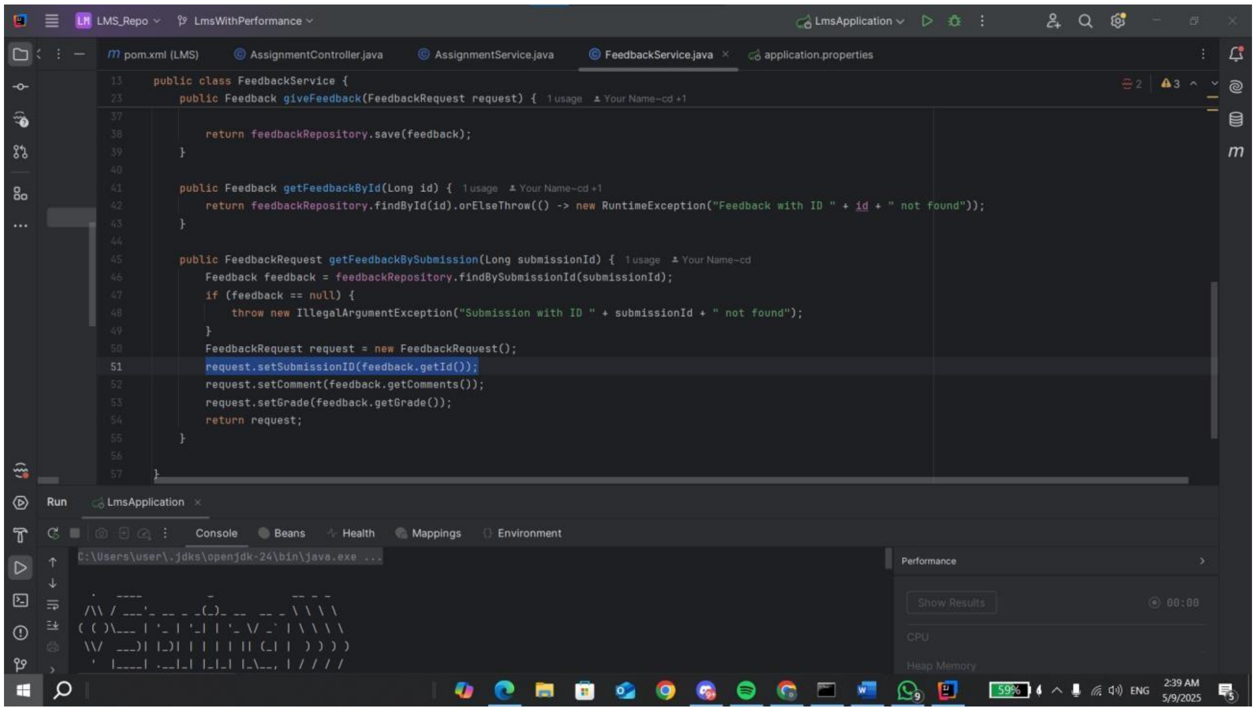
```java
public class FeedbackService {
    public Feedback giveFeedback(FeedbackRequest request) {  1 usage  Your Name~cd +1

        return feedbackRepository.save(feedback);
    }

    public Feedback getFeedbackById(Long id) {  1 usage  Your Name~cd +1
        return feedbackRepository.findById(id).orElseThrow(() -> new RuntimeException("Feedback with ID " + id + " not found"));
    }

    public FeedbackRequest getFeedbackBySubmission(Long submissionId) {  1 usage  Your Name~cd
        Feedback feedback = feedbackRepository.findBySubmissionId(submissionId);
        if (feedback == null) {
            throw new IllegalArgumentException("Submission with ID " + submissionId + " not found");
        }
                                    eedbackRequest();
                                    k.getId());
        request.setComment(feedback.getComments());
        request.setGrade(feedback.getGrade());
        return request;
    }

}
```

```
C:\Users\user\.jdks\openjdk-24\bin\java.exe ...
```

Issue: The IllegalArgumentException message refers to the submission being missing, but the lookup was for feedback. If there’s no feedback record, it may simply mean feedback wasn’t created yet—not that the submission doesn’t exist.

Impact: Misleading error message; genuine “no submission” vs. “no feedback” cases conflated.

Issue 4: Wrong ID Set in DTO



```java
public class FeedbackService {
    public Feedback giveFeedback(FeedbackRequest request) {

        return feedbackRepository.save(feedback);
    }

    public Feedback getFeedbackById(Long id) {
        return feedbackRepository.findById(id).orElseThrow(() -> new RuntimeException("Feedback with ID " + id + " not found"));
    }

    public FeedbackRequest getFeedbackBySubmission(Long submissionId) {
        Feedback feedback = feedbackRepository.findBySubmissionId(submissionId);
        if (feedback == null) {
            throw new IllegalArgumentException("Submission with ID " + submissionId + " not found");
        }
        FeedbackRequest request = new FeedbackRequest();
        request.setSubmissionID(feedback.getId());
        request.setComment(feedback.getComments());
        request.setGrade(feedback.getGrade());
        return request;
    }

}
```

```
C:\Users\user\.jdks\openjdk-24\bin\java.exe ...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
```

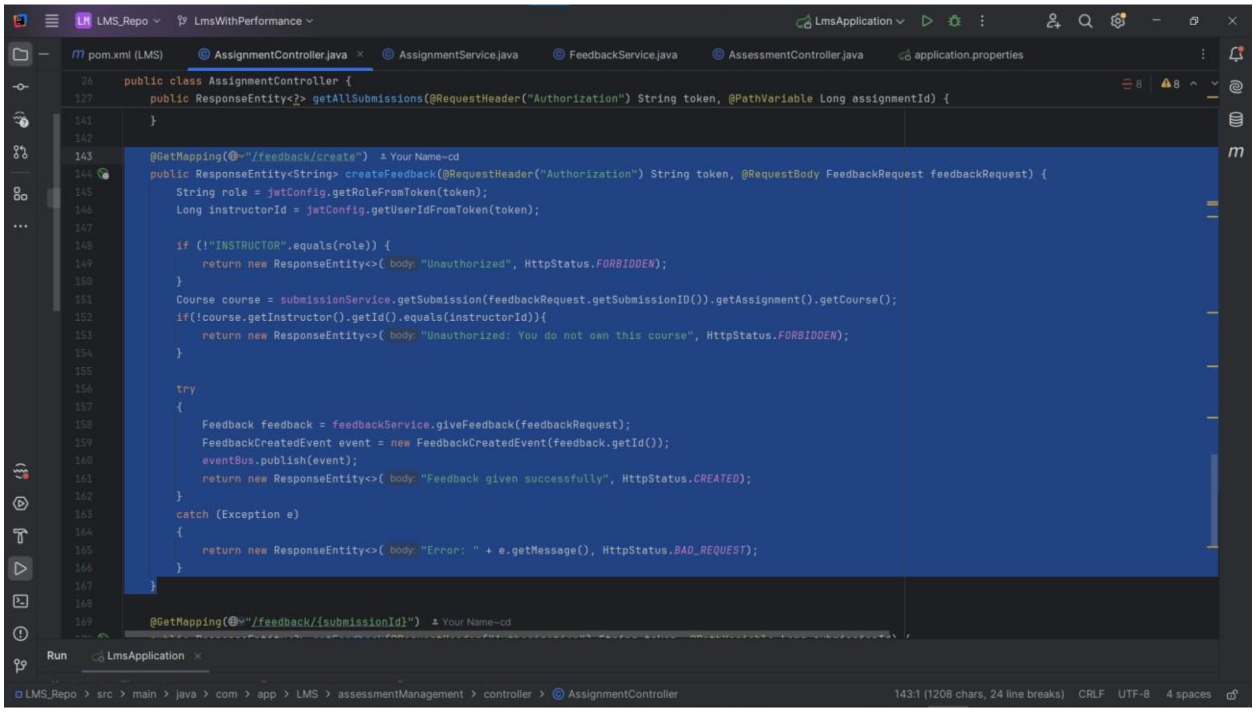
Issue: It sets the feedback's primary key (feedback.getId()) into the submissionID field, instead of the actual submissionId.

Impact: The client receives the feedback ID mislabeled as the submission ID, leading to confusion when updating or correlating records.

Identified Issue in AssignmentController#createFeedback

Issue 5: Incorrect HTTP Verb for Resource Creation

Location:



```java
public class AssignmentController {
    public ResponseEntity<?> getAllSubmissions(@RequestHeader("Authorization") String token, @PathVariable Long assignmentId) {
    }

    @GetMapping("/feedback/create")
    public ResponseEntity<String> createFeedback(@RequestHeader("Authorization") String token, @RequestBody FeedbackRequest feedbackRequest) {
        String role = jwtConfig.getRoleFromToken(token);
        Long instructorId = jwtConfig.getUserIdFromToken(token);

        if (!"INSTRUCTOR".equals(role)) {
            return new ResponseEntity<>( body: "Unauthorized", HttpStatus.FORBIDDEN);
        }
        Course course = submissionService.getSubmission(feedbackRequest.getSubmissionID()).getAssignment().getCourse();
        if(!course.getInstructor().getId().equals(instructorId)){
            return new ResponseEntity<>( body: "Unauthorized: You do not own this course", HttpStatus.FORBIDDEN);
        }

        try
        {
            Feedback feedback = feedbackService.giveFeedback(feedbackRequest);
            FeedbackCreatedEvent event = new FeedbackCreatedEvent(feedback.getId());
            eventBus.publish(event);
            return new ResponseEntity<>( body: "Feedback given successfully", HttpStatus.CREATED);
        }
        catch (Exception e)
        {
            return new ResponseEntity<>( body: "Error: " + e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @GetMapping("/feedback/{submissionId}")
```

LMS_Repo > src > main > java > com > app > LMS > assessmentManagement > controller > AssignmentController   143:1 (1208 chars, 24 line breaks)   CRLF   UTF-8   4 spaces

Description:

The createFeedback endpoint is annotated with @GetMapping, meaning it responds to HTTP GET requests. According to REST principles, GET methods must be **safe** (no side effects) and **idempotent** (multiple identical requests have the same effect). Using GET to create or modify server-side state violates these constraints and can lead to:

- Unexpected data creation when clients (or intermediaries like web crawlers) prefetch or cache GET URLs.
- Violations of HTTP caching rules.
- Security and audit issues, since GET requests are often logged and replayed differently than POST/PUT.

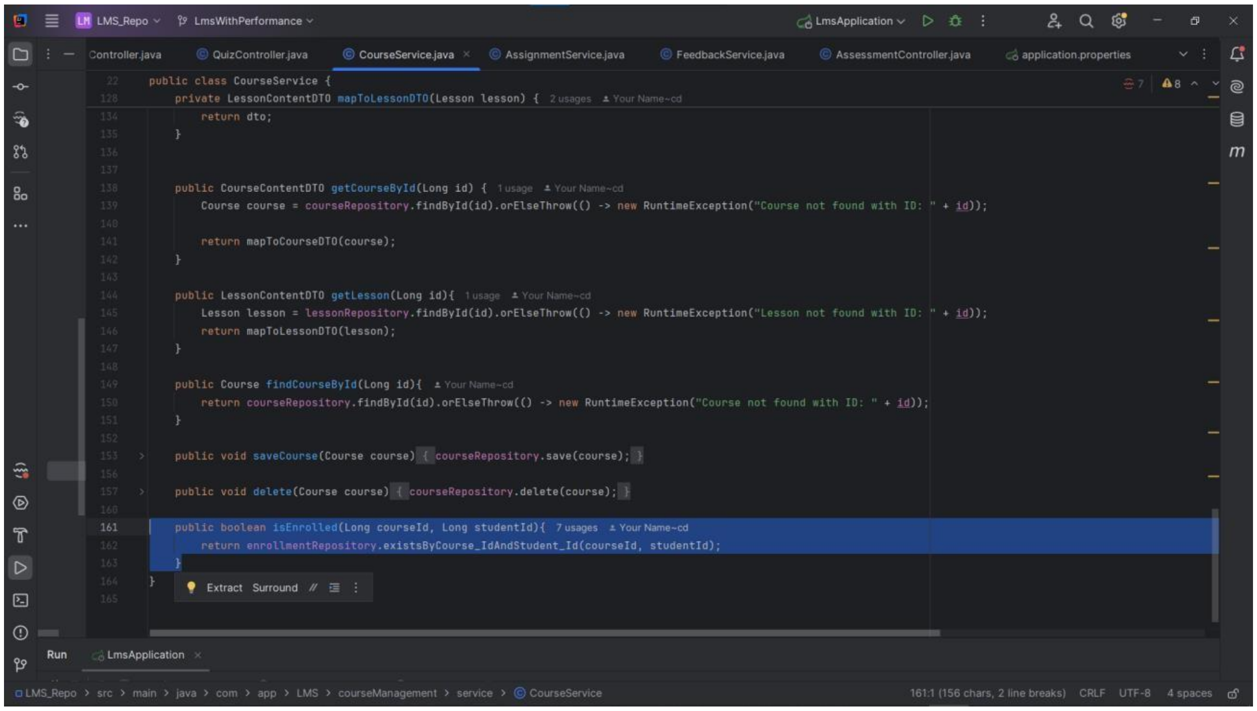
Impact:

- **Unintended Side Effects:** Clients and proxies expecting GET to be read-only may inadvertently trigger feedback creation.
- **Cache Pollution:** Responses may be cached incorrectly, causing stale or duplicated feedback records.
- **Misleading API Semantics:** Breaks client expectations and standard tooling (e.g., Swagger clients assume GET is safe).

Report: Identified Issue in CourseService#isEnrolled

Issue 6: Enrollment Check Doesn't Verify Course Existence

Location:



```java
public class CourseService {
    private LessonContentDTO mapToLessonDTO(Lesson lesson) {  2 usages  Your Name~cd
        return dto;
    }


    public CourseContentDTO getCourseById(Long id) {  1 usage  Your Name~cd
        Course course = courseRepository.findById(id).orElseThrow(() -> new RuntimeException("Course not found with ID: " + id));

        return mapToCourseDTO(course);
    }

    public LessonContentDTO getLesson(Long id){  1 usage  Your Name~cd
        Lesson lesson = lessonRepository.findById(id).orElseThrow(() -> new RuntimeException("Lesson not found with ID: " + id));
        return mapToLessonDTO(lesson);
    }

    public Course findCourseById(Long id){  Your Name~cd
        return courseRepository.findById(id).orElseThrow(() -> new RuntimeException("Course not found with ID: " + id));
    }

    public void saveCourse(Course course) { courseRepository.save(course); }

    public void delete(Course course) { courseRepository.delete(course); }

    public boolean isEnrolled(Long courseId, Long studentId){  7 usages  Your Name~cd
        return enrollmentRepository.existsByCourse_IdAndStudent_Id(courseId, studentId);
    }
}
```

Description:

The isEnrolled method directly queries the EnrollmentRepository to see if a student is enrolled in a given course ID, but it does **not** verify that the course with courseId actually exists. As a result, when clients pass an invalid or non-existent courseId, this method will return false, indistinguishable from the case where the course exists but the student is simply not enrolled.

Impact:

- **Ambiguous Results:** Consumers of the API cannot tell if false means "student not enrolled" or "course not found."
- **Potential Misleading Behavior:** Downstream logic might proceed under the assumption that the course exists, leading to further errors or misleading UI messages.
- **Error Handling Difficulty:** It complicates error reporting and makes it harder to return an accurate HTTP status (e.g., 404 vs. 200).