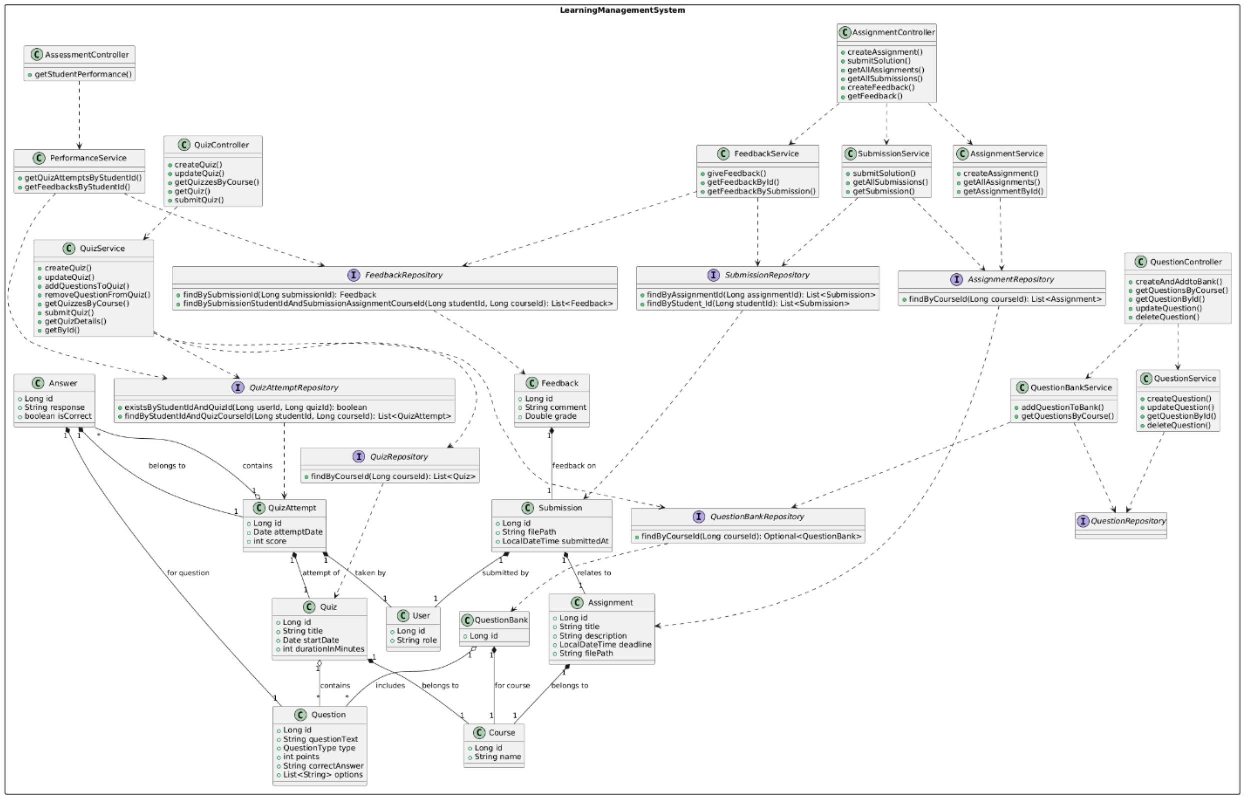
Software Maintenance and Evolution

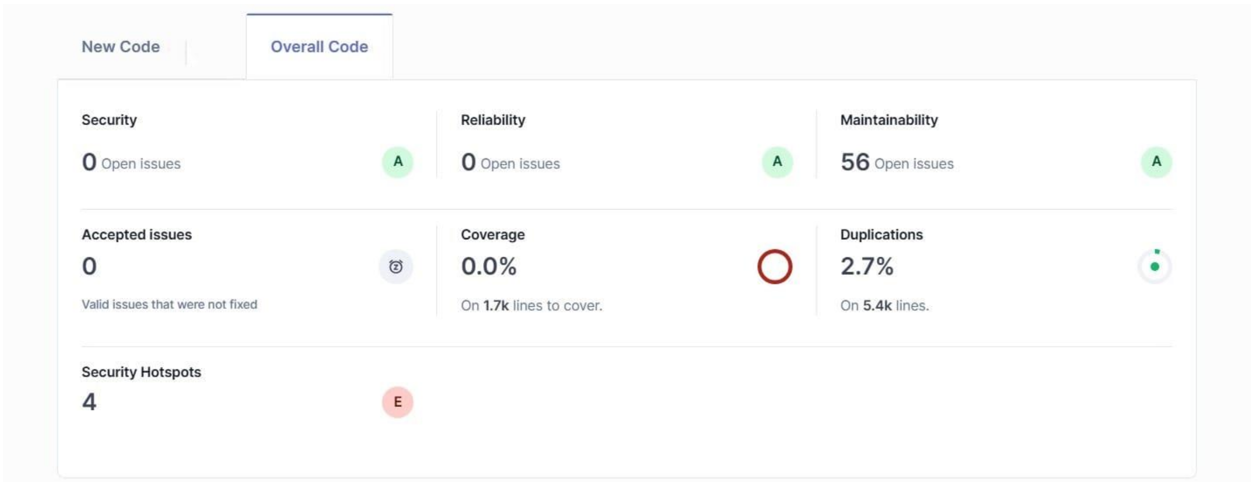
Name	ID	Group
Omar Ibrahim	20226066	S4
Mohab Khaled	20226106	S4
Shahd Kamal	20226051	S4
Zeyad Mohamed	20226158	S4

👉 GITHUB 👈

Class diagram



Sonarqube 2nd analysis after changes were implemented :



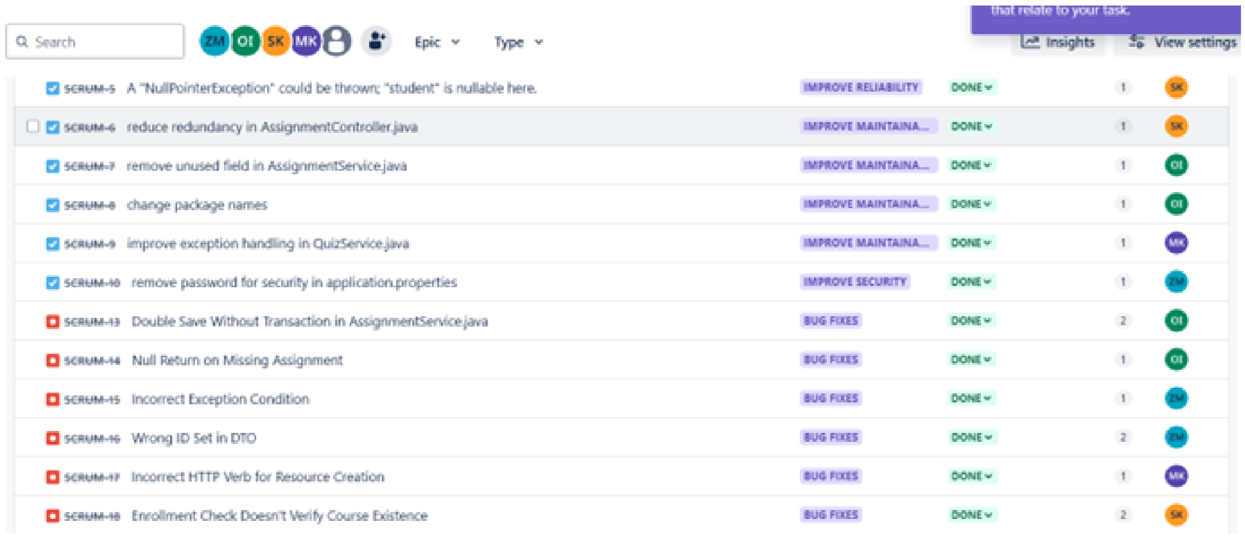
New Code | Overall Code

Security 0 Open issues A	Reliability 0 Open issues A	Maintainability 56 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 5.4k lines.
Security Hotspots 4 E		

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

- Security: from 3 down to zero security issues
- Reliability: from 2 down to zero reliability issues.
- Maintainability: from 154 open issues down to 56.
- 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 after all assigned tasks were done :


that relate to your task.

Search · ZM · OI · SK · MK · Epic · Type · Insights · View settings

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

SCRUM-5: Prevent NPE on student

Before: Student was nullable and there was no checks

After: Added a null check (if (student != null) before use.

SCRUM-6: DRY up AssignmentController

Before: methods repeated strings multiple times

After: Added ROLE_INSTRUCTOR, UNAUTHORIZED,UNAUTHORIZED_OWNERSHIP variables to reuse.

SCRUM-7: Remove Unused Field in AssignmentService

Before: private final SubmissionRepository submissionRepository;

After: Deleted the obsolete SubmissionRepository field (and related getters/setters).

SCRUM-8: Rename Packages to Conventions
Before: Packages like com.app.lms.course and com.app.lms.user didn't match the new naming scheme.

After: Refactored to com.app.lms.courseManagement, com.app.lms.userManagement, etc., and updated all imports.

SCRUM-9: Use Specific Exceptions in QuizService
Before: Quiz service used a generic runtime exceptions

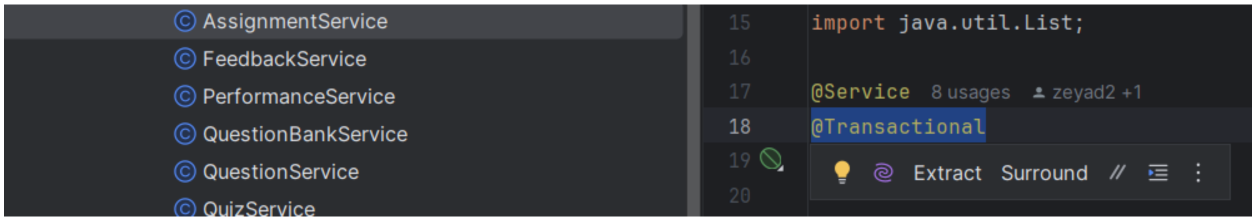
After: Introduced and threw QuizNotFoundException, StudentNotFoundException, and QuizAlreadySubmittedException instead.

SCRUM-10: Externalize security.password
Before: security.password=Actual password lived in application.properties.

After: Removed it from properties, replaced with ${SECURITY_PASSWORD} to load from an environment variable.

Report: Identified Issues in AssignmentService.java

Issue 1: Double Save Without Transaction
After refactoring:



```
import java.util.List;

@Service
@Transactional
```

Before:

The createAssignment method called assignmentRepository.save(...) twice—once to get an ID, then again after setting the file path—without any transactional boundary.

After:

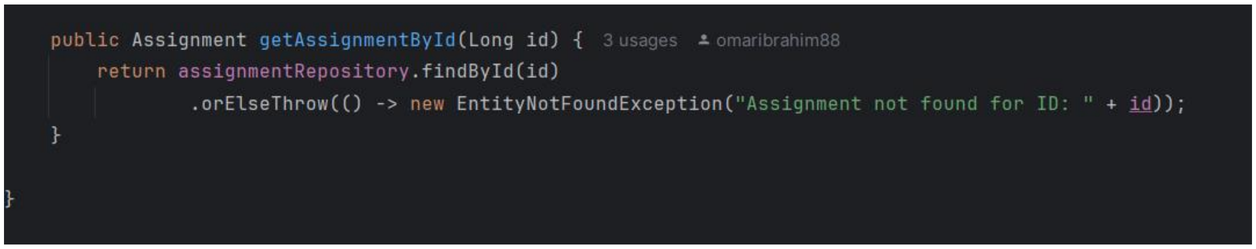
The service class (or method) is annotated with @Transactional, so both the initial save and the second save (with file-path update) occur atomically.

Report:

- Problem: Two separate save() calls without a transaction risked partial commits or orphaned records if the second save or file write failed.
- Fix: Adding @Transactional wraps all database operations (and file I/O) in one atomic unit that rolls back on error.
- Benefit: Guarantees data consistency—no more orphaned assignments or missing file paths.

Issue 2: Null Return on Missing Assignment

After refactoring:



```
    public Assignment getAssignmentById(Long id) {  3 usages  omaribrahim88
        return assignmentRepository.findById(id)
                .orElseThrow(() -> new EntityNotFoundException("Assignment not found for ID: " + id));
    }

}
```

Before:

getAssignmentById(Long id) returned null when the repository lookup failed (orElse(null)).

After:

It now uses orElseThrow(...) to throw EntityNotFoundException with a clear message if the assignment isn't found.

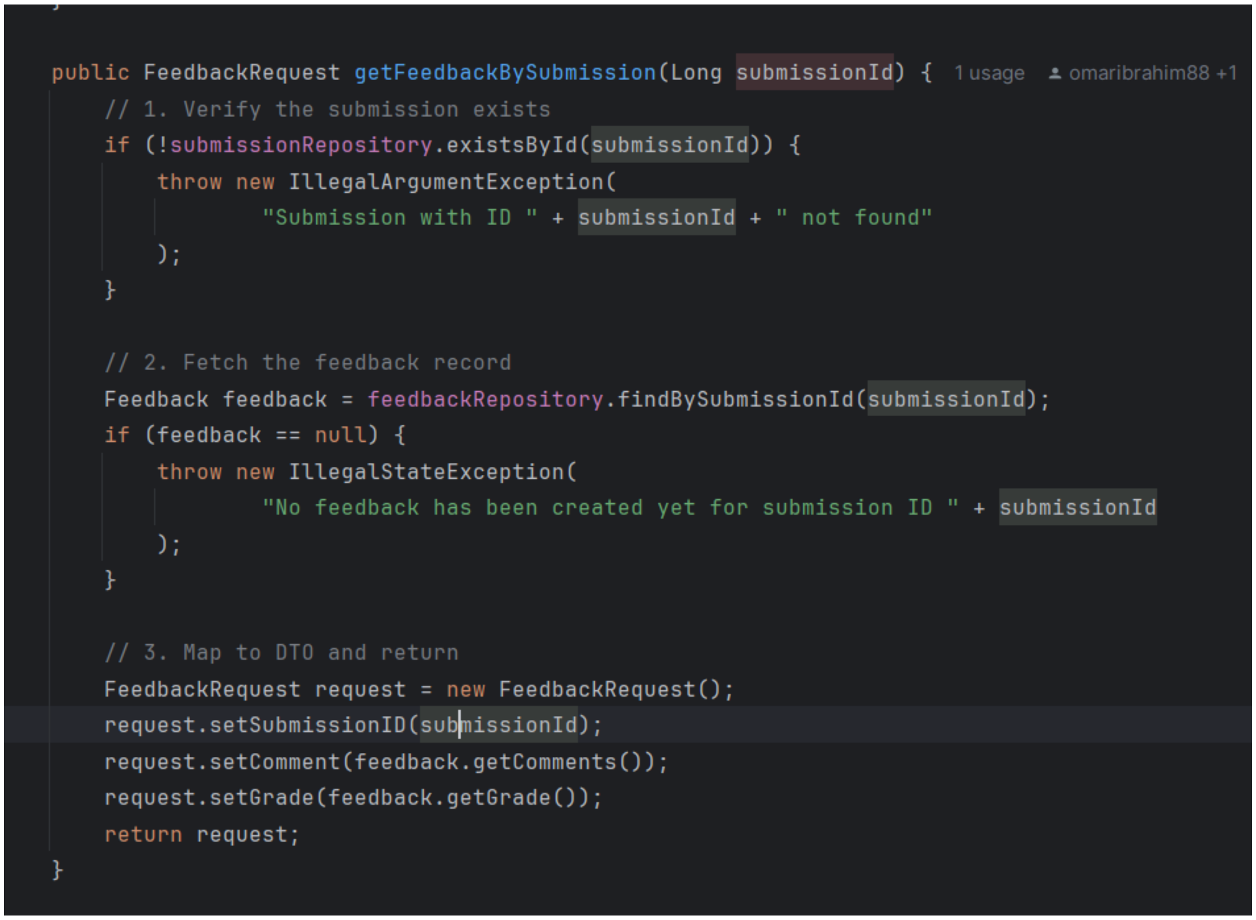
Report:

- **Problem: Returning null forced callers to null-check every call or risk NullPointerException, and gave no context on failure.**
- **Fix: Throwing EntityNotFoundException makes "not found" explicit.**
- **Benefit: Callers receive a clear 404-style error instead of a silent null, preventing NPEs and improving debuggability.**

Identified Issues in FeedbackService.java

Issue 3: Incorrect Exception Condition

After refactoring:



```java
public FeedbackRequest getFeedbackBySubmission(Long submissionId) {  1 usage  omaribrahim88 +1
    // 1. Verify the submission exists
    if (!submissionRepository.existsById(submissionId)) {
        throw new IllegalArgumentException(
                "Submission with ID " + submissionId + " not found"
        );
    }

    // 2. Fetch the feedback record
    Feedback feedback = feedbackRepository.findBySubmissionId(submissionId);
    if (feedback == null) {
        throw new IllegalStateException(
                "No feedback has been created yet for submission ID " + submissionId
        );
    }

    // 3. Map to DTO and return
    FeedbackRequest request = new FeedbackRequest();
    request.setSubmissionID(submissionId);
    request.setComment(feedback.getComments());
    request.setGrade(feedback.getGrade());
    return request;
}
```

Before:

When feedbackRepository.findBySubmissionId(...) returned null, the code threw IllegalArgumentException("Submission with ID ... not found"), conflating "no feedback" with "no submission."

After:

It first checks submissionRepository.existsById(...) and throws if the submission is missing; then, if feedback is null, it throws IllegalStateException("No feedback has been created yet...").

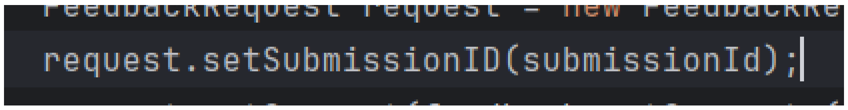
Report:

- **Problem: The original exception message was misleading—lumping together "submission missing" and "feedback missing."**
- **Fix: Splitting into two checks with accurate exception types/messages for each case.**

- Benefit: Distinct handling of “submission not found” versus “feedback not created,” improving clarity and making error responses more precise.

Issue 4: Wrong ID Set in DTO

After refactoring:



```
request.setSubmissionID(submissionId);
```

Before:

The FeedbackRequest DTO called request.setSubmissionID(feedback.getId()), inserting the feedback’s ID into the submission-ID field.

After:

It now correctly uses request.setSubmissionID(submissionId), so the DTO carries the actual submission’s ID.

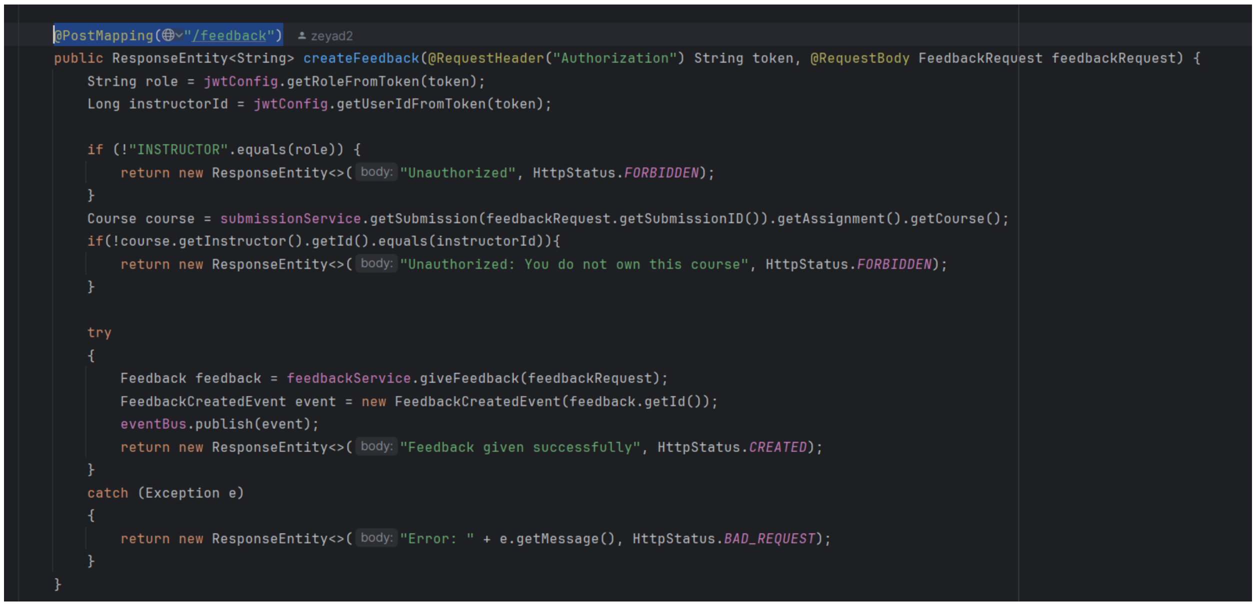
Report:

- **Problem: Mislabeled fields caused clients to receive the feedback record’s PK instead of the submission’s ID, breaking subsequent operations.**
- **Fix: Mapping the real submissionId parameter into the DTO.**
- **Benefit: Ensures clients always get the correct IDs, preserving data integrity and preventing mismatches.**

Identified Issue in AssignmentController#createFeedback

Issue 5: Incorrect HTTP Verb for Resource Creation

After refactoring:



```
@PostMapping("/feedback")  zeyad2
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);
    }
}
```

Before:

The createFeedback endpoint was annotated with @GetMapping("/feedback/create"), using GET for a state-changing operation.

After:

It's been changed to @PostMapping("/feedback"), adhering to REST conventions (POST for create).

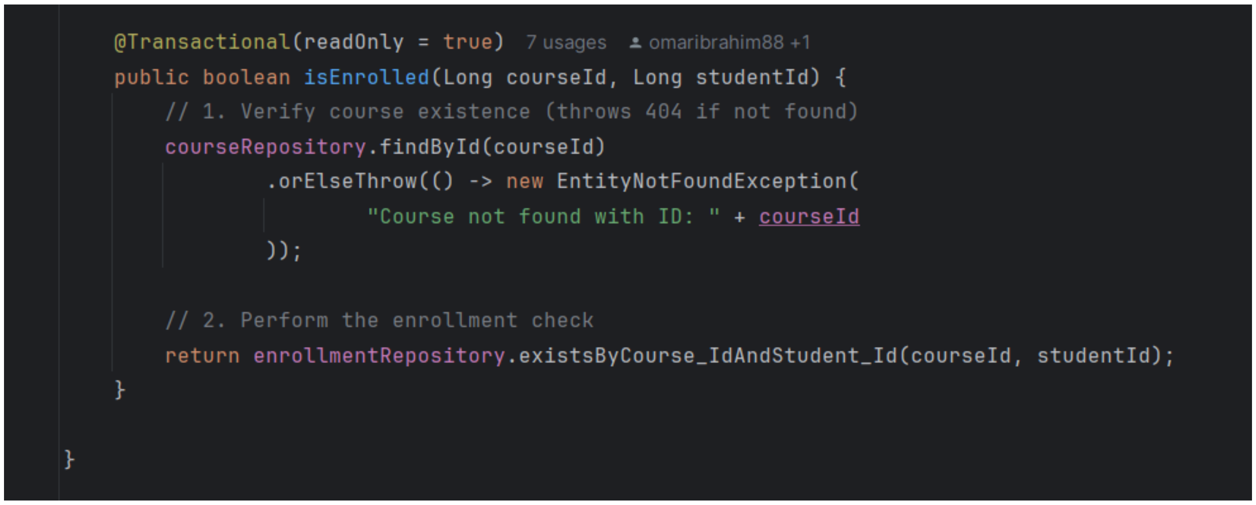
Report:

- **Problem: GET must be safe and idempotent—using it to create data risked accidental writes by crawlers or caches.**
- **Fix: Switching to POST on a resource-oriented URI (/feedback).**
- **Benefit: Prevents unintended side-effects, enables correct caching semantics, and aligns with standard REST tooling.**

Report: Identified Issue in CourseService#isEnrolled

Issue 6: Enrollment Check Doesn't Verify Course Existence

After refactoring:



```
@Transactional(readOnly = true)  7 usages  omaribrahim88 +1
public boolean isEnrolled(Long courseId, Long studentId) {
    // 1. Verify course existence (throws 404 if not found)
    courseRepository.findById(courseId)
            .orElseThrow(() -> new EntityNotFoundException(
                    "Course not found with ID: " + courseId
            ));

    // 2. Perform the enrollment check
    return enrollmentRepository.existsByCourse_IdAndStudent_Id(courseId, studentId);
}

}
```

Before:

isEnrolled(courseId, studentId) returned enrollmentRepository.existsByCourse_IdAndStudent_Id(...) without checking if the course actually exists, so invalid courseId quietly returned false.

After:

It first calls courseRepository.findById(courseId).orElseThrow(...) to throw EntityNotFoundException for a missing course, then performs the enrollment check.

Report:

- **Problem: Returning false for both “course not found” and “student not enrolled” made the two cases indistinguishable.**
- **Fix: Throwing a 404-style exception for a non-existent course before checking enrollment.**
- **Benefit: Consumers can clearly tell “course not found” (404) apart from “not enrolled” (false), improving error handling and user feedback.**

Class diagram after changes:

Bug Fix	Visible in Class Diagram?	Why?
Added `@Transactional`	❌	Annotation doesn't affect structure
Changed `orElse(null)` to `orElseThrow`	❌	Logic change, not structural
Fixed HTTP verb from `@Get` to `@Post`	❌	Routing change, not structural
Improved error messages and checks	❌	Internal logic
Changed mapping logic in DTOs	❌	Not reflected in class diagram
Removed double `.save()` in transactional scope	❌	Logic-level fix

SCRUM Fix	Change Type	Affects Class Diagram?	Reason
SCRUM-5: Prevent NPE on `student`	Internal null checks	No	Only adds guard clauses, no signature changes
SCRUM-6: DRY up `AssignmentController`	Extract constants	No	Moves literals to fields, no API or relation edits
SCRUM-7: Remove unused field in `AssignmentService`	Delete private field	No	Field was unused; no impact on public structure
SCRUM-8: Rename packages to conventions	Package refactor	No	FQCNs changed, but class-level structure remains
SCRUM-9: Use specific exceptions in `QuizService`	Replace exception types	No	Adds custom exceptions; core classes unchanged
SCRUM-10: Externalize `security.password`	Configuration only	No	Property moved out of code; no class edits