Software Testing and Quality Assurance

Report Assignment 2

Linh Trang Nguyen - NGUY1246

Brayden Rhys Little - LITT0113

Prompt History

Write Java codes for Learning Management System. Student : can get their course material, upload by set their assignments, check their grades. Only the enrolled students can access the courses's online exams. Academics can add contents and communicate with students through open student discussion forums. academic calendar: get all the assignment and exam schedules, user can manage to displays assignment and exam schedules in 1 week, 1 month or all semester. LMS have authentication. Every student and academic staff should be authenticated while entering. LMS request authentication after any inactivity of ten minutes after opening the LMS page. A user will be able to logout at any time from the LMS page Separate all Java classes: student, academic, assignment, exam, course, DiscussionForum, Grade, calendar

Here is a simple implementation of the Learning Management System (LMS) in Java with the specified features. Note that this example does not include the full implementation of web authentication, file handling, and other advanced features. It is a simplified version to get you started.

1. **Student.java**

java Copy code

```java
public class Student extends User {
    private String studentId;
```

AcademicCalendar can't add assignments and exams. It only get assignments and exams by in 1 week, 1 month or all semester. Only academic can add the courses' assignments and exams

Assignments and exams have due date. AcademicCalendar will compare with the real time to getAssignments and getExams in 1 week, 1 month or all semester

accessCourseMaterial have to return the materials list

Students upload a file for assignment, not a String content

checkGrades function have to return a String that is grade of student in a specific course

student can get their own String schedule of assignments and exams by using displaySchedule function

Final Recommendations

Overall, the AI generated methods display a basic level of understanding and prompt interpretation. ChatGPT was able to follow all prompts successfully, even if it didn't fully adhere to most of the best practices of coding. It also made a few incorrect assumptions, such as assuming that assignments would be submitted as strings rather than full files. As such, it had to be re-prompted with corrections to these false assumptions. Another area it lacks within is error handling and validation, as many methods lack proper validation and unauthorized access controls. For example, AD01 passed because it correctly threw a Null Pointer Exception, indicating proper handling of null inputs. However, ST03 failed, revealing that the method did not check if the student was enrolled in the course. This lack of validation and control can lead to mishandled exceptions and application crashes. Tests for session related functionalities, such as login and logout, highlight issues with maintaining consistent session states. Test S03 failed because the method allowed multiple logins for the same user, indicating poor session management. Access control checks are also missing, allowing access to assignment submissions and grades for any user. For example, ST05

failed because the method did not restrict grade access for unenrolled students, highlighting the need for better access control mechanisms.

These issues suggest that ChatGPT created the LMS without proper connections between objects, interpreting each criterion or function from the prompts as a self-contained method, rather than as part of an interconnected system. The lack of many-many relationships or intermediate classes that are common in similar systems can make the system confusing without clear connections between objects. Additionally, assumptions that could have been safely made from the prompts such as the need for compiled student schedules and differentiated access levels for students and academic users, were not recognized. Maintenance of generated code would also be challenging due to a lack of documentation and or irrational coding patterns, increasing the effort required for future extensions and debugging. Overall, while automated code generation can speed up development processes, the lack of validation, connectivity, security checks, and consistent coding standards makes it difficult to recommend outside of simple surface-level code generation. As the requirements become more complex, the occurrence and quantity of bugs and crashes increase too quickly to recommend. Further recommendations include the use of a class diagram or ERD, to clearly layout the relationships beforehand. These diagrams could then improve prompting by the user or in some cases also be uploaded to the generative program itself, which may in turn improve the generated code.