Richard Seabridge
CS 320: Software Test/Automation
Southern New Hampshire University
12/15/2024

Summary and Reflections Report

Contact Service: For the contact service, the unit tests focused on validating CRUD operations. Test cases ensured that:

- Contacts could be created with valid names, phone numbers, and addresses.
- Duplicate contacts were not allowed.
- Invalid inputs, such as empty names or incorrectly formatted phone numbers, were rejected.

Task Service: Tests for the task service validated:

- Proper handling of deadlines (e.g., future vs. past deadlines).
- Task prioritization logic for high-priority tasks.

Appointment Service: For appointments, unit tests were designed to check:

- Accurate scheduling of appointments without overlaps.
- Proper handling of invalid dates and times.
- Notifications or conflicts when overlapping schedules occurred

Alignment With Software Requirements

The testing approach was closely aligned with the software requirements:

- Functional requirements, such as the ability to add and retrieve contacts, were tested directly.

- Edge cases, such as overlapping appointments or empty fields, were identified and tested comprehensively.
- Non-functional requirements, such as system stability under edge cases, were ensured.

Quality of JUnit Tests

The effectiveness of the JUnit tests was demonstrated by:

- Achieving a coverage percentage of 90%, as measured by the JaCoCo tool. This coverage indicates that nearly all branches and statements in the code were tested.
- Detecting potential bugs during early development, such as incorrect date formats and null references.

Experience Writing JUnit Tests

Technical Soundness: Writing tests helped ensure technically sound code. For example:

- Used assertions like assertEquals, assertTrue, and assertNotNull to validate correctness.

Reflection

Testing Techniques

Techniques Used:

- **Unit Testing:** Focused on individual modules (e.g., contact service) to verify correctness.
- **Boundary Testing:** Ensured inputs at the edge of validity (e.g., empty strings, future dates) were handled.
- **Exception Testing:** Verified that errors, such as duplicate entries, were thrown appropriately.

Richard Seabridge
CS 320: Software Test/Automation
Southern New Hampshire University
12/15/2024

Techniques Not Used:

- **Integration Testing:** Tests that ensure multiple modules (e.g., contact and task services) work together.
- **Exploratory Testing:** Ad hoc, informal testing to uncover edge cases not explicitly defined.
- **Load Testing:** Tested how the system performs under heavy usage scenarios.

Practical Implications:

- Unit testing is critical for detecting errors early in the development lifecycle.
- Integration testing is valuable for verifying that components interact correctly in larger systems.
- Exploratory testing uncovers hidden bugs in real-world scenarios.

Mindset

Adopting a Software Tester's Mindset:

- **Caution:** Recognized the importance of testing edge cases and interdependencies. For example, ensuring that overlapping appointments trigger a conflict warning.
- Example: A bug was identified where null phone numbers were accepted due to missing validation.

Limiting Bias:

- Tests were reviewed by peers to ensure objectivity.
- Example: A colleague identified missing validation for past dates in task deadlines, which I had overlooked due to over-familiarity with the code.

Commitment to Quality:

Richard Seabridge
CS 320: Software Test/Automation
Southern New Hampshire University
12/15/2024

- Avoided cutting corners by adhering to Test-Driven Development (TDD).
- Example: Writing test cases before implementing features ensured a disciplined approach to development.
- Plans to avoid technical debt:
    1. Regularly refactoring code to maintain readability and efficiency.
    2. Using automated tools to monitor code coverage and maintain high standards.

Conclusion

This project highlighted the importance of structured testing strategies to ensure robust and reliable software. By focusing on unit testing, I ensured that individual components met the requirements while preparing them for integration with larger systems. Reflecting on this experience has reinforced my commitment to maintaining quality and discipline in future projects.