Kelly Perez

08/08/2023

CS 320

Project Two

I rigorously followed the criteria for unit testing Task, Contact, and Appointment assignments in Project One. My JUnit tests covered a lot and found failures, proving my testing method worked. I used extensive unit testing for the Task assignment. The test TaskServiceTest validated that task objects were added, deleted, and updated with accurate characteristics, meeting task initialization requirements.

A screen shot of a computer program

Description automatically generated

For the Contact assignment, I focused on addContact() and deleteContact(), and updateContact().

A screen shot of a computer code

Description automatically generated

I verified contact addition with valid input data in testAddContact() matching the need for reliable contact management. The testDeleteContact() method verified that contacts could be removed safely, meeting data handling requirements. The testUpdateContact() validated any updates made to current contact information.

A screen shot of a computer code

Description automatically generated

Moreover, in the Appointment assignment, I conducted thorough testing with the testAddAndGetAppointment() and testGetAppointmentsForDate() methods. These tests meticulously evaluated appointment creation and conflict detection, effectively addressing the software's primary objective of efficient scheduling. A computer screen shot of a program code

Description automatically generated

Writing JUnit tests provided valuable insights into code functionality and behavior, empowering me to ensure the reliability of my features. Crafting the JUnit tests for the AppointmentService class helped ensure code reliability and functioning. It involved creating scenarios, modeling interactions, and testing results. This method ensured that the code worked as planned and was technically sound and efficient.

To check technical soundness, I created test cases for creating and retrieving appointments using multiple use cases. I created and added an appointment to the service in testAddAndGetAppointment(). I verified data integrity by comparing new and retrieved appointments using assertEquals.





Unit and integration testing were my main software testing methods. Unit testing tested methods and classes individually to ensure their functionality. Integration testing, included in my unit tests, confirmed component interactions. I didn't use regression testing or performance testing to check for unforeseen side effects when making changes. Regression testing would assure code stability throughout updates, while performance testing would evaluate the mobile app's responsiveness and resource utilization. To protect user data in sensitive applications, this project did not use security testing. The benefits of each method depend on project size, requirements, and goals.

I was cautious and diligent as a software tester throughout the assignments. I carefully created tests and implemented feedback to cover multiple scenarios to ensure application reliability. In testing the appointment service, I considered how dates and descriptions affected appointments. I utilized a methodical strategy and objective standards to reduce prejudice. Personal attachment to code might cause developers to miss issues. Overlooking edge cases in a contact form owing to familiarity with its structure. Code quality discipline is crucial. Cutting shortcuts might cause technological challenges and scalability issues. Investing in robust code pays out over time. Technical debt is avoided by code reviews, documentation, and best practices. For instance, slow testing can prevent costly vulnerabilities.