Java Unit Testing

UserTest.Java

Anny Patino

table of contents_

Review Tests

Assist in Debugging

Promote Best Practices

Check to see if the results are correct

Writing and Testing Automation Tools

Holistic Understanding of the System

Introduction_

Unit testing in Java involves creating small, focused tests to verify individual components or methods of your application.

Unit testing is crucial for **Site Reliability Engineering** (SRE) because it helps ensure the stability, reliability, and performance of systems by verifying the correctness of individual components.

While product development teams write unit tests for application features, SREs focus on tests that ensure system reliability, performance, and operational robustness.

UserTest.java_

- **REPUTATION SYSTEM:**
  - Question up-votes increase the questioner's reputation by 5 points.
  - Answer upvotes increase the answerer's reputation by 10 points.
  - Accepted answers give the answerer a 15-point boost.
  - Down-votes on answers reduce the answerer's reputation by 1 point.
- **VOTING RESTRICTIONS:**
  - Users cannot up-vote or down-vote their own questions or answers.
- **ANSWER ACCEPTANCE:**
  - Only the author of a question can accept an answer.
  - Non-authors attempting to accept an answer receive an appropriate exception.

Reputation System Test_

This test verifies:

Up-voting a question increases the questioner's reputation by 5 points.

```
// Validates Questioner's reputation increases by 5 points if  their question is up-voted
@Test
public void questionerReputationIncreasesBy5WhenQuestionIsUpVoted() throws Exception {
    // Act: Voter upVotes the question
    voter.upVote(question);

    // Assert: Questioner's reputation increases by 5
    assertEquals(5, questioner.getReputation());
}
```

Voting Restrictions Test_

This test validates:

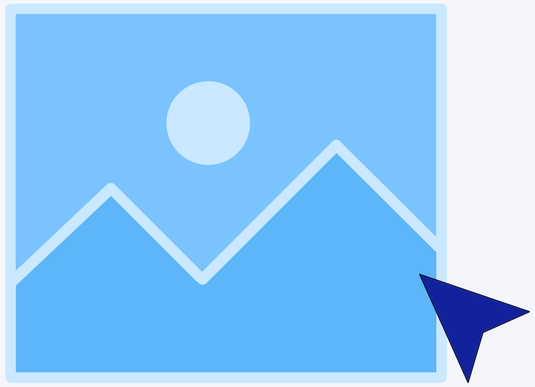
Users cannot up-vote their own questions.

```
// Test for Self-voting Restrictions

// Verify that users cannot upVote their own question
@Test
public void userUpVotesTheirOwnQuestion() throws Exception {
    // Act and Assert: Expect a VotingException with appropriate message
    thrown.expect(VotingException.class);
    thrown.expectMessage("You cannot vote for yourself!");

    questioner.upVote(question);
}
```

Live Demo _



Key Takeaways_

01 Each test should focus on one functionality.

02 Use descriptive names that clearly state the purpose of the test.

03 Arrange-Act-Assert Pattern