Intro to Testing

JUNE 30, 2022

Jennifer Peterson

Let’s Discuss - What is Testing?

Within the realm of software:

1. What does it mean to perform testing?
2. Why is testing important?
3. What kinds of testing are there?
4. How do *you* test your work?

Levels of Testing

- Unit Testing
  - Testing of individual components
- Integration Testing
  - Testing that components work together
- System Testing
  - Testing the entire system
- Acceptance Testing
  - Testing done typically by the customer

Test Code Coverage

- What does code coverage mean?
- Commonly used coverage criteria:

1. **Function coverage** – Has each function in the program been called?
2. **Statement coverage** – Has each statement in the program been executed?
3. **Branch coverage** – Has each branch of each control structure (such as *if* and *case* statements) been executed? For example, given an *if* statement, have both the true and false branches been executed?
4. **Condition coverage** (or predicate coverage) – Has each Boolean sub-expression evaluated both to true and false?

   e.g. if ((x > 0) && (y > 0))

Source: https://en.wikipedia.org/wiki/Code_coverage

Testing Paths

Happy path is a scenario where all provided input is as expected and no error conditions occur.

Unhappy path is any scenario in which error conditions are encountered, such as:

- User provides text input when numbers are expected
- The specified file cannot be found or is locked
- The network is unavailable
- The database doesn’t exist or is corrupted
- The user has insufficient permissions
- etc…

Automated vs. Manual Testing

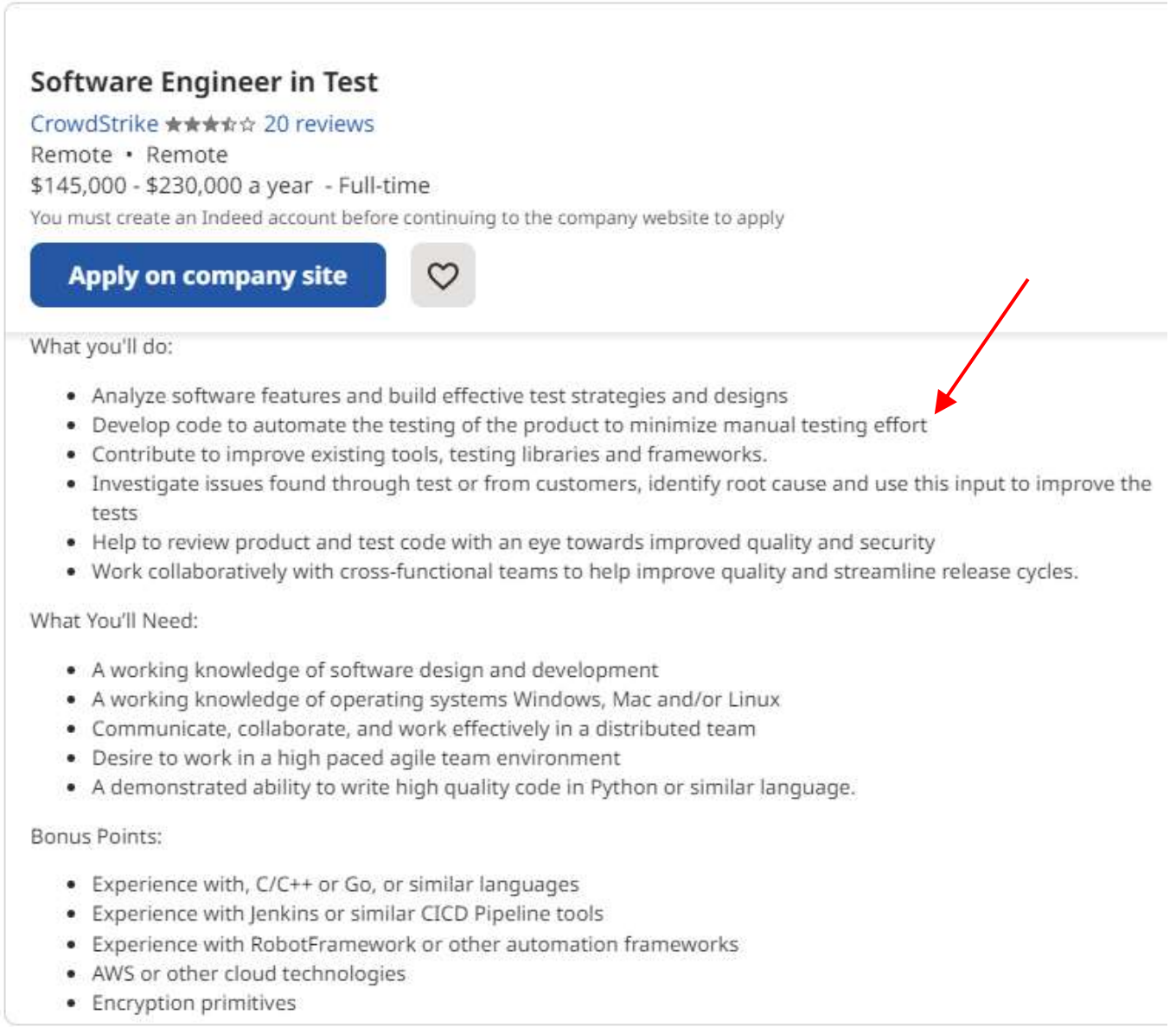
Automated tests are take longer to write initially, but are easier to re-run

Manual tests are faster the first time, but take more effort than automated tests in subsequent runs

Realistically, you should automate as much testing as possible, but there will almost always be some amount of manual testing (particularly at the Acceptance Testing level)

Writing Tests

- Who is responsible for writing tests?
  - Depends on the organization
    - it is good practice for you (the developer) to write tests for your code.
  - Some organizations have dedicated test engineers
    - Entire job is to create tests.
    - This role can sometimes be called a QA or Quality Assurance engineer (though generally QAs have less experience than test engineers).



Software Engineer in Test

CrowdStrike ★★★½☆ 20 reviews

Remote • Remote

$145,000 - $230,000 a year - Full-time

You must create an Indeed account before continuing to the company website to apply

Apply on company site

What you'll do:

- Analyze software features and build effective test strategies and designs
- Develop code to automate the testing of the product to minimize manual testing effort
- Contribute to improve existing tools, testing libraries and frameworks.
- Investigate issues found through test or from customers, identify root cause and use this input to improve the tests
- Help to review product and test code with an eye towards improved quality and security
- Work collaboratively with cross-functional teams to help improve quality and streamline release cycles.

What You'll Need:

- A working knowledge of software design and development
- A working knowledge of operating systems Windows, Mac and/or Linux
- Communicate, collaborate, and work effectively in a distributed team
- Desire to work in a high paced agile team environment
- A demonstrated ability to write high quality code in Python or similar language.

Bonus Points:

- Experience with, C/C++ or Go, or similar languages
- Experience with Jenkins or similar CICD Pipeline tools
- Experience with RobotFramework or other automation frameworks
- AWS or other cloud technologies
- Encryption primitives

SLIDE CREDIT: DAMAN OBEROI

When should tests be written?

- What students typically do → Write tests last minute before turning in the assignment or not at all.
- A better approach is to write tests for any new code before committing it to the repository.
- Another approach is called Test Driven Development (TDD) → Tests are written during the design phase based on the requirements.

Today's Activity

- Modularize your code into functions that can be unit tested.
  - Consider dividing your code in a way that makes sense, but also so that each module has a discrete, testable purpose.
- You will **not** be writing unit tests in this activity; that comes later today. For now, you are preparing to write unit tests by making your code more testable.
- Add comments to your code that identify the kinds of unit tests you will write.
- Join your TA's breakout room and work on the activity.
- Feel free to work with others in your room and ask questions of the group.
- The TA is there to assist you. If you want to ask me a question, come back to the main zoom room.