Software testing theory

Please answer the following questions. This document is designed as a questionnaire regarding Software testing theory, approaches, differences of approaches and various theoretical questions.

**Q: In your opinion what skills are needed for someone to become a software tester?**

Your answer:

An understanding of the tools used for testing and manual testing practices, insatiable knowledge appetite, detail-oriented, good communication skills, problem-solving way of thinking, fast learner, developed logical thinking, feeling for colors and design, empathy, curiosity, and childish spirit, etc.

**Q: What is manual software testing, and how does it differ from automated software testing?**

Your answer:

Manual software testing is performed by a person (mostly by QA), by executing some operations on the tested product without scripts or tools to discover some defects in software under development. Automated software testing implies executing pre-scripted tests, written by a QA engineer or as a part of some tool, with the purpose to trigger the same API request or UI functionality on the emulated browser.

**Q: What are the advantages and benefits of manual testing?**

Your answer:

Manual testing is an activity where the tester needs to be very patient, creative & open-minded. It is good when frequent repetition is not required, or when there is a need for Exploratory Testing, Usability Testing, and Ad-hoc Testing. Manual testing is great to spot some unusual behavior of an app in a process that cannot be tested by a machine. Also if there are some small changes in UI like in id, class, etc, it wouldn't affect the testing process of a manual tester, and there is no need for programming skills for manual testing. No UI test can be automated if it is not performed manually first.

**Q: What are the disadvantages of manual testing?**

Your answer:

Manual Testing is a time-consuming process, and repetitive manual test execution can get boring and error-prone. Cannot be used for Stress testing, Performance testing, etc.

**Q: What are the advantages and disadvantages of automated testing?**

Your answer:

Automated testing is faster, tests can be run repeatedly over a long period (if something needs to be done over 100 times over and over again, it is always a better option to automate it), and there is less room for a mistake. It is best for Repeated Execution, Regression Testing, Load Testing, and Performance Testing

**Q: Can automated testing replace manual testing?**

Your answer:

A lot of things can be automated, but in some close future, no. If not for other reasons, some tasks would be just a waste of effort, time, and resources.

**Q: What types of manual testing are there? Explain each with a sentence.**

Your answer:

Black-Box Testing - Mainly focused on the Input and output of the product without having any knowledge about internal code structure;

White-Box Testing - Analyze the internal structures of the used data structures, internal design, code structure, and the working of the software;

Gray-Box Testing - Gray Box Testing is a software testing technique that is a combination of the Black Box Testing technique and the White Box Testing technique, where the internal structure is partially known.

Unit Testing - Test individual units of software to determine whether they are suitable for use or not;

Integration Testing - The goal of integration testing is to detect any irregularity between the units that are integrated;

System Testing - System testing detects defects within the integrated units and the whole system;

Usability Testing - Usability testing is the practice of testing how to easy a design is to use on a group of representative users;

User Acceptance Testing - The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.

**Q: What is black box testing? Give an example for Black Box testing and describe it in a few sentences.**

Your answer:

In Black-box Testing no knowledge of implementation is required. It involves testing from an external or end-user perspective. The most common functionality that can be used as an example is the "Login" functionality. Test cases will be based on how the end-user interacts with the login page:

\*User should be able to log in with a valid email and password;

\*User should not be able to log in with an invalid email, password, or both, and an error message will be displayed;

\*Links on the page take user to the right page;

\*etc

**Q: How would you approach a new project when you see it for the first time, and have no previous knowledge of how it works?**

Your answer:

Well, it depends if there is the documentation for the project. In case I don't have an approach to the documentation, I'll find a way to get to know the app better. If there is a Help section or tool tips and error messages, I'll check if the message matches the functionality. The easiest for the start is to check input fields, links, and buttons (corner cases, special characters, etc).

**Q: What are the different types of Software testing? Explain the difference.**

Your answer:

Functional Testing and Non-functional Testing. Functional Testing verifies each function/feature of the software and mainly involves black box testing, and it is not concerned with the source code of the application. Non-functional Testing verifies non-functional aspects like performance, usability, reliability, etc. A good example of a Non-functional test would be to check how many people can simultaneously login into a software.

**Q: When should testing end? Elaborate.**

Your answer:

As long as the app is used it is going to be tested. After all functionalities of an app are developed and tested/verified by QA, it continues to be tested as it has been used by end users.

**Q: What kind of testing do you use in your current position?**

Your answer:

Currently, I'm doing Manual and Automation UI and API testing.

**Q: When you see a project or a feature for the first time, during your testing you see something that doesn’t seem right. How do you go forward?**

Your answer:

Firstly I'll check the documentation to see if that behavior is expected. If there is no access to the documentation, I'll discuss it with PO or team members. If the behavior is not expected functionality, I'll open the ticket and report the bug with a description of the expected behavior, current behavior, environments, version, steps to reproduce, and a screenshot of the defect.