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:

There are various skills required to carry out the QA role successfully. Here are some key skills for someone to become a software tester:

- **Problem solving skills** – Whenever there are issues in project, a unique, smart and efficient approach will be required to solve it.

- **Communication skills** – A software tester must have good verbal and written communication skills. There is a lot of collaboration required in this role with stakeholders, developers, QA's, product owners, designers, clients where communication plays a vital role. Expressing ideas and concepts is very important along with active listening.

- **Attention to detail** – A software tester must be able to spot minor details and imperfections in the code and the systems. The primary motive is to identify issues by digging deep and scrutinizing to ensure quality standards. It is very important for the tester to have a clear understanding of the requirements and observe the impact of changed in the application's behavior.

- **Empathy** – Understanding the client's priorities is an essential part of showing empathy. Software testers must be able to ask questions to learn what is important from their perspective.

- **Analytical thinking** – This is one of the most important skill for a software tester which can enhance the quality of the tester's work. Break down the problem into smaller segments to examine every element properly and fetch the best possible solution. During testing, one must analyze the product properly on the first go as this will give proper understanding of the functionality.

- **Focus on Quality** – A software tester should never compromise on quality aspects of the application alongside the deadlines. This can be done by taking ownership and ensuring regular follow-up process with regards to the bug for fixing and retesting.

- **Instinctive reporting and documentation** - Reporting and adequate documentation are essential to ensure that all the updates are tracked.

- **Highly adaptable** – As we work in an agile environment where the environment is quite dynamic with frequently changing requirements, a tester should be very flexible and adaptive. Also, with the fast-growing technologies, tester should keep checking the way of working and upgrade skills.

- **Understanding the business and its customers** - The most crucial thing in testing is to understand the customer expectations and business requirements of the product.

- **Good at planning and execution** – It is very important to have a plan or strategy to start testing any feature therefore, planning is an important skill.

- **Being both diplomatic and assertive** – A tester should always follow a diplomatic approach while communicating with developers or stakeholders to avoid resistance. Yet, being assertive is also very important.

- **Prioritize** – As testers work for more than one project at a time, it is very important to prioritize the work as there would be lot of pressure and continuous testing requests. Planning daily tasks really helps to prioritize and it will not impact the quality of testing.

- **Knowledge of test management and bug tracking tools** – Testrail and Jira
- **Knowledge of source control system** – Git, GitHub, Bitbucket
- **Backend skills** – Postman, Rest Assured for API testing
- **Database and SQL skills**
- **Automation and coding skills** – Selenium, Cypress, Java, JavaScript, Cucumber, different frameworks and approaches for automation like hybrid, data-driven, BDD.

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

Your answer:
Manual testing is a software testing process in which test cases are executed manually without using any automation tool. The purpose of manual testing is to identify bugs, issues, or defects in the application. The key concept of is to ensure that the application is error free, and it is working as per the specified functional requirements.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual

testing concepts does not require knowledge of any testing tool. One of the testing fundamentals is “100% Automation is not possible “which makes Manual Testing imperative.

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

Your answer:

- Manual testing allows testers to monitor the quality of a product throughout its development cycle by spending more time familiarizing themselves with its features and functions. This helps maintain their knowledge of the project, which will be useful if issues arise after changes have been made to the code. This is one of the reasons why manual testing is recommended first to test a scenario before considering automation.
- Manual testing gives testers the freedom to use their expertise to emulate the user experience.
- Manual testing can help testers use their expertise in detecting bugs.
- With manual testing, there is the flexibility advantage where adjustments can be done quickly and on the go.
- The tester can quickly access visual elements like text, layout, and other components to identify UI and UX problems.

Q: What are the disadvantages of manual testing?

Your answer:

- Testers may make mistakes sometimes, so we cannot expect more accuracy in manual testing.
- Testing with a large amount of data is impractical.
- Executing the same tests, again and again, is time consuming process as well as tedious.
- Manual testing is not suitable for large scale and time bounded projects.

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

Your answer:

Advantages of automated testing:

- It improves the coverage of testing as execution of test cases is faster than manual test execution.

- It reduces the dependability of testing on the availability of the QA engineers and has less chances of error hence are more reliable.

- It provides good test coverage as automated tests can be executed all time in 24/7 environment.

- It takes far less resources in execution as compared to manual testing.

- It helps in testing which is not possible without automation such as reliability testing, stress testing, load and performance testing.

- It acts as a test data generator and produces maximum test data to cover a large number of input and expected output for result comparison.

- It can also lead to reduced costs. When tests are automated, the need for manual testers is reduced. Also, the time needed to execute tests is reduced which leads to save both time and money.

Disadvantages of automated testing:

- Automated tests can take longer to be implemented than manual tests.

- Automated tests can sometimes fail even when there is no actual issue present. For example, this can be the case if the test contains an error or is not comprehensive enough to cover all its intended use cases. Similarly, tests may generate false negatives if they are designed only to verify that something exists and not that it works as expected.

- Debugging the test script can be challenging sometimes. If there is any error in the test script, sometimes it may lead to deadly consequences.

- Designing a comprehensive suite of automated tests is not a small task. They need to be reliable enough that they can be run frequently and consistently without giving false positives or negatives. On the other hand, test scripts must be maintainable enough to adapt to changes in your application.

- All types of testing cannot be automated. For example, Usability testing,

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

Your answer:

No, automation testing can never take over manual testing completely. We need both manual testing and automation testing. Whatsoever the feature or a product is, it should be first tested manually and then can be automated. Manual testing handles complex test cases, while automated testing handles simpler, more repetitive tests. In manual testing, there is a lower

risk for false negatives. Repetitive, high-frequency tests can easily be automated but tests should not be automated if they occur infrequently or are subject to change.

Here are some instances when we would want manual testing instead of automated testing:

- Complex test scenarios that are not efficient and sometimes not feasible to automate
- Test scenarios that are only being validated once in a while
- Exploratory testing is done to uncover things that automation tests can never find
- Automating usability tests is just not possible

Manual testing helps us understand the entire problem and explore other angles of tests with flexibility. Automated testing helps saves time in the long run by accomplishing many tests in a short time. So, manual testing is still important. But adding automated testing makes manual tests more efficient.

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

Your answer:

Types of manual testing are explained below:

1. **Black box testing** – This type of testing is also known as behavioral testing which analyzes the application's behavior from the end user's perspective. It is not concerned with the internal structure of the application and is done by the testers. Black box testing can be further divided into functional and non-functional testing.

- Functional Testing
- Smoke Testing
- Regression Testing
- User Acceptance Testing
- Usability Testing
- Localization Testing
- Exploratory Testing
- Integration Testing
- System Testing
- UI Testing

2. **White box** – This is also known as glass box or transparent testing. White box testing is approach in which testers is familiar with the internal code or structure of the application and verify the internal processes and integrations with external systems. This is an essential part of automated build processes in CI/CD pipelines.

3. **Gray box** - This test approach is the combination of both white box and black box testing techniques. The main aim of this approach is to identify any bugs present either due to inappropriate usage or any structural flaws.

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

Your answer:

Black box testing is a type of software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Some examples of black box testing are:

The specification is input field accepts only integer values and prints it. To test this, we will select a positive or a negative integer as test data.

Verifying that it is possible to log in using correct user credentials, and not possible to log in using wrong credentials.

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:

This is always an interesting opportunity for a QA to expand knowledge and get involve in the product from the business perspective. I would perform the following steps to be involved in the new project:

1. Firstly, analyze the project documentation and do some research related to the type of project.
2. Understand the front end and backend functionality of the application and ensure the UI requirements comparing it with designs.
3. Document the findings or questions of the research to discuss in the meeting
4. Organize the brainstorm session together with the stakeholders, developers and PO of the team.
5. Discuss the requirements, order of actions, goals, timeline, estimations, roles and responsibility of the project.
6. Do some learnings in parallel if needed.
7. Evaluate in-depth the essential approaches to establishing the QA process, benefits, disadvantages, and challenges of each strategy, as well as detailed steps and best practices of the QA process and procedures.

8. Align with the QA team to discuss the process and strategies.
9. Be involved in the sprint ceremonies like daily standups, planning, refinement and retrospective of the project as a QA
10. Review functional and non-functional project requirements
11. Analyze potential risks (e.g. tight timelines, changing requirements, etc.) and create a risk mitigation plan.
12. Prepare a testing strategy together with the developers and PO.
13. Design test scenarios and test cases in a test management tool
14. Write scripts for automation testing (if aligned with the business requirements)
15. Incorporate tests into CI/CD pipeline to speed up the testing process.

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

Your answer:

There are various types of software testing that can be used to ensure that the software is bug free. Each type of testing has its own features, advantages, and disadvantages as well. The explanation along with different types of software testing are described below:

1. **Functional Testing**
   Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. There are four main types of functional testing:

   i) Unit testing – It is a type of software which is done on an individual methods and functions of the classes, components, or modules used by the application. Unit tests are generally run very quickly by a continuous integration server and is done by the developers not testers.

   ii) Integration testing – It is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole. The focus of this type of testing is to find the defect on interface, communication, and data flow among modules. You need to perform fewer integration tests than unit tests.

   iii) System testing - System testing is types of testing where tester evaluates the whole system against the specified requirements.

      (a) End to End testing - It is the functional testing of the entire software system. When you test the complete software system, such testing is called end-to-end testing. You need to perform fewer end-to-end tests than integration tests.

(b) Black box testing – It is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects.

(c) Smoke testing - It is performed to verify that basic and critical functionality of the system under test is working fine at a very high level. Whenever a new build is provided by the development team, then the tester validates the build and ensures that no major issue exists. The tester will ensure that the build is stable, and a detailed level of testing will be carried out further.

(d) Sanity testing - It is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test.

(e) Happy path testing - The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions.

(f) Monkey testing - It is carried out by a tester by entering random input values without any knowledge or understanding of the application. The objective is to check if an application or system gets crashed by providing random input values or data. No test cases are scripted, and it is not necessary to be aware of the full functionality of the system.

iv) Acceptance Testing - It is a type of testing where clients, customer or stakeholders test the software with real time business scenarios.

2) Non-Functional Testing

Non-functional testing is the testing of non-functional aspects of an application, such as performance, reliability, usability, security, and so on. Non-functional tests are performed after the functional tests. With non-functional testing, we can improve the software's quality to a great extent. Functional tests also improve the quality, but with non-functional tests, we have the opportunity to make the software even better.

There are several types of non-functional testing, such as:

i. Performance Testing – It is testing of an application's stability and response time by applying load.

a. Load Testing – It is testing of an application's stability and response time by applying load, which is equal to or less than the designed number of users for an application.

b. Stress Testing – It is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

c. Scalability testing- It is testing an application's stability and response time by applying load, which is more than the designed number of users for an application.

d. Volume testing - It is testing an application's stability and response time by transferring a large volume of data to the database. Basically, it tests the capacity of the database to handle the data.

ii. Usability Testing – It is testing an application from the user's perspective to check the look and feel and user-friendliness.

iii. Exploratory Testing- It is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application.

iv. Cross browser testing- It is testing an application on different browsers, operating systems, mobile devices to see look and feel and performance.

v. Accessibility Testing – It is to determine whether the software or application is accessible for disabled people or not.

vi. Adhoc Testing – This type of testing is performed on an adhoc basis with no reference to the test case and also without any plan or documentation.

Q: When should testing end? Elaborate.

Your answer:
I believe testing is a never-ending process in a positive way. However, it can be said as when you are confident enough with your testing for a feature, you can stop testing.
I consider the following checklist to determine whether I have enough confidence for completing the testing.

- When the release deadlines or testing deadlines have reached
- When all the planned test cases are executed with some prescribed pass percentage

- When the code coverage and functionality requirements come to the desired level
- When all the bugs related to that feature are fixed

It is important to pay attention to finer details and errors in the feature of the application, with the aim of finding out why doesn't the functionality work as intended. It is not possible to find all the bugs that may exist, but the team will make decisions on when to release the product anyway.

Another thing I would like to add is when certain bugs do appear, we need to make sure they are fixed once and for all. It won't be a good quality product if the error keeps reappearing. We should predict how many bugs are there so we can find them in order to have a desirable level of confidence that the feature is ready to go live.

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

Your answer:

I am involved in all the internal and external projects of vanmoof which gives me an opportunity to perform various types of testing such as:

- Functional testing (Manual and Automation)
- API testing (Postman and Rest Assured)
- Smoke and Sanity Testing
- Regression Testing
- Performance Testing
- Acceptance Testing
- Accessibility Testing
- End to end Testing
- Integration Testing
- UI Testing
- Exploratory Testing
- Adhoc Testing
- Cross Browser 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:

If I find something in a feature or a project that doesn't seem right for example BUG, I will do the following steps:

- Start by breaking the problem into smaller chunks.
- Gather information, do some research related to the feature/project to understand more about it through the documentation and reaching out to the relevant people.
- Prepare a document with the findings along with the approach to fix that before meeting the team

- Organize a brainstorming session with the developers, stakeholders and PO.
- Create a ticket for that in Jira with all the necessary information like screenshots, logs, steps to reproduce, actual result and expected results.
- Prioritize the bug
- Estimate the bug together with the team
- Add a test case in the regression suite for that to avoid the bug in future.