1. **Importance of quality in a fast-paced delivery environment**

Answer:

The influence of software is far bigger these days. It contributes to upscale life standards, enhances the business prospects, and evolves a smart world. In addition to the industrial /commercial impact, today software plays critical roles across all the walkways of common people. However, all it requires is the smooth and fail-proof functioning of the software. This is where Software quality comes into picture. The importance of software testing and quality assurance is of high value in a software development cycle. Both processes refine the whole process and ensure superior quality to the product. Also, it reduces maintenance costs and provides better usability and enhanced functionality.

2. **What is shifting left - Pros and Cons of shifting left quality**

Answer:

Instead of taking a backseat during the development process, testing is planned and begins earlier in the SDLC (therefore "shifts left"). It could even start before a single line of code is written.

Pros:

- Lower development and testing cost.
- Increase efficiency and quality.
- Compete more fiercely

Cons:

- Easier said than done
- Risks of bottlenecks

3. **Exploratory testing and automated testing - Pros and Cons and how they can be synergized**

Answer:

Exploratory Testing is a style of software testing where there is less of a structure and a specified process. In this testing, the tester has more personal freedom and responsibility to utilize their skills and knowledge to optimize the quality of their work.

Pros of Exploratory Testing:

- Ability to analyse the application as it will be used when it is live
- Helps in finding usability & UI issues
- The testing outcomes depend on testers' creativity, experience, & skill
- Project owners can get insights which were not possible to get with scripted tests

Cons of Exploratory Testing:

- There is no documentation for reference
- A bias depending on the tester can exist
- Critical bugs can be missed

However, In the automated Testing approach, the testers follow the script which includes well-documented test cases and test steps. There is no deviation from the script. The testers can write scripts in several programming and scripting languages. They can choose to execute the test cases manually or with the help of automated tools.

Pros of automated Testing

- Well-suited for automation
- Good in finding functional defects
- Standardized documentation helps in repeatability and tracking

Cons of automated Testing

- There may be non-standard results across individual testers
- Fewer bugs are found
- Focuses on limited areas constrained by the script

Conclusion

For comprehensive testing of an application, the combination of exploratory testing and automated testing should be used. This will help the testing team to harness the benefits of both the approaches and ensure a high-quality application is released to the market.

4. Checklists for an efficient automation framework

Answer:

- Reusable methods or page classes – Create reusable methods wherever you discover repeatable code. Don't duplicate an equivalent thing multiple test.
- Data driven – Test data like URLs / Usernames and Passwords are maintained in properties file or Excel files. Do not hard code everywhere.
- Explicit waits – Thread sleep delays everywhere in test scenarios. Also reduce the performance. So, attempt to use Explicit waits.
- Variables names should be meaning full.
- Try to use public API's rather than creating more utility files from scratch.
- Reporting – Do not print results using System.out.println. Always use Reporting mechanisms.
- Headless test execution support when there is a necessity
- Do not hard code absolute paths given to files utilized in the framework, instead of just putting the files into a folder relative to the framework.
- Data should read from test scenarios but not in page classes.
- Try to reduce Unnecessary program loops within the code.
- Test framework should organize into well-defined packages
- Pages – Where page classes reside
- Test – Where test reside
- Utility – Where utility classes reside. like reporting and file reading classes
- Documentation on deploying the test framework
- Logging facility for frameworks when something goes wrong
- Base driver support to run in multiple browsers
- Good naming conventions for page class and test class naming
- Tests should be independent when executing
- Detailed reports on test executions and failures
- Use design patterns and principals
- Use BDD – But this is often not mandatory always
- Screen shots on failures - Helps failure investigation easy.
- Use dependency management like Maven for Java, Nuget for .net, PIP for Python

5. <u>Share any experience where you have driven quality process improvements</u>

Answer:

In my current and previous organizations, I have on-boarded many tools and technologies that has changed the traditional way of testing and made the testing process more evolved. I have on-boarded and implemented tools like Elastic Search, Logstash, and Kibana, SonarQube, UI Path, Docker, Zerenium, Appium, Browserstack, RestAssured, Cloud (AWS), Jenkins, Python, Jira API's , OWASP ZAP, JMeter and other CI/CD related tools. With these tools and technologies, the testing that I and my team performed was not only of more quality, but also of less cost & less risk. Overall, testing Team happened to be more efficient than it used to be earlier.

For more details, please have a look at Key Contributions and Accomplishments section at each organization I have worked so far in my Resume.