**1)What is the difference between Automated and Manual testing in Software Development?**

In manual testing, testers execute test cases manually without using any automation tools. They interact with software as how end users would, identifying test results and recoding results, while in Automation testing, they are executed using automation tools or scripts. These scripts perform pre-defined actions on the software, comparing the actual results and reports the results.

**In Terms of Speed and Efficiency:** Manual testing is exhaustive and prone to human error especially for large and complex software applications. Automation testing on the other hand is more efficient and less time consuming. Large test cases and repetitive testing can be done in a short period of time.

**Accuracy:** It heavily relies on the skills and experience of the tester. While in Automation testing, it is more consistent and accurate and it eliminates the scope of error while executing.

**Test coverage:** While the testers maybe creative in exploratory testing, the scope of human availability and time constrain limit the coverage. Automated testing can achieve broader test coverage as it can execute large number of test cases quickly and repeatedly.

**Initial Setup:** Manual testing requires minimum setup as it mainly involves the tester knowledge and expertise. Automation testing requires significant setup time to create test scripts and setting up the environment.

**Usability and User experience:** Manual testing is better suited for testing aspects related to user experience, usability and visual aesthetics where human judgement plays a crucial role. In Automation testing, it is more suitable for functional and regression testing focusing on repeated verification of specific functionalities.

**2)Explore the common automation testing tools in the market?**

The most common and widely used automation testing tools are Selenium, Appium, Cucumber and TestNG.

**Selenium:** It is the most widely used open-source automation testing framework for web application. It supports various programming languages such as Java, Python and C# and more. It provides a suite of tools like Selenium Web driver for browser automation and Selenium IDE for record and play-back functionality.

**Appium:** It is an open-source automation tool for mobile application, both native and hybrid on IOS and Android platforms. It allows testers to write test scripts using popular programming languages such as Java, Javascript and Python. It supports both devices and simulators.

**Cucumber:** It is a behavior driven development framework that enables collaboration between Developers, testers and Business stakeholders. It uses simple, user readable language to read test scenarios that can be easily understood for non-technical people.

**TestNG:** It is an another type of testing framework for Java Applications offering more flexibility and features. It supports parallel test execution, data driven testing and configuration through XML files.

**3)What is Cross browser testing?**

It is a type of software testing that ensures a web application or website works correctly and consistently across different web browsers. Chrome, Firefox, Edge and Safari can interpret and display web pages differently. These differences may lead to varying behaviors, layout issues, or functional problems while accessing the web application from different browsers.

The primary goal of cross browser testing is to ensure there are no inconsistencies while accessing the same web application through different web browsers.

**Browser Capability:** Different browsers implement web standards differentially. One application works perfect on one type of browser while encounter bugs and issues in another. This is resolved in cross browser testing.

**User Experience:** Inconsistent rendering across browsers can result in varying user experiences. Cross browser testing ensures that users get a consistent and functional experience regardless of their browser choices.

**Mobile Devices:** With the increasing mobile phones, cross browser testing also extends to mobile browsers on IOS and Android platforms to ensure the website is responsive and functional on different devices.

**5)What is TDD and BDD?**

TDD (Test-Driven Development) and BDD (Behavior-Driven Development) are two popular software development methodologies that focus on writing tests before or alongside writing the actual code. Both methodologies aim to improve software quality, foster collaboration between team members, and provide clear requirements through automated tests. However, they differ in their approach and the level of abstraction they use for writing tests.

1. Test-Driven Development (TDD):

TDD is a development approach in which developers write tests for the desired functionality before writing the actual code to implement that functionality. The TDD process typically follows these steps:

1. Writing the Test: In TDD, the developer starts by writing a test for a specific piece of functionality. The test is written in a testing framework like JUnit for Java or unit test for Python.

2. Executing the test: After writing the test, the developer runs the test, and it should fail since the functionality has not been implemented yet.

3. Write the Code: The next step is to write the minimum amount of code required to make the test pass. The code is written with the sole purpose of passing the test.

4. Refactor: Once the test passes, the developer refactors the code to improve its design and maintainability while ensuring that all tests continue to pass.

5. Repeat: The process is then repeated for the next piece of functionality, and the cycle continues.

TDD helps ensure that code is testable, encourages developers to focus on writing simple and modular code, and provides a safety net for future code changes. It also acts as documentation, demonstrating how the code is expected to be used.

**2. Behavior-Driven Development (BDD):**

BDD is an extension of TDD that emphasizes collaboration between developers, testers, and business stakeholders. BDD aims to improve communication by expressing the desired behavior of the software in a more human-readable format. The process involves the following steps:

1. Define Behavior: In BDD, the team discusses and defines the behavior of the software in a structured, human-readable language called Gherkin. Gherkin uses keywords like Given, When, and Then to describe the system's behavior in a specific scenario.

2. Write Scenarios: Once the behavior is defined, the team writes test scenarios using Gherkin syntax. These scenarios serve as specifications for the software's behavior.

3. Implement Step Definitions: Each Gherkin step is mapped to a step definition, which is written in the programming language of choice. Step definitions define the actual code that executes the behavior described in the scenario.

4. Run the Tests: The automated tests are executed, and the results are used to verify that the software behaves as expected.

BDD encourages collaboration between different stakeholders and helps ensure that the software's behavior aligns with business requirements. It also facilitates a shared understanding of the system's features and functionality.