TEST AUTOMATION

1. What is Test Automation?

Test Automation is the process of using special software tools to automatically run tests on a software application to check whether it is working correctly.

Instead of a human manually testing every feature of the software, automation uses scripts or programs to perform the testing steps and validate the results.

In short:

Manual Testing → Human tests software step by step
Automation Testing → Computer tests software automatically using scripts

---

2. Why do we use Test Automation?

Testing is a crucial part of software development. Without testing, we can't know if our application is working properly. But manual testing has some problems:

- **Time-consuming:** Testing every feature manually takes a lot of time, especially if the software is big.
- **Reduces human error:** Automated scripts follow predefined steps exactly. This eliminates mistakes caused by manual execution.
- **Repetitive tasks:** Regression testing (testing existing features after a change) is repetitive and boring manually.
- **Not suitable for frequent releases:** Modern software often releases updates frequently (Agile/Scrum). Manual testing can't keep up.
- **Faster execution:** Automated tests run much faster than manual testing. This helps in quick feedback and faster releases.

Automation solves these problems:

- Tests run **faster**.
- Tests run **accurately**, with no human mistakes.
- Tests can be **reused** for every software update.
- You can run tests **24/7**, even overnight.

Test Automation is a way to use software tools to test other software automatically. It saves time, reduces errors, and makes testing large projects manageable. It's not for everything, but it's essential for modern software development, especially in Agile and fast-release environments.

3. Manual Testing vs Automation Testing

Feature	**Manual Testing**	**Automation Testing**
Who performs the test	Human tester	Computer (using scripts/tools)
Speed	Slow	Fast
Accuracy	May have errors	Very accurate
Repeatability	Needs human effort every time	Can be reused multiple times
Cost	Low initial cost, high over time	High initial cost (tools, scripts), low over time
Best for	Exploratory testing, new features	Regression testing, repetitive tests, large projects

Example:

Imagine you have an app with a login page.

- **Manual Testing:** You open the app, enter username and password, click login, check if login works. You do this every time the app updates.
- **Automation Testing:** You write a script once to perform login. Next time, you just run the script, and it automatically tests login hundreds of times without human effort.

---

4. How Test Automation Works

Test automation usually involves these steps:

1. **Identify what to automate**
   Not everything should be automated. Usually repetitive, critical, or high-risk tests are automated.
2. **Choose a tool**
   Examples: Selenium, QTP, TestComplete, Cypress, Robot Framework, etc.
3. **Write test scripts**
   These are programs that tell the software exactly what to do and what to check.
4. **Run tests**
   The automation tool executes the test scripts.
5. **Compare results**
   The tool checks if the application behaves as expected.
6. **Report results**
   Test automation tools generate reports showing which tests passed or failed.

---

5. Advantages of Automation Testing

1. **Faster execution:** Tests run in minutes instead of hours.
2. **Reusability:** Test scripts can be reused for multiple versions of the software.
3. **Accuracy:** No human errors.
4. **Cost-effective in the long run:** Saves effort over repeated testing cycles.
5. **Supports Continuous Integration (CI/CD):** Automation helps test every new build automatically.
6. **Test large projects efficiently:** Especially useful for apps with many features.

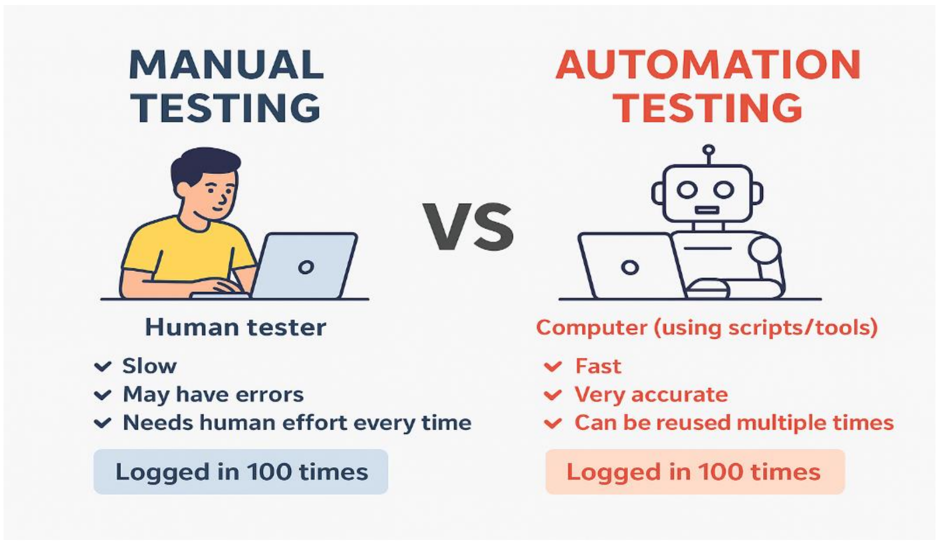
---

6. SCENARIOS THAT CAN BE AUTOMATED

- **Regression test cases:** Regression tests are executed repeatedly to ensure existing functionality is not broken. Automation saves time by running large regression suites efficiently.
- **Smoke and Sanity testing:** These tests validate critical functionalities after each build or fix. Automation helps in quick verification and faster feedback.
- **Data-driven testing:** The same test case is executed with multiple sets of data. Automation handles large data combinations easily.
- **Cross-browser testing:** Tests need to be executed on multiple browsers and versions. Automation ensures consistent behaviour across all browsers.
- **Test cases executed frequently:** Frequently executed test cases are ideal for automation. Automating them reduces manual effort and speeds up testing cycles.

SCENARIOS THAT CANNOT BE AUTOMATED

- **Exploratory testing:** Exploratory testing depends on tester creativity and intuition. It cannot be fully scripted or automated.
- **Usability testing:** Usability testing focuses on user experience and visual comfort. Human judgment is required to evaluate it effectively.
- **Captcha verification:** Captchas are designed to prevent automation. Automating them violates their purpose and is unreliable.
- **OTP validation:** OTPs are dynamic and valid only once. Automation cannot reliably capture and reuse them.
- **One-time test cases:** Test cases executed only once do not justify automation effort. Manual testing is more practical in such cases.
- **Frequent UI-changing features:** Features with frequent UI changes require constant script updates. Automation maintenance becomes costly and inefficient.



AUTOMATION BEST PRACTICES

1. Choose the Right Test Cases to Automate

Not all tests should be automated. Focus on:

- **Repetitive Tests**: Tests you run frequently (e.g., regression tests).
- **Stable Features**: Features that don't change often.
- **High-Risk Areas**: Critical workflows where failures can be costly.
- **Data-Driven Tests**: Tests that run with multiple sets of input data.

Avoid automating:

- Tests that require frequent UI changes.
- Exploratory or ad-hoc testing.
- One-time or low-priority tests.

2. Maintain Clear and Modular Test Scripts

- **Use Page Object Model (POM)** for Selenium or equivalent design patterns.
- Separate **test scripts**, **locators**, and **test data**.
- Keep **functions reusable**.

3. Use Reliable Locators

- Prefer **ID** or **Name** over XPath or CSS when possible.
- Avoid absolute XPaths.
- Use **relative locators** or unique CSS selectors for maintainability

4. Keep Test Data Separate

- Store test data in **Excel, CSV, JSON, or databases**.
- Use **parameterization** to avoid hardcoding.

5. Maintain Test Scripts

- Regularly **review and update** scripts with UI changes.
- Keep the framework **clean and organized**.

Notes by - Ramya R

COMMON MISCONCEPTIONS ABOUT AUTOMATION TESTING

1. Automated Testing Can Replace Manual Testing Completely

Misconception: Many believe that once automated tests are in place, manual testing is no longer needed.
Reality: Automated testing is excellent for repetitive, predictable tasks like regression testing, smoke testing, and data-driven scenarios. However, manual testing is crucial for exploratory testing, usability testing, and scenarios that require human judgment.

2. Automation Is Cheap and Quick

Misconception: Automation is often assumed to be a fast and low-cost solution.
Reality: Initially, automation requires significant investment in tools, frameworks, test data preparation, and skilled resources. ROI is achieved only over time when repeated testing reduces effort.

3. Automated Tests Are Always Reliable

Misconception: If a test script passes, the application is bug-free.
Reality: Automated tests are only as good as their design. Poorly written tests, flaky locators, or outdated scripts can lead to false positives/negatives.

4. All Tests Should Be Automated

Misconception: Every test case should be automated to save time.
Reality: Not all tests are worth automating. Tests that run infrequently, require complex setup, or change frequently may be better off as manual tests.

5. Automation Eliminates Bugs

Misconception: If you have automated tests, your software will be bug-free.
Reality: Automated testing catches known issues, but it cannot discover new, unexpected bugs without human intuition or creative test scenarios.

6. Automation Works Without Maintenance

Misconception: Once scripts are written, they run forever without updates.
Reality: Applications evolve, so automated scripts need regular maintenance to adapt to UI changes, updated workflows, or new features.

FUNCTIONAL AUTOMATION TOOLS

These are the tools that are used to test what the application does(Features, workflows, business logic)

• Selenium	Web-application automation Languages: Java, Python, C#, JavaScript, Ruby
• Playwright	Fast, reliable, UI automation for web Languages: JavaScript, TypeScript, Python, Java, C#
• Cypress	Web automation(JavaScript based applications) Languages: JavaScript, TypeScript
• Robot framework	Keyword driven automation Languages: Python (core), Java (via libraries)
• UFT	Functional testing of desktop and web applications Languages: VBScript
• TestComplete	Functional/UI testing for Web, desktop and mobile apps Languages: JavaScript, Python, VBScript, JScript, DelphiScript
• Appium	Mobile applications Languages: Java, Python, JavaScript, C#, Ruby
• Postman	API functional testing Languages: JavaScript (for scripting)
• RestAssured	API testing automation Languages: Java

NON-FUNCTIONAL AUTOMATION TOOLS

These are the tools which will check how the application behaves, not what it does

• Performance/Load testing	Jmeeter, loadrunner
• Security testing	Nessus, Burp Suite
• Compatibility testing	Browser stack