Testing Automation

What is automation?

- Testing automation involves using software tools to execute pre-scripted tests on a software application before it is released into production.
- It can
  - Enter test data
  - Compare expected and actual results
  - Generate detailed test reports

Examples of Testing Automation Tools

Selenium

- Open-source framework for web testing.
- Supports multiple browsers

Appium

- Open-source tool for mobile app testing.

JUnit

- Popular unit testing framework for Java applications.

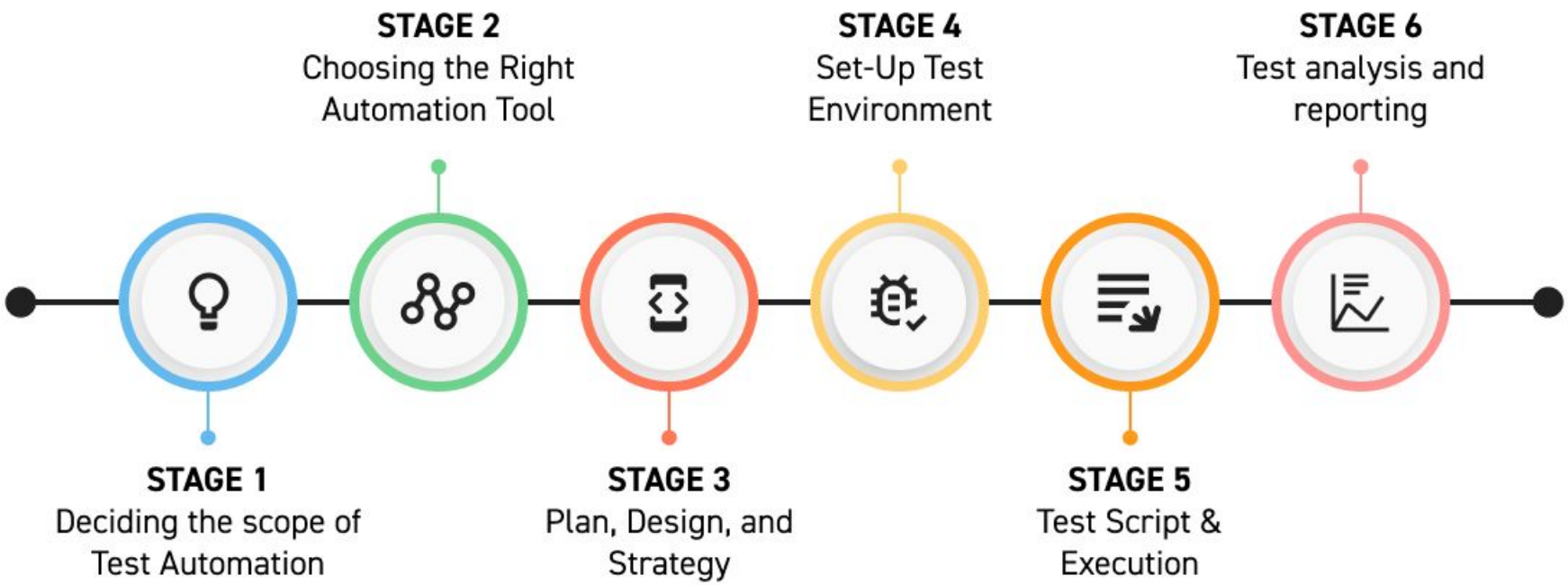
Cypress

- Focused on JavaScript/TypeScript frameworks.

Benefits of Automation

- **Challenges with Manual Testing**:
  - Time-consuming and repetitive
  - Prone to human error
  - Limited test coverage
- **Benefits of Automation**:
  - Increases efficiency and speed
  - Enables large-scale testing
  - Improves consistency and reliability

6 stages of **Automation Testing Life Cycle (ATLC)**



Life Cycle Phases

Phase 1: Deciding the scope of Test Automation:

- Decide which modules of the applications can be automated and which not.
- Which test cases can be automated and how to automated them?

Phase 2: Choosing The Right Automation Tool:

- Create an automation test strategy.
- Define testing tools, framework, and environment.

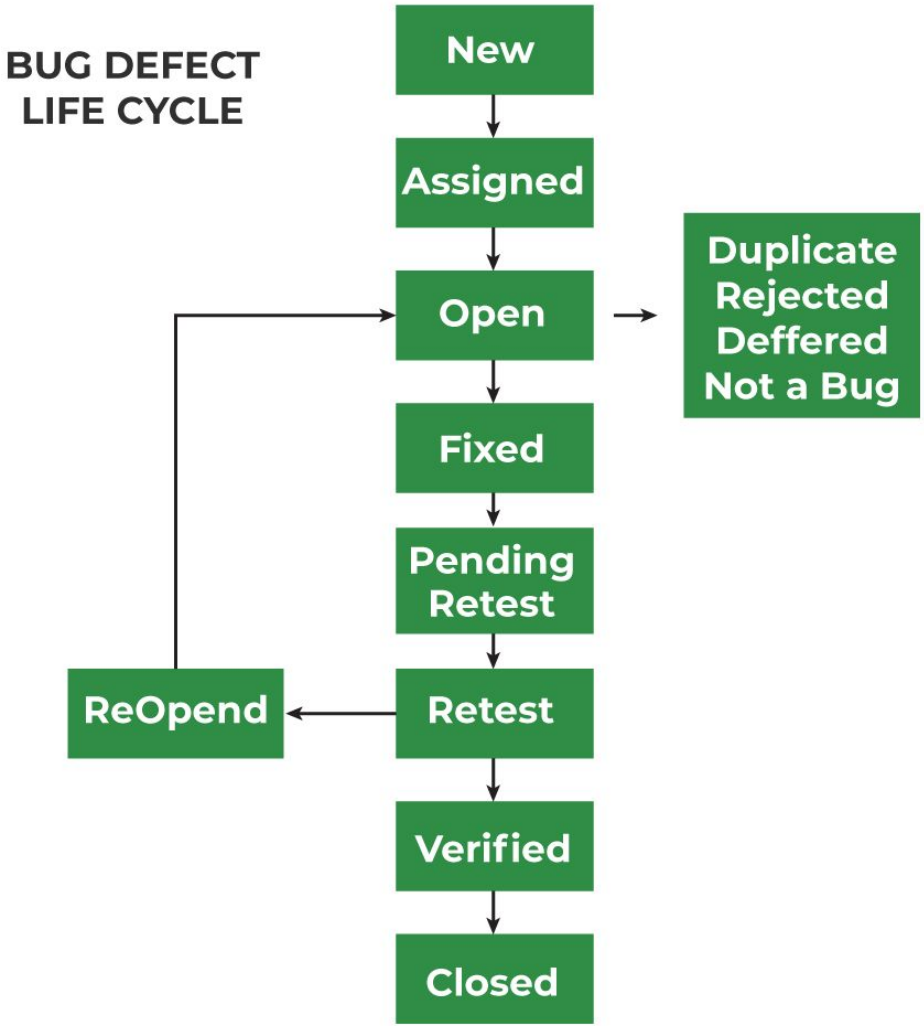
Phase 3: Plan, Design, and Strategy:

- Design automated test cases based on the identified scope.
- Develop test scripts using selected tools.

- **Phase 4: Setting Up the Test Environment**
  - Hardware and Software Setup
  - Test Data and Tools
  - Connections and Integration
- **Phase 5: Automation Test Script Development + Execution**
  - Run scripts in numerous conditions
  - Using Data to Test
  - Dealing with Errors
- **Phase 6: Analysis + Generation of Test Results**
  - Figure out why the tests failed if any
  - Check to see if all of the essential parts have been tested
  - Propose methods to make things better

Defect Life Cycle

- A defect/bug life cycle is the sequence of steps a bug or defect goes through from its identification to its resolution in software development.
- The objective is to easily communicate the current status of the defect and make the defect-fixing process efficient.


BUG DEFECT
LIFE CYCLE
New
Assigned
Open
Duplicate
Rejected
Deffered
Not a Bug
Fixed
Pending
Retest
ReOpend
Retest
Verified
Closed

Life Cycle

New: When any new defect is identified by the tester, it falls in the ‘New’ state.

Assigned: When the defect is assigned to the developer team the status of the bug changes to the ‘Assigned’ state.

Open: In this ‘Open’ state the defect is being addressed by the developer team

Fixed: After necessary changes of codes or after fixing identified bug developer team marks the state as ‘Fixed’.

Pending Request: The developer team passes the new code to the testing team for retesting.

Retest: The tester starts work of retesting the defect to check whether the defect is fixed by the developer or not.

Reopen: If the tester team found that the bug continues like previously even after the developer team has fixed the bug, then the status of the bug is again changed to ‘Reopened’.

Verified: If the tester does not find any kind of defect/bug then the bug is fixed and the status assigned is ‘Verified’.

Closed: If the bug has been resolved and it does not persist then they mark the defect as a ‘Closed’ state.

Additional Phases

1. Rejected: If the developer team rejects a defect if they feel that defect is not considered a genuine defect, and then they mark the status as ‘Rejected’. The cause of rejection may be any of these three i.e Duplicate Defect, NOT a Defect, Non-Reproducible.

2. Deferred: All defects have a bad impact on developed software and also they have a level based on their impact on software. If the developer team feels that the defect that is identified is not a prime priority and it can get fixed in further updates or releases then the developer team can mark the status as ‘Deferred’. This means from the current defect life cycle it will be terminated.

3. Duplicate: Sometimes it may happen that the defect is repeated twice or the defect is the same as any other defect then it is marked as a ‘Duplicate’ state and then the defect is ‘Rejected’.

4. Not a Defect: If the defect has no impact or effect on other functions of the software then it is marked as ‘NOT A DEFECT’ state and ‘Rejected’.

Regression Testing

It involves retesting the previously tested functionalities to verify that recent code changes haven’t adversely affected the existing features.

When to do regression testing?

- When new functionality is added to the system and the code has been modified.
- When some defect has been identified in the software and the code is debugged to fix it.
- When the code is modified to optimize its working.

Types of Regression Testing

- **Retest-all regression testing:** Every test scenario is re-executed using all previous test cases to compare against previous testing outcomes. It is conducted when all existing tests on new code need to be rerun to uncover regressions.
- **Corrective regression testing:** Used when the source code has not been altered or updated. Reuses the existing test cases.
- **Selective regression testing:** Focuses on retesting only the parts of an application that are likely to be affected by new code changes.
- **Progressive Regression Testing:** Involves using new testing scenarios when the product specifications are modified.

Benefits of Regression Testing

- Ensuring Software Stability
- Detecting and Preventing Frequent Defects
- Validating the Impact of Code Modifications
- Mitigating Risks Associated With Software Upgrades