TheAutomationEngineer

QA Cheat Sheets

The Complete Set

All You Need

1.Selenium Cheet Sheet

2.API Cheet Sheet

3.Framework Cheat Sheet

4.Git Cheet Sheet

5.Xpath Cheet Sheet

6.TestNG Cheet Sheet

7.Manual Testing Cheet Sheet

By Achal Singh

TheAutomationEngineer

Selenium Cheat Sheet

Driver Initialization basics

Chrome	WebDriver driver = new ChromeDriver();
Firefox	WebDriver driver = new FirefoxDriver();
Edge	WebDriver driver = new EdgeDriver();
Safari	WebDriver driver = new SafariDriver();

Selenium Locators

Find element by ID

```
WebElement element = driver.findElement(By.id("element_id"));
```

Find element by Name

```
WebElement element = driver.findElement(By.name("element_name"));
```

Find element by Class Name

```
WebElement element = driver.findElement(By.className("element_class"));
```

Find element by XPath

```
WebElement element = driver.findElement(By.xpath("//div[@id='element_id']"));
```

Find element by CSS Selector

```
WebElement element = driver.findElement(By.cssSelector("div#element_id"));
```

Selenium Operations

Open URL	driver.get("https://www.example.com");
Go back	driver.navigate().back();
Go forward	driver.navigate().forward();
Refresh page	driver.navigate().refresh();

Implicit Wait

```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Explicit Wait

```
WebElement element = (new WebDriverWait(driver,
10)).until(ExpectedConditions.elementToBeClickable(By.id("element_id")));
```

Selenium Element Interactions

Type text into input field	element.sendKeys("Text to type");
Click on element	element.click();
Clear input field	element.clear();
Get element text	String elementText = element.getText();

Get attribute value

```
String attributeValue = element.getAttribute("attribute_name");
```

Handling Frames

Switch to frame by ID or Name

```
driver.switchTo().frame("frame_id_or_name");
```

Switch back to main content

```
driver.switchTo().defaultContent();
```

Switch to frame by index

```
driver.switchTo().frame(0);
```

Wait for frame to be available and switch to it

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("frame_id")));
```

Handling Alerts

Switch to an alert and accept it

```
Alert alert = driver.switchTo().alert();
alert.accept();
```

Dismiss an alert alert.dismiss();

Get the text of an alert String alertText = alert.getText();

Input text in an alert prompt

```
alert.sendKeys("Text for prompt");
alert.accept();
```

Handling Windows

Get current window handle

```
String mainWindowHandle = driver.getWindowHandle();
```

Get all window handles

```
Set<String> allWindowHandles = driver.getWindowHandles();
```

Switch to a window by handle

```
String windowHandle = "window_handle";
driver.switchTo().window(windowHandle);
```

Switch back to the main window

```
driver.switchTo().window(mainWindowHandle);
```

Taking Screenshot

Take a screenshot of the whole page

```
File screenshotFile = ((TakesScreenshot)
driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(screenshotFile, new File("screenshot.png"));
```

Take a screenshot of a specific element

```
File elementScreenshotFile =
element.getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(elementScreenshotFile, new
File("element_screenshot.png"));
```

TheAutomationEngineer

API Cheat Sheet

API Test Cases

- Verify `status code` is 200 OK
- Verify response format (`JSON/XML`)
- Verify all expected fields present
- Verify correct data for each field
- Verify response time within limits
- Verify request `parameters` processed
- Verify HTTP method (`GET/POST/PUT/DELETE`)
- Verify `endpoint URL` is correct
- Verify `response headers` correct
- Verify `error handling` for missing methods
- Verify error for non-existent resources

HTTP Status Codes

1xx: Info

100	Continue
101	Switching

2xx: Success

200	OK
201	Created
202	Accepted
204	No Content

3xx: Redirect

301	Moved Permanently
302	Found
304	Not Modified

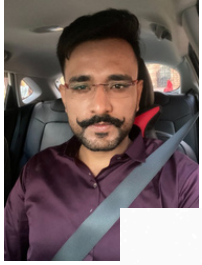
4xx: Client Error

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed

5xx: Server Error

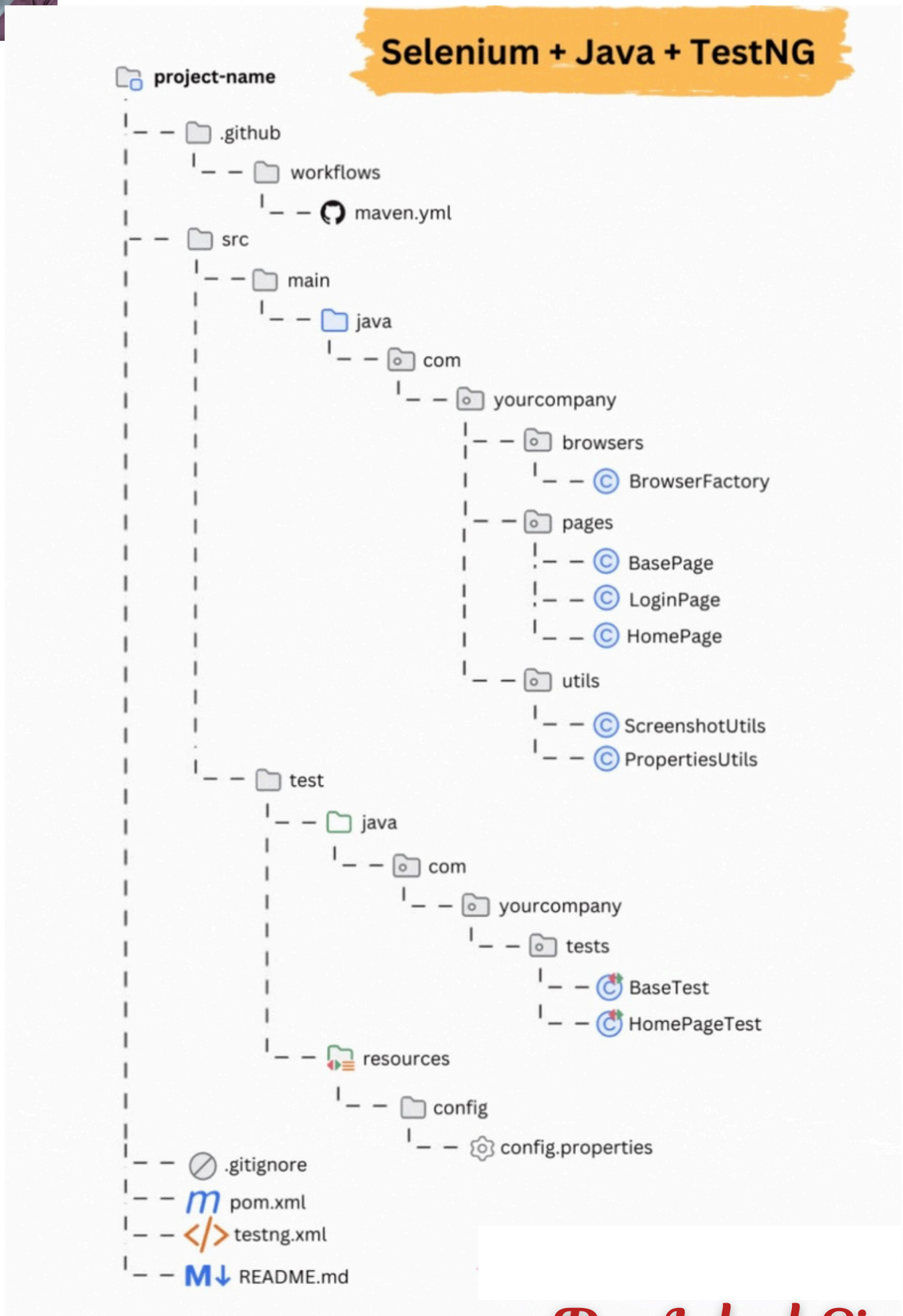
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

By Achal Singh



TheAutomationEngineer

Selenium + Java + TestNG



By Achal Singh

TheAutomationEngineer

Git Commands Cheat Sheet

Git Commands Cheat Sheet

```
# Git setup commands
$ git config --global user.name "<name>"          # Set username
$ git config --global user.email "<email>"        # Set email
$ git config --list                               # View all config settings

# Git repository commands
$ git init                                        # Initialize a new Git repository
$ git clone <repo-url>                            # Clone a repository
$ git remote add origin <url>                     # Add remote origin
$ git remote -v                                   # Show remote URLs

# Git basic snapshotting
$ git status                                      # Check the status of files
$ git add <file>                                  # Add a file to staging
$ git add .                                       # Add all files to staging
$ git commit -m "<message>"                       # Commit staged changes
$ git commit -am "<message>"                      # Add & commit in one step

# Git branching
$ git branch                                      # List branches
$ git branch <branch-name>                        # Create a new branch
$ git checkout <branch-name>                      # Switch to a branch
$ git checkout -b <branch-name>                   # Create and switch to new branch
$ git merge <branch-name>                         # Merge branch into the current branch
$ git branch -d <branch-name>                     # Delete a branch

# Git updating & publishing
$ git pull                                        # Fetch and merge changes from remote
$ git push                                        # Push changes to remote
$ git push -u origin <branch-name>                # Push new branch to remote

# Git undoing changes
$ git reset <file>                                # Unstage a file
$ git checkout -- <file>                          # Discard changes in a file
$ git revert <commit-id>                          # Revert a commit

# Git log & history
$ git log                                         # Show commit logs
$ git log --oneline                               # Show condensed logs
$ git diff                                        # Show changes between commit, branches,files

# Git stash commands
$ git stash                                       # Stash current changes
$ git stash pop                                   # Apply stashed changes
$ git stash list                                  # List all stashes

# Git tag commands
$ git tag                                         # List tags
$ git tag <tag-name>                              # Create a new tag
$ git push origin <tag-name>                      # Push a tag to remote
```

By Achal Singh

TheAutomationEngineer

XPath cheat sheet

Types of XPath

1. **Absolute XPath**
   /html/body/div/div/section/section
   /div/div/div/input
2. **Relative XPath**
   //*[@id='row1']/input

XPath formula

//tag[@attribute='value']

Example:
//div[@class='round-button']

SYNTAX

EXPLAINATION:

- **/** — **Absolute XPATH** - Starts at the top of the DOM, or a direct decendant (child)
- **//** — **Relative XPATH** - Looks anywhere on the page. Starts at any element on the page with this tag, or an indirect decendant
- **div** — *Example of an element tag*
- **[ ]** — **Predicates** - Used to find a specific node or a node with a specific value
- **@** — *Attribute*
- **="** — Specific attribute value to search for
- **.** — Uses the node that is in context
- **..** — Selects the parent of the current node

/ vs //

/ - short for child node
// - short for descendant or self node

at the beginning of xpath
/ - selects a root element
// - selects element anywhere on the page

in the middle of xpath
/ - selects child of the element
// - selects descendant of the element

// vs .//

dot introduces a relative location path, starting at the context node

Examples:
WebElement parentElement =
driver.findElement(By.id("someId"));

1. By childLocator1 = By.xpath("//input");
   parentElement.findElement(childLocator1);
   This will ignore parentElement and will search for input element anywhere on the page
2. By childLocator2 = By.xpath(".//input");
   parentElement.findElement(childLocator2);
   This will search input element that is descendant of the parentElement

Text Function

** Full element text **
//div[text()='Full element text']

Contains Function

Work with attribute values
** **
//button[contains(@id,'username')]
And with text
** Lets learn how to automate tests **
//div[contains(text(),'how to automate')]

Starts-With Function

Work with attribute values
<input class="input-field">
//input[starts-with(@class,'input')]
And with text
** This page is created to be able to reproduce the most common Selenium Exceptions. **
//p[starts-with(text(),'This page is created')]")

not Function

//div[not(@id='login')]
//a[not(text()='Click here')]
//input[not(contains(@class,'input'))]
//p[not(starts-with(text(),'Selenium')]

XPath Operators

Using **'OR'**
//button[@name='Add' or @name='Remove']
Using **'AND'**
//button[@id and @class='btn'and @style and @name='Add']
//button[@id][@class='btn'][@style][@name='Add']

XPath wildcards

//*[@class] - *Element with any tag that has 'class' attribute*
//button[@*='btn'] - *Any button element where any attribute has value 'btn'*
//div[@*] - *Div element that has any attribute*

Index

//tag[index]
//h5[2]
- get second element with tag h5
//tag1[index1]/tag2[index2]
(//div[@class='row'])[3]/h5[2]
- find third div element that has class row, and then get second h5 direct child
(//tag1[@attribute='value']/tag2)[index]
(//div[@class='row']/input[@class='text'])[2]
- get all input elements with class text that are children of any div elements with class row, and then get second element from that list

Position functions

position()=2 works same way as index [2]
//h5[position()=2] same as //h5[2]

Operators we can use with position

position()=2	Equal
position()!=2	Not equal
position()>2	Greater than
position()>=2	Greater than or equal to
position()<2	Less than
position()<=2	Less than or equal to

last() - *get last element from the list*
//h5[last()]
We can also use subtraction with the last function
//h5[last()-1]

Finding elements relative to other elements

//div[./input]
Find div element that has input child
//input[parent::div[@id='row2']]
The same as //div[@id='row2']/input

Selecting Several Paths

Use the vertical bar to combine two or more XPath expressions into one
//div[@id='row1']/button |
//div[@id='row1']/input
//h2 | //h5 | //p

SVG elements

To get to SVG element, use wildcard in place of tag name, and use name function for the SVG element tag
//*[name()='svg']//*[name()='rect' and @transform]
//*[name()='rect' and contains(@transform,'rotate (45.0')]

XPath axes

Formula:
axisname::nodetag[predicate]

XPath axes:
ancestor:: ancestor
Selects all ancestors of the current nodes
descendant:: descendant
Selects all children, grand-children etc... of the current node
parent:: parent *Only the parent of the current node*
following-sibling:: *Siblings after the current node*
preceding-sibling:: *Siblings before the current node*

Examples:
//button[@id='btn']/parent::div
Find div parent of button element with id "btn"
//button[@id='btn']/following-sibling::label
Find label sibling that is located after button element with id "btn"
//button[@id='btn']/preceding-sibling::label
Find label sibling that is located before button element with id "btn"
//button[@id='btn']/parent::div/following-sibling::div/div
combination of few axes in the same expression

By Achal Singh

Java - TestNG Cheat Sheet

TheAutomationEngineer

TestNG Annotations

- **@Test** - Denotes a test method.
- **@BeforeClass** - Runs once before the first test method in the current class.
- **@AfterClass** - Runs once after all the test methods in the current class.
- **@BeforeMethod** - Runs before each test method.
- **@AfterMethod** - Runs after each test method.
- **@BeforeSuite** - Runs once before all tests in the suite.
- **@AfterSuite** - Runs once before all tests within the current <test> tag.
- **@AfterTest** - Runs once after all tests within the current <test> tag.

TestNG Attributes

- **priority** - Set to prioritize the test method for execution.
- **enabled** - Set to ignore the test method for execution.
- **invocationCount** - Set to run the test method multiple times.
- **invocationtimeOut** - Maximum time period provided the tests case for all invocations count set.
- **timeout** - Total time period to the test case to completely execute its test case.
- **dependsOnMethods** - Inject dependency to the test methods.
- **groups** - To categorize the test case to execute under a specified group.
- **dependsOnGroups** - Inject dependency to the test methods based on the groups to execute.

TestNG Assertions

- **assertEqual(String actual, String expected);**
  Asserts the two strings are equal or not
- **assertEquals(Boolean actual, Boolean expected);**
  Asserts that two boolean values are equal
- **assertTrue(boolean condition);**
  Asserts that the condition is true.
- **assertFalse(boolean condition);**
  Asserts that condition is false.
- **assertsNull(Object object);**
  Asserts that an object is null
- **assertsNotNull(Object object);**
  Asserts that an object is not null.
- **assertsSame(Object object);**
  Asserts that two objects refer to the same object
- **assertsNotSame(Object actual, Object expected);**
  Asserts that two objects do not refer to the same objects.

TestNG Configuration

- **testng.xml**
  XML file to configure TestNG tests.
- **<suite>**
  Represents a suite of tests.
- **<test>**
  Represents a test within a suite.
- **<classes>**
  Specifies the classes to be included in a test.
- **<packages>**
  Specifies the packages to be included in a test.
- **<methods>**
  Specifies the methods to be included in a test.
- **<groups>**
  Specifies the groups to be included in a test.
- **<listeners>**
  Specifies the listeners to be used in a test.

By Achal Singh

TheAutomationEngineer

Selenium Java Folder Structure

```
selenium-ecommerce-framework/
├── src/
│   ├── main/
│   │   └── java/
│   │       ├── pages/
│   │       │   ├── LoginPage.java
│   │       │   ├── HomePage.java
│   │       │   ├── ProductPage.java
│   │       │   ├── CartPage.java
│   │       │   ├── CheckoutPage.java
│   │       │   └── OrderConfirmationPage.java
│   │       │
│   │       └── utils/
│   │           ├── DriverFactory.java
│   │           ├── ConfigReader.java
│   │           ├── TestDataReader.java
│   │           └── ScreenshotUtil.java
│   │
│   └── test/
│       └── java/
│           ├── stepdefinitions/
│           │   ├── LoginSteps.java
│           │   ├── HomeSteps.java
│           │   ├── ProductSteps.java
│           │   ├── CartSteps.java
│           │   ├── CheckoutSteps.java
│           │   └── OrderConfirmationSteps.java
│           │
│           ├── runners/
│           │   └── TestRunner.java
│           │
│           └── hooks/
│               └── Hooks.java
│
├── features/
│   ├── login.feature
│   ├── home.feature
│   ├── product.feature
│   ├── cart.feature
│   ├── checkout.feature
│   └── order_confirmation.feature
│
├── config/
│   └── config.properties
│
├── testdata/
│   ├── users.csv
│   ├── products.json
│   └── shippingDetails.xlsx
│
├── reports/
│   ├── extent/
│   │   └── index.html
│   └── cucumber/
│       └── cucumber.html
│
├── pom.xml
├── testng.xml
└── README.md
```

🟡 Gherkin Features

- Specifies the tests using Gherkin syntax (Given–When–Then)
- Focuses on user behavior and expected outcomes.

🔴 **Step Definitions**

- Files defining Cucumber steps (Given, When, Then)
- bridge feature files to the application under test, performing actions and assertions.

🔷 **Page Object Models**

- Each Java class represents a specific page of the application.
- It contains locators (for elements) and methods (for actions).

🟤 Test Data

- Contains various formats of test data,
- User Login Credentials
- Product IDs
- Shipping Details

🟢 Reports

- Generated after test execution, including detailed Cucumber
- Extent reports to track test status, screenshots, and execution logs.

By Achal Singh

TESTING TECHNIQUES

Smoke Testing
Quick round of basic functionality tests

Sanity Testing
Focused testing on specific module aftral fix

Regression Testing
Re-running tests to ensure existing functionality still works

Retesting
Testing failed tests after defects are fixed

Exploratory Testing
Testing without predefined cases or documentation

System Testing
Testing the complete, integrated system

UAT
Final validation by the end-users

Alpha Testing
Testing performed in house by thy QA team

Bug
Mismatch between expected and actual behavior

Test Case
Step-by-step set of actions to validate functionality

Test Scenario
High-level idea of what to test

Test Plan
Document outlining scope, strategy, and schedule of testing

Ad-hoc Testing
Informal, unstructured testing to find defects

Compatibility Testing
Checks app behavior across different environments

Security Testing
Ensures the app is safe from vulnerabilities

TheAutomationEngineer

30 QA TERMS EVERY TESTER SHOULD KNOW

Whether you're just starting in testing or you're already knee-deep in bugs, these terms are must-know. Save this for quick reference!

1. **Test Scenario**
   High-level functionaality to test
2. **Test Case**
   Step-by-step actions to verify a scenario
3. **Test Suite**
   Collection of test cases for a module
4. **Test Data**
   Inputs used fortesting
5. **Test Plan**
   Stralagy, scope, and schedule of testing
6. **RTM (Requirement Traceability Matrix)**
   Mapping requirements to st cases
7. **Funetional Testing**
   Valldates business funcionalities
8. **Non-Functional Testing**
   Focuses on performance, security, usability
9. **Exploratory Testing**
   Simultaneous leaming and testing
10. **Smoke Testing**
    Basic check of cittical features
11. **Sanity Testing**
    Quick checke after minor changes
12. **Regression Testing**
    Ensures new changes dont break existing features
13. **Retesting**
    Verifying a bug fix
14. **Defect Life Cycle**
    Bug statue journey fron discovery to closure
15. **Severity**
    Impact of the defect
16. **Priority**
    Urgency to fix the defect
17. **BVA (Boundary Value Analysis)**—Testing edge values of input ranges
18. **EP (Equivalence Partitioning)**
    Grouping inputs into valid/invalid sets
20. **QA (Quality Assurance**
    Process sriented, prevention of defects

BONUS TERMS YOU SHOULDN'T MISS!

21. **QC (Quality Control)**
    Product-oriented, identifying defects
22. **Bug Leakage**
    Bugs missed during testing
23. **Bug Release**
    known bugs intentionally released
29. **Bug Leakage**
    Checking during testing
29. **Integration Testing**
    Validaring interactions between modules
30. **E2E (End-to-End) Testing**
    Testing complete application flow from start to finish

By Achal Singh