**Development Plan For Tester**

**1. Research all key tools / technologies - gather a full list by asking industry ppl, network and seek.**

* **Test Management**: JIRA, TestRail, Zephyr
* **Bug Tracking**: JIRA, Bugzilla
* **Automation Tools**: Selenium, Postman, Cypress (basic exposure)
* **API Testing**: Postman, REST Assured (concepts only)
* **Version Control**: Git / GitHub
* **CI/CD Basics**: Jenkins, GitHub Actions (awareness only)
* **Databases**: SQL basics (MySQL, PostgreSQL)
* **Basic Programming**: Python or Java fundamentals
* **Agile/Scrum Methodology**: Confluence, Jira Boards
* **Cloud Awareness**: AWS or Azure (very basic awareness, not deep)

**2. Create a nice table of the above list and identify all possible resoureces links**

|  |  |  |
| --- | --- | --- |
| **Tool / Concept** | **LinkedIn Learning Course (Preferred)** | **Free Resources (YouTube / Docs)** |
| **JIRA** | Learning Jira Software | Automation step by step |
| **Zephyr** |  | Automation step by step |
| Jmeter |  | Automation step by step |
| **Selenium** | Selenium Essential Training | Automation step by step |
| **Postman** | Postman Essential Training | Automation step by step |
| OWsap Zap |  | [SecNet Lab](https://www.youtube.com/@SecNetLab) |
| **SQL Basics** |  | W3Schools SQL |
| **GitHub** |  | GitHub Docs |
| **Jenkins (CI/CD)** |  | Automation step by step |
| **Agile** | Agile testing |  |
| **Cloud Basics (AWS)** |  |  |

**3. Create a two weeks plan to cover all section 2 identified courses/videos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Day** | **Tool / Topic** | **Learning Resource** | **Task** | **Notes to Capture (4 lines)** |
| 1 | Testing Foundations | LinkedIn | Methods | Learn manual vs automation testing. Understand SDLC & STLC. Note defect lifecycle and test levels. Summarize key testing types. |
| 2 | JIRA Basics | LinkedIn: Learning Jira Software + YouTube | Create a JIRA project, add a task | Practice creating project board. Add task, bug, and assign user. Learn workflow status. Note difference between task and story. |
| 3 | Zephyr | YouTube | Write 2 sample test cases | Explore test case structure. Add steps, expected result. Link to JIRA issue. Note how to execute manually. |
| 4 | Zephyr | YouTube | Perform automation test | Learn automation integration setup. Execute automated test cycle. Review report and logs. Note pass/fail status meaning. |
| 5 | OWASP ZAP | YouTube | Perform Active Scan, find alerts | Copy website URL and run scan. View vulnerabilities & severity. Learn to interpret alerts. Note how to export report. |
| 6 | SoapUI | YouTube | Perform request | Create a new SOAP project. Send sample request and review response. Understand XML format. Note response codes. |
| 7 | JMeter | YouTube | HTTP Request | Add Thread Group + HTTP Sampler. Run test and view results tree. Measure response time. Note performance metrics. |
| 8 | Postman | LinkedIn: Postman Essential Training | Send a GET request to public API | Understand request structure. Send GET call and view response JSON. Check status 200. Note key headers and params. |
| 9 | Postman | Postman YouTube | Run a POST request | Create POST request with body data. Send to test API endpoint. Observe status 201/409. Note environment variable usage. |
| 10 | Selenium Basics | LinkedIn: Selenium Essential Training | Open browser → Google → Click search | Setup Selenium & WebDriver. Automate browser open and search. Capture element locators. Note test script flow. |
| 11 | SQL Basics | YouTube | Run SELECT \* FROM users; query | Learn basic SQL syntax. Run queries in DB tool. View table results. Note WHERE and ORDER BY usage. |
| 12 | GitHub | YouTube | Create repo and commit | Initialize repository. Add README file. Commit and push via Git Bash. Note commit message format. |

|  |  |  |  |
| --- | --- | --- | --- |
| Day | Tool / Topic | Learning Resource | Task (“Hello World”) |
| 13 | Cypress Basic |  | Run sample Cypress test |
| 14 | Jenkins / CI Basics | DevOps CI/CD Foundations | Run sample Jenkins build (or watch demo if setup hard) |
| 15 | Agile / Scrum | Agile Foundations | Write sprint backlog (mock) |
| 16 | Cloud Awareness (AWS) | Cloud Computing Foundations | Launch AWS Free Tier account |
| 17 | GitLab |  |  |
| 18 | Browser Developer Tools |  |  |
| 19 | VirtualBox, VMware |  |  |
| 20 | TestRail |  |  |
| 21 | Fiddler |  |  |
| 22 | LoadRunner |  |  |

**4. C1: 20 Common Basic Tech Q & A (Entry-Level Tester)**

Testing Fundamentals

1. What is the difference between verification and validation?

Verification : Checking if we built the product correctly, following the design or plan (e.g., checking code against the design).

Validation : Checking if we built the right product that the user actually needs (e.g., checking if the software meets user needs).

1. What are functional vs non-functional testing? Give examples.

Functional testing : Tests *what* the system feature does (e.g., checking if login works).

Non-functional testing : Tests *how* the system works (e.g., speed, security, usability).

1. What is the Software Testing Life Cycle (STLC)?

It is a sequence of steps folowed during the testing process to ensure the software is tested correctly and meets quality standards. STLC covers phases like requirement analysis, test planning, test case design, test execution, defect reporting, and closure.

1. Explain smoke vs regression testing.

S**moke Testing**: A quick test done before full testing to check if the main functions of the software are working properly.

**Regression Testing** :Testing to make sure new changes or updates haven’t broken any existing features of the software

1. Difference between manual and automation testing?

Manual Testing is perform manually by a human without using any tools, to find bugs.

Automation Testing perform by tools or scripts, where instructions are given to automatically check for defects.

1. What are positive test cases vs negative test cases?

**Positive Test Case** : Tests the software with valid input to check if it works as expected.

**Negative Test Case** : Tests the software with invalid or unexpected input to check if it handles errors properly

Defects & Bug Tracking

1. What is a defect/bug, and how is it different from error and failure?

Bug/defect: mismatch or difference between expected and actual results.

Error: human mistake in code/design.

Failure: system doesn’t perform the expected function.

1. Explain defect life cycle.

Defect Life Cycle → The journey of a defect/bug from the time it is found until it is fixed and closed.

Steps in Defect Life Cycle :

New → Defect is reported for the first time.

Assigned → Defect is assigned to a developer to fix.

Open → Developer starts working on the defect.

Fixed/Resolved → Developer fixes the defect.

Retest → Tester checks if the defect is fixed.

Closed → Defect is confirmed fixed and closed.

Reopen → If the defect still exists, it is reopened.

1. What key details should you include in a bug report?

**Bug ID/Title** – A short, clear name for the bug.

**Description** – What the bug is and what went wrong. Step-by-step instructions to show the bug.

**Expected Result** – What should happen.

**Actual Result** – What actually happened.

**Severity/Priority** – How serious or urgent the bug is.

**Environment** – Where the bug occurred (OS, browser, device, etc.).

**Attachments** – Screenshots, logs, or videos showing the bug.

Test Design & Execution

1. What is a test case? Components?

**Test Case** : A document with steps, inputs, and expected results to check if a feature works correctly.

**Test Component** : A specific part or module of the software that is tested, like a login page or payment feature

1. Difference between test plan and test strategy?

**Test Plan** : A document that describes what to test, how to test, who will test, and when to test. It’s a detailed plan for the testing process.

**Test Strategy** : A high-level document that describes the overall approach and goals of testing, including tools, techniques, and standards to follow.

1. What is boundary value analysis? Example?

.Boundary Value Analysis (BVA) : A testing technique where you test the values at the edges or boundaries of input ranges, because errors often occur at the limits.

Example: If an input accepts 1 to 10, you test 0, 1, 10, 11 to check for defects.

1. What is equivalence partitioning?

Equivalence Partitioning (EP) : A testing technique where input data is divided into groups (partitions) that are expected to behave the same. You test one value from each group instead of all possible values.

Example: If an input accepts 1–10:

Valid partition: 1–10 : test input 5

Invalid partitions: <1: test input 0 & for >10 : test input 11.

Agile & Collaboration

1. What role does QA play in Agile methodology?

QA is involved throughout the development process, not just at the end. They collaborate with developers and product owners and help define acceptance criteria and user stories. They perform continuous testing during each sprint and ensure the product meets quality standards before release.

1. What happens in a daily stand-up?

A short daily meeting where each team member Share progress, plan the day, and highlight problems or blockers.

Tools & Basics

1. What is JIRA used for in testing?

JIRA is used for managing tasks, user stories, test cases, and defect tracking in Agile projects.

1. What is API testing, and why is it important?

API Testing the Application Programming Interface (API) to check if it works correctly, returns the right data, and handles requests properly.

It ensures the backend works correctly, systems communicate properly, and bugs are caught early before affecting user.

1. What is SQL, and why do testers need to know it?

SQL is used to query databases.

Testers need it to check and verify data.

1. What is Selenium, and how does it help in testing?

Selenium is a tool that automates testing of web applications to save time and reduce manual effort.

1. Why is version control (like Git) useful for testers?

Version control helps testers track changes, test the right version, and collaborate with developers easily.

**C2: Resume/Project-specific Technical Q&A (Entry-Level Tester)**

**JMeter (Performance Testing)**

1. What is JMeter and when would you use it?  
 JMeter is an open-source tool used for performance, load, and stress testing of web applications, APIs, and servers. I would use it to simulate multiple users accessing the system at the same time to check how the application performs under load.

2. How do you create a basic test in JMeter?  
Steps:

Create a Test Plan.

Add a Thread Group (define number of users, ramp-up time, loop count).

Add a Sampler (e.g., HTTP Request).

Add Listeners (View Results Tree, Summary Report) to see results.

Run the test.

3. What is the difference between load testing and stress testing in JMeter?  
 Load testing checks how the system performs under expected user load.

Stress testing pushes beyond normal limits to see when the system breaks.

**Jira (Bug/Project Management)**

4. How do you log a bug in Jira?  
 Steps:

Go to the project → click Create.

Select Issue Type = Bug.

Enter details: Summary, Description (steps to reproduce, expected vs actual), Severity/Priority, Environment, Attach screenshot/log.

Assign to developer.

5. What is the difference between Epic, Story, Task, and Bug in Jira?

Epic: High level of requirement.

Story: Smaller functionality under Epic.

Task: Work item assigned to team members.

Bug: Defect/issue in the system.

6. How do you track the status of a bug in Jira?  
 By checking its workflow (Open → In Progress → In Review → Resolved → Closed). Status updates show progress.

**Zephyr (Test Management in Jira)**

7. What is Zephyr used for?  
 Zephyr is a Jira plugin used for test case management. It helps create, organize, execute, and track test cases directly inside Jira.

8. How do you execute a test case in Zephyr?

Go to the test cycle.

Select the test case.  
Select Test script  
Select Automate test

Execute → choose status (Pass, Fail, Blocked, WIP).

Attach evidence (screenshots, logs).

8. Can Zephyr be linked with Jira issues?  
 Yes, test cases in Zephyr can be linked to Jira Stories/Bugs so you can trace requirements and defects.

9. Can you explain how you created and managed a test case in JIRA/Zephyr?

In my practice project, I set up Zephyr inside JIRA. I created a test case with steps, expected results, and linked it to a user story.

**OWASP ZAP (Security Testing)**

10. What is OWASP ZAP used for?  
 It’s an open-source tool for finding security vulnerabilities in web applications, like SQL injection, XSS, or authentication issues.

11. What is the difference between Active Scan and Passive Scan in ZAP?

Passive Scan: Monitors traffic without modifying requests (safe, non-intrusive).

Active Scan: Actively attacks the app with payloads to find vulnerabilities.

12. Give an example of a common vulnerability you can find with ZAP.  
 Cross-Site Scripting (XSS): injecting malicious scripts into input fields.

**Postman (API Testing)**

13. How do you test an API in Postman?

Select request type (GET, POST, PUT, DELETE).

Enter API URL.

Add headers (e.g., Content-Type, Authorization).

Add request body (if needed).

Send → check response (status code, body, time).

14. What is the difference between GET and POST request in API testing?

GET: Retrieves data (safe, no body).

POST: Submits data (has request body, modifies server).

15. How do you automate API tests in Postman?  
 Using Collections + Test scripts. You can write JavaScript in the “Tests” tab for assertions (e.g., check status code = 200).

**SoapUI (Basic API Testing for SOAP/REST)**

16. What is SoapUI used for?  
 It’s a tool for testing SOAP and REST APIs. It allows functional, security, and load testing.

17. What’s the difference between SoapUI and Postman?

SoapUI: Supports both SOAP & REST APIs, more advanced for enterprise testing.

Postman: Easier, mostly REST-focused, great for quick/manual testing.

18. How do you create a SOAP request in SoapUI?

Create a new SOAP project.

Add the WSDL URL.

Select operation → SoapUI auto-generates request XML.

Enter parameters → Send → check response.

**Automation Tools (Selenium )**

**19. What is Selenium?**  
Selenium is an open-source automation tool used for testing web applications across different browsers and platforms. It supports multiple programming languages like Java, Python, and C#.

**20. What are the main components of Selenium?**

* **Selenium IDE** – Record & playback tool for beginners.
* **Selenium WebDriver** – Automates browsers by directly communicating with them.
* **Selenium Grid** – Runs tests in parallel on multiple machines/browsers.

**21. Which programming language do you use with Selenium?**  
I mainly use Pythonbecause it’s easy to learn, supports multiple libraries, and integrates well with testing frameworks like PyTest .

**22. How did you create your first Selenium script (using PyTest & PyCharm)?**

I created my first Selenium automation script in **PyCharm** using **PyTest** as the test framework.  
I started by installing the required packages: pip install selenium pytest webdriver-manager

Then, I wrote a simple test :

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from webdriver\_manager.chrome import ChromeDriverManager

import pytest

def test\_google\_search():

driver = webdriver.Chrome(ChromeDriverManager().install())

driver.get("https://www.google.com")

search\_box = driver.find\_element(By.NAME, "q")

search\_box.send\_keys("Selenium Python")

search\_box.send\_keys(Keys.RETURN)

assert "Selenium" in driver.title

driver.quit()

I ran it. When the browser opened, typed the text, and verified the title, it confirmed my setup worked.

**23. What challenges did you face while using Selenium?**  
Initially, setting up the Chrome driver path was tricky. I solved it by installing **webdriver-manager** (Python) and managing browser versions properly. After setup, I focused on improving locators and waiting strategies.

**24. What frameworks can be integrated with Selenium?**

* **PyTest / Unittest** (Python)
* **TestNG / JUnit** (Java)  
  They help in managing test suites, setup/teardown, and reporting.