1- MONTH QA FRESHER TRAINING GUIDE

Role: QA Trainee (Fresher)
Duration: 4 Weeks (5 days per week)

How to Use This Document

- Follow **each day in order** (Day 1 → Day 20).
- Every day has:
    - **What you will learn (Theory)**
    - **What you will do (Practical Task)**
    - **Small Checkpoint / Assignment**
- Keep a **notebook or Google Doc** to store:
    - Notes
    - Test cases
    - Bugs you find
    - Questions to ask your mentor

WEEK 1 – QA BASICS, SDLC, STLC & TEST CASES

Day 1 – Introduction to QA & SDLC

Goal: Understand what QA is and how software is developed.

1. Learn (Theory)

- What is Software Testing? Why is it important?
- Difference between **Developer** and **Tester**.
- What is **SDLC (Software Development Life Cycle)**?
    - Common models: **Waterfall, Agile (Scrum)**.

2. Do (Practical Task)

- Search and read about **Waterfall vs Agile** (any basic article or video).
- Write in your notes:
    - 3 differences between Waterfall and Agile.
    - One real-life example where Agile would be better.

3. Checkpoint (End-of-day)

- Can you explain SDLC in 2–3 sentences?
- Can you list the main phases: **Requirement → Design → Development → Testing → Deployment → Maintenance**?

Day 2 – STLC & Role of QA

Goal: Learn how testing itself has a lifecycle.

1. Learn (Theory)

- What is **STLC (Software Testing Life Cycle)**?
  STLC Common Phases:

    1. Requirement Analysis
    2. Test Planning
    3. Test Case Design
    4. Test Environment Setup
    5. Test Execution
    6. Test Closure

- Difference between **QA** and **QC**.
- What is a **Defect Life Cycle** (New → Open → Fixed → Retest → Closed, etc.)?

2. Do (Practical Task)

- Draw a simple STLC flow in your notebook.
- Beside each phase, write **one line** about what happens there.

3. Checkpoint

- Be able to **describe STLC** to your mentor in simple English.

Day 3 – Understanding Requirements & Test Scenarios

Goal: Learn how to read requirements and think of what to test.

1. Learn (Theory)

- What is a **Requirement**? (BRD, SRS – basic idea only)
- What is a **Test Scenario** vs **Test Case**?

2. Do (Practical Task)
Use this simple requirement:

> "The system should allow the user to log in using email and password."

- Write **at least 5 test scenarios** for this login feature.
  Example:
  - Verify login with valid email and password.
  - Verify login with invalid password.
  - etc.

3. Checkpoint

- Share your test scenarios with mentor / senior and ask:
  - "Did I miss any important scenario?"

Day 4 – Writing Test Cases (Step-by-step)

Goal: Start writing proper test cases.

1. Learn (Theory)

- What is a **Test Case**?
- Basic fields in a test case:
    - Test Case ID
    - Test Scenario / Title
    - Precondition
    - Test Steps
    - Test Data
    - Expected Result
    - Actual Result
    - Status (Pass/Fail)

2. Do (Practical Task)

- Open any common app: e.g., **Gmail login / any sample login page**.
- Write **10 test cases** for the login screen in a table (Excel or Google Sheet).

3. Checkpoint

- Show your test cases to a mentor/senior for feedback.

Day 5 – Test Plan Basics & Weekly Review

Goal: Understand what a Test Plan is and revise Week 1.

1. Learn (Theory)

- What is a **Test Plan**?
- Basic contents (at high level):
    - Scope
    - Features to be tested
    - Test strategy (what types of testing)
    - Resources, tools
    - Risks & assumptions

2. Do (Practical Task)

- Create a **mini test plan** for the Login feature in a simple document:
    - What you will test (e.g., valid/invalid login, UI checks).
    - What you will not test (e.g., performance).
    - Assumptions (e.g., user already registered).

3. Weekly Assignment (Weekend)

- Select **any one app** (mobile or web): e.g., WhatsApp, Amazon, Swiggy, etc.
    - Write:
        - 10 test cases
        - 3–5 test scenarios
        - 1 short test summary (what you tested + what issues you found)

WEEK 2 – TESTING TYPES, DESIGN TECHNIQUES & BUG REPORTING

Day 6 – Types of Testing

Goal: Get familiar with different testing types.

1. Learn (Theory)

- **Functional vs Non-Functional Testing.**
- Some important types:
    - Smoke, Sanity, Regression
    - Integration Testing
    - System Testing
    - UAT (User Acceptance Testing)

2. Do (Practical Task)

- Pick any app (e.g., e-commerce).
- Write 5 examples of **functional tests** and 5 of **non-functional tests**.

3. Checkpoint

- Can you explain difference between **Smoke** and **Regression** testing?

Day 7 – Static, Dynamic & Exploratory Testing

Goal: Learn different approaches to testing.

1. Learn (Theory)

- **Static Testing:** Review documents, requirements, test cases.
- **Dynamic Testing:** Actually executing the application.
- What is **Exploratory Testing**?

2. Do (Practical Task)

- Do a **30-minute exploratory testing session** on any website (like a demo shop).
- Note:
    - What you tried
    - What issues you found
    - Any usability problems

3. Checkpoint

- Convert at least 3 exploratory findings into **proper test cases**.

Day 8 – Test Design: Equivalence Partitioning (EP)

Goal: Learn how to reduce the number of test cases smartly.

1. Learn (Theory)

- What is **Equivalence Partitioning (EP)**?
    - Dividing input data into valid and invalid groups.

2. Do (Practical Task)
Example:

Age field accepts values from 18 to 60.

- Identify valid and invalid partitions and write test cases:
    - Less than 18
    - Between 18–60
    - More than 60

3. Checkpoint

- Create EP-based test cases for:
    - A password field (8–20 characters).

Day 9 – Boundary Value Analysis (BVA)

Goal: Learn another key test design technique.

1. Learn (Theory)

- What is **Boundary Value Analysis (BVA)**?
- Test values at the **edges**, like min-1, min, min+1, max-1, max, max+1.

2. Do (Practical Task)
Using the same **Age 18–60** example:

- Find the boundary values and write test cases:
    - 17, 18, 19, 59, 60, 61

3. Checkpoint

- Apply BVA to:
    - Username (length 3–10 characters).

Day 10 – Bug Reporting Basics (with JIRA Concept)

Goal: Learn how to log a clear bug.

1. Learn (Theory)

- What is a **Defect/Bug**?
- Key fields in a bug report:
    - Summary (short title)
    - Description
    - Steps to Reproduce
    - Expected Result
    - Actual Result
    - Severity & Priority
- What is **JIRA** (brief intro as a bug tracking tool)?

2. Do (Practical Task)

- Take any bug you found earlier (from exploratory testing).
- Write a bug report in this format in a Word/Google Doc.

3. Weekly Assignment (Weekend)

- Find **5 issues/bugs** in any app or website.
- For each bug, write:
    - Steps to reproduce
    - Expected vs Actual result
    - Severity (Low/Medium/High)
    - Priority (Low/Medium/High)

WEEK 3 – TOOLS (JIRA, TEST MANAGEMENT) & DATABASE BASICS

Day 11 – JIRA: Issue Tracking (Hands-on Concept)

Goal: Understand how QA uses JIRA.

1. Learn (Theory)

- What is a **Story**, **Task**, and **Bug** in JIRA?
- Workflow example: Open → In Progress → Resolved → Closed.

2. Do (Practical Task)

- If you have access to JIRA:
    - Create a sample bug.
    - Add proper summary, description & status.
- If no access, do the same in Excel/Doc using JIRA-like columns.

3. Checkpoint

- Be able to show at least **3 sample bugs** properly formatted.

Day 12 – Test Management Tool (TestRail/Zephyr or Excel)

Goal: Learn where test cases are stored and managed.

1. Learn (Theory)

- What is a **Test Suite** and **Test Run**?
- Why test management is important.

2. Do (Practical Task)

- Take your **Login test cases** from Week 1.
- Organize them into:
    - One **Test Suite** (e.g., “Authentication”).
    - Create a **Test Run** and mark Pass/Fail (even in Excel).

3. Checkpoint

- Be able to explain “How do you organize test cases?”

Day 13 – SQL Basics for Testers (Part 1)

Goal: Learn to read data from the database.

1. Learn (Theory)

- What is a **Database** & **Table**?
- Basic SQL commands:
    - SELECT
    - FROM
    - WHERE

2. Do (Practical Task)

- Practice simple queries (even on online SQL practice sites, if available):
    - SELECT * FROM customers;
    - SELECT name, email FROM customers WHERE city = 'Delhi';

3. Checkpoint

- Be able to write **3 SQL SELECT** queries on your own.

Day 14 – SQL Basics for Testers (Part 2)

Goal: Use SQL in testing scenarios.

1. Learn (Theory)

- Concept of **JOIN** (simple understanding).
- Why testers use SQL (to check data is saved correctly).

2. Do (Practical Task)

- Think of a test scenario:
    - “User places an order on an e-commerce app.”
- Write:
    - How you would check the order data in DB (e.g., by order_id).

3. Checkpoint

- Write at least 2 test cases that include **DB validation** as a step.

Day 15 – Web Testing & DevTools Basics

Goal: Observe UI, elements, and basic browser tools.

1. Learn (Theory)

- What is **UI Testing** and **Usability Testing**?
- Browser Developer Tools (right-click → Inspect).

2. Do (Practical Task)

- Open any website.
- Use **Inspect** to:
    - Highlight different UI elements (button, input, label).
    - Observe attributes like id, name, class.

3. Weekly Assignment (Weekend)

- Pick any one complete feature (e.g., “Search and Add to Cart”).
    - Write:
        - Test scenarios
        - Test cases
        - At least 2 DB validation test ideas
        - List of bugs (if you find any)

WEEK 4 – AUTOMATION BASICS, API TESTING & FINAL PROJECT

Day 16 – Introduction to Automation & Selenium

Goal: Understand what automation is and when to use it.

1. Learn (Theory)

- What is **Automation Testing**?
- Difference between **Manual** and **Automation** testing.
- When to automate and when not.

2. Do (Practical Task)

- Watch one beginner video or read basics on **Selenium WebDriver**.
- List:
    - What is a locator?
    - Types of locators (id, name, xpath, etc.).

3. Checkpoint

- Be able to answer: “Which test cases are good candidates for automation?”

Day 17 – Selenium Locators (Conceptual + Light Hands-on)

Goal: Get familiar with how elements are identified.

1. Learn (Theory)

- Brief understanding of:
    - `driver.findElement(By.id(""))` etc.
- Focus mainly on **concept**, not coding perfection.

2. Do (Practical Task)

- Open any web page, use Inspect, and list locators for:
    - Username field
    - Password field
    - Login button
- Note their **id/name/xpath** values (if visible).

3. Checkpoint

- Explain how a script would “find” an element using `id`.

Day 18 – API Testing Basics (Using Postman Conceptually)

Goal: Understand what APIs are and how to test them.

1. Learn (Theory)

- What is an **API**?
- What is **GET** vs **POST** request?
- Response status codes (200, 400, 404, 500).

2. Do (Practical Task)

- If you have **Postman** or any API testing tool:
    - Call a sample public API (like a dummy users list).
    - Check:
        - Status code
        - Response body
- If not, just read sample API request/response examples.

3. Checkpoint

- Be able to answer: “How do you test an API?”

Day 19 – End-to-End Testing Simulation

Goal: Put everything together like a real project.

1. Task for the Day
Pick **one feature** (for example: *User Registration* or *Place an Order*) and do:

1. Understand the feature (assume simple requirements).
2. Write:
    - Test scenarios
    - Test cases
3. Execute your test cases on any real or demo app.
4. Log bugs (in JIRA or in Excel).
5. Think where DB and API testing would fit in (conceptually).

2. Output

- A mini test pack for that feature.

Day 20 – Final Review, Documentation

Goal: Summarize your learning and prepare for review.

1. Final Project (to complete & submit)
Prepare a **Final QA Package** that includes:

1. One feature fully tested (e.g., Login, Registration, Add to Cart).
2. Documents:
    - Test Scenarios
    - Test Cases
    - Bug List
    - Short Test Summary Report
3. Optional:
    - Sample SQL queries used
    - Sample API test (if done)

2. Review Prep Checklist
Be ready to answer:

- What is SDLC & STLC?
- Difference between:
    - Smoke vs Sanity
    - Regression vs Retesting
    - Severity vs Priority
- Explain one bug you found and how you reported it.
- Show your test cases and explain your thought process.

What You Will Have After 1 Month

By following this document daily, you (the fresher) will have:

- 50+ **manual test cases** written by you
- 20+ **bug reports**
- Basic understanding of:
    - SDLC, STLC, Test Types
    - Test Design Techniques (EP, BVA)
    - Bug Reporting
    - Test Management concepts
    - SQL basics
    - Basic idea of Automation & API testing
- One **mini end-to-end testing project** to show in interviews

Extra Tips for Freshers

- Ask questions if anything is confusing: senior team members want to help!
- Take notes and review them daily.
- Try practicing tasks on free demo apps/websites.
- Discuss your findings and mistakes in daily standups/check-ins.
- Try group exercises – learning from peers is powerful.

Remember: Every step gets you closer to being a skilled QA engineer. Stay curious, be hands-on, and take the week seriously. Your questions and learning now will pay off throughout your career!