### ****Practical Example of Implementing Shift-Left in a Project****

Let’s take a **real-world project**: **Developing an Online Banking Application** with a **Shift-Left Approach**.

### ****Project Scenario****

Your team is building a new feature: **"User Fund Transfer"** (Send money between accounts).  
Normally, testing would happen **after** development. But with **Shift-Left**, testing starts **much earlier** in the SDLC.

**Step-by-Step Implementation of Shift-Left**

**1. Testers Involved in Requirement Phase (Early Testing 🚀)**

🔹 **Traditional Approach:** Testers wait until the development is complete.  
🔹 **Shift-Left Approach:** Testers participate in requirement discussions.

👉 **Example:**

* The business analyst says: "The user should be able to transfer money securely."
* The tester asks:  
  ✅ "What if the user enters an invalid account number?"  
  ✅ "What if the balance is insufficient?"  
  ✅ "What if the user session expires?"  
  ✅ "What if there is a network failure mid-transaction?"

📌 **Result:** Requirements are refined **before** coding starts, reducing future bugs.

**2. Developers Write Test Cases Before Code (TDD Implementation)**

🔹 **Traditional Approach:** Developers write code first, then hand it to testers.  
🔹 **Shift-Left Approach:** Developers write **unit tests** first before coding.

👉 **Example (Test-Driven Development - TDD)**

* Before implementing **"Transfer Money"**, a developer writes test cases:

@Test

public void testTransferFailsForInsufficientBalance() {

assertThrows(InsufficientBalanceException.class, () -> {

accountService.transfer(12345, 67890, 10000.00); // Trying to send $10,000 with only $500 in account

});

}

@Test

public void testTransferSucceedsWithSufficientBalance() {

boolean result = accountService.transfer(12345, 67890, 100.00);

assertTrue(result);

}

📌 **Result:**

* The developer must write code that **passes these tests**.
* Bugs are caught **before** sending to testers.

**3. Automating API Testing in CI/CD Pipeline**

🔹 **Traditional Approach:** API testing happens **after development**.  
🔹 **Shift-Left Approach:** Automated API tests run **whenever code changes**.

👉 **Example (Using Postman or RestAssured for API Testing)**

@Test

public void testAPIFundTransfer() {

given()

.contentType("application/json")

.body("{\"fromAccount\": 12345, \"toAccount\": 67890, \"amount\": 100.00 }")

.when()

.post("/api/transfer")

.then()

.statusCode(200)

.body("message", equalTo("Transfer Successful"));

}

📌 **Result:**

* Every time a developer pushes code, API tests **automatically verify** the functionality.

**4. Security Testing (Shift-Left DevSecOps)**

🔹 **Traditional Approach:** Security testing is done **after** development.  
🔹 **Shift-Left Approach:** Security vulnerabilities are **tested early** in CI/CD.

👉 **Example (Using OWASP ZAP for Security Scanning)**

* Before merging code, ZAP scans for security issues like **SQL Injection & XSS attacks**.

📌 **Result:**

* Security issues are **fixed early**, reducing risks in production.

**5. Performance Testing Early**

🔹 **Traditional Approach:** Performance testing is done **just before release**.  
🔹 **Shift-Left Approach:** Performance is tested **on early versions** of the app.

👉 **Example (Using JMeter for Load Testing)**

* A JMeter script simulates **1,000 users transferring money at once**.
* If response time > 3 seconds, developers optimize **before release**.

📌 **Result:**

* No last-minute surprises about **slow performance**.

**Final Outcome of Shift-Left in This Project**

✅ Bugs are **prevented**, not just **found later**.  
✅ Developers & testers **collaborate early**, reducing miscommunication.  
✅ Automated tests in CI/CD catch issues **as soon as they occur**.  
✅ Security & performance issues are **addressed before release**.

Top of Form



**Practical Example: How QA Can Use Shift-Left Approach**

Let’s take a **real-world example** where a **QA team** applies the **Shift-Left approach** to test an **E-commerce Checkout Process** **(Online Shopping Website)**.

**Project Scenario:**

Your company is developing a new feature: **"Checkout & Payment Process"** for an online shopping website.

🔹 **Traditional Approach (Without Shift-Left)**:

* QA starts testing **after** development is complete.
* Bugs (e.g., wrong total price, payment failures) are found **late** in the SDLC.
* Fixing issues **causes delays** and increases **costs**.

🔹 **Shift-Left Approach (QA Involves Early)**:

* QA starts testing **from the requirement phase** instead of waiting for development.
* Bugs are **prevented** instead of being found late.
* Automated tests **run in CI/CD** from the beginning.

## ****Step-by-Step Shift-Left Implementation by QA in this Project****

### ****1️⃣ QA Reviews Requirements to Find Gaps (Early Testing)****

📌 **Before Development Starts, QA Analyzes Requirements**

🔹 **Requirement:**

* "The checkout page should allow users to review their cart, enter shipping details, select a payment method, and confirm the order."

🔹 **How QA Uses Shift-Left:**

* QA asks **critical questions** to uncover missing details:  
  ❓ "What happens if the payment gateway fails?"  
  ❓ "What if a user enters an invalid credit card?"  
  ❓ "What if a product goes out of stock during checkout?"  
  ❓ "How does the system handle discounts and promo codes?"

✅ **Outcome:**

* Developers refine requirements **before coding** to avoid future defects.

### ****2️⃣ QA Defines Test Cases Early (Before Coding Starts)****

📌 **QA Writes Test Scenarios While Developers Write Code (Parallel Work)**

🔹 **Example Test Cases QA Creates:**  
✅ "Verify checkout completes successfully with valid payment."  
✅ "Ensure expired credit card shows an error message."  
✅ "Check if applying a promo code updates the total amount correctly."  
✅ "Ensure order is saved in the database after successful payment."

💡 **Shift-Left Approach:** Instead of waiting for developers, **QA writes these cases before coding is done.**

✅ **Outcome:**

* Developers understand expected behaviors **before development finishes**, reducing defects.

### ****3️⃣ Automate Tests in CI/CD (Immediate Feedback)****

📌 **QA Integrates Automated Tests in CI/CD Pipeline to Run on Every Code Commit**

🔹 **Traditional Approach (Without Shift-Left):**

* Developers complete a sprint → QA starts manual testing → Bugs found → Rework.
* This delays releases.

🔹 **Shift-Left Approach (QA Automates Tests Early):**

* When developers push code, automated tests **immediately** check the checkout flow.
* If a test fails, the build **does not proceed** → Developer fixes the issue **immediately**.

👉 **Example Automated UI Test Using Selenium (QA Writes Tests Early)**

python

from selenium import webdriver

from selenium.webdriver.common.by import By

# Launch browser

driver = webdriver.Chrome()

driver.get("https://shoppingwebsite.com/checkout")

# Enter shipping details

driver.find\_element(By.ID, "address").send\_keys("123 Street, New York")

driver.find\_element(By.ID, "zipcode").send\_keys("10001")

# Select Payment Method & Complete Checkout

driver.find\_element(By.ID, "creditCardNumber").send\_keys("4111111111111111")

driver.find\_element(By.ID, "checkoutButton").click()

# Verify success message

assert "Order Confirmed" in driver.page\_source

driver.quit()

✅ **Outcome:**

* **Every time a developer updates the checkout code, this test runs automatically**.
* If checkout **fails**, QA gets an alert **immediately** instead of discovering the issue days later.

### ****4️⃣ API Testing Shift-Left (Before UI is Ready)****

📌 **QA Tests APIs Before UI is Developed to Detect Issues Early**

🔹 **Example:**

* Instead of waiting for the UI to be ready, QA **directly tests checkout APIs** using **Postman or RestAssured**.

👉 **Example API Test Using Postman/Newman**

{

"method": "POST",

"url": "https://api.shoppingwebsite.com/checkout",

"body": {

"items": [{"id": 101, "quantity": 2}],

"payment\_method": "credit\_card",

"shipping\_address": "123 Street, New York"

},

"tests": {

"Status code is 200": "responseCode.code === 200",

"Order ID is returned": "pm.expect(responseBody).to.have.property('order\_id')"

}

}

✅ **Outcome:**

* Bugs in checkout APIs are **fixed early** before UI development even starts.

### ****5️⃣ Performance Testing Before Release****

📌 **QA Tests Performance Early (Instead of Last-Minute Testing)**

🔹 **Example:**

* QA runs **JMeter tests** to simulate **1,000 users checking out at the same time**.
* Detects slow performance **early**, preventing server crashes in production.

👉 **Example JMeter Test:**

* Simulates 1,000 users clicking **"Checkout"** simultaneously.
* Identifies if the checkout process **slows down or crashes**.

✅ **Outcome:**

* QA ensures the site **handles high traffic before release**, preventing downtime.

### ****6️⃣ Shift Security Testing Left (Prevent Breaches Early)****

📌 **QA Conducts Early Security Testing Instead of Waiting for Penetration Tests**

🔹 **Example:**

* QA runs **OWASP ZAP scans** on checkout pages to detect **SQL Injection, XSS attacks, etc.**
* Developers fix security vulnerabilities **before the site goes live**.

👉 **Example Security Test:**

* OWASP ZAP detects **insecure credit card data storage**.
* Developers encrypt payment details **before release**, reducing data breach risks.

✅ **Outcome:**

* Security issues are **fixed early**, reducing last-minute risks.

## ****📌 Final Outcome: How QA Benefits from Shift-Left?****

| **Without Shift-Left** | **With Shift-Left** |
| --- | --- |
| QA starts after development | QA starts during requirements |
| Late defect detection | Bugs prevented early |
| Delayed bug fixes | Faster fixes with early feedback |
| High testing costs | Cost reduction by early detection |
| Security & performance issues found late | Security & performance tested early |

Bottom of Form