Simulating the **entire SQA journey (Weeks 1–3)** using the same **Food Delivery App**.

Let’s start with **Week 1: Foundations**, and I’ll guide you step-by-step — with PM-style explanations, QA tasks, and practical examples for each day using the same app.

**Week 1: SQA Foundations — Connected Story (Food Delivery App)**

Theme: Build the *foundation of quality* before any development starts.

**🔹 Day 1: Introduction to SQA & SDLC**

🗣️ QA Lead (you):

“Before we start testing, let’s understand what SQA is.”

**What We Did:**

* Defined **QA (Quality Assurance)** as *process-oriented* → prevent defects
* Defined **QC (Quality Control)** as *product-oriented* → find defects
* Reviewed the **SDLC (Software Development Life Cycle)**:
  + Planning → Analysis → Design → Dev → Testing → Deployment → Maintenance
  + We chose **Agile model** for this project

**Food App Example:**

* QA team joins from day one — sits with PM & devs in planning
* Our QA is responsible for reviewing requirements & ensuring testability

**🔹 Day 2: Software Standards & Models**

🗣️ QA Lead:

“We don’t test randomly — we follow **standards and models**.”

**What We Did:**

* Learned about **IEEE 829 / IEEE 830** (SRS, test plan formats)
* Explored **CMMI levels** (we aim for Level 2: Managed)
* Introduced **ISO 9001** → Process-driven quality

**Food App Example:**

* Created documentation using **IEEE 830** structure
* Used naming standards for all JIRA tickets
* QA process included review checkpoints before code moved forward

**Task:** Selected our own app (Food Delivery) and started a **QA checklist**

**🔹 Day 3: Requirements Analysis & Traceability**

🗣️ QA Lead:

“Before writing test cases, we need to **verify the requirements**.”

**What We Did:**

* Read the **SRS document** for the food app
* Checked for characteristics of good requirements: **Clear, Testable, Unambiguous**
* Created a **Traceability Matrix**:
  + Links **each requirement to its test case**
  + Helps ensure full coverage & trace missing areas

**Food App Example:**

| **Requirement ID** | **Description** | **Linked Test Case** |
| --- | --- | --- |
| RQ-1 | Users must log in using email | TC-01 |
| RQ-2 | Users must see list of restaurants | TC-02 |
| RQ-3 | Add to cart must update total | TC-03 |

**Tool:** Excel or Google Sheet  
**Task:** Started traceability matrix for login, cart, and order features

**🔹 Day 4: Test Planning & Strategy**

🗣️ QA Lead:

“Let’s plan how we’ll test — not just what.”

**What We Did:**

* Wrote a **Test Plan** with:
  + **Objectives**: Ensure critical paths like placing an order are bug-free
  + **Scope**: Functional testing for cart, checkout, login
  + **Environment**: Testing on Chrome + mobile view
  + **Entry/Exit Criteria**:
    - Entry: All stories marked ‘Ready for QA’
    - Exit: All critical test cases passed, no P1 bugs

**Food App Example:**

* Created test environments on local + staging
* Used a shared Test Plan doc (Google Docs or Notion)

**Task:** Filled in a simplified test strategy template for demo day

**🔹 Day 5: Test Case Design Techniques**

🗣️ QA Lead:

“Time to start writing test cases using techniques like **BVA**, **EP**, and **Decision Tables**.”

**What We Did:**

* Learned **Equivalence Partitioning**:
  + Input: marks (0–100) → Valid = 0–100, Invalid = <0 or >100
* Learned **Boundary Value Analysis**:
  + Test values: 0, 1, 99, 100, 101
* Created **Decision Tables**:
  + E.g., coupon applied + cart total > X → Free delivery

**Food App Example:**

📄 **Test Case: Add to Cart**

| **TC ID** | **Description** | **Steps** | **Expected** |
| --- | --- | --- | --- |
| TC-03 | Cart total updates correctly | Add 2 items, check total | Total should equal item price sum |

**Tool**: Excel / Google Sheets  
**Task**: Wrote 5 manual test cases using all 3 techniques

**Week 2: Testing Types & Automation Basics** using the same **Food Delivery App** example.

This week is where your QA team **actually gets hands-on with different types of testing** — unit, integration, system, performance, and even automation. I’ll present this in the same “PM/QA Lead explains to intern” format with connected dots + practical examples.

**Week 2: Testing Types & Automation Basics**

📦 Theme: Apply testing types on real features to validate quality from every angle.

**🔹 Day 6 – Unit & Integration Testing**

🗣️ QA Lead:

“Before the full app is tested, developers test individual pieces — that's **unit testing**. Then we check how modules connect — that’s **integration testing**.”

**🧪 Unit Testing Example (Python – calculate\_total function):**

def calculate\_total(cart\_items):

return sum(item['price'] for item in cart\_items)

✅ Test using pytest:

def test\_calculate\_total():

cart = [{'price': 100}, {'price': 200}]

assert calculate\_total(cart) == 300

**🔗 Integration Testing Example:**

Testing **add-to-cart** + **checkout total** together.

* User adds item to cart
* Cart total should update
* Checkout page should show the same total

We used mock data and created a test that simulates both modules together

**🔹 Day 7 – System & Acceptance Testing**

🗣️ QA Lead:

“Now we test the app **as a whole system** — just like a real user would. That’s system testing. Then we simulate a client test — that’s UAT.”

**🎯 Black-Box Testing (System)**

We don’t look at the code — only the inputs/outputs.

✅ Food App example:

* Open login form → enter valid credentials → expect redirect
* Add item to cart → check that price updates

**🧪 White-Box Testing (Code Coverage)**

For example, testing all if-else paths in a promo logic:

if total > 500:

apply\_discount()

else:

no\_discount()

✅ We used coverage.py + pytest to ensure all paths were covered.

**🧪 Acceptance Testing**

We asked a **mock client** to place a test order and give feedback:

* Was the app easy to use?
* Did all steps work?
* What was confusing?

✅ Their feedback was used for UI improvements

**🔹 Day 8 – Non-Functional Testing**

🗣️ QA Lead:

“We also test how the app behaves under stress, and how secure or user-friendly it is.”

**🏋️ Load & Performance Testing**

Used **JMeter** to simulate 50 users placing orders at once.  
Measured:

* API response time
* Cart loading delay
* Failure rate under load

**🔐 Security Testing (Basic)**

* Tested for **input validation** (e.g., SQL injection attempts)
* Checked for **password encryption** in login

**👀 Usability Testing**

* Asked users: “Can you place an order without instruction?”
* Measured:
  + Time to complete task
  + Confusion or blockers
  + Suggestions for layout

✅ We recorded findings in a **UX Test Report**

**🔹 Day 9 – Manual Testing Lifecycle**

🗣️ QA Lead:

“Now let’s follow a complete manual testing lifecycle.”

| **Stage** |  | **Example** |
| --- | --- | --- |
| 📄 Test Case Writing |  | Login, Add to Cart, Checkout |
| ✅ Execution |  | Ran each test on staging |
| 🐞 Defect Logging |  | Found bug in cart total logic |
| 📊 Reporting |  | Used TestLink/JIRA to track status |

✅ Task:

* Wrote 5 test cases for cart + checkout
* Logged 2 bugs in JIRA (UI mismatch, incorrect total)

**🔹 Day 10 – Intro to Automation Testing**

🗣️ QA Lead:

“Some tests are repetitive — we **automate** them using tools like Selenium.”

**🧪 What We Automated:**

* ✅ Login flow
* ✅ Add item to cart
* ✅ Check final total

📜 Tools used:

* Selenium with Python
* GitHub for test script repo

✅ Also introduced **API Testing** with Postman:

* Checked /api/login, /api/order
* Validated response code = 200, correct JSON keys

Perfect, Fahad! Let’s now complete the **SQA Journey Week 3** — focused on **Advanced QA with Selenium**, reporting, metrics, and full QA lifecycle — using the **same Food Delivery App**.

We’ll follow the same intern-friendly narration format, connected examples, and practical tasks.

## Week 3: Advanced QA Practices with Selenium

📦 **Theme**: Automate, measure, integrate, and deliver high-quality software.

### 🔹 ****Day 11 – Selenium Basics: Web Automation****

🗣️ QA Lead:

“We started writing our first automated tests using **Selenium with Python**.”

#### 🛠️ What We Did:

* Installed selenium + chromedriver
* Wrote a script to automate login flow

from selenium import webdriver

from selenium.webdriver.common.by import By

driver = webdriver.Chrome()

driver.get("http://localhost:8501")

driver.find\_element(By.ID, "username").send\_keys("testuser")

driver.find\_element(By.ID, "password").send\_keys("test123")

driver.find\_element(By.TAG\_NAME, "button").click()

assert "Dashboard" in driver.page\_source

✅ **Task:** Team automated login + cart addition in Streamlit/HTML food app

### 🔹 ****Day 12 – Selenium Advanced: Data-Driven + Regression Testing****

🗣️ QA Lead:

“Now we made our test reusable using **data-driven testing** and planned for **regression**.”

#### 📊 Data-Driven Example:

import pytest

@pytest.mark.parametrize("username,password", [

("user1", "pass1"),

("admin", "wrongpass"),

])

def test\_login(username, password):

# Use selenium steps to test different inputs

#### 🔁 Regression Strategy:

Whenever a change is made (e.g. coupon logic), we re-run **core test cases** like:

* Login
* Add to cart
* Checkout
* Total with/without discount

✅ **Task:** Created a regression test suite in Python + Git

### 🔹 ****Day 13 – QA Metrics & Reporting****

🗣️ QA Lead:

“We don’t just test — we **measure** how effective we are.”

#### 🔢 Key Metrics:

| **Metric** | **Food App Example** |
| --- | --- |
| ✅ **Test Coverage** | Used coverage.py to check which functions are tested |
| 🐞 **Defect Density** | Logged 10 bugs across 2000 LOC → 5 bugs/1K LOC |
| 📈 **Test Execution Rate** | 80% test cases passed in Sprint 1 |
| 📊 **Dashboard Reporting** | Used JIRA dashboard + Excel for summary chart |

✅ **Task:** Built an Excel dashboard showing defects by module & status

### 🔹 ****Day 14 – Continuous Testing & Agile QA****

🗣️ QA Lead:

“We now integrated our tests into a **CI/CD pipeline** — so they run automatically on every push.”

#### 🧪 What We Did:

* Used **GitHub Actions** to auto-run Selenium tests on push
* Shifted **left** — started testing in early dev phases
* Did **Exploratory Testing**:
  + Tried random input on cart total field (e.g., -50, string)
  + Found layout bugs in mobile view using DevTools

✅ **Task:** Performed 5 exploratory tests using browser DevTools + logs

### 🔹 ****Day 15 – End-to-End QA Simulation****

🗣️ QA Lead:

“Final day! We combined **manual + automated QA** into a full testing cycle.”

**Week 1,2,3 Summary — SQA Foundations Connected**

You now saw how a QA team:

* Understands software process models and standards
* Verifies & validates requirements before testing
* Creates documentation: SRS review, Test Plan, Traceability
* Uses testing techniques to write solid test cases
* Starts early in the lifecycle to **prevent bugs**, not just catch them
* Tests at every level (unit → integration → system → UAT)
* Validates both **functionality** and **non-functional behavior**
* Uses both **manual and automation** testing
* Logs and tracks defects in a process
* Starts API and performance testing early
* Mastered automation with Selenium
* Used data-driven & regression strategy
* Tracked real-world QA metrics
* Practiced CI/CD & shift-left QA
* Delivered a complete end-to-end QA cycle