Fake Store API Testing

Project Documentation

Project Overview

This project demonstrates comprehensive API testing for the Fake Store API as part of the ITI Graduation Project. It combines both manual and automation testing, using industry-standard tools and best practices to ensure API functionality, performance, and reliability.

Project Tech Stack

- Manual Testing: Postman, Newman

- Automation: Rest Assured, TestNG

- Build Tool: Maven

- Reporting: Allure, SLF4J (Logger)

- Language: Java

- Structure: Modular (per feature)

- Logging: Custom LoggerUtil + SLF4J

Project Structure

```
FakeStoreAPI-Testing/
├──── BaseTest/
│     └──── BaseTest.java        # Contains common setup for Rest Assured
├──── Products/
│     └──── Product_Test.java     # Tests for product endpoints
├──── Users/
│     └──── Users_Test.java       # Tests for user endpoints
├──── Carts/
│     └──── Carts_Test.java       # Tests for cart endpoints
├──── Auth/
│     └──── Login_Test.java       # Login/authentication tests
├──── EndToEnd/
│     └──── EndToEnd_Test.java     # Complete flow (user > product > cart > delete)
├──── models/
│     ├──── Product.java
│     ├──── User.java
│     ├──── Name.java
│     ├──── Cart.java
│     └──── ProductInCart.java
├──── utils/
│     └──── LoggerUtil.java       # Custom SLF4J logger utility
├──── testng.xml                 # TestNG suite
└──── pom.xml                 # Maven dependencies
```

Modules & Test Coverage

Products Module:

- CRUD + Filter + Sort

Users Module:

- CRUD + Sort/Limit

Carts Module:

- CRUD + Filter by Date/User

Auth Module:

- Login

End-to-End:

- Full user-product-cart lifecycle

🧪 Testing Types

- Status Code Validation

- Response Body Checks

- Response Time Limits

- Data Assertions

- Dynamic Data Handling

- Negative Testing

🔁 POJOs and Serialization

Used clean POJOs to structure both request and response data.
Handled using Rest Assured's built-in serialization/deserialization.

Logging & Reporting

Logging: SLF4J + LoggerUtil used for modular logs

Reporting: Allure reporting for beautiful interactive HTML reports

End-to-End Flow (Realistic Simulation)

1. Create User

2. Create Product

3. Create Cart

4. Validate Cart

5. Delete Cart > Product > User

Challenges & Solutions

Authentication Enforcement | Requires token | Added static login test

Changing Data/IDs | Dynamic updates | Used POJOs to manage data

Inconsistent Product Structure | Varies in cart | Deserialization with flexibility

Rate Limiting | Some endpoints slow | Used .time(lessThan())

Data Deletion Failures | Deleted needed data | Cleanup only at end

Manual Testing with Postman

Postman Collection + Newman CLI runs

Command: newman run fakestore_collection.json

🧠 Lessons Learned

- Modular design

- Real-world data modeling

- Avoiding flaky tests

- Best practices for logging/reporting

🙌 Credits

Developed with pride as part of the ITI Graduation Project

Supervised by ITI Team

By: Mohamed Kamal

🧠 Quote of the Project

“Quality is never an accident; it is always the result of intelligent effort.” — John Ruskin