Testing Online Shop Application

EN.601.422/622: Software Testing and Debugging - Spring 2023

SUT Introduction

SUT Description: is the **Online Shop Web Application**: A full-stack web application for online shopping, focused on supporting local shops or outlet chains.

Potential Customer: Developers trying to build a website and/or shoppers

Purpose: Enable customers to purchase products online with home delivery, email integration, and more

App Type: Cross-platform web app

Platforms: Accessible on all major web browsers

Number lines of code: About 2500~

Github: https://github.com/singhanshika311/wipro_capstone_project_c7_b2

SUT Key Features and Technologies

User types: Administrators and Customers

Backend Technologies: Spring Boot, MySQL, and JPA

Frontend Technologies: Angular 8 and Angular Material UI component library

Two Major Components:

1. RESTful web services for server-side operations
2. User-friendly interface designed using Angular and Angular Material UI components

Existing state of codebase

Existing tests: Completely untested with no existing tests

User Documentation: Very ineffective and missing major components to run the application

Code Comments: Code almost entirely undocumented

GUI Testing

Model user interactions with the site

- GUI testing on the Customer side

- GUI testing on the Admin side

- Tool: **Selenium**

GUI Testing Live Demo

Major Faults:

1. User
   a. Duplicate username registration
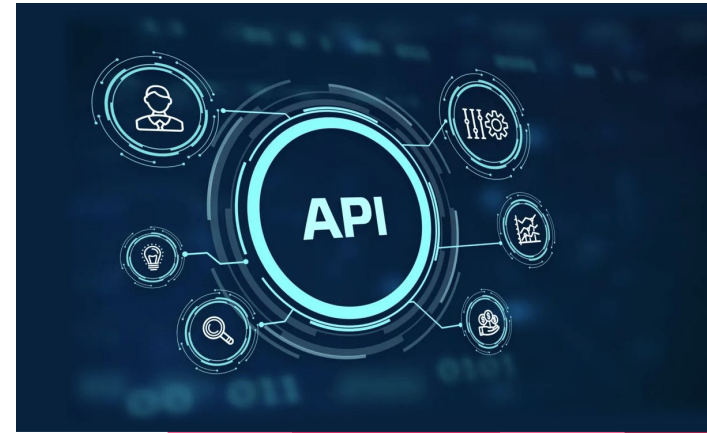   b. Add Product without the price
   c. Payment Method
2. Admin
   a. View Order, Upload Image and Send Email Function
   b. Duplicated Categories Name

API testing

Analyzes API to verify that it fulfills its expected functionality, security, performance and reliability

- API testing

  - Admin, Category, Customer, Order, Product, Wishlist

- Tool: **RestAssured**

API Testing Live Demo

Major Faults:

1. Wishlist API return Status Code <500> for GET and POST

2. Admin API return Status Code <405>

    a. Authorization Issue

Blackbox Testing

Equivalence partitioning

Blackbox Testing Live Demo

Major Faults:

1. In many services: the author does not check about null object being as an input and accept it.

2. In many services: when an invalid id has been entered, the author will just throw an NoSuchElementException

Blackbox tests run: 62, Failures: 6, Errors: 0, Skipped: 0

Whitebox Testing

Achieving 100% LC by unit testing the models, services, controller, and dtos

Why 100% Line coverage?

- The backend code had no unit tests
    - There were many "latent faults" or faults that were non-obvious and difficult or impossible to trigger by a user
- There was no documentation or comments describing expected behavior
    - Therefore, LC required less interpretation of the intended behavior of the methods
- <100% BC because some of the branches were impossible to execute because of errors

Final LC: 100%

Final BC: 93%

Whitebox tests run: 162, Failures: 3, Errors: 9, Skipped: 0

Jacoco Coverage Report

com.homecommerce.services

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
CustomerService		100%		80%	2	12	0	14	0	7	0	1
AdminService		100%		87%	1	9	0	14	0	5	0	1
CartService		100%		100%	0	8	0	13	0	7	0	1
ProductService		100%		n/a	0	8	0	13	0	8	0	1
WishlistService		100%		100%	0	6	0	10	0	5	0	1
OrderService		100%		n/a	0	6	0	9	0	6	0	1
CategoryService		100%		n/a	0	5	0	7	0	5	0	1
OrderDetailService		100%		n/a	0	4	0	5	0	4	0	1
PaymentService		100%		n/a	0	3	0	3	0	3	0	1
AddressService		100%		n/a	0	3	0	3	0	3	0	1
Total	0 of 420	100%	3 of 22	86%	3	64	0	91	0	53	0	10

com.homecommerce.models

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Customer		100%		n/a	0	17	0	27	0	17	0	1
Order		100%		n/a	0	18	0	30	0	18	0	1
Product		100%		n/a	0	15	0	24	0	15	0	1
Payment		100%		n/a	0	12	0	19	0	12	0	1
Address		100%		n/a	0	12	0	17	0	12	0	1
Cart		100%		n/a	0	10	0	16	0	10	0	1
Admin		100%		n/a	0	9	0	17	0	9	0	1
OrderDetails		100%		n/a	0	10	0	14	0	10	0	1
Wishlist		100%		n/a	0	8	0	12	0	8	0	1
Category		100%		n/a	0	6	0	11	0	6	0	1
Total	0 of 662	100%	0 of 0	n/a	0	117	0	187	0	117	0	10

com.homecommerce.controller

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
OrdersController		100%		100%	0	8	0	35	0	6	0	1
ProductController		100%		100%	0	11	0	25	0	7	0	1
CustomerController		100%		100%	0	7	0	14	0	6	0	1
CartController		100%		100%	0	6	0	11	0	5	0	1
WishlistController		100%		100%	0	5	0	9	0	4	0	1
FileController		100%		100%	0	3	0	11	0	2	0	1
AdminController		100%		100%	0	4	0	8	0	3	0	1
CategoryController		100%		n/a	0	4	0	6	0	4	0	1
Total	0 of 465	100%	0 of 22	100%	0	48	0	119	0	37	0	8

com.homecommerce.dtos

| Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
LoginDTO		100%		n/a	0	8	0	11	0	8	0	1
Response		100%		100%	0	6	0	12	0	4	0	1
CartDTO		100%		n/a	0	11	0	16	0	11	0	1
OrderDetailsDTO		100%		n/a	0	8	0	13	0	8	0	1
PlaceOrderDTO		100%		n/a	0	9	0	13	0	9	0	1
ProductDTO		100%		n/a	0	4	0	7	0	4	0	1
OrderResponseDTO		100%		n/a	0	5	0	7	0	5	0	1
WishlistDTO		100%		n/a	0	3	0	4	0	3	0	1
Total	0 of 245	100%	0 of 4	100%	0	54	0	83	0	52	0	8

Whitebox Testing Live Demo

Major Faults:

- Admin Class
  - **Security vulnerability**: The password of the admin user is transmitted in plain text and there is also a tostring for password
  - **Bug**: The updateAdmin() method fails on Null inputs which would crash the site
  - **Bug**: The AdminService constructor fails on certain Null or uninitialized arguments.
- Wishlist Class
  - **Bug**: The findByuserid method is broken. There isn't a way to attach a wishlist to a customer
  - **Bug:** wishlistController.listall method is broken because it can't fetch wishlists by user id
- Customer Class
  - **Bug / Security vulnerability:** In the updateProfile method if the password is empty or null a NullPointerException is thrown

Load Testing

Using JMeter to test load on endpoints

Graph
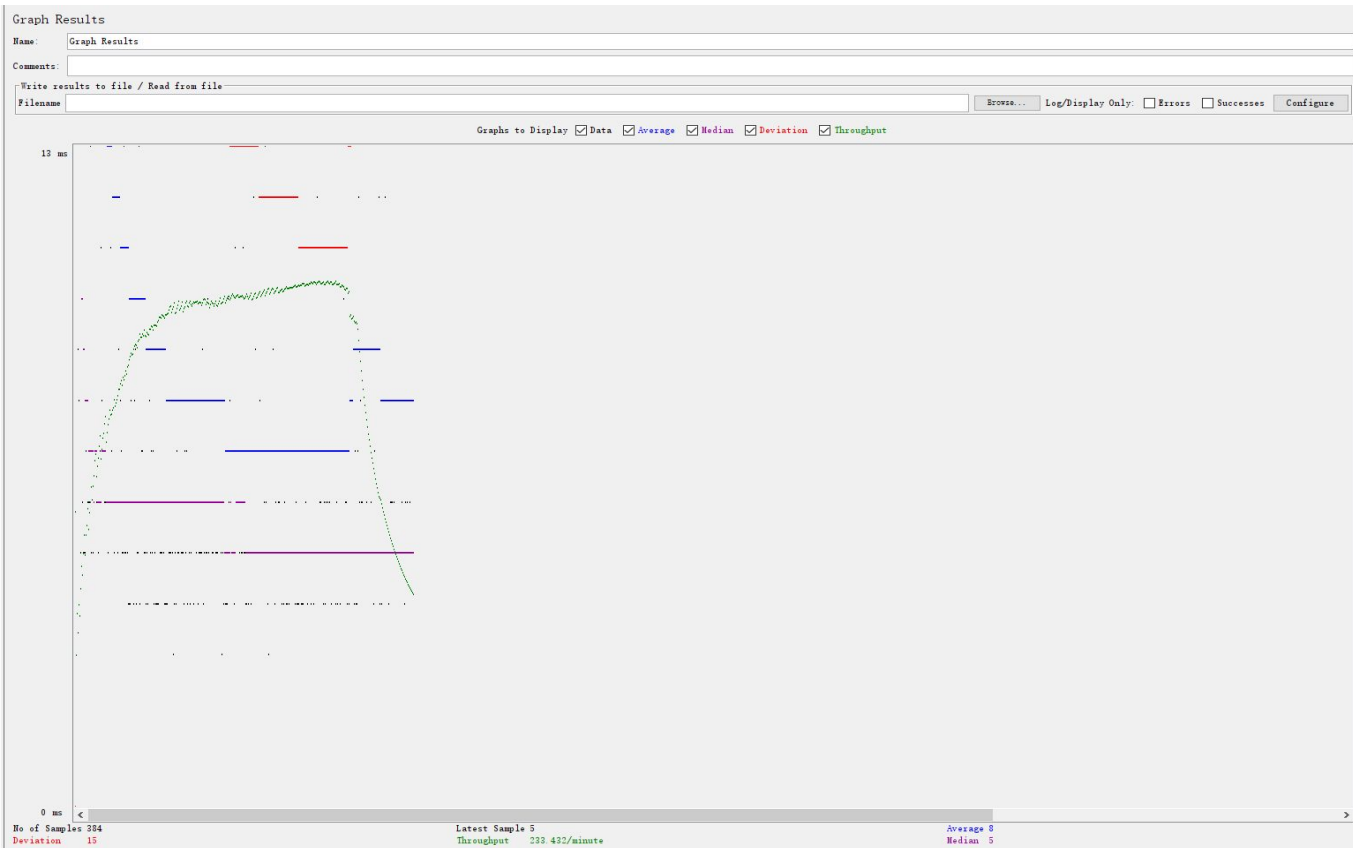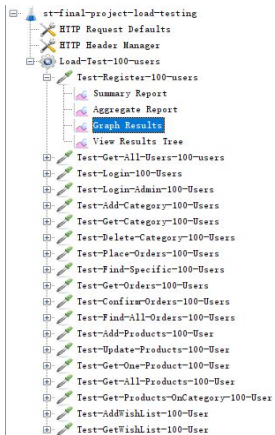
Conclusions

Testing Summary

Gui Testing

- Learn how to test usability issues, compatibility issues and functional defects from the user perspectives

Api Testing

- Learn to analyze the results of the testing process to identify any bugs, defects, or issues.

Load Testing:

- Learned how to use Jmeter and analyze load testing metrics

Whitebox testing

- 100% LC and 93% BC
- Learned more complicated mocking and injection strategies for files, services, controllers, etc.

Total Test Cases Written: > 300

Total Major Faults Discovered: 34

Questions?