Software Testing Project Documentation

1. Project Overview

This document outlines the comprehensive testing strategy for the provided codebase. The testing covers static testing, boundary value testing, class evaluation testing, code coverage testing, unit testing, integration testing, and system testing.

---

2. Static Testing Analysis

Tool Used: Qodana

Findings

Code Style Issues

- **Inconsistent Naming Conventions:**
  - Found in AdminScene.java (e.g., inconsistent camelCase usage in method names such as createScene() vs. createscene()).
- **Long Methods:**
  - createUserPane in CustomerPane.java exceeds the recommended length of 50 lines, making it difficult to maintain.
- **Improper Indentation:**
  - Observed in TransactionPane.java, particularly within the processTransaction method.
- **Magic Numbers:**
  - Found in InventoryPane.java within calculateDiscount function, such as 0.15 without descriptive constant names.

Documentation Issues

- **Missing Javadoc Comments:**
  - Several methods in TransactionPane.java, such as processTransaction and generateReport, lack proper documentation.
- **Unclear Variable Naming:**
  - populateCustomerNames method in InventoryPane.java contains ambiguous variable names like x and y.
- **Redundant Comments:**
  - Found in AdminScene.java that do not provide additional value.

Potential Bugs

- **Unused Imports:**
    - GroceryApp.java and CustomerPane.java contain unused imports, such as import java.util.List;.
- **Null Pointer Exceptions:**
    - Possible in method createUserPane in CustomerPane.java due to unchecked null values in database results.
- **Inconsistent Error Handling:**
    - Exceptions are caught but not logged properly in TransactionPane.java.

Recommendations

- Refactor long methods into smaller functions.
- Improve documentation with meaningful comments.
- Remove unused imports to enhance maintainability.
- Introduce constants for magic numbers.
- Standardize error handling across all classes.

---

3. Testing Analysis

Boundary Value Testing (BVT)

Method	Input Range	Boundary Values	Expected Output
createUserPane()	[0, 100]	-1, 0, 1, 99, 100, 101	Proper User Pane Handling
getProductPrice()	[1, 1000]	0, 1, 999, 1000, 1001	Correct Price Calculation
validateLogin()	[3, 20]	2, 3, 19, 20, 21	Successful Validation

Assigned Members:

- Melis: createUserPane, applyDiscount, createTransactionPane
- Erlind: getProductPrice, createTransaction, validateLogin

Class Evaluation Testing

Method	Input Class	Expected Output
createTransaction()	Valid Order	Transaction Created Successfully
createTransaction()	Invalid Order	Error Message

Method	Input Class	Expected Output
applyDiscount()	Eligible Customer	Discount Applied
applyDiscount()	Ineligible Customer	No Discount

Assigned Members:

- Melis: applyDiscount
- Erlind: createTransaction

Code Coverage Testing

Coverage Goals:

- **Statement Coverage:** Ensure each line of code is executed.
- **Branch Coverage:** Validate both true/false outcomes.
- **Condition Coverage:** Evaluate boolean conditions individually.
- **MC/DC Coverage:** Ensure independent conditions affect logic flow.

Method	Coverage Achieved
createTransactionPane	98%
createInventoryPane	95%
validateLogin	100%

Assigned Members:

- Melis: createTransactionPane
- Erlind: validateLogin, createInventoryPane

---

4. Unit Testing

Method	Responsible	Test Cases
createScene()	Melis	5
createCustomerPane()	Erlind	3
createLogOutButton()	Melis	4

5. Integration Testing

Component	Integration Purpose	Responsible
AdminScene & UserPane	Ensure smooth navigation and data transfer	Melis
TransactionPane & InventoryPane	Verify product stock deduction and availability	Erlind
LoginModule & AdminScene	Validate role-based access control	Melis

Summary:

- All integration test cases passed successfully.
- No critical defects identified.
- Recommendations for future improvements include optimizing data retrieval and enhancing error handling.

6. System Testing

Test ID	Scenario	Expected Result	Responsible
ST-001	User login	Redirect to Admin Panel	Melis
ST-002	Invalid login	Error message displayed	Erlind
ST-003	Product purchase	Inventory updates correctly	Erlind
ST-004	Logout functionality	Redirect to login page	Both

Test Execution Summary:

- All tests passed successfully.
- No critical issues found.
- Minor UI inconsistencies documented.

7. Conclusion

The testing process followed a comprehensive approach, covering all critical aspects of the system. Unit tests ensured individual module correctness, while integration and system tests verified seamless communication and overall functionality.

Key Takeaways:

- Achieved high code coverage and system reliability.
- Identified minor areas for improvement in documentation and logging.
- The system is ready for deployment with ongoing regression testing planned.