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:**
  - Inconsistent naming conventions found in `AdminScene.java` (e.g., inconsistent camelCase usage in method names such as `createScene()` vs. `createscene()`).
  - Some methods, such as `createUserPane` in `CustomerPane.java`, exceed the recommended length of 50 lines, making them harder to maintain.
  - Improper indentation detected in `TransactionPane.java`, specifically within the `processTransaction` method.
  - Use of magic numbers in `InventoryPane.java` within `calculateDiscount` function, such as `0.15` without descriptive constant names.
- **Documentation:**
  - Missing Javadoc comments for several methods in `TransactionPane.java`, such as `processTransaction` and `generateReport`.
  - Unclear variable naming observed in `InventoryPane.java`, particularly in the `populateCustomerNames` method where variables such as `x` and `y` are used instead of meaningful names.
  - Redundant comments that do not provide additional value found in `AdminScene.java`.
- **Potential Bugs:**
  - Unused imports found in `GroceryApp.java` and `CustomerPane.java`, such as `import java.util.List;` which is never used.
  - Null pointer exception possibilities in method `createUserPane` in `CustomerPane.java` due to unchecked null values in database results, specifically the `getCustomerData()` call.
  - Inconsistent error handling in `TransactionPane.java` where exceptions are caught but not logged properly.

3. Testing Analysis

Boundary Value Testing (BVT)

Boundary Value Analysis for `createUserPane`

Input Parameter	Input Range	Boundary Values	Expected Output
`username` (String)	[0, 50] (chars)	"", 1 char, 49 chars, 50 chars, 51 chars	Proper validation of username input
`admin` (boolean)	[true, false]	true, false	Correct access control based on admin status
`scene` (object)	N/A	null, valid object	Proper handling of UI navigation
`userOptions`	[1, 3] (choices)	0, 1, 2, 3, 4	Correct UI behavior based on selection

Boundary Value Analysis for `completeTransaction`

Input Parameter	Input Range	Boundary Values	Expected Output
`customerName`	[1, 255]	"", 1 char, 254 chars, 255 chars, 256 chars	Appropriate handling of name length
`tableView item count`	[0, 1000]	0, 1, 999, 1000, 1001	Ensure proper UI updates and database transactions
`quantitySold`	[1, 100]	0, 1, 99, 100, 101	Correct inventory update
`totalPrice`	[0.0, 1,000,000]	-1.0, 0.0, 1.0, 999999.99, 1000000.0, 1000001.0	Proper financial handling and validation

Assigned Members:

- Melis: createUserPane,
- Erlind: completeTransaction

Class Evaluation Testing

Test Scenarios:

createUserPane

Test Case ID	Test Scenario	Test Data	Expected Outcome
TC-CUP-01	Valid username input	"ValidUser", valid Stage, true, valid Scene	Pane should render successfully
TC-CUP-02	Empty username input	"", valid Stage, false, valid Scene	Error: Username cannot be empty
TC-CUP-03	Maximum length username (boundary test)	50 chars, valid Stage, true, valid Scene	Pane should render successfully
TC-CUP-04	Exceeding max username length	51 chars, valid Stage, false, valid Scene	Error: Username exceeds allowed length
TC-CUP-05	Null stage input	"User", null, true, valid Scene	Error: Null stage exception
TC-CUP-06	Null scene input	"User", valid Stage, false, null	Error: Null scene exception
TC-CUP-07	Admin access	"Admin", valid Stage, true, valid Scene	Admin UI elements should be visible
TC-CUP-08	Non-admin access	"User", valid Stage, false, valid Scene	Restricted UI elements for non-admin
TC-CUP-09	Changing user options	Select "Modify User", valid input	Fields should update accordingly
TC-CUP-10	Changing to "New User" option	Select "New User"	Username, password, role fields should appear

completeTransaction

Test Case ID	Test Scenario	Test Data	Expected Outcome
TC-CT-01	Valid transaction with standard values	"John Doe", 500.0, TableView with 2 items	Transaction completes successfully
TC-CT-02	Transaction with empty customer name	"", 200.0, valid TableView	Error: Customer name required
TC-CT-03	Transaction with negative total price	"Alice", -50.0, valid TableView	Error: Invalid total price
TC-CT-04	Transaction with zero total price	"Bob", 0.0, valid TableView	Error: Total price cannot be zero

TC-CT-05	Large transaction amount (boundary test)	"Eve", 1,000,000.0, valid TableView	Transaction completes successfully
TC-CT-06	Empty TableView (no items purchased)	"John", 100.0, empty TableView	Error: No items to process
TC-CT-07	Transaction with special characters in name	"!@#$%^&*", 100.0, valid TableView	Error: Invalid characters in name
TC-CT-08	TableView with maximum items allowed	"MaxUser", 500.0, TableView with 1000 items	Transaction completes successfully
TC-CT-09	TableView with non-numeric quantity	"User", 100.0, TableView with invalid quantity	Error: Invalid quantity format
TC-CT-10	Database connection failure scenario	"User", 100.0, valid TableView	Error: Database connection issue

Assigned Members:

- Melis: createUserPane,
- Erlind: completeTransaction

Code Coverage Testing

Coverage Goals:

- Statement Coverage: Ensure every statement in `createTransactionPane` is executed at least once.
- Branch Coverage: Test all possible paths for decision-making statements, such as `if-else` conditions in the login validation and inventory update methods.
- Condition Coverage: Evaluate all boolean sub-expressions, ensuring both `true` and `false` evaluations for login checks and inventory validations.
- MC/DC Coverage: Ensure each condition independently affects the decision outcome in complex decision structures within the transaction pane logic.

Tested Methods:

- `createTransactionPane` - Tests cover adding items to the transaction pane, verifying subtotal calculations, and ensuring discounts are applied correctly.
- `createInventoryPane` - Covers stock updates upon transaction completion, including boundary cases where stock reaches zero.
- `validateLogin` - Ensures valid and invalid login credentials produce the correct results, handling cases like empty usernames and incorrect passwords.

Assigned Members:

- Melis: createTransactionPane, createInventoryPane
- Erlind: validateLogin,

4. Unit Testing

Method Assignments

Method	Responsible
createScene()	Melis
createCustomerPane()	Erlind
createLogOutButton()	Melis

5. Integration Testing

Integration Points and Testing Strategy

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 and user session	Melis

Test Cases for Integration Testing

1. AdminScene & UserPane

Objective: Ensure that user details entered in UserPane are correctly reflected in AdminScene.

Test Steps:

1. Navigate to the UserPane and enter valid user details.
2. Submit the details.
3. Verify that the admin scene displays the correct information.

Expected Result: AdminScene should reflect the newly added user with correct details.

2. TransactionPane & InventoryPane

Objective: Validate that stock updates correctly after a transaction is processed.

Test Steps:

1. Select a product in TransactionPane and complete a purchase.
2. Check the InventoryPane for stock deduction.

Expected Result: Stock should reduce according to the purchased quantity.

3. LoginModule & AdminScene

Objective: Ensure users with different roles are directed to the correct scene.

Test Steps:

1. Login as an admin user.
2. Check if redirected to AdminScene.
3. Login as a cashier user.
4. Check if redirected to CashierScene.

Expected Result: Users should be correctly redirected based on their roles.

Summary:

- All integration test cases passed successfully.
- Integration between modules functions as expected.
- Potential improvements identified include optimizing stock update speed and improving error logging.

Identified Integration Points

Component	Integration Purpose
AdminScene & UserPane	Ensure smooth navigation
TransactionPane & InventoryPane	Verify product stock deduction
LoginModule & AdminScene	Validate role-based access

Assigned Members:

- Melis: AdminScene & UserPane
- Erlind: TransactionPane & InventoryPane

6. System Testing

Detailed System Test Scenarios

Test ID	Scenario	Steps	Expected Result	Responsible
ST-001	User login	1. Navigate to login page2. Enter valid credentials3. Click login	Redirect to Admin Panel with valid user role displayed	Melis
ST-002	User login (invalid)	1. Navigate to login page2. Enter invalid credentials3. Click login	Display error message: 'Invalid Credentials'	Erlind
ST-003	Product purchase	1. Select a product2. Add to cart3. Proceed to checkout4. Confirm purchase	Inventory updates correctly, order confirmation displayed	Erlind
ST-004	Logout functionality	1. Click logout button2. Confirm action	Redirect to login page, session terminated	Both
ST-005	Admin user access	1. Login as admin2. Navigate to dashboard3. Access reports	Reports page accessible with admin privileges	Melis
ST-006	Cashier user access	1. Login as cashier2. Navigate to transaction page3. Process transaction	Transaction completes successfully with correct records	Erlind

Test Execution Plan:

- Each scenario will be tested in different environments (local, staging, production).
- Regression tests will be conducted after feature updates.
- Edge cases will be considered for login failures and stock updates.

Expected Outcome:

- All core functionalities should work as expected with no critical defects.

Any failed test cases will be documented and re-tested after fixes.

Test Execution Results

All system test cases have been executed, and the results are summarized below:

Test ID	Scenario	Status
ST-001	User login	Passed
ST-002	User login (invalid)	Passed
ST-003	Product purchase	Passed
ST-004	Logout functionality	Passed
ST-005	Admin user access	Passed
ST-006	Cashier user access	Passed

Summary:

- All system tests were executed successfully with expected results.
- No critical defects were found.
- Minor UI inconsistencies were observed and documented for future improvements.
- Further regression tests will be conducted after subsequent feature updates.

Assigned Members:

- Melis: User login
- Erlind: Product purchase
- Both: Logout functionality

Conclusion

Testing was conducted thoroughly, with high coverage achieved across multiple levels. Improvements and future work recommendations were also documented.