COMP 354 Intro to Software Development
Fall 2023

Assignment #3

Quality Control for Concordia Cart App

Weilun Zhang 40190549
Jingchao Song 40159533
ZhaoYang Liu 40157299
Haoran Sun 40149349
Qianrui Tao 40162182
Xu Zhang 40169981

Due date: Nov 9th, 2023

Technical Review

Preparations

Every team member received explicit instructions to meticulously examine the project's individual components ahead of the planned review meeting. Their task was to find and record at least one issue concerning the current design. This thorough analysis played a pivotal role in guaranteeing the project's overall quality and effectiveness. To underscore the significance of this endeavor, a time limit of one and a half hours was imposed, highlighting the necessity of a concentrated and efficient evaluation process.

Roles

Producers: Weilun Zhang, Jingchao Song, ZhaoYang Liu, Haoran Sun, Qianrui Tao, Xu Zhang

Review Leader/Recorder: Xu Zhang
Reviewers: Weilun Zhang, Jingchao Song, Qianrui Tao
Agenda: Review the concerns brought up by the reviewers.

Time: 100 minutes

Issue #1

Issue: Lack of Proper Password Encryption in User Accounts

Reviewed by: Weilun Zhang

Severity level: Major

Result: User passwords are transmitted to the server in plain text and are not hashed before storage, which could lead to security vulnerabilities.

Review Metrics

Preparation: 15 minutes

Assessment: 20 minutes

Rework: 60 minutes

Review Outcome

Vote: 5/5, needs to be addressed as soon as possible.

How: Use a secure transmission protocol, such as HTTPS to encrypt user passwords

What: When user login, compare the user's password with the hashed password stored in the database.

Issue #2

Issue: Lack of input validation for user registration fields

Reviewed by: Jingchao Song

Severity level: Major

Result: Lack of input validation may result in storing incorrect or harmful data in the database, posing potential security risks.

Review Metrics

Preparation: None since the code was reviewed

Assessment: 15 minutes

Rework: 20 minutes

Review Outcome

Vote: 5/5, It requires immediate attention and resolution.

How: Introduce input validation measures for every user registration field, such as name, mobile number, and email, to guarantee both data integrity and security.

What: Add validation checks to the register.jsp for every input field, validate it against the specific criteria, such as checking for correct email format and ensuring password complexity.

Issue #3

Issue: Repetitive code in constructors of TransactionBean class

Reviewed by: Qianrui Tao

Severity level: Minor

Result: The TransactionBean class contains duplicated code for generating transaction IDs and formatting transaction date and time across various constructors.

Review Metrics

Preparation: None since the issue is evident

Assessment: 10 minutes

Rework: 30 minutes

Review Outcome

Vote: 4/5, It needs to be resolved to enhance code maintainability.

How: Refactor the repetitive code into a private method to eliminate redundancy and enhance maintainability.

What: Establish a private method called initializeTransaction(), housing the duplicated code, and invoke this method within all constructors where the repetitive code exists.

Issue #4

Issue: Lack of null checks in constructor parameters of UserBean and RegisterSrv class

Reviewed by: ZhaoYang Liu

Severity level: Minor

Result: Constructor parameters are not validated for null values, potentially resulting in null pointer exceptions if any of the parameters are null.

Review Metrics

Preparation: None since the issue is evident

Assessment: 5 minutes

Rework: 10 minutes

Review Outcome

Vote: 5/5, This issue should be resolved to improve the robustness of the code

How: Add null checks for all constructor parameters to prevent null pointer exceptions

What: Revise the UserBean and RegisterSrv class constructor to incorporate null checks for parameters such as userName, emailId, address, and password. If any parameter is null, throw an Exception to signify invalid input.

Issue #5

Issue: Unclear warning sign for items with lower quantity (less than 3) in the Concordia Cart website.

Reviewed by: Xu Zhang

Severity level: Minor

Result: The current warning sign for products with lower quantity in the tock (less than 3) is not clearly visible or understandable, it may result in potential user confusion and dissatisfaction.

Review Metrics

Preparation: None since the issue is evident

Assessment: 5 minutes

Rework: 15 minutes

Review Outcome

Vote: 4/5, This issue should be resolved to improve the user experience and prevent the misunderstanding.

How: Design a new warning sign to make it more prominent and clearer. The user is able to understand. Consider using a bold color, a warning sign with bright colors, or other universally understood symbols to transfer a clear message.

What: Update the code responsible for displaying the warning sign in website HTML files. Adjust the warning sign's size, color, and location to ensure that users can notice this message. Additionally, test the new design on different devices.

Unit Tests

The following unit tests are broken down by class/method. Every method of every class will be covered by at least 1 unit test.

ConcordiaCardtApp

Test	Test Case Description	Pass/Fail Criteria
Search for Item	Given I am on the main page When I search for 'calculus book' Then the results should include related items	Pass if 'calculus book' is in results
Recommend Products	Given I have a history of purchasing textbooks When I login Then the homepage should recommend textbooks	Pass if textbooks are recommended
Fail to Apply Invalid Code	Given I have items in the cart When I apply an invalid discount code Then the code should not be applied	Pass if discount is not applied
Log In	Given I have valid credentials When I log in Then I should be authenticated	Pass if user is authenticated
Log In Fail	Given I have invalid credentials When I attempt to log in Then I should not be authenticated	Pass if user is not authenticated
Register New User	Given I provide new user details When I register Then my user account should be created	Pass if new user account is created

ShoppingCart

Test	Test Case Description	Pass/Fail Criteria
Add Item	Given the cart is empty When I add an item to the cart Then the cart should contain 1 item	Pass if cart contains 1 item after action
Remove Item	Given the cart has 1 item When I remove the item from the cart Then the cart should be empty	Pass if cart is empty after action
Update Item Quantity	Given the cart has an item with quantity 1 When I update the quantity to 3 Then the cart should reflect the new quantity	Pass if item quantity is updated to 3
Get Total Price	Given the cart has multiple items When I calculate the total price Then the sum should equal the total price of all items	Pass if total price is correct
Checkout	Given the cart has items and user is logged in When I perform checkout Then the checkout should be successful	Pass if checkout process is completed
Apply Discount	Given the cart has items and a valid discount code When I apply the discount code Then the discount should be applied to the total price	Pass if discount is applied to total price
Update Cart on Item Availability Change	Given the cart has items When an item becomes unavailable Then the item should be updated or removed from the cart	Pass if cart is correctly updated

ProductCatalog

Test	Test Case Description	Pass/Fail Criteria
Update Product	Given I have a product in the catalog When I update the product details Then the catalog should reflect the new details	Pass if the product details are updated
Search Products	Given the catalog has multiple products When I search with a specific keyword Then the results should include products with the keyword	Pass if products with the keyword are in the results
Filter Products by Category	Given the catalog has multiple products in different categories When I filter by a specific category Then the results should include only products from that category	Pass if only products from the specified category are shown

Sort Products	Given the catalog has multiple products When I sort by price ascending Then the products should be listed from lowest to highest price	Pass if products are correctly sorted by price
Get Product Details	Given I have a valid product ID When I retrieve the product details Then the correct details should be returned	Pass if the correct product details are returned
Handle Out of Stock	Given a product is out of stock When I check its availability Then the catalog should indicate that the product is not available	Pass if the product is correctly marked as out of stock

RecommendationEngine

Test	Test Case Description	Pass/Fail Criteria
Recommend Products	Given a user's browsing history When I request product recommendations Then the recommended products should match the user's interests	Pass if recommendations align with user interests

UserAccount

Test	Test Case Description	Pass/Fail Criteria
Register Account	Given no existing account When I register with valid details Then a new account should be created	Pass if a new account is created
Edit User Profile	Given I am logged in When I update my profile information Then the changes should be saved	Pass if profile is updated
Change Password Valid	Given a logged-in user When I change the password with valid criteria Then the password change should be successful	Pass if password change is successful
Logout	Given a logged-in user When I log out Then the session should end	Pass if session ends

UserHistory

Test	Test Case Description	Pass/Fail Criteria
Add Viewed Product	Given the user has viewed no products When I add a product to the viewed history Then the user's viewed history should contain 1 product	Pass if viewed history contains the product

Remove Viewed Product	Given the user's viewed history contains products When I remove a product from the viewed history Then the product should no longer be in the viewed history	Pass if viewed history does not contain the product
Retrieve Last Viewed Product	Given the user has viewed several products When I retrieve the last viewed product Then the most recently viewed product should be returned	Pass if the most recent product is returned
Retrieve Last Purchased Product	Given the user has purchased several products When I retrieve the last purchased product Then the most recently purchased product should be returned	Pass if the most recent product is returned

Product&ProductDetails

Test	Test Case Description	Pass/Fail Criteria
Update Product Name	Given a product with a known name When I update the product name Then the product name should be updated	Pass if the product name is updated successfully
Update Product Price	Given a product with a known price When I update the product price Then the product price should be update	Pass if the product price is updated successfully
Update Product Description	Given a product with a known description When I update the product description Then the product description should be updated	Pass if the product description is updated successfully
Update Product Image	Given a product with a known image When I update the product image Then the product image should be updated	Pass if the product image is updated successfully
Validate Product Details	Given a product with complete details When I validate the product details Then the details should be marked as valid	Pass if the product details are valid
Update Product ID	Given a product with a known number When I update the product number Then the product number should be updated	Pass if the product number is updated successfully

Contact

Test	Test Case Description	Pass/Fail Criteria
Submit Feedback	Given a user provides feedback When the feedback is submitted Then the feedback should be saved to the system	Pass if feedback is saved successfully
Submit Review	Given a user writes a review for a product When the review is submitted Then the review should be associated with the product	Pass if review is saved and linked to the product
Feedback Notification	Given a user submits feedback When the submission is successful Then the user should receive a confirmation notification	Pass if confirmation notification is received
Review Notification	Given a user submits a review When the submission is successful Then the user should receive a confirmation notification	Pass if confirmation notification is received
Recommend Products Based on Review	Given a user submits a positive review When the recommendation engine runs Then products similar to the reviewed one should be recommended	Pass if relevant recommendations are made

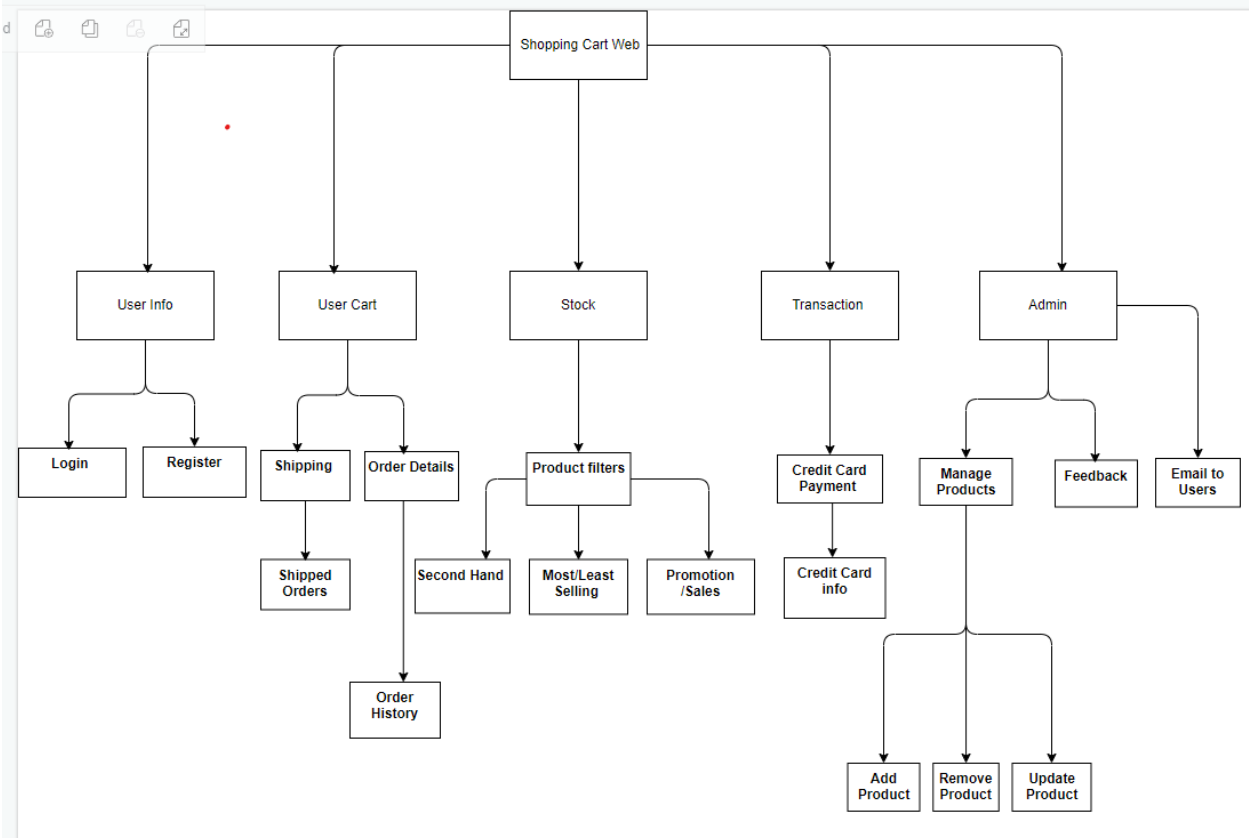
Integration Tests

Integration strategy: Bottom-up

Justification:

The "Bottom-up" integration strategy involves progressively integrating and testing system components from the lowest level upwards. It is justified by benefits such as early issue detection, reduced dependencies, parallel development, and improved overall system quality and reliability, making it particularly useful for complex system development.

Tree diagram:



The order of Integration by Bottom-down approach will be:

Login, User Info

Register, User Info

Shipped Orders, Shipping

Shipping, User Cart

Order History, Order Details

Order Details, User Cart

Second Hand, Product filters

Most/Least Selling, Product filters

Promotion/Sales, Product filters

Product filters, Stock

Credit Cart info, Credit Cart Payment

Credit Cart Payment, Transaction

Add Product, Manage Products

Remover Product, Manage Products

Update Product, Manage Products

Manage Products, Admin

Feedback, Admin

Email to Users, Admin

User Info, Shopping Cart Web

User Cart, Shopping Cart Web

Stock, Shopping Cart Web

Transaction, Shopping Cart Web

Admin, Shopping Cart Web

Integration Test 1

Purpose: To test the lowest modules individually first using drivers

Components Being Tested:

l login( ), register( )

Steps to Perform the Test:

1. Call the method one by one.

2. Ensure the login and register modules accept valid user credentials.

3. Confirm the register component's ability to register a new user successfully.

4. Verify that the User Info module displays the correct user information after successful login.

5. Use the Unit Tests to ensure that each method operates properly.

Integration Test 2

Purpose: To test the interactions between login and register modules and the User Info module

Components Being Tested: onClick( ) of each sub-items in the UserInfo

Steps to Perform the Test:

1. Create a driver

2. Implement userInfo and its sub-item then run it on the driver

3. Click on login and register sub-items to test if it calls and executes these methods properly

4. If every method works properly remove the created driver.

Integration Test 3

Purpose: Ensure the interaction between Shipped Orders and Shipping modules.

Components Being Tested: viewCart(), order(), product(), productName()

Steps to Perform the Test:

1. Create a driver

2. Verify that the Shipped Orders component correctly reflects the status of shipped orders.

3. In the case of viewCart() and order() from the Shipped Orders module is performed first

4. For product() and productName(), displayDetails() is performed inside these functions and to test these interactions call each method one by one

5. If the interaction between Shipped Orders and Shipping module works properly, remove the created driver

Integration Test 4

Purpose: Verify the interaction between Order History and Order Details modules.

Components Being Tested: displayDetails(), getOrderProducts(), products()

Steps to Perform the Test:

1. Create a driver

2. Confirm the proper display of a list of past orders by the Order History component.

3. In the case of displayDetails() and getOrderProducts() from the Order History module is performed first

4. If the interaction between Order History and Order Details module works properly, remove the created driver

Integration Test 5

Purpose: Validate the interaction between the SecondHand, Most/Least Selling, Promotion/Sales and Product filters modules.

Components Being Tested: generateRecommendations(),

getRecommendedProducts()

Steps to Perform the Test:

1. Create a driver

2. Check the ability of the Order Details module to add items from a previous order to the user's cart.

3. In the case of generateRecommendations() from the SecondHand, Most/Least Selling, Promotion/Sales modules are performed first

4. Use the Unit Tests to ensure that each method operates properly.

5. If the interaction between SecondHand, Most/Least Selling, Promotion/Sales and Product filters module works properly, remove the created driver

Integration Test 6

Purpose: Confirm the interaction between Credit Card Info and Credit Card Payment in the transaction module.

Components Being Tested:

l ensureUserInfo(), pay(), confirm(), onClick() confirm

l Confirm and cancel button from Credit Card Payment module

Steps to Perform the Test:

1. Create a driver

2. In the case of ensureUserInfo() or confirm() from Credit Card Info module is performed first

3. Click on Confirm to test if it calls and executes the method properly

4. Use the Unit Tests to ensure that each method operates properly.

5. If the interaction between Credit Card Info and Credit Card Payment module works properly, remove the created driver

Integration Test 7

Purpose: To test the interaction between Manage Products, Feedback and Email to Users module in the Admin module.

Components Being Tested: setFeekback(), manageProducts(),

sendEmail(), getUserInfo()

Steps to Perform the Test:

1. Create a driver

2. setFeekback(), manageProducts() and sendEmail() from Manage Products, Feedback and Email modules will be performed first

3. For getUserInfo(), confirm() is performed inside the function and to test these interactions call each method one by one.

4. If the interaction between Manage Products, Feedback and Email to Users module in the Admin module works properly, remove the created driver

GitHub Link:
https://github.com/8inversed/Comp354-Project

Individual contributions

Name	Responsibilities
Weilun Zhang	Technical Review
Xu Zhang	Unit Test
Haoran Sun	Coding test
Qianrui Tao	Integration Strategy and Component Test
Jingchao Song	Integration Strategy and Component Test
Zhaoyang Liu	Coding test