Software Quality Engineering

Assignment – 1: Jira Manual Testing

Course Instructor

Madam Uzma Mahar

Submitted by

Abdullah Daoud…….(22I-2626)

Section

SE-E

Date

Monday, September 16, 2024

Fall 2024

Department of Software Engineering

FAST – National University of Computer & Emerging Sciences

Islamabad Campus

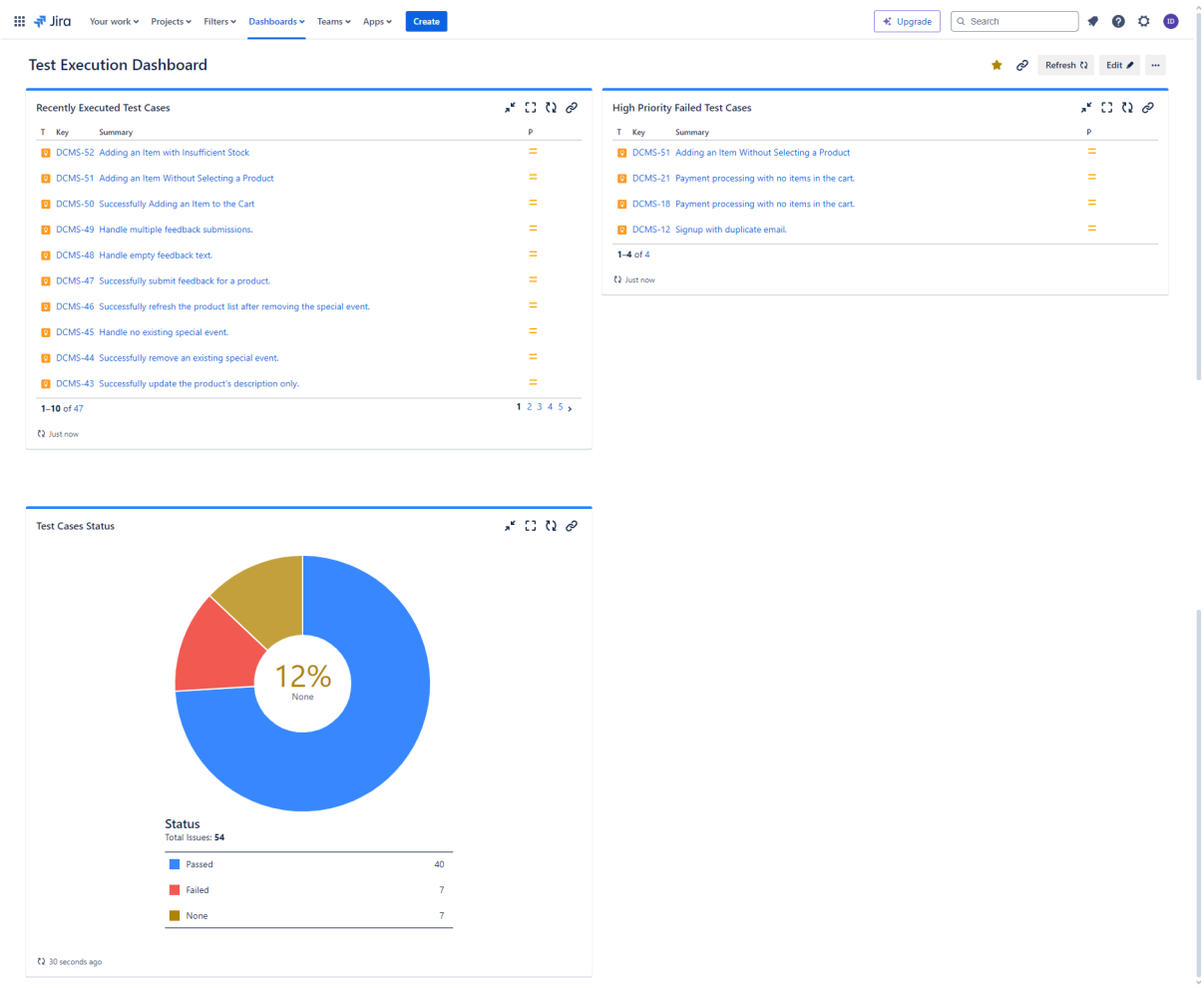
Table of Contents

1. Project for Assignment

The project used for this assignment was the Spring 2024 semester's Database System's 2 person group project titled "DaVinCheese Café", which is a Café management system built with C# and SQL Server

2. Jira Dashboard Screenshots and Explanation

(i). Test Execution Dashboard



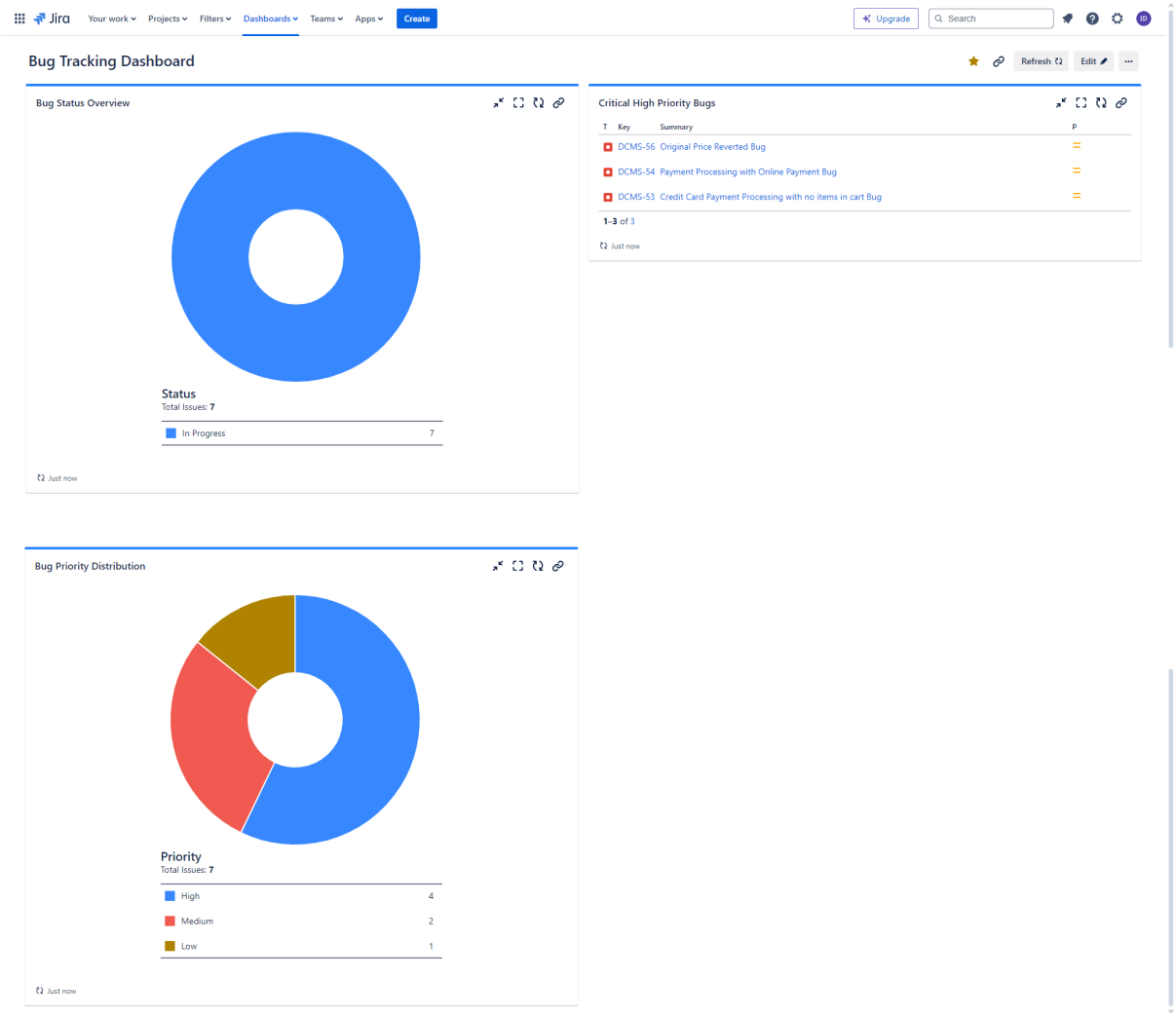
The test execution dashboard consists of 3 main gadgets used to represent important information discovered about test cases:

1st gadget represents a list of the most recently executed test cases. It represents the test cases that were most recently tested and marked as done

2nd gadget represents a list of high-priority failed test cases. It represents the test cases whose status was failed meaning the test case was failed and whose priority was high. From the screenshots, there were only 4 high-priority failed test cases

3rd gadget represents a pie chart that gives information about the amount of test cases that were passed and failed. From the screenshots, there were a total of 47 test cases, 40 of which were passed and 7 of which were failed

(ii). Bug Tracking Dashboard



The bug-tracking dashboard consists of 3 main gadgets used to represent important information discovered about bugs for test cases:

1st gadget represents a pie chart for giving an overview of the bugs' status. From the screenshots, the bug status is “In Progress”, since in this assignment we have only identified the bugs and reported them so that they can be worked on in the future

2nd gadget represents a list of critical high-priority bugs. It represents the bugs whose priority was high and whose severity was critical. From the screenshots, there were only 3 critical high-priority failed bugs

3rd gadget represents a pie chart that gives information about the priority distribution of bugs. From the screenshots, there were a total of 7 bugs reported, 4 of which were high, 2 of which were medium, and 1 of which was low

3. Analysis of Test Execution Results

After manually testing 47 test cases for various functions, the pass/fail ratio was **40 passed** and **7 failed**, resulting in a **pass rate of approximately 85%**. Most of the failed cases were related to issues like data validation, incorrect business logic, and inadequate handling of edge cases

The **trends** identified from the failed test cases suggest that the application struggles with:

- Proper validation of user input (e.g., duplicate emails, empty feedback, zero quantity).
- Correctly managing states and transitions, such as reverting prices after removing discounts.
- Ensuring processes only continue when prerequisites are met (e.g., payment with an empty cart).

The majority of the successfully passed test cases indicate that basic functionalities (adding items to the cart, editing menu items, and giving feedback) work as expected. However, the **failed test cases** demonstrate areas where user actions do not align with business rules or where there are logical errors in the implementation

4. Complexity of Test Scenarios

The test scenarios covered various aspects of the application, including:

- **User Interactions:** Like signing up, providing feedback, and adding items to the cart
- **Data Operations:** Involving database interactions for storing and retrieving information (e.g., menu item edits, discount application, etc.)
- **Input Validation:** Ensuring correct and meaningful data is provided by users in different forms
- **Edge Cases:** Such as attempting payments without items in the cart and ensuring discounts are reverted properly

The complexity of these test scenarios stems from their need to **simulate real-world usage** of the application, which involves handling unexpected input and managing multiple dependent states. The scenarios that are covered, together provide comprehensive coverage of the application's core functions. However, the identified bugs suggest there may still be gaps in input validation, error handling, state management, and business logic execution

5. Summary of Critical Bugs

Three critical bugs were identified during testing:

1. **Online Payment with No Items in Cart:**
    - **Impact:** This bug allows users to proceed with online payment transactions even when the cart is empty, leading to potential billing issues, customer dissatisfaction, and possible financial discrepancies
    - **Proposed Solution:** Introduce a check before proceeding with any payment method to ensure that the cart contains at least one item. If the cart is empty, display an error message and prevent payment
2. **Credit Card Payment with No Items in Cart:**
    - **Impact:** Similar to the online payment bug, this bug permits transactions with an empty cart, which is not valid behavior and can confuse users or result in billing errors
    - **Proposed Solution:** Add validation to the credit card payment process to ensure the cart has items before allowing the transaction. If no items are found, prompt the user to add items to the cart
3. **Original Price Not Reverted After Removing Discount:**
    - **Impact:** When a special discount is removed, product prices are not reverted to their original values, which affects pricing accuracy and can lead to revenue loss or customer complaints
    - **Proposed Solution:** Store the original prices before applying any discounts. When removing a discount, use these stored values to reset the prices to their correct amounts

These bugs are **high-priority** as they directly affect the **financial operations** of the application and could disrupt user experience by a big margin

6. Reflection on the Testing Process and Lessons Learned

The testing process was thorough in covering various functional aspects of the application. It included typical use cases as well as edge cases, which were essential in identifying critical bugs. However, the manual testing of 47 cases highlighted a few areas for improvement:

- **Importance of Validation:** The failed test cases underscore the need for robust input validation, especially in forms, payment processes, and data operations. Enhancing validation logic could prevent most of these bugs
- **Edge Case Handling:** Critical bugs like allowing payments with an empty cart indicate that edge cases need more focused testing and handling in the code. In future tests, more attention should be given to scenarios that might seem unlikely but can break the application's core functionalities
- **State Management:** The issue with prices not reverting after removing a discount suggests that state management (e.g., maintaining original values) needs careful consideration during both coding and testing
- **Automation:** Given the number of test cases, automating the testing process would have saved time and increased accuracy in identifying failures. Automated tests could quickly validate input fields, transitions, and complex scenarios like the ones that failed in this cycle
- **Prioritize Critical Functionalities:** Testing payment processes and pricing mechanisms should be a priority since these are central to the application's success and user trust.

In summary, while most functionalities work as intended, critical bugs highlight the need for improved validation and state management. Addressing these issues will enhance both the robustness and usability of the application. Moving forward, focusing on input validation, complex state changes, and automating repetitive test cases will be key to refining the application