

Test Summary Report

Of ToDo Web App

**Version 1.0**

**24 Nov 2023**

**BAJRA TECHNOLOGIES**

**Prepared By: Sampada Suwal**

**Date of Submission: 16th January 2024**

**Table Of Contents**

[1 About Project 3](#_Toc156283106)

[2 Executive Summary 3](#_Toc156283107)

[3 Test Objective 4](#_Toc156283108)

[4 Test Strategy for Todo WebApp: 4](#_Toc156283109)

[5 Modules Information 5](#_Toc156283110)

[6 Test Report 6](#_Toc156283111)

[7 Overall Conclusion 11](#_Toc156283112)

# About Project

|  |  |
| --- | --- |
| Project Name | ToDo WebApp |
| Version | **1.0** |
| Testing Period | **10th January – 15th January 2024** |
| Test Environment | **Windows 11, JDK 8.0, Eclipse IDE using Selenium in Java, Chrome Driver** |

# Executive Summary

**Project Overview:**

Todo WebApp is a web-based application designed to empower users with the ability to create tasks and lists, providing a streamlined task management experience. The application aims to enhance productivity and organization in a user-friendly manner.

**Key Features:**

1. Task Creation: Users can easily create tasks, specifying details such as task name, priority, and due date.
2. List Creation: The application allows users to create lists, enabling them to categorize and organize tasks efficiently.
3. Date and Time Functionality: Todo WebApp supports date and time settings for tasks, facilitating better scheduling and planning.

**Executive Brief of Testing:**

During the testing phase, Todo WebApp exhibited commendable performance in delivering its core features. The task creation and list creation functionalities proved to be intuitive and user-friendly. Users can seamlessly create, edit, and manage tasks, providing a flexible platform for personal and professional use.

**Testing Highlights:**

1. Positive and negative test scenarios were conducted for user registration, task creation, and list creation.
2. The application demonstrated reliability and consistency in handling various user inputs.
3. Date and time-related functionalities were thoroughly validated, ensuring accurate scheduling.

# Test Objective

The primary objective of testing Todo WebApp is to ensure the delivery of a high-quality and reliable task management application that meets user expectations and fulfills its intended purpose. The testing process encompasses various aspects, each serving a specific goal:

**Functional Testing:**

1. Task Creation: Verify that users can create tasks with accurate details, including task names, priorities, and due dates.
2. List Creation: Confirm that users can create lists to categorize and organize tasks effectively.
3. Date and Time Functionality: Ensure that the application correctly handles date and time settings for tasks.

**Usability Testing:**

1. Evaluate the overall user interface and experience to ensure it is intuitive and user-friendly.
2. Validate that users can easily navigate through the application and perform essential actions without confusion.

**Performance Testing:**

1. Assess the application's responsiveness under different load conditions to ensure optimal performance.
2. Check for any latency issues or delays in task and list creation processes.

.

**Error Handling and Recovery Testing:**

1. Test the application's response to invalid inputs to ensure proper error messages and user guidance.

# Test Strategy for Todo WebApp:

The test strategy for Todo WebApp involves a combination of exploratory testing, manual test case generation, and subsequent automation using Selenium in Java. This comprehensive approach ensures thorough coverage of functional, usability, performance, security, and compatibility aspects.

**Exploratory Testing:**

1. Objective: Explored the application organically to identify potential defects, usability issues, and areas of improvement.
2. Execution: Manually interact with the application without predefined test scripts.
3. Benefits: Uncover unexpected issues, assess the user experience, and gather insights for further testing.

**Manual Test Case Generation:**

1. Objective: Systematically document positive and negative test scenarios based on exploratory testing findings.
2. Process:

* Categorize identified scenarios into positive and negative cases.
* Document detailed steps, expected results, and preconditions.
* Prioritize test cases based on critical functionalities and user interactions.

**Automation Using Selenium in Java:**

1. Objective: Automate repetitive and time-consuming test cases to ensure efficient and consistent execution.
2. Tool Selection: Utilize Selenium WebDriver with Java bindings for browser automation.

**Scripting Approach:**

1. Develop test scripts to automate positive and negative scenarios.
2. Implement data-driven testing for diverse test data sets.
3. Implemented positive and negative test for each field for create account

**Logging:**

1. Uploaded Screenshots of errors displayed in the fields of the modules present in the system.

# Modules Information

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Modules | Sub-Modules | No. of Test Cases | Executed Test Cases |
| User Authentication | Create Account | 16 | 15 |
| Login/Logout |
| Forget Password |
| Create Tasks |  | 2 | 2 |
| Create Lists |  | 2 | 2 |
| Home Page | Dashboard | 2 | 2 |
| Sign Up and Login |
| Reset Password |
| Forget Password |

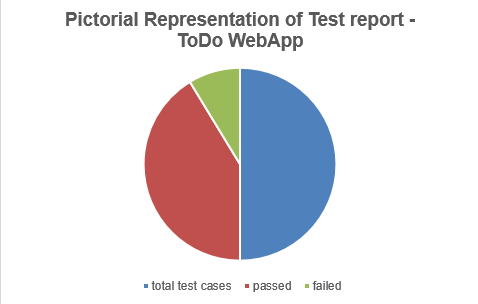
# Test Report

|  |  |
| --- | --- |
| Total Test Scenarios including both positive and negative | 23 |
| Passed | 19 |
| Failed | 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Priority** | **Test Descriptions** | **Expected Results** | **Actual Results** | **Test Status** |
| TC\_001 | High | **Positive Test Case - Create Account with all valid details**  Verify successful creation of new user account | 1. The user account is successfully created. 2. The system redirects to the login page. 3. The user can now log in using the newly created credentials. | Same as expected Results | PASS |
| TC\_002 | High | **Positive test Case-** Login using registered email ID | 1. System redirects to dashboard page 2. User can access the features of the system | Same as expected Results | PASS |
| TC\_003 | High | **Negative Test Case –** Login using non-registered email ID | * System displays email doesn’t exist error message * User cannot access the features of the system | Same as expected Results | PASS |
| TC\_004 | High | **Negative Test Case - Create Account with blank details**  Verify how the system handles when the input data is not provided | 1. System displays an error message indicating email is already in use. 2. User is not allowed to proceed with the registration where next step is creating password | Same as expected Results | PASS |
| TC\_005 | Medium | **Negative Test Case - Create account with Existing Email**  Verify the system handles existing email during account creation. | 1. System displays an error message indicating email is already in use. 2. User is not allowed to proceed with the registration where next step is creating password | Existing email was allowed to proceed with registration | FAIL |
| TC\_006 | Medium | **Negative Test Case –**  **Create account with an email w/o a domain**  Verify the system handles email without domain during account creation | 1. System displays an error showing invalid email 2. User is not allowed to proceed with registration where next step is creating password. | Same as expected Results | PASS |
| TC\_007 | Medium | **Negative Test Case –**  **Create account with an email with invalid domain**  Verify the system handles email without domain during account creation | 1. System displays an error showing email already exists. 2. User is not allowed to proceed with registration where next step is creating password. | Same as expected Results | PASS |
| TC\_008 | Medium | **Negative Test Case –**  **Create account with an email without proper top-level domain**  Verify the system handles email without domain during account creation | 1. System displays an error showing email already exists. 2. User is not allowed to proceed with registration where next step is creating password. | Same as expected Results | PASS |
| TC\_009 | High | **Negative Test Case –**  **Create account with Invalid name (name containing number)**  Verify the system handles invalid name during account creation | 1. System should display to user that Name can only contain letters in the real time when user inputs name with number. 2. User is not allowed to proceed with registration where next step is creating password. | Same as expected Results | PASS |
| TC\_010 | High | **Negative Test Case –**  **Create account with Invalid name (name containing special character)**  Verify the system handles invalid name during account creation | 1. System should display to user that Name can only contain letters in the real time when user inputs name with special character. 2. User is not allowed to proceed with registration where next step is creating password. | Same as expected Results | PASS |
| TC\_011 | High | **Negative Test Case-** Create account using invalid phone number (number containing more than 10 digits- IN CASE OF “Nepal”)  Verify system handles the invalid phone number or not | 1. User is not allowed to proceed with registration 2. System should prompt error displaying user to enter valid phone number | Same as expected Results | PASS |
| TC\_012 | High | **Positive Test Case –**  **Create Account using number of Norway (8 digits)**  Verify system handles the phone number of 8 digits when selected country “Norway” | User is allowed to proceed with registration | Same as expected Results | PASS |
| TC\_013 | Medium | **Negative Test Case –**  **Create account with Date of birth of future date**  Verify system handles the “future date” for Date of Birth during account creation | 1. User is not allowed to proceed with registration 2. System displays error of invalid Date of Birth | System Failed to validate the future value of Date of Birth | FAIL |
| TC\_014 | Medium | **Negative Test Case –**  **Create account with Date of birth of non-existing month**  Verify system handles the “non existing month” for Date of Birth during account creation | 1. User is not allowed to proceed with registration 2. System displays error of invalid Date of Birth displaying that “month” doesn’t exist | System Failed to validate the INVALID value of Date of Birth | FAIL |
| TC\_015 | High | **Positive Test Case-**  **Forget Password**  Verify how system behaves when existing email ID is sent to reset password | 1. System displays that email for reset has been sent. 2. User is allowed to reset their password using instructions in their email | Same as expected Results | PASS |
| TC\_016 | High | **Negative Test Case-**  **Forget Password**  Verify how system behaves when non registered email ID is sent to reset password | 1. System displays that email doesn’t exist error message 2. User is not allowed to reset their password as email is not registered. | Same as expected Results | PASS |
| TC\_017 | High | **Positive Test Case-**  **Create Task using valid task title** | 1. Task is created successfully 2. Created task should be displayed at “Today’s Task” section | Same as expected Results | PASS |
| TC\_018 | High | **Positive Test Case-**  **Create Task using invalid task title** | * Task is not created * System should display invalid task title error message | Same as expected Results | PASS |
| TC\_019 | High | **Positive Test Case-**  **Create List using valid list detail** | * List is created * Created list should be displayed in “List Names” section and as well as in “Choose list” option of “Create Task” * User can choose the list when he wishes to create another task | Same as expected Results | PASS |
| TC\_020 | High | **Positive Test Case-**  **Create List using invalid list detail** | * List is not created * System displays invalid list error message | Same as expected Results | PASS |
| TC\_021 | High | **Positive test Case – user logouts from the system** | * User is logged out of the system * System redirects to Home Page | Same as expected Results | PASS |
| TC\_022 | High | **Positive Test Case- Accessing dashboard from Home page after log out** | * System displays “alert message” prompting user to log in first to access the dashboard * After user clicks on “login” alert , system redirects to login page | Same as expected Results | PASS |
| TC\_023 | High | **Positive Test Case – Accessing Reset Password from home page** | * System redirects to login page and validates the credentials * After passing the credentials, password reset interface shall shown | Login page was redirected but resetting functionality was absent | FAIL |

# Overall Conclusion

|  |  |
| --- | --- |
| Total Test Scenarios including both positive and negative | 23 |
| Passed | 19 |
| Failed | 4 |



The testing efforts on Todo WebApp comprised a total of 23 test cases, out of which 19 passed successfully, while 4 test cases failed. The analysis of these results provides valuable insights into the application's strengths, weaknesses, and areas that require attention.

**Successful Aspects:**

1. Functional Validation: The majority of test cases passed, indicating that the core functionalities of Todo WebApp are working as intended. Users can create tasks, lists, and manage them effectively.
2. Positive Test Scenarios: Test cases related to positive scenarios, such as creating a new account, adding tasks, and managing lists, demonstrated the application's reliability.

**Areas for Improvement:**

1. Failed Test Cases: The four failed test cases highlight specific areas that need attention. Analyzing the nature of these failures will provide insights into potential bugs or usability issues.
2. Negative Test Scenarios: Failed test cases in negative scenarios may indicate weaknesses in error handling, input validation, or user feedback. Focusing on these areas can improve the application's resilience.
3. Usability Concerns: In-depth analysis of user interactions, especially in scenarios where test cases failed, can uncover usability concerns. Addressing these concerns enhances the overall user experience.

**Recommendations:**

1. Bug Fixing: Investigate and address the root causes of failed test cases. Implement bug fixes to ensure a more robust application.
2. Usability Enhancements: Consider incorporating user feedback and observations from failed scenarios to enhance the application's usability.
3. Error Handling: Strengthen error-handling mechanisms, particularly in scenarios where negative test cases failed, to provide clear and informative feedback to users.
4. Performance Optimization: Evaluate the performance of the application, especially in scenarios where larger data sets or complex operations may impact responsiveness.