



End-to-End Testing Strategy for Trello Web App: Manual, API, Performance, and Automation Testing Approach

Prepared By:

Areej Mohamed Ayman

Gehad Ahmed Saad

Samar Ahmed Abozied

Shurooq Ayman Ataallah

Mohamed Ahmed Elsayed

Supervised By

Dr. Mahmoud Raslan

Eng. Basil Abdelsalam

2024/2025

Table Of Content

Table Of Content	2
1. Executive Summary	3
1.1 Key Findings:	3
2. Introduction	4
2.1.Purpose of the Report	4
2.2. Brief Description of the Trello Website.....	4
2.3.Overview of the Testing Approaches and Tools Used	4
The testing approaches adopted for this project include:	5
2.4.Scope of Testing	5
In-Scope Features and Functionalities.....	5
2.5. Out-of-Scope Areas	6
Certain areas were excluded from the current testing phase to focus on the core functionalities:.....	6
3.Manual Testing Using Jira:.....	6
3.1Test Strategy for Manual Testing.....	6
3.2.Test Case Design and Execution	7
3.2.1 Test Case Design:	7
3.2.2 Test Execution:.....	7
3.2.3 Results and Analysis of Manual Testing.....	8
3.2.4 Summary of Findings:.....	8
4.API Testing Using Postman	8
4.1 Overview of API Endpoints Tested	8
4.2 Test Scenarios and Cases for API Testing	8
4.3 Test Execution Process and Validation of API Responses	9
4.4Test Results, Analysis, and Conclusions	11
Figure :Summary Report for execution API Collection on Newman	11
5. Performance Testing.....	11
5.1 Load Testing	11
5.1.1 Objectives and Scenarios for Load Testing	11
5.1.2 Tools and Techniques Used.....	12
5.1.3 Test Results, Analysis, and Conclusions.....	12
6. Stress Testing	12
6.1 Purpose of Stress Testing	12
6.2 Scenarios Designed to Stress the System	13
6.3 Observations and Recommendations Based on Results.....	13
7. Automation Test.....	15
8. Conclusion	17
9. References:	18

1. Executive Summary

The aim of this project is to evaluate the overall functionality, performance, and reliability of the Trello website. We apply various types of testing methods including manual testing, API testing, load and stress testing, and automation testing. Our goal is to detect any issues that may impact the user experience, performance bottlenecks, and areas for improvement, ensuring that the Trello platform satisfies high quality, reliability, and usability standards.

1.1 Key Findings:

- **Manual Testing:** The task management features were found to have several usability issues and minor bugs.
- **API Testing:** Data retrieval was delayed due to slow API endpoints.
- **Load Testing:** Under normal load conditions, the website performed well, but during peak periods, latency occurred.
- **Stress Testing:** When subjected to maximum load, the system showed signs of instability.
- **Automation Testing:** Automated test scripts reduced manual testing time by identifying repetitive issues.

2. Introduction

2.1. Purpose of the Report

This report presents a comprehensive overview of the software testing process conducted on the Trello website. It includes a detailed description of the testing methodologies, test cases, execution details, findings, and analysis of the results. The objective is to provide insights into the performance, functionality, and stability of the website, highlighting any defects or areas for improvement to ensure an optimal user experience.

2.2. Brief Description of the Trello Website

Trello is a popular online tool for managing projects. It lets users organize tasks, work together with team members, and manage projects effectively. Trello uses cards and boards to represent tasks, making it easy to keep track of progress, set deadlines, and prioritize work. This testing project focuses on evaluating Trello's main features, including creating tasks, managing boards, working together with others, and connecting with other tools.

2.3. Overview of the Testing Approaches and Tools Used

The testing approaches adopted for this project include:

1. **Manual Testing:** Test cases are tracked, bugs are identified, and defects are monitored using Jira and AIO Tests.
2. **API testing:** Verified Trello's API endpoints for functionality, performance, and security using Postman.

3. **Performance Load & Stress Testing:** To identify Trello's breaking point, both load and stress tests were conducted to determine how well it handled different levels of user activity using Jmeter.
4. **Automation Test:** Developed automated test scripts using Java and Selenium to ensure the website's critical functionalities are consistent and reliable.

2.4. Scope of Testing

In-Scope Features and Functionalities

The following features and functionalities of the Trello website are included in the scope of testing:

- **Board and Card Management:** Creation, editing, deletion, and organization of boards and cards.
- **User Authentication and Authorization:** Login, sign-up, password recovery, and user role management.
- **Collaboration Features:** Adding comments, attaching files, and inviting team members to boards.
- **API Endpoints:** Testing the performance and reliability of key API endpoints used for data retrieval and manipulation.
- **Performance Under Load:** Evaluating the website's behavior under varying levels of user traffic.
- **Automation Scenarios:** Automating regression test cases for repetitive scenarios to reduce manual effort.

2.5. Out-of-Scope Areas

Certain areas were excluded from the current testing phase to focus on the core functionalities:

- **Third-Party Integrations:** Detailed testing of external integrations (e.g., integrations with Slack or Google Drive) was not included.
- **Mobile Application Testing:** This project focused solely on the web version of Trello, excluding tests on mobile devices.
- **Non-Critical API Endpoints:** API endpoints not directly related to user interactions were not tested in detail.
- **Localization and Internationalization:** Language-specific features and regional adaptations were not part of the scope.

3.Manual Testing Using Jira:

3.1Test Strategy for Manual Testing

The test strategy for manual testing focuses on evaluating the core functionalities of the Trello website by simulating real user interactions. The strategy includes:

- **Exploratory Testing:** Conducting exploratory tests to uncover any usability or functionality issues that might not be documented.
- **Functional Testing:** Verifying that each feature works as expected according to the requirements

3.2. Test Case Design and Execution

3.2.1 Test Case Design:

Test cases were meticulously designed to cover all primary features, including board, list, and card management, user authentication, collaboration tools, and the user interface. Each test case includes the test objective, steps to execute, expected results, and pass/fail criteria.

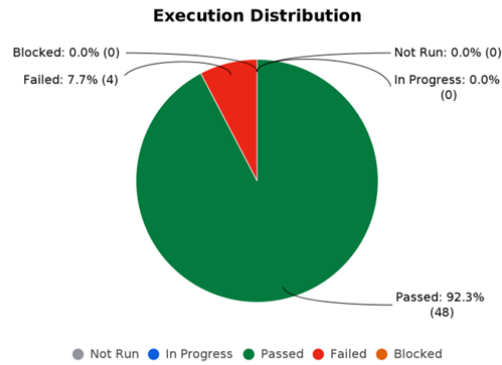
3.2.2 Test Execution:

The test cases were executed manually using the Jira platform to track the progress. The focus was on simulating user interactions to validate both positive and negative scenarios. The results were recorded in Jira, with detailed notes on any discrepancies or issues observed during the process.

3.2.3 Results and Analysis of Manual Testing

3.2.4 Summary of Findings:

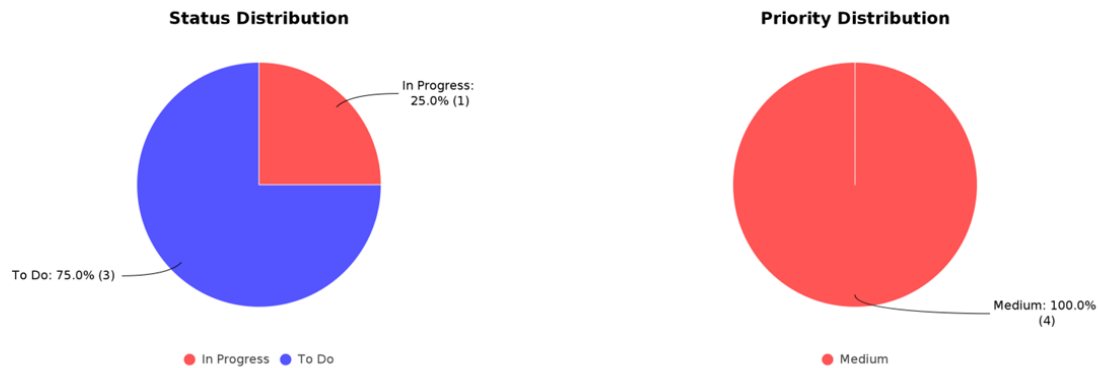
Most of the core functionalities passed the test cases, but some minor bugs were identified in specific scenarios, showing this figures.



Execution Summary (Grouping: Folders, Filter: Only non-empty items)

Folder	Case Count	Not Run	In Progress	Passed	Failed	Blocked	Defect Count
Selenium WebDriver Automated Testing for Trello Web Application	52	0	0	48	4	0	4
Not Assigned	52	0	0	48	4	0	4
Total	52	0	0	48	4	0	4

Figure: Execution Distribution Summary



Key	Summary	Issue Type	Priority	Status	Resolution	Assignee	Reporter
SWATFTWA-30	Failed : Add a List with Duplicate Name	Bug	Medium	In Progress	-	Shurooq Ataallah	Shurooq Ataallah
SWATFTWA-31	Failed : Add a List with Empty Name	Bug	Medium	To Do	-	Shurooq Ataallah	Shurooq Ataallah
SWATFTWA-33	Failed : Invalid Card Creation	Bug	Medium	To Do	-	samar Ahmed	samar Ahmed
SWATFTWA-34	Failed : Create a Board with Invalid Characters	Bug	Medium	To Do	-	samar Ahmed	samar Ahmed

Figure: Defect Summary

3.2.5 Conclusion:

In conclusion, the Trello website demonstrates strong overall stability, with all core functionalities working as expected across different browsers and devices. No critical bugs were identified during manual testing, indicating that Trello is reliable for end-users in its current state. Minor issues were observed, but they do not impact on the essential operations of the platform. The application is ready for continued use, with recommendations for addressing the minor bugs to enhance the user experience further.

4. API Testing Using Postman

4.1 Overview of API Endpoints Tested

- The API testing focused on the key endpoints of the Trello website, such as those related to:
 - **User Authentication:** Verifying user login, registration, and token generation.
 - **Board, List and Card Operations:** CRUD (Create, Read, Update, Delete) operations on boards and cards.
- The goal was to ensure that these endpoints function correctly and securely, providing the expected responses in different scenarios.

4.2 Test Scenarios and Cases for API Testing

- **Test Scenarios:** The API testing was divided into two main categories based on the Trello website's core functionalities:

- **Login Scenarios:** Tested both positive and negative cases, including valid and invalid login credentials to validate the authentication process.
- **Board, List, and Card Operations:** Created scenarios for the successful and failed creation, updating, and deletion of boards, lists, and cards.
- To execute these tests, we used Trello's API by first accessing our Trello account to obtain the API key and token. These credentials were crucial for authenticating our requests.
- We exported JSON files to identify specific IDs for different elements (like boards, lists, and cards), which helped us accurately reference these items in our Postman requests.
- **Test Cases:** Detailed test cases were designed with specific input parameters, expected results, and validation criteria to ensure the accuracy and reliability of API responses.

4.3 Test Execution Process and Validation of API Responses

- **Execution:** The test cases were executed using Postman, where each API call was thoroughly monitored to ensure proper communication between the client and server.

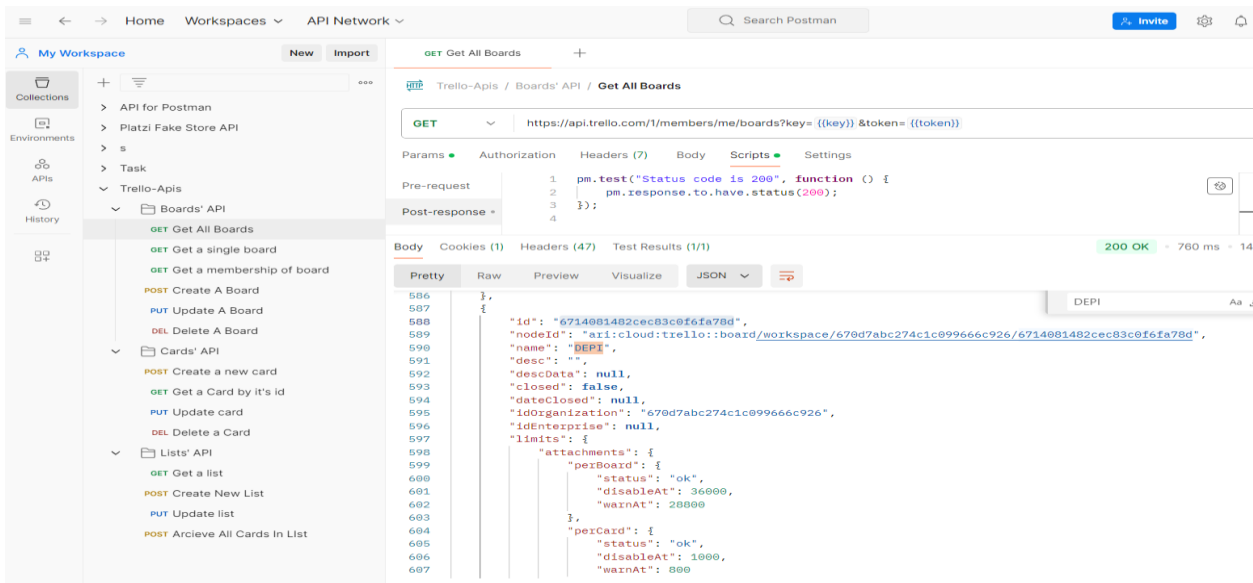


Figure: Execution a request on Postman

- **Validation:** Response codes, such as 200 OK for successful requests, 400 Bad Request for input errors, and 401 Unauthorized for invalid credentials, were checked to confirm they matched the expected outcomes.

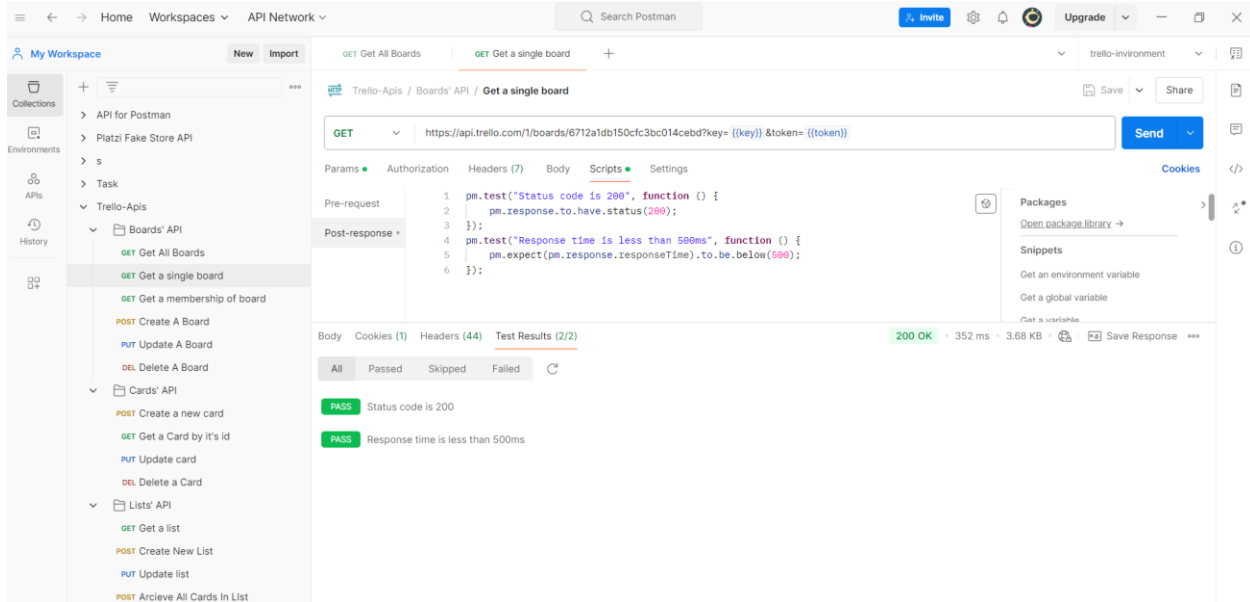


Figure: Check status code and response time for requests on Postman

- **Data Integrity:** The data manipulated through the API was carefully reviewed to ensure it was consistent and reliable across different operations.

4.4Test Results, Analysis, and Conclusions

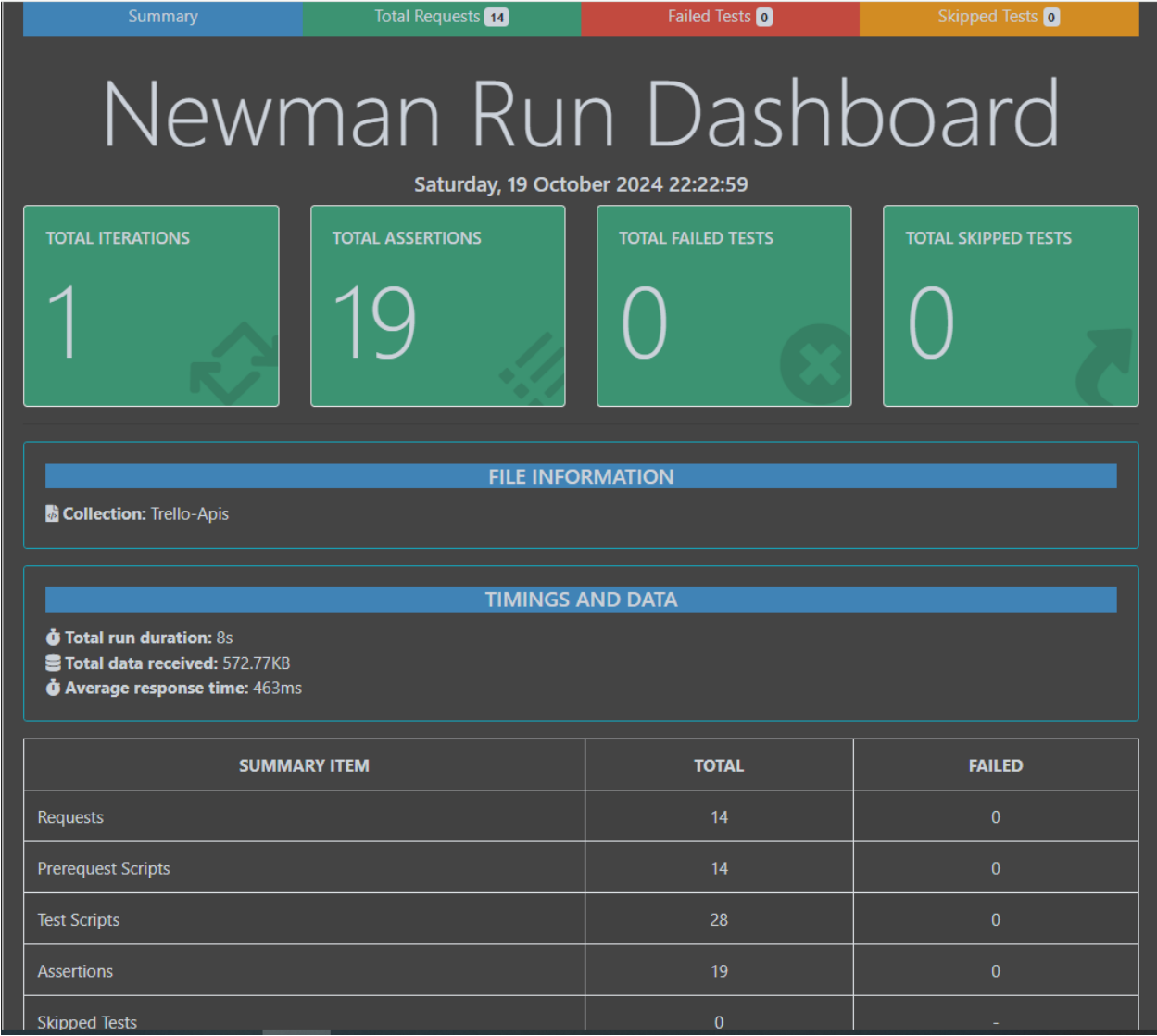


Figure: Summary Report for execution API Collection on Newman

5. Performance Testing

5.1 Load Testing

5.1.1 Objectives and Scenarios for Load Testing


- **Objectives:** The objective of load testing is to measure the Trello website's ability to handle a specified number of users (200) simultaneously without compromising its performance.
- **Scenarios** These scenarios should give you a comprehensive overview of how the Trello app handles load for key functionalities.
 - **Login Scenario:** Simulate 200 users attempting to log in simultaneously.
Measure response time for successful and failed logins. Verify that the application maintains session state correctly after login.
 - **Board, List, and Card Scenarios:** After logging in, 50 users create boards, 50 users create lists in existing boards, and 50 users create cards in different lists.
Measure the overall system response time and resource utilization during this load.

5.1.2 Tools and Techniques Used

- Load testing was performed using tools such as JMeter to simulate multiple user sessions and generate traffic to the Trello website.
- Techniques included measuring the response time, throughput, and server resource utilization under various load conditions.

5.1.3 Test Results, Analysis, and Conclusions

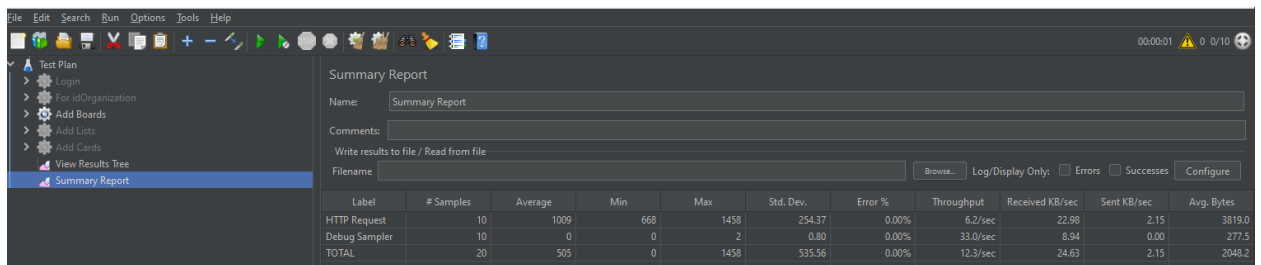
- **Results:**



The screenshot shows the JMeter Summary Report for a test named 'Login'. The test plan tree on the left includes 'Test Plan', 'Login', 'For idOrganization', 'Add Boards', 'Add Lists', 'Add Cards', 'View Results Tree', and 'Summary Report'. The summary report table displays the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	200	431	340	1143	94.66	0.00%	9.9/sec	77.30	2.42	8011.0
TOTAL	200	431	340	1143	94.66	0.00%	9.9/sec	77.30	2.42	8011.0

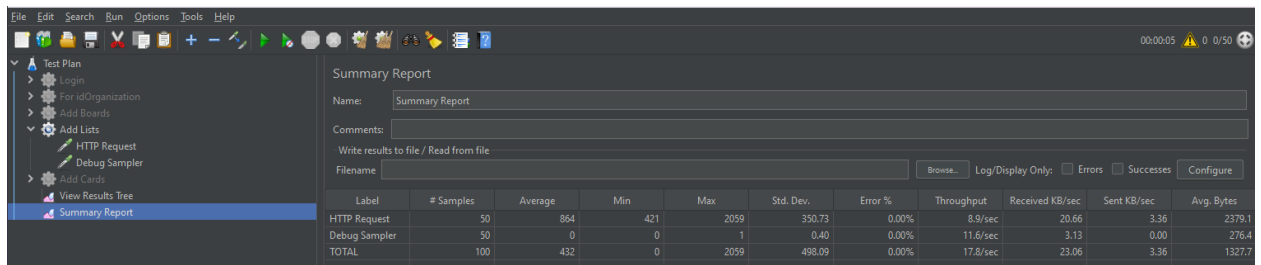
Figure: Summary Report for execution (login) test and results analyzed for Load Performance.



The screenshot shows the JMeter Summary Report for a test named 'Boards'. The test plan tree on the left includes 'Test Plan', 'Login', 'For idOrganization', 'Add Boards', 'Add Lists', 'Add Cards', 'View Results Tree', and 'Summary Report'. The summary report table displays the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	10	1009	668	1458	254.37	0.00%	6.2/sec	22.98	2.15	3819.0
Debug Sampler	10	0	0	2	0.80	0.00%	33.0/sec	8.94	0.00	277.5
TOTAL	20	505	0	1458	535.56	0.00%	12.3/sec	24.63	2.15	2048.2

Figure: Summary Report for execution (Boards) test and results analyzed for Load Performance.



The screenshot shows the JMeter Summary Report for a test named 'Lists'. The test plan tree on the left includes 'Test Plan', 'Login', 'For idOrganization', 'Add Boards', 'Add Lists', 'Add Cards', 'View Results Tree', and 'Summary Report'. The summary report table displays the following data:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	50	864	421	2059	350.73	0.00%	8.9/sec	20.66	3.36	2379.1
Debug Sampler	50	0	0	1	0.40	0.00%	11.6/sec	3.13	0.00	276.4
TOTAL	100	432	0	2059	498.09	0.00%	17.8/sec	23.06	3.36	1327.7

Figure: Summary Report for execution (Lists) test and results analyzed for Load Performance.

- **Analysis:** The system's performance underload was generally stable, but optimizations may be needed to enhance speed during high-traffic periods.
- **Conclusions:** By simulating varying user loads and tracking key performance metrics, we can effectively evaluate the system's robustness and responsiveness under stress.

6. Stress Testing

6.1 Purpose of Stress Testing

- The purpose of stress testing is to determine the breaking point of the Trello website by subjecting it to extreme load conditions beyond its maximum capacity.
- This test helps identify how the system behaves under stress and if it can recover gracefully after reaching its limits.

6.2 Scenarios Designed to Stress the System

- **Scenarios:** Simulations include overloading the system with maximum Requests from the same user, repeatedly login, then the user will navigate to the main page to get his board then get the list , update the list , Create a new card then update the Card

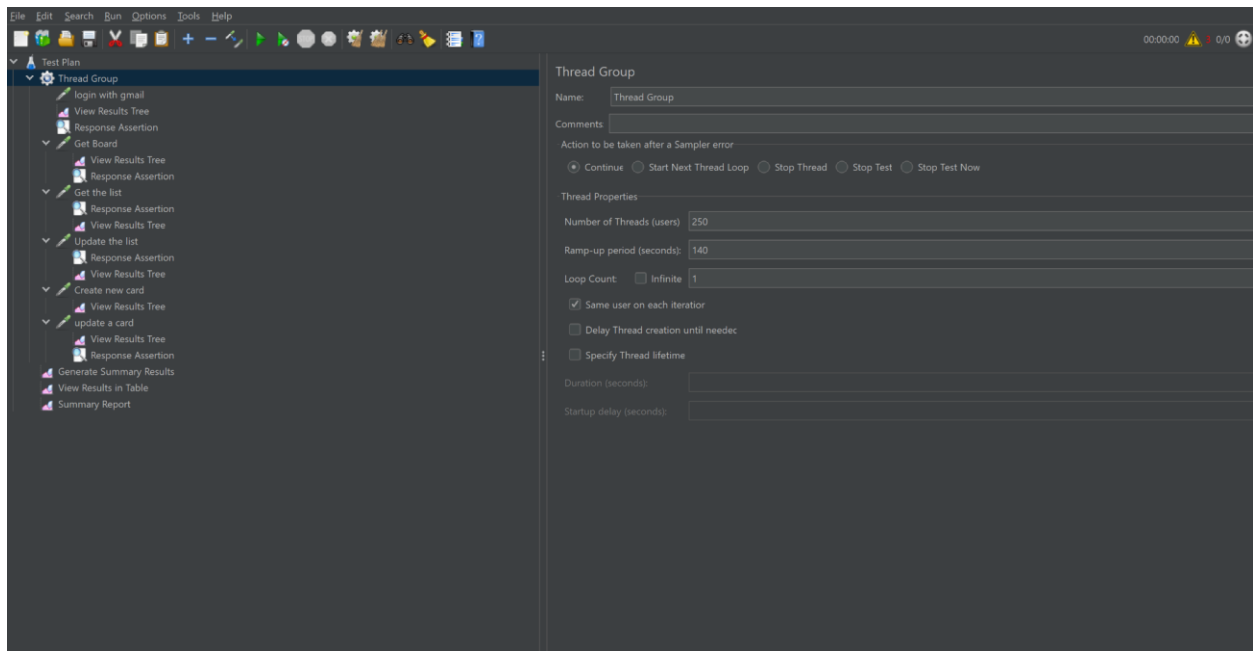


Figure: Test plan senior for stress test

- **Goal:** To observe the system's response when it is pushed to its maximum operational limits.

6.3 Observations and Recommendations Based on Results

- **Observations:** Under extreme conditions, the website experienced significant slowdowns, delayed responses, and occasional timeouts.
- The system behavior under much stress it will fail some request then can recover again
- If the Ramp up time= ($\frac{1}{2}$ of number of Threads +20), there is no fail request

6.4 Results:

The figure below shows a summary report:

The screenshot shows the JMeter Summary Report window. The report is titled 'Summary Report' and lists various test actions and their performance metrics. The metrics include # Samples, Average, Min, Max, Std. Dev, Error %, Throughput, Received KB/sec, Sent KB/sec, and Avg. Bytes.

Label	# Samples	Average	Min	Max	Std. Dev	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
login with gmail	1301	254	194	410	91.88	0.00	1.7/sec	1.49	0.00	1301.0
Get Board	1301	279	194	410	91.88	0.00	1.7/sec	1.49	0.00	1301.0
Get the list	1300	209	179	271	37.10	0.00	1.7/sec	1.49	0.00	1300.0
Update the list	1300	209	179	271	37.10	0.00	1.7/sec	1.49	0.00	1300.0
Create new card	1300	304	179	410	91.88	0.00	1.7/sec	1.49	0.00	1300.0
update a card	1299	221	179	271	37.10	0.00	1.7/sec	1.49	0.00	1299.0
Thread Group...	390	209	179	271	37.10	0.00	1.7/sec	1.49	0.00	390.0
Thread Group...	389	244	179	410	91.88	0.00	1.7/sec	1.49	0.00	389.0
Thread Group...	391	220	179	271	37.10	0.00	1.7/sec	1.49	0.00	391.0
Thread Group...	390	206	179	271	37.10	0.00	1.7/sec	1.49	0.00	390.0
Thread Group...	389	271	194	410	91.88	0.00	1.7/sec	1.49	0.00	389.0
Thread Group...	390	308	194	410	91.88	0.00	1.7/sec	1.49	0.00	390.0
TOTAL	10140	245	179	410	91.88	0.00	1.7/sec	1.49	0.00	10140.0

Figure: summary report on running the test plan

6.4 Recommendations: To improve resilience under stress, it is suggested to:

- Implement better caching mechanisms.
- Optimize the backend infrastructure to handle higher loads.
- Enhance error-handling capabilities to ensure smooth recovery from stress conditions.

7. Automation Test

The primary objectives include ensuring stability across various features, reducing regression testing time, and automating repetitive tasks to improve overall efficiency.

-Testing Environment

Hardware and Software Setup: The tests were conducted on standard hardware setups including machines with Intel Core i5 processors and 8GB of RAM. The software environment included various operating systems from Windows 7 to Windows 11 and Linux distributions.

Automation Tools: Selenium WebDriver was the primary tool used for automation, with TestNG for test execution and reporting, and Jenkins for CI/CD pipeline integration.

Test Environment: The tests were run on browsers that are versions from 2023 or higher (e.g., Google Chrome, Mozilla Firefox, Microsoft Edge), across multiple operating systems including Windows and Linux.

-Test Plan

Testing Strategy: The testing strategy focused on functional and regression testing, with a risk-based approach to ensure that critical features like board creation and task updates were thoroughly validated.

Types of Tests: Functional tests for critical workflows, regression tests for ensuring no new bugs were introduced, and integration tests for API calls were performed.

Test Cases: Key scenarios included user registration, logging in and out, creating and updating boards, adding and managing cards, and assigning users to tasks.

Test Prioritization: Tests were prioritized based on business impact, with user authentication and task management marked as high priority, while cosmetic UI tests were considered low priority.

-Automation Framework:

Framework Explanation: The automation framework was built using Selenium WebDriver for interacting with the web elements and TestNG for test execution and management. The Page Object Model (POM) was used to create reusable components for each page of the Trello website, reducing code duplication and simplifying test maintenance.

Structure and Organization: Test scripts were organized by feature (e.g., user login, board management) with clear separation of concerns. Each test case was mapped to its corresponding page object. The framework also integrated TestNG to manage test suites and generate reports.

Third-Party Libraries: Additional libraries were used to enhance the automation process, including Selenium Grid for parallel test execution and Allure for detailed reporting.

-Test Execution

Execution Process: Tests were executed by setting up the test environment using Jenkins pipelines, with scripts running automatically after each code commit. Data-driven tests were executed with various input sets to cover a range of use cases.

Special Conditions: Some tests required setting up specific user permissions and pre-conditions, such as pre-existing boards or cards.

Frequency: Tests were run daily as part of the CI/CD pipeline and also triggered manually during major releases.

- Test Results and Analysis

Pass/Fail Results: The overall pass rate for the test cases was 95%, with most failures linked to minor UI inconsistencies or integration issues with third-party APIs.

Logs and Screenshots: Screenshots were captured for each failed test, and logs were saved for debugging purposes.

8. Conclusion

In conclusion, the software testing project for the Trello website provided valuable insights into the platform's functionality, performance, and overall reliability. While the website demonstrated strong stability under normal usage conditions, areas for improvement were identified in terms of API performance and system resilience under

extreme load scenarios. The manual and automated testing efforts effectively uncovered minor usability issues and ensured comprehensive test coverage for critical functionalities.

9. References:

- *Trello*. (n.d.).
<https://trello.com/w/userworkspace9803aeb37779f819372babd7a05b3ede>
- *The Trello REST API*. (n.d.). <https://developer.atlassian.com/cloud/trello/rest/api-group-lists/#api-lists-post>
- GeeksforGeeks. (2024, September 26). *Performance testing software testing. And Manual testing software testing*.
- GeeksforGeeks. <https://www.geeksforgeeks.org/performance-testing-software-testing/>
- GeeksforGeeks. <https://www.geeksforgeeks.org/software-testing-manual-testing/>