

Mini-Project 1: E-Commerce Website Functional Testing

Objective: To validate the functional requirements of an e-commerce website's checkout process.

Tasks:

- Review the requirement documentation for the checkout process.
- Write test cases for a complete checkout flow, including adding items to the cart, applying discounts, and processing payment.
- Perform a smoke test on the key features of the checkout process.
- Log test results and report any defects found using a tool like JIRA.
- Tag the reported issues appropriately and assign them to the developer role in JIRA.

1. Review the requirement documentation for the checkout process of Snapdeal e-commerce website.

Overview

The checkout process on the Snapdeal e-commerce website involves several steps that users must follow to complete their purchase. This document outlines the requirements for the checkout process, including adding items to the cart, applying discounts, selecting a shipping address, entering payment information, and confirming the order.

Functional Requirements

- 1. Adding Items to Cart:** Users should be able to add items to their shopping cart from the product detail page.
- 2. Viewing Cart:** Users should be able to view items in their cart.
- 3. Applying Discounts:** Users should be able to apply discount codes to their order.
- 4. Proceeding to Checkout:** Users should be able to proceed to the checkout from the cart page.
- 5. Selecting Shipping Address:** Users should be able to select or add a shipping address during checkout.
- 6. Entering Payment Information:** Users should be able to enter payment information.
- 7. Reviewing Order:** Users should be able to review their order before placing it.
- 8. Placing Order:** Users should be able to place their order.
- 9. Order Confirmation:** Users should receive an order confirmation.
- 10. Viewing Order History:** Users should be able to view their order history.

Acceptance Criteria:

The UI should follow standard design principles and be user-friendly.

Users should be able to complete the checkout process with minimal clicks and inputs.

Assumptions and Dependencies

The user is already registered and logged into the Snapdeal website.

The product catalog and inventory systems are functioning correctly.

The payment gateway is integrated and operational.

Glossary

Cart: A virtual container that holds items the user intends to purchase.

Checkout: The process of finalizing a purchase, including entering shipping and payment information.

Discount Code: A code that provides a discount on the total purchase amount.

COD: Cash on Delivery, a payment method where the user pays for the item upon delivery.

PCI-DSS: Payment Card Industry Data Security Standard, a set of security standards designed to protect card information during and after a financial transaction.

2. Write test cases for a complete checkout flow, including adding items to the cart, applying discounts, and processing payment.

Test Case ID	Test Case	Test Case Description	Pre-Requisite	Test Data	Test Steps	Expected Result	Actual Result	PASS/FAIL	Defect ID
#Snapdeal_TC001	Add Item to Cart	Verify that a user can add item to the cart	User is logged in and on a product page	Item ID:12345	1.Click the "Add to Cart" button on the product page 2.Go to the cart page	The item should be added to the cart and visible on the cart page.	As Expected	Pass	
#Snapdeal_TC002	Apply Valid Discount Code	Verify that a valid discount code is applied Successfully	User has an item in the cart and a valid discount code	Discount Code:CUHGSY32799	1.Go to the cart page 2.Enter the valid discount code in the "Discount Code" field 3.Click the "Apply" button	The discount should be applied, and the total amount should be reduced accordingly.	As Expected	Pass	
#Snapdeal_TC003	Apply Invalid Discount Code	Verify that an invalid discount code is not applied	User has an item in the cart and an invalid discount code	Discount Code:jqwj89	1.Go to the cart page 2.Enter the invalid discount code in the "Discount Code" field 3.Click the "Apply" button	The system should display an error message indicating the discount code is invalid, and the total amount should remain unchanged	As Expected	Pass	
#Snapdeal_TC004	Proceed to Checkout	Verify that the user can proceed to the checkout page	User has an item in the cart		1.Go to the cart page 2.Click the "Proceed to Checkout" button	The user should be redirected to the checkout page	As Expected	Pass	
#Snapdeal_TC005	Select Shipping Address	Verify that the user can select a shipping address	User is on the checkout page and has multiple saved addresses	User has a saved address	1.On the checkout page, select a shipping address from the saved addresses list	The selected address should be highlighted, and the shipping details should update accordingly	As Expected	Pass	
#Snapdeal_TC006	Enter Payment Information (Credit Card)	Verify that the user can enter credit card information	User is on the payment section of the checkout page	Card Number: 4111111111111111, Expiry: 12/25, CVV: 123	1.Select "Credit Card" as the payment method 2.Enter valid credit card details (card number, expiration date, CVV) 3.Click the "Pay Now" button	The payment should be processed successfully, and the user should be redirected to the order confirmation page	As Expected	Pass	
#Snapdeal_TC007	Enter Invalid Payment Information (Credit Card)	Verify that the system handles invalid credit card information	User is on the payment section of the checkout page	Card Number: 1234567890123456, Expiry: 12/20, CVV: 999	1.Select "Credit Card" as the payment method 2.Enter invalid credit card details (e.g., incorrect card number) 3.Click the "Pay Now" button	The system should display an error message indicating the payment information is invalid	As NotExpected	Fail	#Def_001
#Snapdeal_TC008	Place Order with COD	Verify that the user can place an order using Cash on Delivery (COD)	User is on the payment section of the checkout page	Payment Method: COD	1.Select "Cash on Delivery" as the payment method 2.Click the "Place Order" button	The order should be placed successfully, and the user should be redirected to the order confirmation page	As Expected	Pass	
#Snapdeal_TC009	Order Confirmation Page	Verify that the order confirmation page displays correct order details	User has successfully placed an order	Order ID: 78901	1.Complete the checkout process 2.Check the order confirmation page	The order confirmation page should display the correct order details (order number, item details, shipping address, payment method)	As Expected	Pass	
#Snapdeal_TC010	Verify Order in Order History	Verify that the order appears in the user's order history	User has successfully placed an order	User Account: testuser@gmail.com	1.Go to the user profile 2.Navigate to the order history section	The newly placed order should appear in the order history with the correct details	As Expected	Pass	
#Snapdeal_TC011	Add Multiple Items to cart	Verify adding multiple items to the cart	User should be logged in	Item IDs: 12345, 67890,85632	1.Navigate to the Snapdeal website 2.Add multiple items to the cart 3.Verify each item is listed in the cart	All items are successfully added to the cart	As Expected	Pass	
#Snapdeal_TC012	Update cart	Update item quantity in the cart	Item in cart	Item ID: 12345, Quantity: 2	1.Navigate to the cart 2.Change the quantity of an item 3.Verify the cart updates with the new quantity	The cart is updated with the correct item quantity	As Expected	Pass	
#Snapdeal_TC013	Apply Discount	Apply expired discount code	Item in cart	Discount Code: EXPIRED50	1.Navigate to the cart 2.Enter an expired discount code	The system displays an error message indicating the discount code is expired	As NotExpected	Fail	#Def_002
#Snapdeal_TC014	Empty Cart	Verify behavior when placing an order with an empty cart	No Item in Cart		1.Attempt to proceed to checkout with an empty cart	The system displays a message indicating that the cart is empty and the user cannot proceed	As NotExpected	Fail	#Def_003

3.Perform a smoke test on the key features of the checkout process.

Performing a smoke test on the key features of the checkout process on Snapdeal involves verifying that the essential functions work as expected. Here's a guide on how to create and execute a smoke test using Jira, particularly with Zephyr Squad for test management.

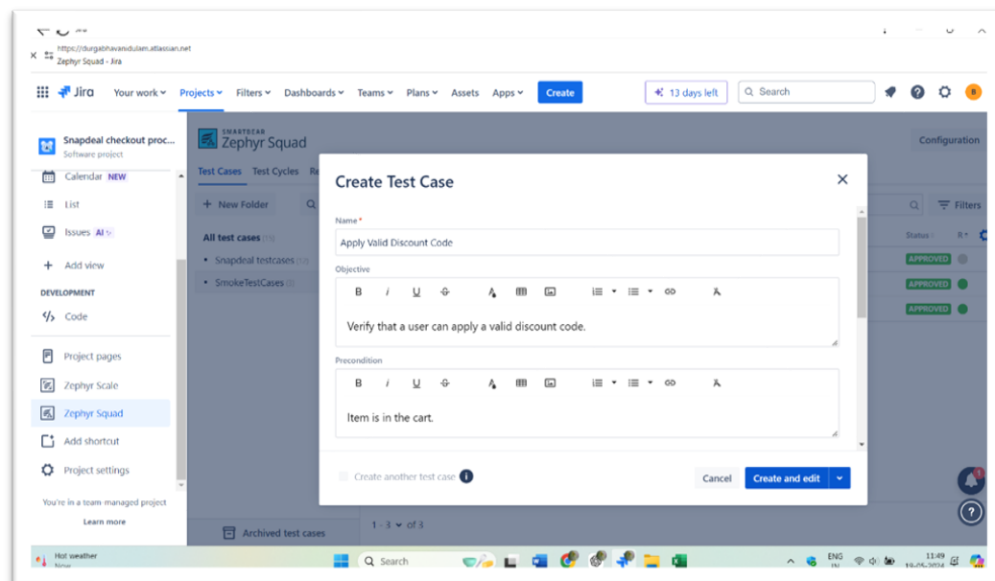
Step 1: Define the Smoke Test Cases

Smoke test cases for the checkout process should cover the core functionalities such as adding items to the cart, applying discount codes, proceeding to checkout, entering shipping details, entering payment information, and placing the order.

Step 2: Create and Manage Smoke Test Cases in Jira

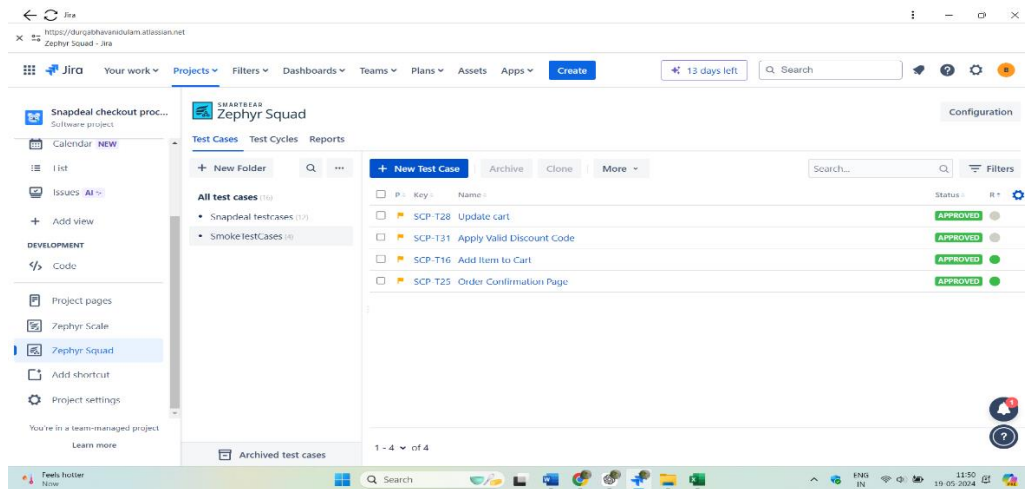
1.Create Test Cases:

- Navigate to the project in Jira.
- Go to the “Tests” tab (Zephyr Squad).
- Click on “Create Test” to add a new test case.
- Fill in the details for each test case (ID, Summary, Pre-condition, Test Steps, Expected Result).



2.Organize Test Cases:

- Group all smoke test cases under a folder or label them appropriately (e.g., "Smoke Test").
- This helps in easily identifying and executing smoke tests.



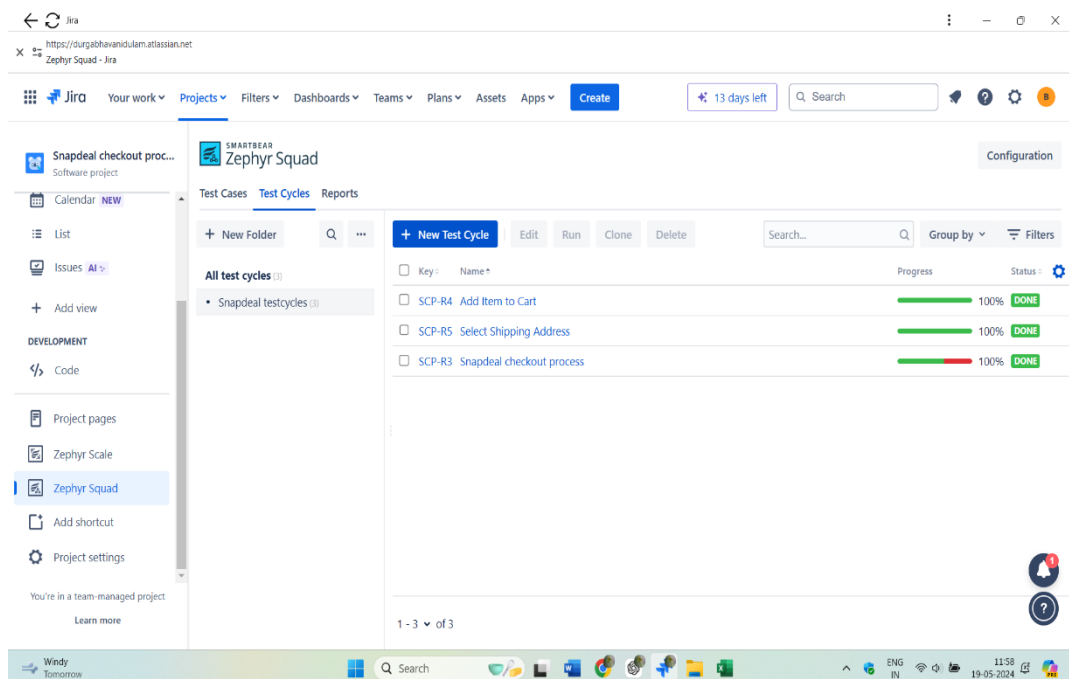
Step 3: Create a Test Cycle for Smoke Testing

1. Create a New Test Cycle:

- Go to “Test Cycles” in Zephyr Squad.
- Click on the “Create Cycle” button.
- Name the cycle (e.g., “Smoke Test - Checkout Process”).
- Set the build version, environment, and start/end dates if necessary.

2. Add Test Cases to the Cycle:

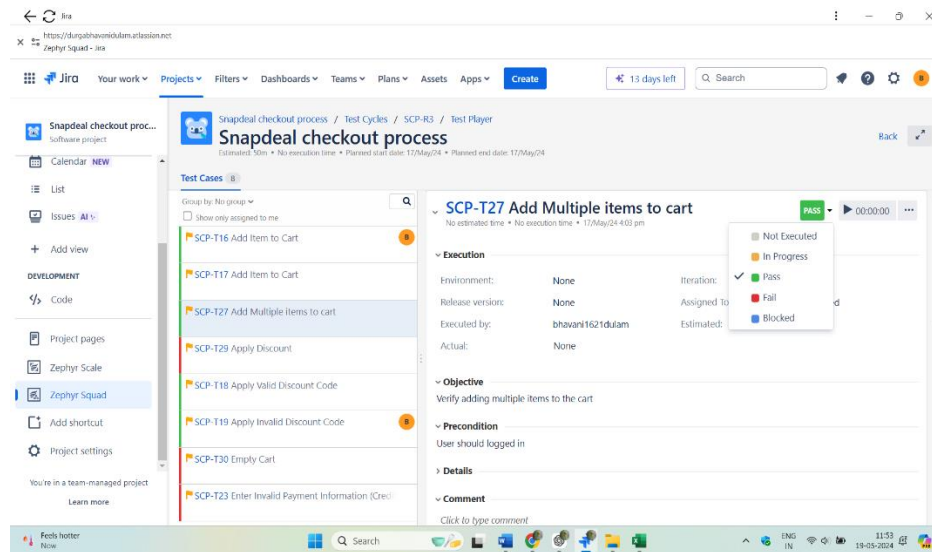
- Open the created test cycle.
- Click on the “Add Tests” button.
- Search for the smoke test cases by their IDs or summaries.
- Select the test cases and click “Add” to include them in the test cycle.



Step 4: Execute the Smoke Test Cases

1. Execute Test Cases:

- Navigate to the test cycle.
- Click on a test case to open it.
- Click on the “Execute” button.
- Select the execution status (Pass/Fail) based on the test outcome.
- Provide detailed comments or attach evidence if necessary.
- Save the execution results.



4. Log test results and report any defects found using a tool like JIRA.

Logging test results and reporting defects in Jira using Zephyr Squad involves systematically documenting the outcomes of test executions and creating detailed bug reports for any issues discovered. Here's a step-by-step guide on how to log test results and report defects using Jira:

Step 1: Execute Test Cases

- Navigate to Test Cycles
- Open the Test Cycle
- Execute a Test Case

Step 2: Report Defects

1. Identify a Defect:

- During test execution, if a test case fails or an unexpected behavior is observed, note down the issue.

2. Create a New Bug in Jira:

- Navigate to the “Create” button in the top navigation bar of Jira.
- Select “Bug” as the issue type.

3.Fill in Bug Details:

- **Summary:** Provide a concise title for the bug.
- **Description:** Detail the steps to reproduce the bug, expected results, and actual results.
- **Components:** Assign relevant components (e.g., Payment Gateway, Checkout Process).
- **Labels:** Add relevant labels for easy filtering (e.g., checkout, payment, high-priority).
- **Priority:** Set the priority level (e.g., High, Medium, Low).
- **Assignee:** Assign the bug to a developer or leave it for triage.

4.Link the Bug to the Test Case:

- In the bug creation screen, find the “Linked Issues” section.
- Link the bug to the test case where the defect was found.

5.Attach Evidence:

- Attach any relevant screenshots, logs, or files that can help the developer understand the issue.

Step 3: Review and Track Progress

1.Monitor Bug Status:

- Use Jira dashboards to monitor the status of reported bugs.
- Follow up with assigned developers if necessary to ensure timely resolution.

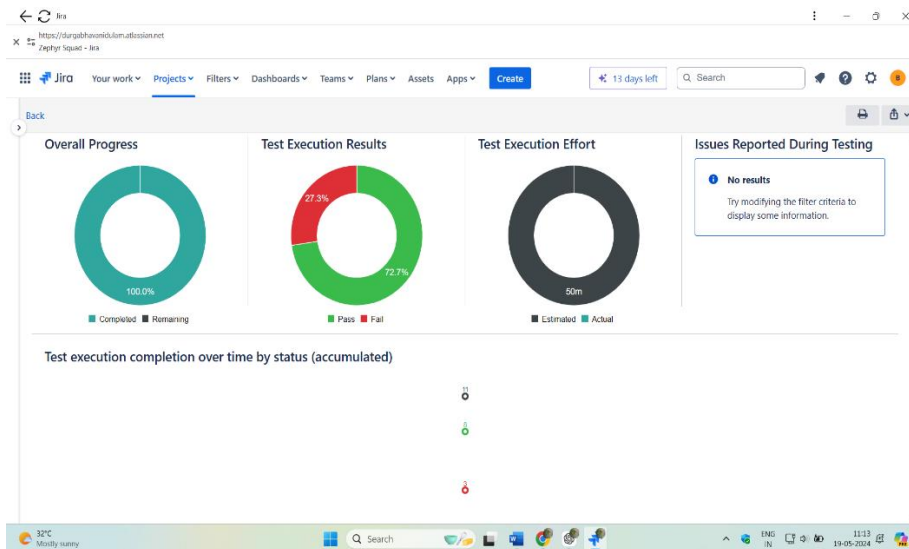
2.Update Test Cases:

- Once a bug is fixed, update the related test case to include regression tests to verify the fix.
- Re-execute the test cases to ensure the defects are resolved.

Step 4: Reporting

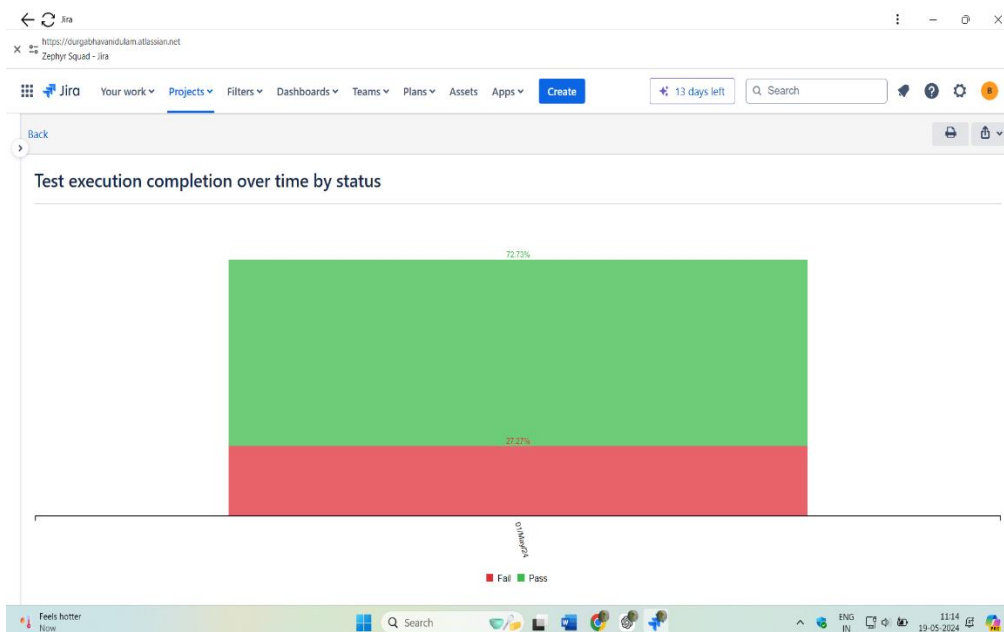
1.Generate Reports:

- Use Zephyr Squad’s reporting features to generate execution and defect summary reports.
- Track key metrics like the number of tests executed, passed, failed, and the status of reported bugs.



2.Share Reports with Stakeholders:

- Share the reports with project stakeholders to keep them informed about the testing progress and any critical issues.



By following these steps, you can systematically log test results and report defects in Jira using Zephyr Squad, ensuring a thorough and organized approach to testing the Snapdeal checkout process.

5.Tag the reported issues appropriately and assign them to the developer role in JIRA.

To effectively manage and track reported issues in Jira, tagging and assigning them appropriately is crucial. Here's a step-by-step guide on how to tag reported issues and assign them to developers:

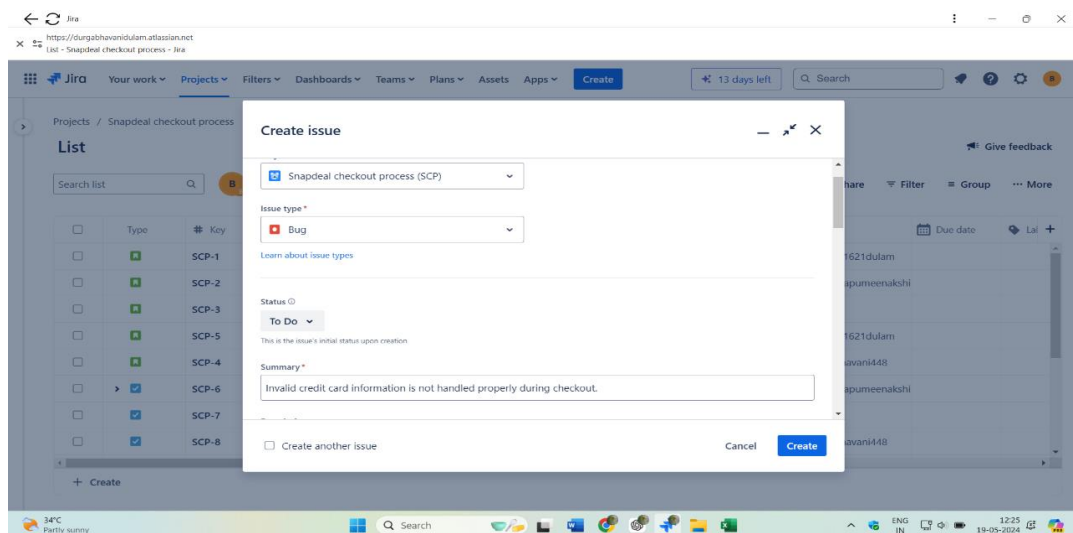
Step 1: Create and Tag Issues

1.Log in to Jira:

- Navigate to your Jira instance and log in with your credentials.

2.Create a New Issue:

- Click on the “Create” button located in the top navigation bar.
- Fill in the required fields such as Project, Issue Type (e.g., Bug), Summary, and Description.



3.Tag the Issue:

- **Labels:** Use the “Labels” field to add relevant tags. Labels help in categorizing and filtering issues. For example, you might use labels like checkout, high-priority, payment, etc.
- **Components:** If your project uses components (subsections of the project), assign the appropriate component to the issue. This helps in further categorizing and routing the issue.
- **Priority:** Set the priority of the issue (e.g., High, Medium, Low) to indicate its urgency.

4. Assign the Issue:

- **Assignee:** In the “Assignee” field, select the developer responsible for addressing the issue. If you’re not sure who to assign it to, you can assign it to the project lead or leave it unassigned for triage.

5. Save the Issue:

- Click on the “Create” button to save the issue.

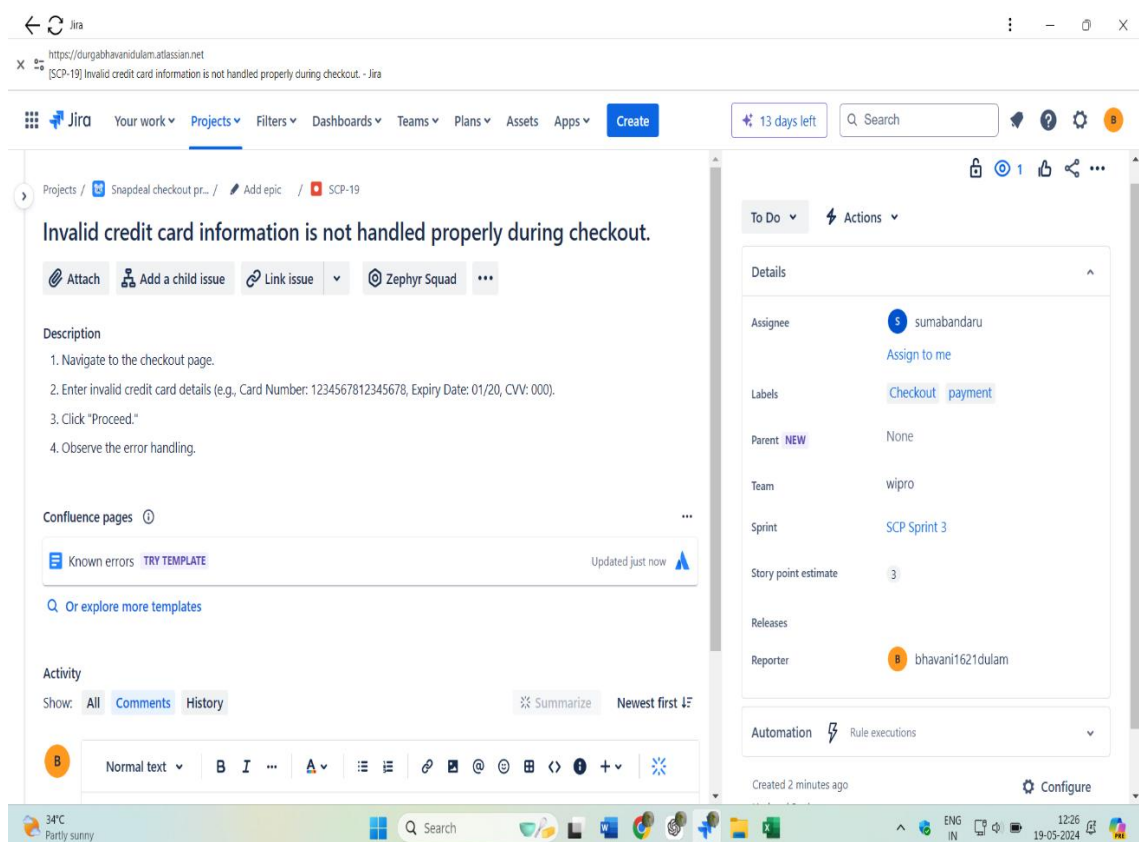
Step 2: Assigning Issues to Developers

1. Navigate to the Issue:

- Find the issue you want to assign. You can search for it using the issue key or summary in the search bar.

2. Assign the Issue:

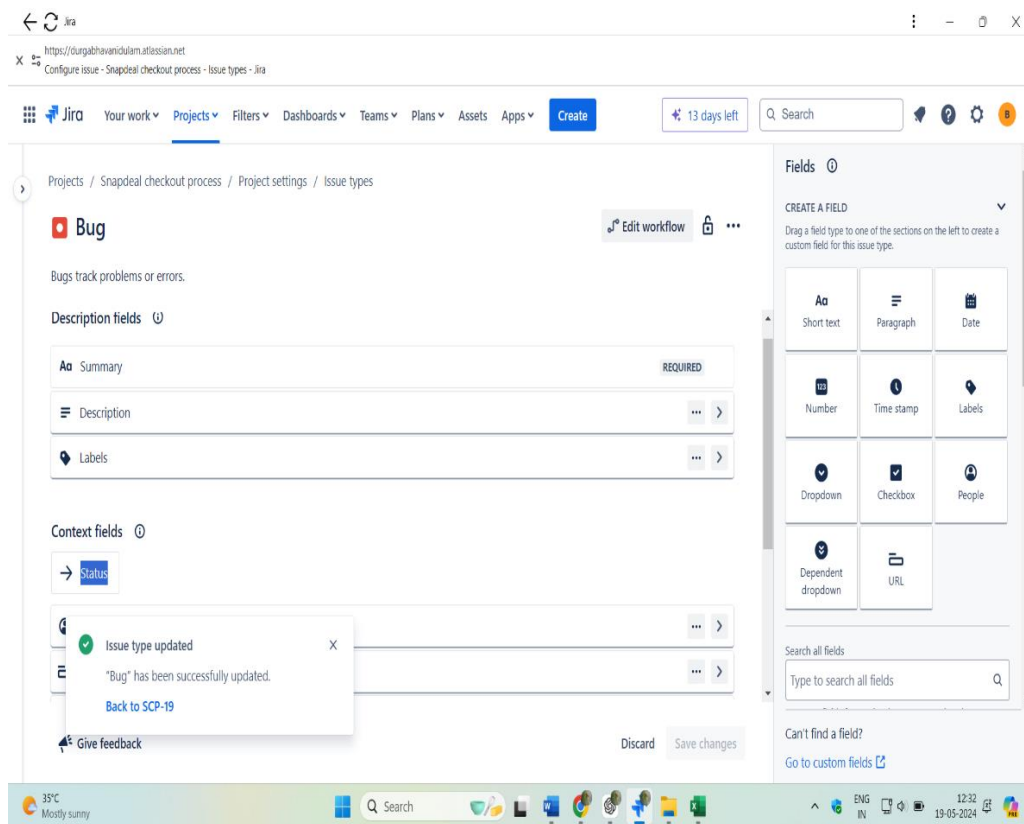
- Open the issue.
- In the “Assignee” field, type the name of the developer you want to assign the issue to. Select their name from the dropdown list.
- Click on the “Assign” button to update the issue.



Step 3: Managing and Tagging Existing Issues

1. Bulk Edit Issues (if needed):

- Navigate to the “Issues” section.
- Use filters to find the issues you want to tag or assign. You can use JQL (Jira Query Language) to filter issues based on various criteria.
- Select the issues you want to edit and click on the “More” button.
- Choose “Bulk Change” and follow the steps to update the labels, components, or assignee for multiple issues at once.



Mini-Project 2: Performance Testing on a Landing Page

Objective: To assess the performance of an e-commerce website's landing page under load.

Tasks:

- Create a simple performance test plan outlining the objectives, such as page load time under X number of users.
- Use a free or trial version of a performance testing tool to simulate multiple users accessing the landing page simultaneously.
- Record the performance data and analyze it against the performance objectives.
- Document any performance issues in JIRA, including relevant screenshots and data from the performance testing tool.

1. Performance Test Plan for Snapdeal E-commerce Website

1. Introduction

This document outlines the performance test plan for the Snapdeal e-commerce website. The primary objective is to assess the website's performance, specifically focusing on page load time under a simulated load of 500 concurrent users.

2. Objectives

- **Measure Page Load Time:** Ensure that the homepage and key product pages load within an acceptable time frame under a load of 500 concurrent users.
- **Identify Bottlenecks:** Determine any performance bottlenecks that occur under the specified load.
- **Scalability Assessment:** Evaluate how well the website scales when subjected to increasing user loads.
- **User Experience Assurance:** Ensure that the user experience remains optimal under peak load conditions.

3. Scope of Testing

I. Pages to be Tested:

- Homepage
- Product Category Pages
- Product Detail Pages
- Search Results Page
- Checkout Page

II. Key Metrics:

- Page Load Time
- Time to First Byte (TTFB)

- Response Time
- Error Rate
- Throughput (requests per second)

4. Test Environment

I. Hardware:

- Load testing will be conducted using a cloud-based load testing tool (e.g., JMeter, LoadRunner, or a SaaS solution like BlazeMeter).
- Test servers will mirror the production environment in terms of configuration and capacity.

II. Software:

- Web servers (Apache, Nginx, etc.)
- Application servers
- Database servers

III. Network:

- Bandwidth and latency similar to real-world user conditions.

5. Test Data

I. User Profiles:

- Simulate a diverse range of users, including both logged-in and guest users.
- Include different browsing behaviors (e.g., casual browsing, search, adding to cart, checking out).

II. Data Load:

- Use realistic data sets to mimic actual user interactions and product database entries.

6. Test Scenarios

- Scenario 1: Homepage Load Test
- Scenario 2: Product Category Page Test
- Scenario 3: Product Detail Page Test
- Scenario 4: Search Functionality Test
- Scenario 5: Checkout Process Test

7. Performance Acceptance Criteria

- **Page Load Time:** All key pages should load within 3 seconds under the specified load.
- **Error Rate:** Less than 1% errors across all requests.
- **Response Time:** 95th percentile response time should be under 2 seconds.
- **Throughput:** System should handle at least 100 requests per second consistently.

8. Test Execution

I. Pre-test Checklist:

- Ensure the test environment is isolated and does not affect production.
- Verify that monitoring tools are in place and configured correctly.
- Confirm that test data is prepared and uploaded.

II. Execution Steps:

- Run each scenario individually, gradually ramping up the load to 500 users.
- Monitor and record the performance metrics.
- Identify any performance issues and bottlenecks.

9. Monitoring and Reporting

I. Tools:

- Use monitoring tools like New Relic, Datadog, or similar to capture detailed performance metrics.
- Collect logs and error reports for analysis.

II. Reporting:

- Generate a detailed report summarizing the test results.
- Include recommendations for performance improvements based on the findings.

10. Post-Test Activities

I. Analysis:

- Analyze test results to identify trends, bottlenecks, and areas for improvement.
- Correlate performance issues with specific parts of the system.

II. Recommendations:

- Provide actionable recommendations for optimizing performance.
- Plan for iterative testing to validate improvements.

III. Retesting:

- Schedule follow-up tests to verify that performance issues have been resolved and improvements are effective.

Conclusion

This performance test plan outlines a structured approach to evaluate the Snapdeal e-commerce website's performance under a load of 500 concurrent users. By following this plan, we aim to ensure that the website maintains optimal performance and provides a seamless user experience even during peak usage.

2. Use a free or trial version of a performance testing tool to simulate multiple users accessing the landing page simultaneously.

Using JMeter to simulate multiple users accessing the Snapdeal landing page involves several steps.

Step-by-Step Guide to Simulate Multiple Users Using JMeter

1. Install JMeter

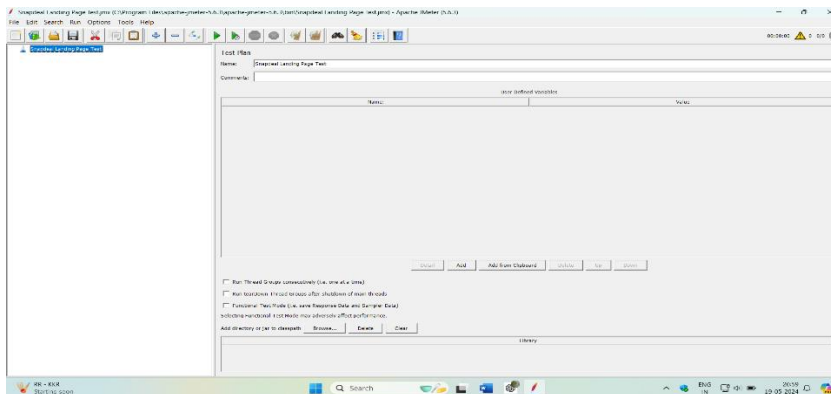
- Download and install JMeter from the Apache JMeter website.
- Extract the downloaded zip/tar file to a suitable location on your machine.

2. Open JMeter

- Navigate to the bin directory of the JMeter installation.
- Launch JMeter by running the `jmeter.bat` (Windows) or `jmeter.sh` (Linux/Mac) file.

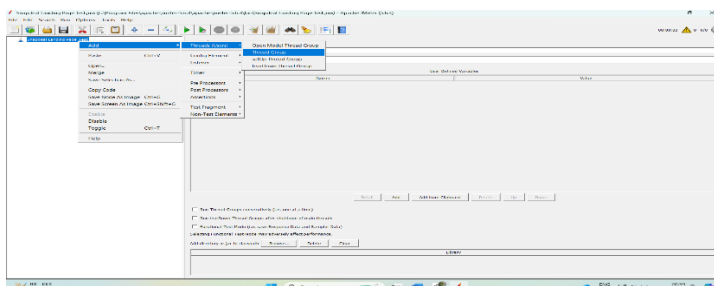
3. Create a Test Plan

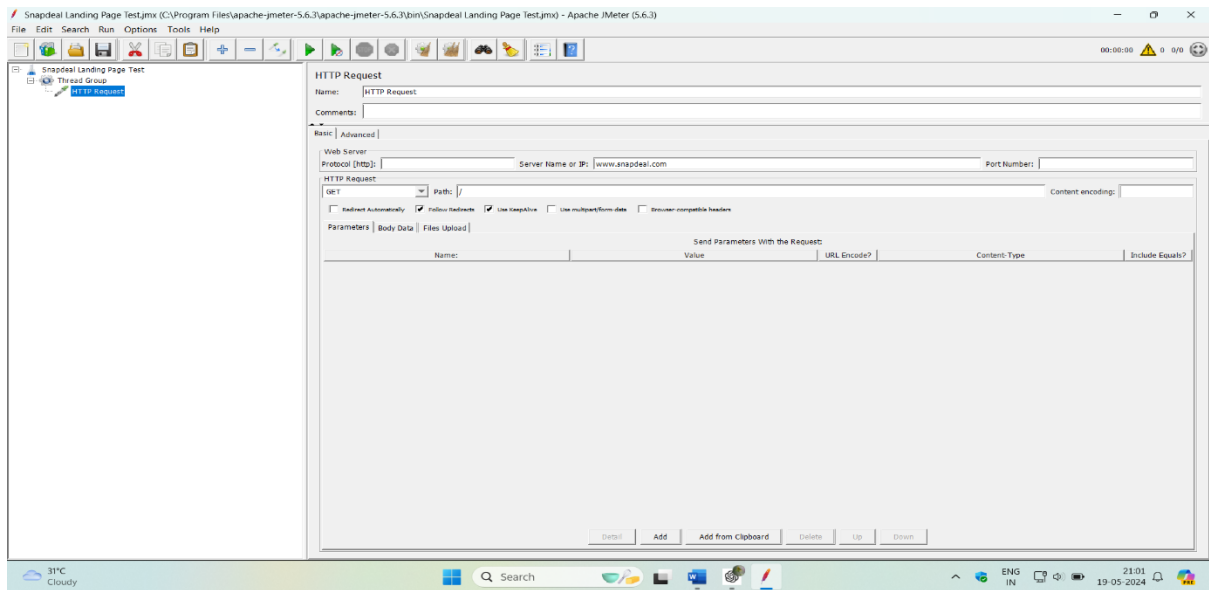
- In JMeter, create a new Test Plan by clicking on File > New.
- Rename the Test Plan to something meaningful, e.g., "Snapdeal Landing Page Test".



4. Add Thread Group

- Right-click on the Test Plan, then add a Thread Group: Add > Threads (Users) > Thread Group.
- **Configure the Thread Group:**
- **Number of Threads (Users):** 500 (simulates 500 concurrent users).
- **Ramp-Up Period (in seconds):** 100 (this means it will take 100 seconds to start all 500 users, which adds 5 users per second).
- **Loop Count:** 1 (each user will perform the test once).





6. Add Listener to View Results

- Right-click on the Thread Group, then add a Listener to view results: Add > Listener > View Results in Table.
- Optionally, add other listeners such as View Results Tree, Summary Report, or Aggregate Report for more detailed analysis.

7. Configure and Run the Test

- Save your test plan by clicking File > Save As.
- Click the green Start button to begin the test.

8. Analyze Results

- Once the test is completed, view the results in the configured listeners.
- Analyze the data to understand the performance metrics such as response times, throughput, error rates, etc.

Conclusion

By following these steps, you can use JMeter to simulate 500 users accessing the Snapdeal landing page simultaneously. This will help you evaluate the performance of the landing page under load and identify any potential bottlenecks. Make sure to monitor the server's performance during the test to gather comprehensive data for analysis.

3. Record the performance data and analyze it against the performance objectives.

To record and analyze the performance data against the performance objectives, you would typically use JMeter's built-in listeners and possibly additional tools for more in-depth analysis. Here's how you can do it:

1. Recording Performance Data:

I. Run the Test:

Start the test in JMeter by clicking the green "Start" button.

II. Monitor Test Progress:

During the test execution, JMeter will display real-time data in the configured listeners (e.g., View Results in Table, Summary Report).

III. Collect Performance Metrics:

JMeter captures various performance metrics such as response times, throughput, error rates, etc., for each HTTP request.

IV. Save Test Results:

After the test completes, save the test results in a file format (e.g., .jtl) for further analysis.

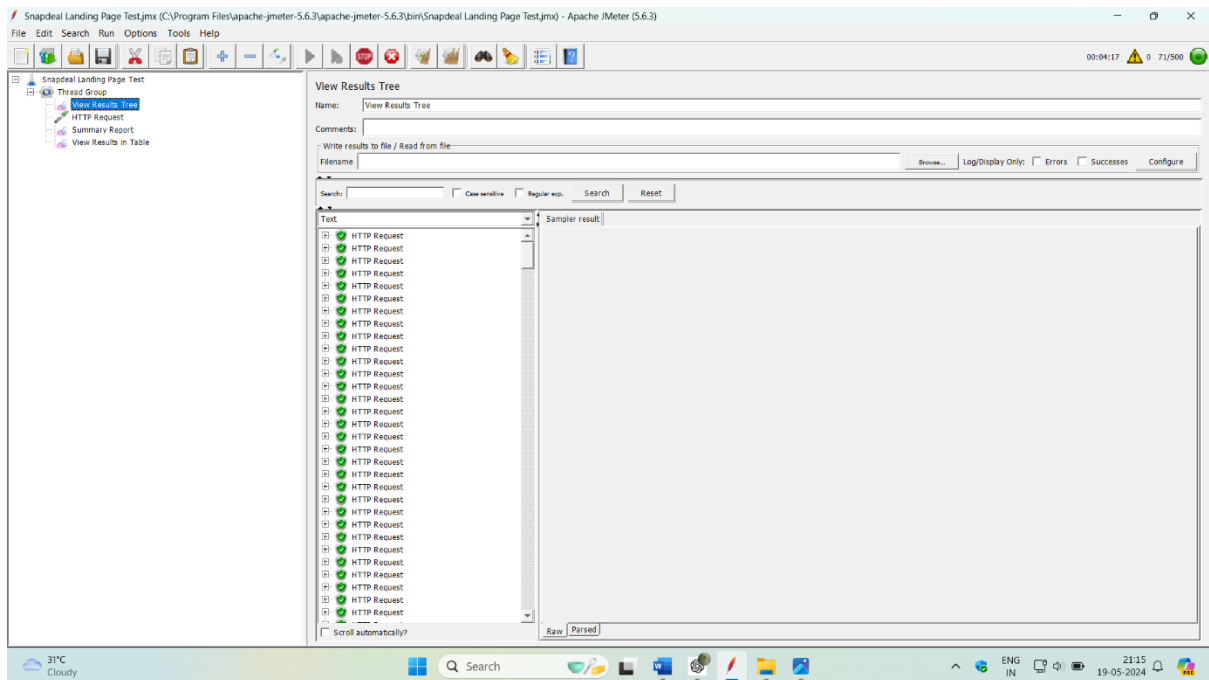
2. Analyzing Performance Data:

I. Open Test Results:

Open the saved test results file (.jtl) in JMeter or any compatible tool.

II. Review Summary Reports:

Use JMeter's Summary Report listener to get an overview of key performance metrics such as average response time, throughput, error rate, etc.



III. Check Response Times:

Analyze individual HTTP request/response times to identify any outliers or slow-performing requests.

IV. Assess Throughput:

Check the throughput to see how many requests per second the system handled during the test.

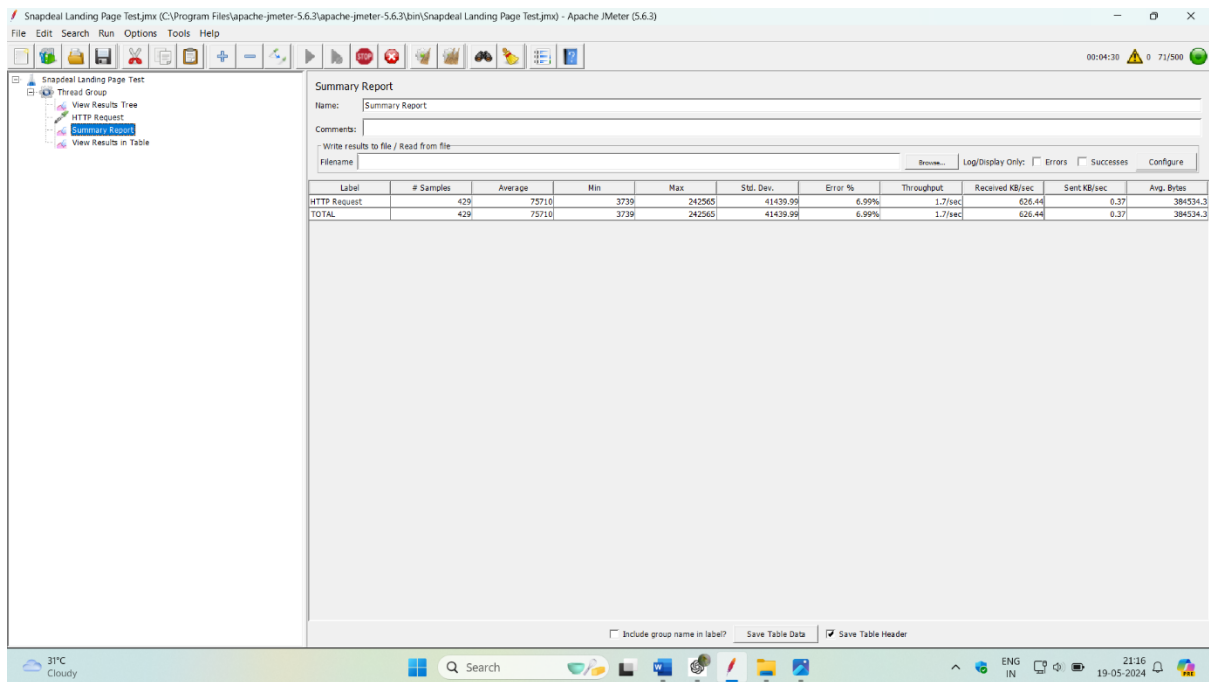
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	21:11:33.020	Thread Group 1-1	HTTP Request	3739	✓	413196	238	409	230
2	21:11:33.235	Thread Group 1-2	HTTP Request	4057	✓	413195	238	226	51
3	21:11:35.821	Thread Group 1-15	HTTP Request	7675	✓	413203	238	5476	394
4	21:11:35.627	Thread Group 1-4	HTTP Request	11636	✓	413203	238	400	70
5	21:11:43.019	Thread Group 1-51	HTTP Request	14231	✓	413204	238	709	606
6	21:11:43.222	Thread Group 1-52	HTTP Request	19266	✓	413204	238	511	446
7	21:11:36.620	Thread Group 1-19	HTTP Request	25977	✓	413204	238	7231	195
8	21:11:37.421	Thread Group 1-23	HTTP Request	25471	✓	413204	238	6445	239
9	21:11:38.219	Thread Group 1-27	HTTP Request	24709	✓	413204	238	5641	224
10	21:11:37.821	Thread Group 1-25	HTTP Request	25364	✓	413203	238	5849	144
11	21:11:42.831	Thread Group 1-50	HTTP Request	20483	✓	413204	238	903	804
12	21:11:38.021	Thread Group 1-26	HTTP Request	25366	✓	413204	238	5918	235
13	21:11:41.236	Thread Group 1-42	HTTP Request	23518	✓	413203	238	2400	115
14	21:11:40.026	Thread Group 1-36	HTTP Request	25485	✓	413203	238	3847	529
15	21:11:33.832	Thread Group 1-5	HTTP Request	33070	✓	413204	238	325	104
16	21:11:33.423	Thread Group 1-3	HTTP Request	34200	✓	413204	238	211	44
17	21:11:42.627	Thread Group 1-49	HTTP Request	25477	✓	413204	238	1182	1106
18	21:11:36.420	Thread Group 1-18	HTTP Request	31843	✓	413204	238	656	311
19	21:11:34.421	Thread Group 1-8	HTTP Request	34149	✓	413203	238	9216	699
20	21:11:34.022	Thread Group 1-6	HTTP Request	34886	✓	413203	238	1205	63
21	21:11:40.621	Thread Group 1-39	HTTP Request	28684	✓	413204	238	4567	298
22	21:11:35.321	Thread Group 1-13	HTTP Request	36331	✓	413204	238	8547	713
23	21:11:43.421	Thread Group 1-53	HTTP Request	28204	✓	413204	238	388	287
24	21:11:35.021	Thread Group 1-11	HTTP Request	39049	✓	413203	238	8616	654
25	21:11:34.221	Thread Group 1-7	HTTP Request	40161	✓	413204	238	914	126
26	21:11:50.421	Thread Group 1-88	HTTP Request	24449	✓	413204	238	4202	2080
27	21:11:37.220	Thread Group 1-22	HTTP Request	38567	✓	413204	238	4134	76
28	21:11:37.621	Thread Group 1-24	HTTP Request	38627	✓	413204	238	7114	129
29	21:11:39.020	Thread Group 1-31	HTTP Request	37320	✓	413204	238	6477	470
30	21:11:45.220	Thread Group 1-62	HTTP Request	33147	✓	413204	238	2558	1907
31	21:11:42.422	Thread Group 1-48	HTTP Request	36404	✓	413204	238	1345	1279
32	21:11:44.021	Thread Group 1-56	HTTP Request	34878	✓	413204	238	1589	714
33	21:11:35.421	Thread Group 1-13	HTTP Request	43527	✓	413204	238	15253	546

V. Evaluate Error Rate:

Examine the error rate to see if there were any failed requests or unexpected errors during the test.

VI. Compare Against Performance Objectives:

Compare the recorded performance metrics against the predefined performance objectives outlined in the test plan.



The screenshot shows the Apache JMeter 5.6.3 Summary Report for a test named 'Snapdeal Landing Page Test'. The report is displayed in a web browser window. The left sidebar shows the test structure with 'Thread Group' and 'HTTP Request' under 'Summary Report'. The main area displays a table of performance metrics. The table has columns for Label, # Samples, Average, Min, Max, Std. Dev., Error %, Throughput, Received KB/sec, Sent KB/sec, and Avg. Bites. The data shows 429 samples for the HTTP Request, with an average response time of 75710 ms and an error rate of 6.99%.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bites
HTTP Request	429	75710	3739	242565	41439.99	6.99%	1.7/sec	626.44	0.37	384534.3
TOTAL	429	75710	3739	242565	41439.99	6.99%	1.7/sec	626.44	0.37	384534.3

Conclusion:

By recording and analyzing the performance data against the predefined objectives, you can determine whether the Snapdeal landing page meets the required performance standards under a load of 500 concurrent users. If the objectives are met, the performance is considered acceptable. If not, further optimization and tuning may be necessary to improve the performance of the website.