Software Testing Graduation Project

OpenCart, GoREST, Demoblaze testing

Mohamed Abdelhaq
Ghada Amr
Hesham Ahmed
Menna Ibrahim
Randa Mamdouh

Test Planning

Test Planning

Project Objective:

Validate the functionality and reliability of OpenCart, API testing for GoREST, UI automation and performance of Demoblaze

Scope:

- User flows (Register, Login, Cart, Checkout)
- Product browsing & comparison
- Store navigation & account features

Testing Approach

- Agile methodology
- Manual, Database (OpenCart)
- API (GoRest)
- Automation and Performance (Demoblaze)

Test Planning

Entry / Exit Criteria

- Entry: Approved requirements, test cases ready
- Exit: Full execution, major defects resolved

Tools & Environment

- Excel, Jira, MySQL, Postman, GitHub, Selenium, JMeter
- Windows 11, MacOS Sequoia 15.7.1
- Firefox, Chrome, Safari

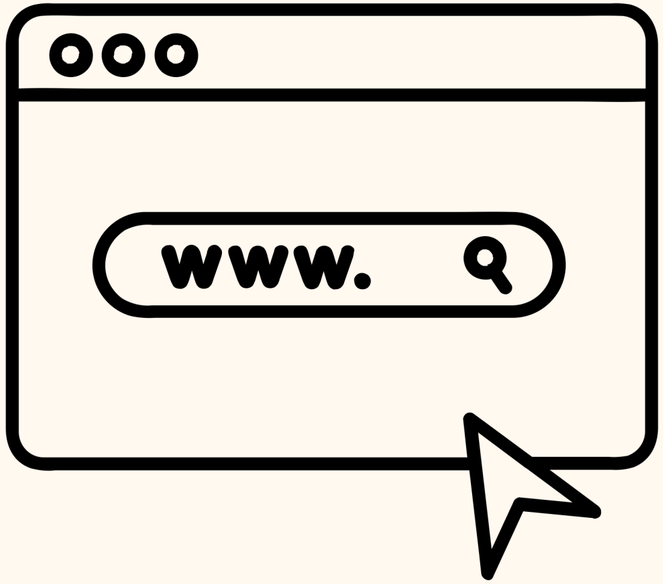
Test Schedule

- Manual → Database → API → Automation → Performance

Manual Testing

Manual Testing Approach

OpenCart ECommerce Demo



• **Registration** • **Login** • **Forgot Password** • **My Account Page** • **Product List Page**	• **Product Details Page** • **Compare Page** • **Cart** • **Checkout** • **Admin Page**

Functional testing and **exploratory** testing were done.

The tests were meticulously detailed, then subsequently refined to enhance efficiency while maintaining comprehensive coverage.

Examples of Manual Test Cases

Test ID	Test Case Description	Pre-Condition	Roles	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created I
CART_01	Add a single product to cart	User is logged in User is on the product deta	Customer	product name = "iPhone"	1- Go to the website: https://demo.opencart.com/en-gb?ro 2- Add one product to cart (ex, iPhone) 3- Observe that the item is added successfully	Product added successfully and visible in cart	As expected	Passed	Highest	Menna
CART_02	Add multiple different products to	User is logged in User is on the product deta	Customer	products name = "iPhone","	1- Go to the website: https://demo.opencart.com/en-gb?ro 2- Add 3 products to cart 3- Observe that the item is added successfully	All items appear in the cart	Clicking the "Add to Cart" icon r	Failed	Highest	Menna
CART_03	Add same product twice to cart	User is logged in User is on the product deta	Customer	product name = "iPhone"	1- Go to the website: https://demo.opencart.com/en-gb?ro 2- Add the same product twice to the cart (ex, iPhone) 3- Observe that the item quantity is increased successfully	Quantity increases instead of duplicate entry	As expected	Passed	High	Menna
CART_04	Add out-of-stock item to cart	User is logged in User is on the product deta	Customer	out of stock item	1- Go to the website: https://demo.opencart.com/en-gb?ro 2- Add out of stock item 3- Observe the cart page	System shows "Out of stock" message	As expected	Passed	High	Menna
CART_05	Check product details in cart	User is logged in User is on the product deta Items added	Customer	product name = "iPhone"	1- Go to the website: https://demo.opencart.com/en-gb?ro 2- Add products to cart (ex, iPhone) 3- Click cart icon 4- Observe that the items details (name, image, price, qua	Each product shows name, image, price, quantit	As expected	Passed	Highest	Menna
CART_06	View empty cart message	User is logged in The cart is empty	Customer		1- Go to the website: https://demo.opencart.com/en-gb?ro 2- keep the cart empty 3- Click cart icon 4- Observe the cart page	Message "Your cart is empty" displayed	As expected	Passed	High	Menna
CART_07	Verify cart icon count	User is logged in Add/remove items	Customer	prod						
CART_08	Increase product quantity	User is logged in Add items to cart	Customer	prod						
CART_09	Decrease product quantity	User is logged in Add items to cart	Customer	prod						
CART_10	Update quantity manually	User is logged in Add items to cart	Customer	prod						

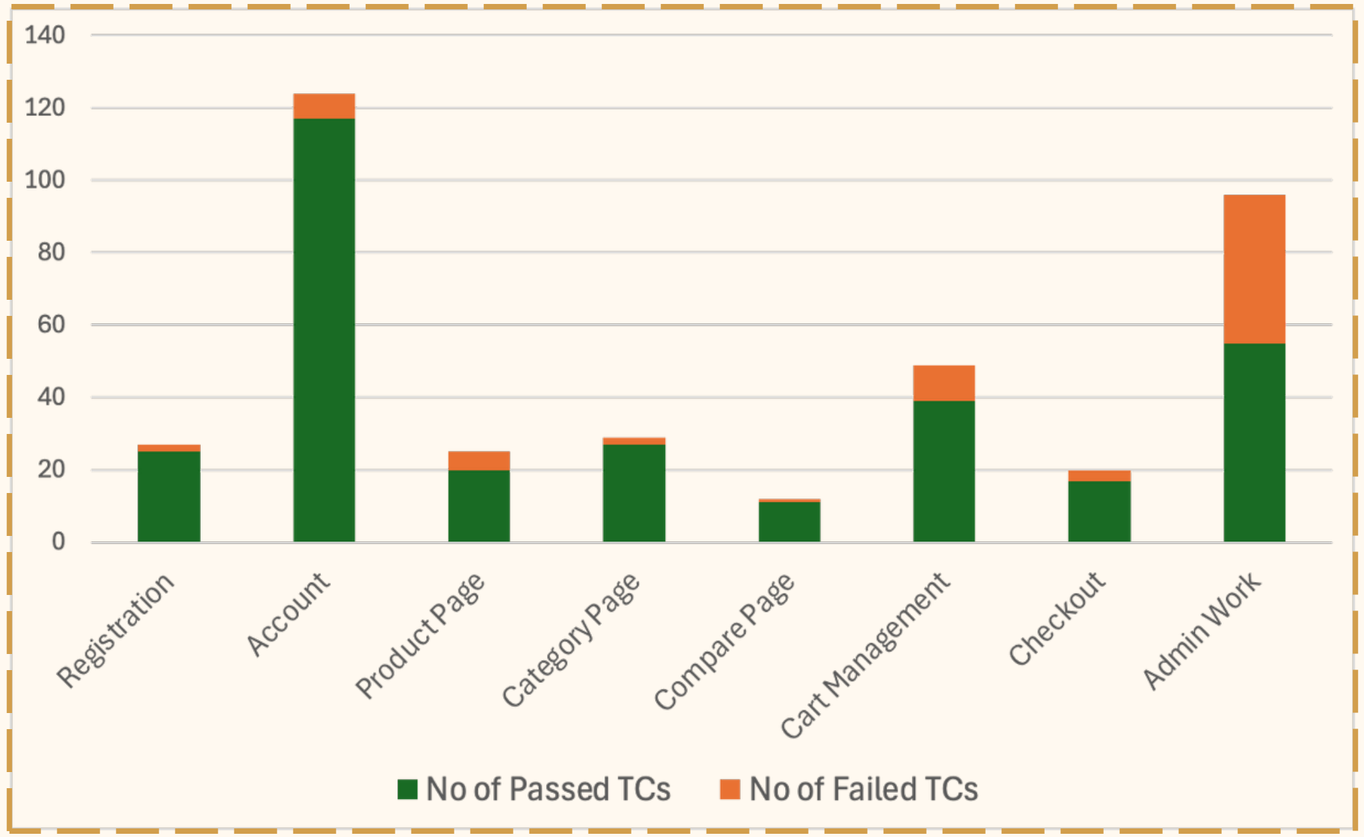
Test ID	Test Case Description	Pre-Condition	Roles	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created By
REG_01	verify that "Register" option exists in the Account drop		Customer		1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register"	"Register" option exists in the Account dropdown menu a	As expected	Passed	High	Ghada Amr
REG_02	verify that the registration page contains First Name, L		Customer	First Name: tester Last Name: tester Email: testerfortesting@grr Password: testing	1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register" 4- Enter First Name, Last Name, Email and Password	The registration page has input fields for the First Name,	As expected	Passed	Highest	Ghada Amr
REG_03	verify that the password field hides characters		Customer	First Name: tester Last Name: tester Email: testerfortesting@grr Password: testing	1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register" 4- Enter Password	Characters typed in the password field should be hidden	As expected	Passed	Medium	Ghada Amr
REG_04	verify that an error message appeas if any of the First		Customer	First Name: tester Last Name: tester Email: testerfortesting@grr Password: testing	1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register" 4- Enter First Name, Last Name, Email and Password 5- Leave First Name field empty 6- Click on the checkbox next to "I have read and agree to the Privac 7- Click on "Continue" button 8- Repeat steps 5 to 7 leaving the Last Name field empty	"First Name must be between 1 and 32 characters!" mes "Last Name must be between 1 and 32 characters!" mes	As expected	Passed	High	Ghada Amr
REG_05	verify that an error message appeas if Email is invalid		Customer	First Name: tester Last Name: tester Email: testerfortesting@grr Password: testing	1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register" 4- Enter First Name, Last Name and Password 5- Leave Email field empty 6- Click on the checkbox next to "I have read and agree to the Privac 7- Click on "Continue" button 8- Enter an invalid Email (testingfortester@grr....la) then click on "Co	"E-Mail Address does not appear to be valid!" error mess	As expected	Passed	High	Ghada Amr
REG_06	verify that an error message appeas if Password field		Customer	First Name: tester Last Name: tester Email: testerfortesting@grr Password:	1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register" 4- Enter First Name, Last Name and Email 5- Leave Password field empty 6- Click on the checkbox next to "I have read and agree to the Privac 7- Click on "Continue" button 8- Repeat steps 5 to 7 using a long password (testingfortestinglovertesterfortesting2025)	"Password must be between 6 and 40 characters!" error	As expected	Passed	Highest	Ghada Amr
REG_07	verify that the user can check to subscribe to the news		Customer	First Name: tester Last Name: tester Email: testerfortesting@grr Password: testing	1- Go to the website: https://demo.opencart.com/en-gb?route=comm 2- Click on "Account" in the header at the top of the page 3- Click on "Register" 4- Click the checkbox next to the"Subscribe" under "Newsletter"	The user can check the subscribe option to subscribe to	As expected	Passed	Low	Ghada Amr

Representative bugs uncovered during testing

Defect #	Bug Description	Status	Assigned To	Raised By	Tested on Version	Test Case I	Steps to Reproduce	Expected Result	Actual Result	Priority	Severity
BUG_PP_01	Incorrect product images displayed when clicking alternat	open	Developer	Mohamed	win 11 pro - chron	PP_02	1. Go to the website: https://demo.opencart.com/ 2. Open the product page for a product with multiple in 3. Look for thumbnail/alternate view images below/nea 4. Click each thumbnail to change the main image	Alternate thumbnails are displayed and c	Other product Images show	High	Major
BUG_PP_02	Discounted price not clearly visible or formatted	open	Developer	Mohamed	win 11 pro - chron	PP_12	1. Go to the website: https://demo.opencart.com/ 2. Open product page for discounted product 3. Observe original price and discounted price present	Discounted price is displayed and origina	Discounted price is not disp	Medium	Moderate
BUG_PP_03	Quantity field allows invalid input without showing validati	open	Developer	Mohamed	win 11 pro - chron	PP_15	1. Go to the website: https://demo.opencart.com/ 2. Open product page 3. Enter a numeric value in quantity field and try to adc 4. Enter invalid input ("test") and verify validation	Quantity accepts numbers; invalid input p	No error massage	High	Major
BUG_PP_04	Invalid quantity can be added to cart without validation	open	Developer	Mohamed	win 11 pro - chron	PP_16	1. Go to the website: https://demo.opencart.com/ 2. Open product page 3. Enter (1) quantity and click Add to Cart 4. Observe validation/error messages	System shows appropriate validation mes	added to cart withour error r	High	Critical
BUG_PP_05	"Write Review" link is unresponsive	open	Developer	Mohamed	win 11 pro - chron	PP_23	1. Go to the website: https://demo.opencart.com/ 2. Open product page 3. Click on 'Write a review' link 4. Observe redirect to review tab or form	Write review link opens the review form/ta	not clickabe	Medium	Major
BUG_CMP_0	Alternate images not updating on compare page	open	Developer	Mohamed			1. Go to the website: https://demo.opencart.com/ 2. Open product page 2. On Compare page,change product images				
BUG_CP_01	Incorrect category products shown on Desktops page	open	Developer	Mohamed							
BUG_CP_02	Category filter not returning products	open	Developer	Mohamed							

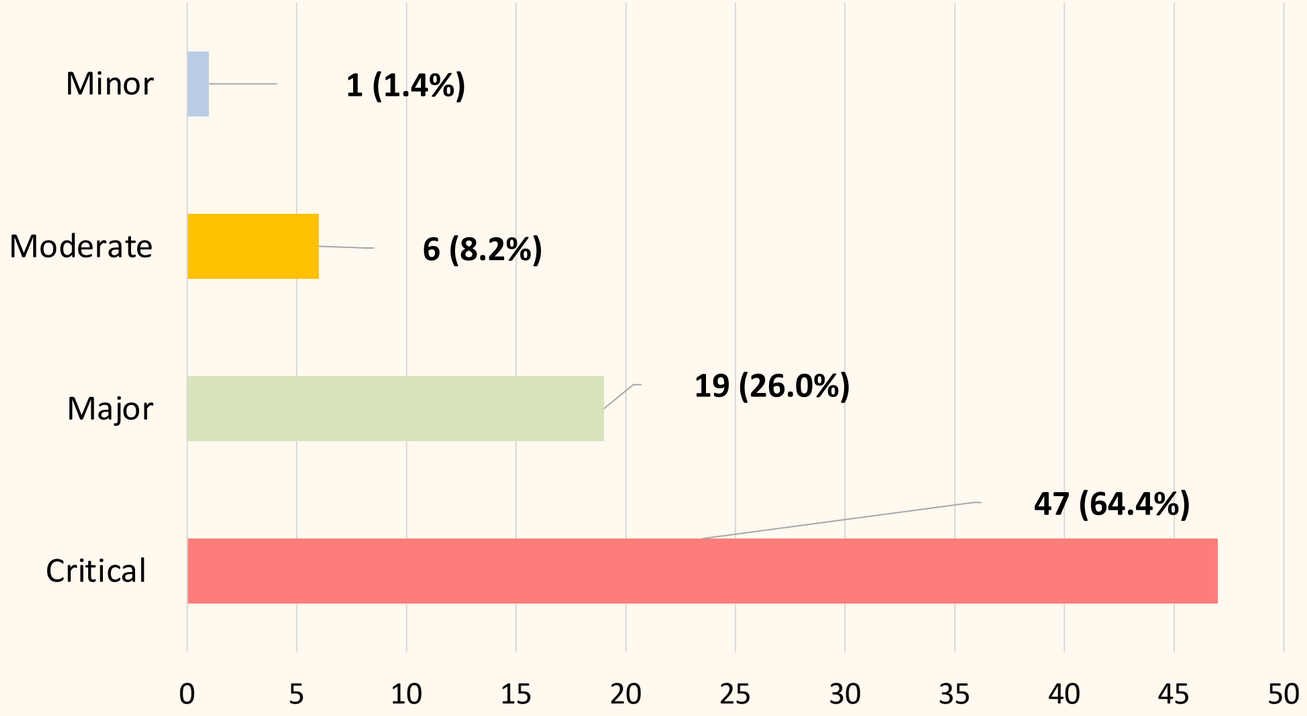
Defect #	Bug Description	Status	Assigned To	Raised By	Tested on Version	Test Case ID	Steps to Reproduce	Expected Result	Actual Result	Priority	Severity
BUG_ADMIN_01	Error message display (object object)	New	Developer	Randa	Chrome v141	TC_Order_002	1. Scroll to the Order History section. 2. Select the Processing status from the "Order Status" dropdown. 3. Enter a comment in the "Comment" field. 4. Click the "Add History" button.	The page refreshes (or updates), the new history entry (including status and comment) appears in the Order History table, and the main Order Status field reflects the new status (Processing).	Error message display (object object)	High	Critical
BUG_ADMIN_02	The table reloads and displays the message "No results!".	New	Developer	Randa	Chrome v141	TC_Order_005	1. Enter the partial customer name (kathy) into the 'Customer' filter field. 2. Click the 'Filter' button.	The table reloads, showing all orders placed by "Kathy Durr" (and any other customer containing the text 'kathy').	The table reloads and displays the message "No results!".	High	Critical
BUG_ADMIN_03	The table reloads and displays the message "No results!".	New	Developer	Randa	Chrome v141	TC_Order_006	1. Select the status Processing from the 'Order Status' dropdown. 2. Click the 'Filter' button.	The table reloads, and every order in the list has the status Processing.	The table reloads and displays the message "No results!".	Highest	Critical
BUG_ADMIN_04	the system display message "Warning: You do not have permission to modify orders!"	New	Developer	Randa	Chrome v141	TC_Order_008	1. Click the 'Add Order' button. 2. Do not fill out any required fields (First Name, Last Name, E-Mail, Telephone). 3. Click the 'Continue' button at the bottom.	Validation errors are displayed next to the missing required fields, and the user remains on Tab 1: Customer Details.	the system display message "Warning: You do not have permission to modify orders!"	Highest	Critical
BUG_ADMIN_05	no action happend	New	Developer	Randa	Chrome v141	TC_ADD_CUST_001	1. Fill out all required fields on the General tab. 2. Click the blue 'Save' button.	The user is redirected to the Customer List with a success message. The new customer is listed as a standard user.	no action happend	High	Critical
BUG_ADMIN_06	no action happend	New	Developer	Randa	Chrome v141	TC_ADD_CUST_002	1. Leave any required field on the General tab empty. 2. Click the 'Save' button.	Validation error messages are displayed next to the missing required fields, and the customer is NOT saved.	no action happend	Highest	Critical
BUG_ADMIN_07	The table reloads and displays the message "No results!".	New	Developer	Randa	Chrome v141	TC_CUST_003	1. Enter the e-mail into the 'E-Mail' filter field. 2. Click the 'Filter' button.	Only the customer matching that e-mail address is displayed.	The table reloads and displays the message "No results!".	High	Critical
BUG_ADMIN_08	The table reloads and displays the message "No results!".	New	Developer	Randa	Chrome v141	TC_CUST_005	1. Select 'Disabled' from the 'Status' dropdown. 2. Click the 'Filter' button.	Only customers with a 'Disabled' status are displayed.	The table reloads and displays the message "No results!".	High	Critical
BUG_ADMIN_09	the system display message "Warning: You do not have permission to modify extension!"	New	Developer	Randa	Chrome v141	TC_EXT_02	1. Navigate to the relevant category (Shipping). 2. Click the 'Install' button (green plus icon) next to the target extension.	A success message is displayed. The 'Install' button is replaced by the 'Uninstall' (red minus) and 'Edit' (blue pencil) buttons.	the system display message "Warning: You do not have permission to modify extension!"	High	Critical
BUG_ADMIN_10	the system display message "Warning: You do not have permission to modify extension!"	New	Developer	Randa	Chrome v141	TC_EXT_03	1. Navigate to the relevant category (Shipping). 2. Click the 'Uninstall' button (red minus icon) next to the target extension.	A success message is displayed. The 'Uninstall' and 'Edit' buttons are replaced by the 'Install' button.	the system display message "Warning: You do not have permission to modify extension!"	High	Critical

Manual Testing Pass Rates



No	Module Name	Pass Rate %
1	Registration	92.6
2	Account	94.4
3	Product Page	80.0
4	Category Page	93.1
5	Compare Page	91.7
6	Cart Management	79.6
7	Checkout	85.0
8	Admin Work	57.3
	Total	81.4

Bug Severity Breakdown



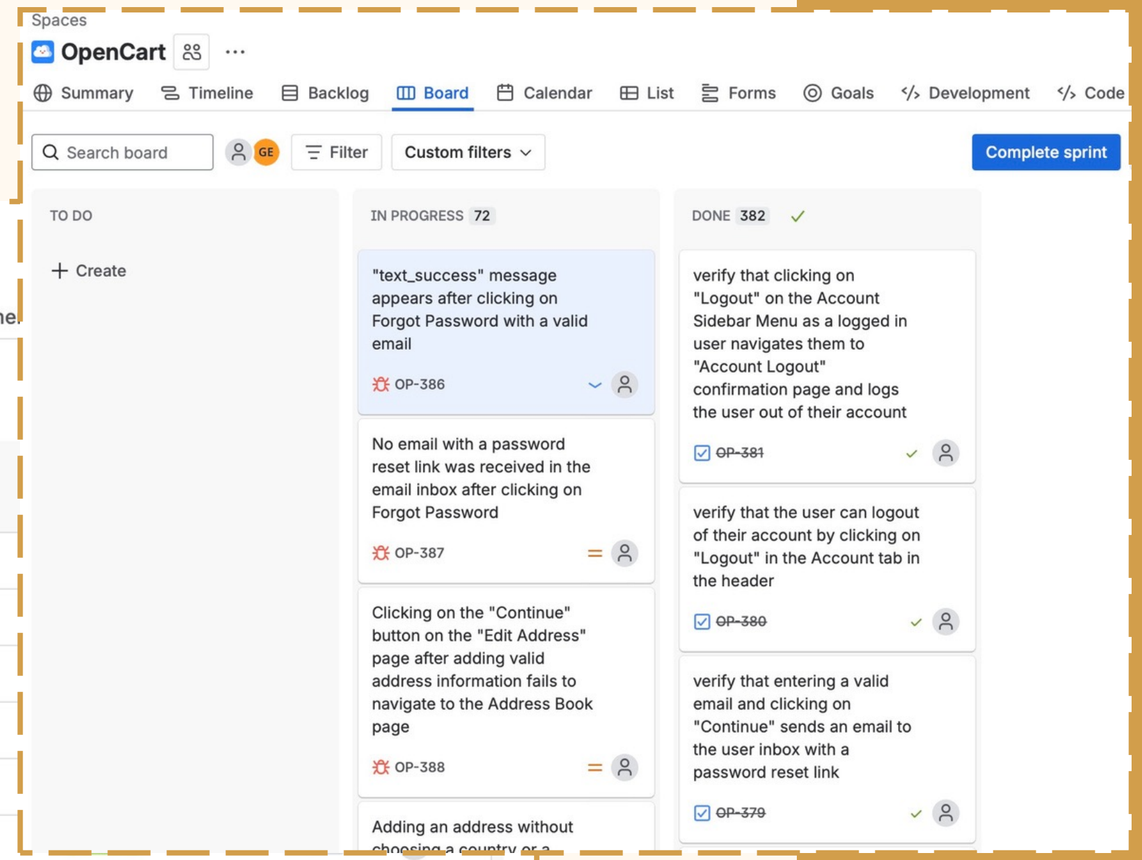
Bug Severity	No	%
Critical	47	64.4
Major	19	26.0
Moderate	6	8.2
Minor	1	1.4
Total	73	100

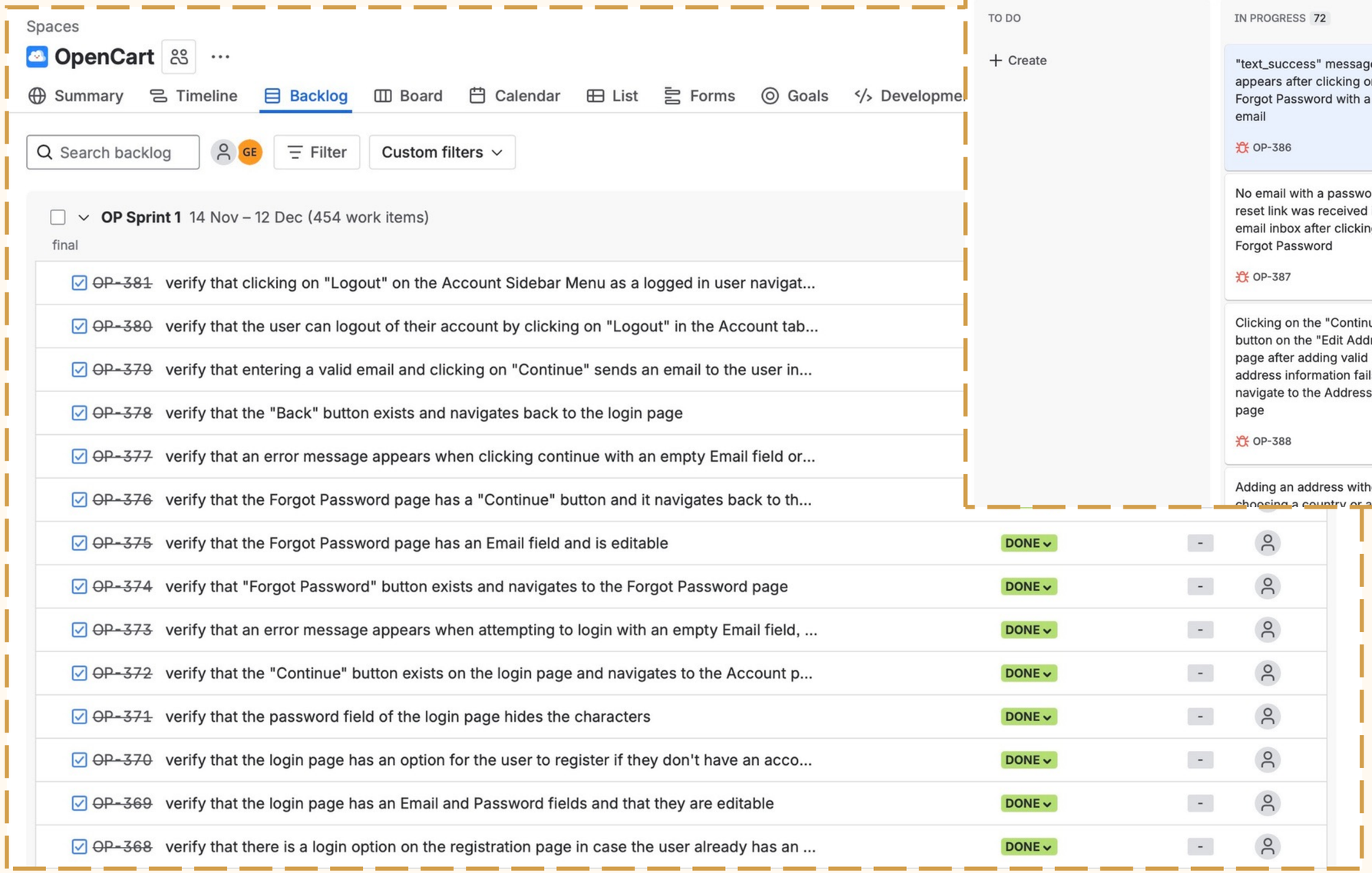
Jira Integration

- Used to **log** and track all defects during testing
- Supports retesting and **updating** bug status

Jira Software

Jira Integration





Jira Integration

Each bug includes:

- Test ID
- Pre-condition
- Test Data
- Steps to reproduce
- Expected Result
- Actual Result
- Priority

Add epic / OP-386

"text_success" message appears after clicking on Forgot Password with a valid email

Key details

Description

Entering a valid email then clicking on Forgot Password displays a message stating “text_success” instead of “Reset email sent successfully”

Test ID

FP_03

Pre-Condition

User has a valid email

Roles

Tester

Test Data

Email: testerfortesting@grr.la

Steps/Actions

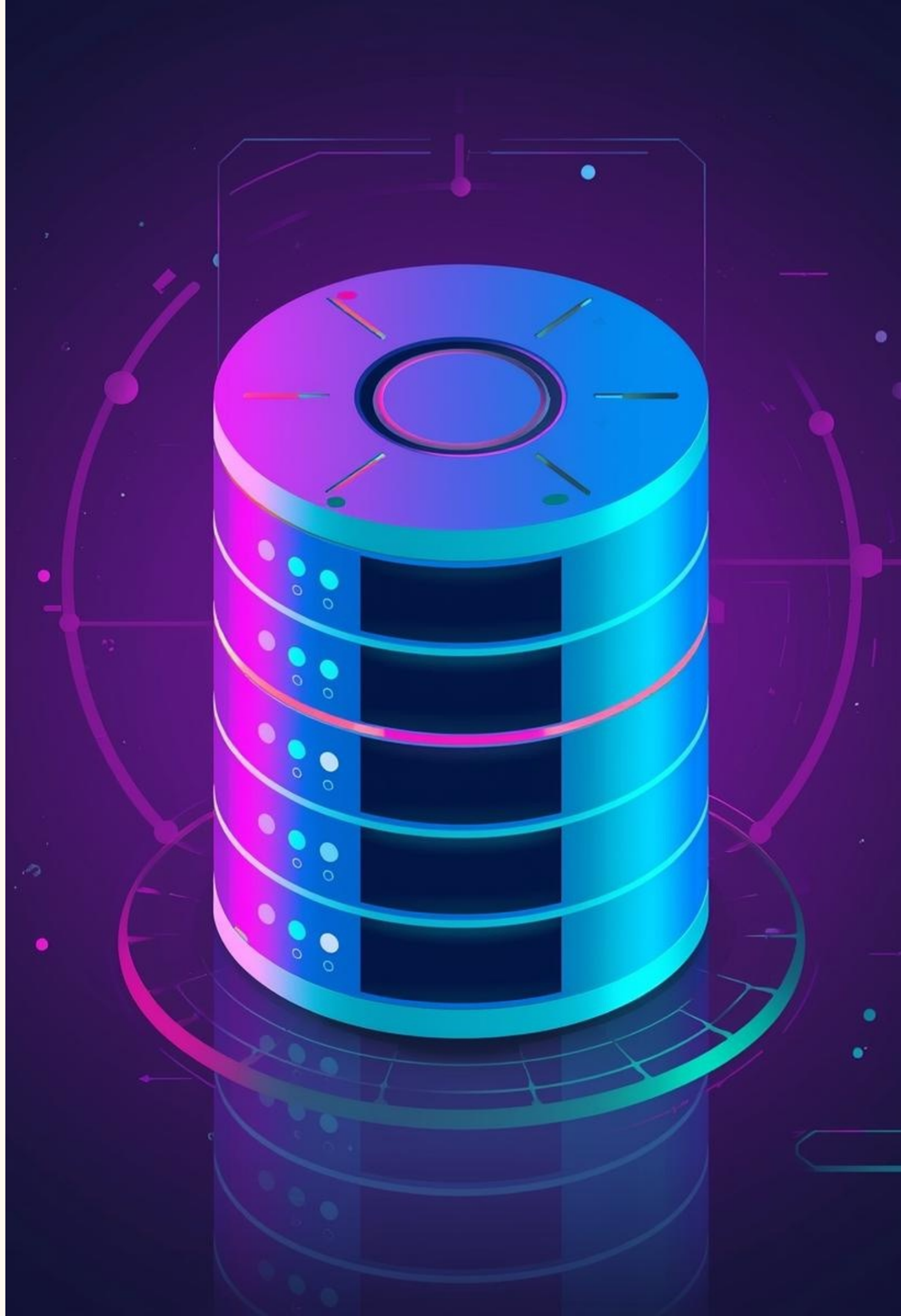
1- Go to the website: https://demo.opencart.com/en-gb?route=common/home
2- Click on "Account" in the header at the top of the page
3- Click on "Login"
4- Click on "Forgot Password"

Database Testing

Database Testing Approach

OpenCart ECommerce Demo

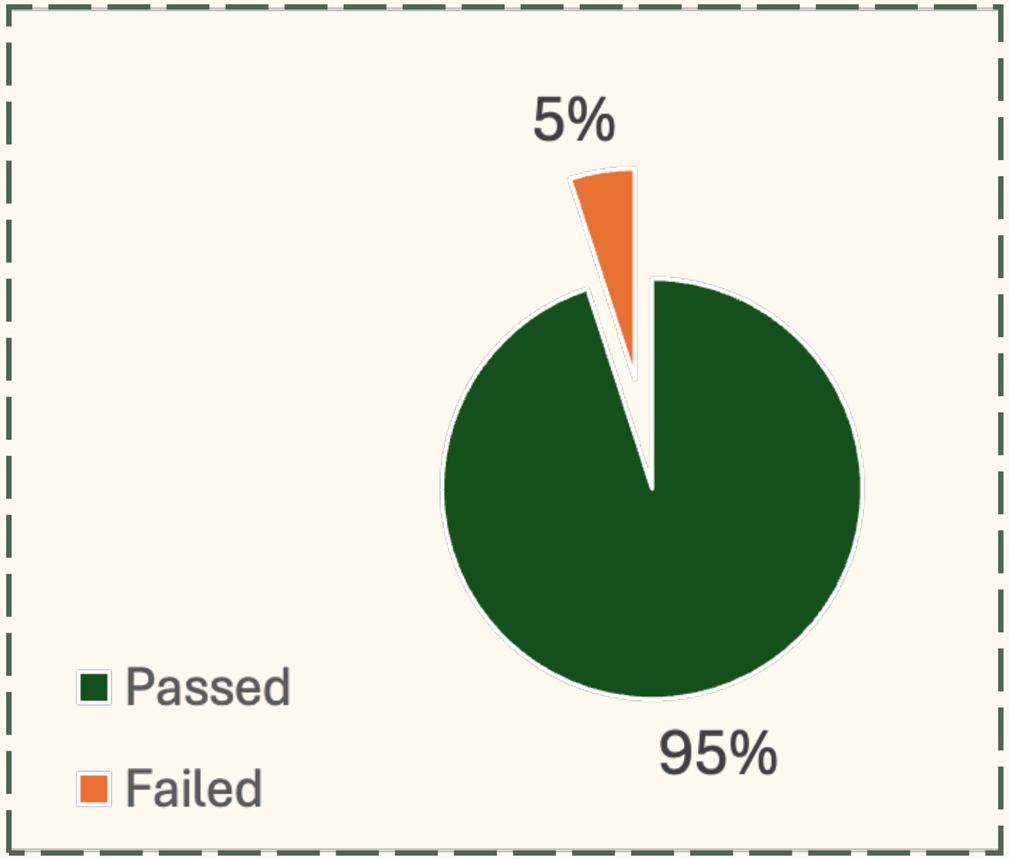
- Used **MySQL** to verify data consistency between frontend actions and MySQL tables
- Checked key tables: **Customers, Address, Products, Orders**
- Validated inserts, updates, deletions, and key relationships
- Cross-checked UI/API results with stored data

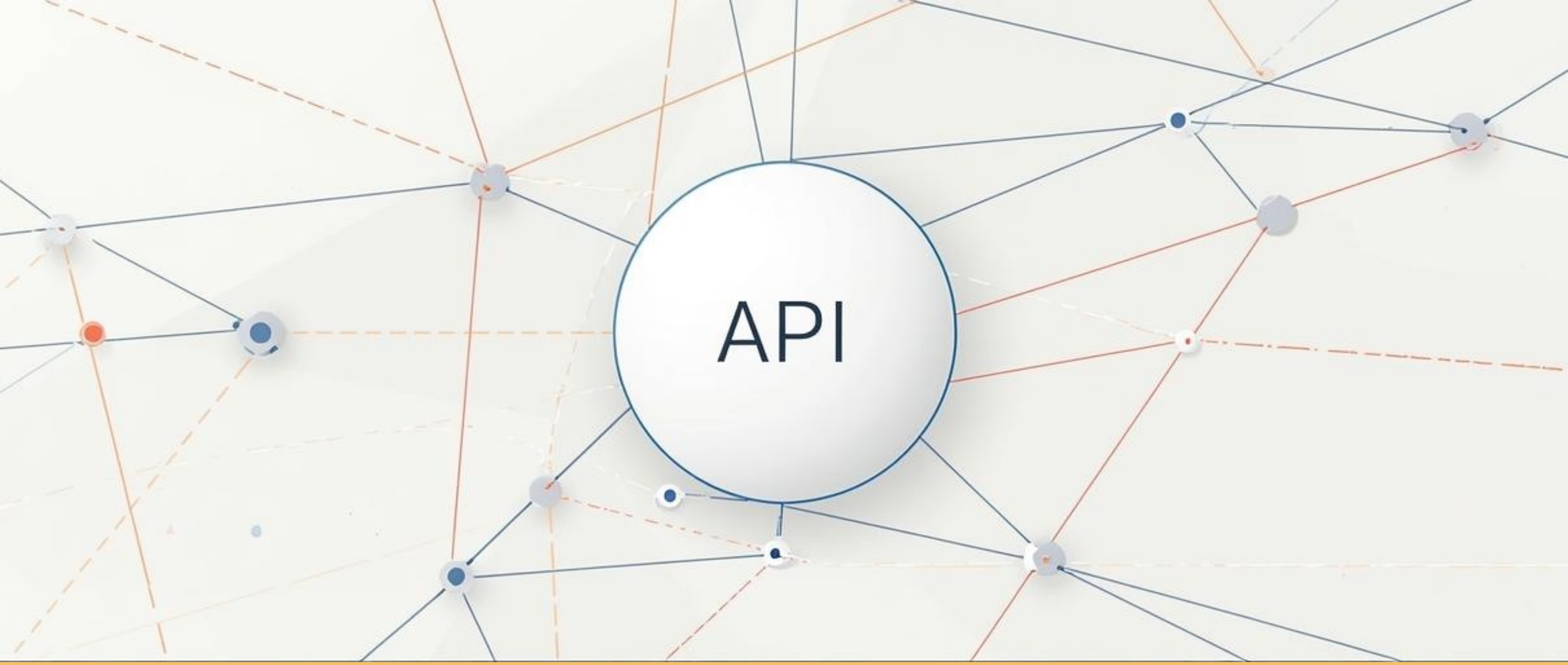


Database test cases

Test ID	Test Case Description	Pre-Condition	Roles	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created By
DB_CUS_01	Verify customers have first name	Database open	DB	oc_customer	Run: SELECT * FROM oc_customer WHERE firstname IS NULL OR firstname='';	Query should return zero rows.	As expected	Passed	High	Mohamed
DB_CUS_02	Verify customers have last name	Database open	DB	oc_customer	Run: SELECT * FROM oc_customer WHERE lastname IS NULL OR lastname='';	Query should return zero rows.	As expected	Passed	Medium	Mohamed
DB_CUS_03	Check customers without related address	Database open	DB	oc_customer, oc_address	Run: SELECT c.customer_id FROM oc_customer c LEFT JOIN oc_address a ON c.customer_id=a.custor	Every customer should have at least on	As expected	Passed	Medium	Mohamed
DB_CUS_04	Verify duplicate customers by name	Database open	DB	oc_customer	Run: SELECT c1.customer_id, c1.firstname, c1.lastname, c1.email FROM oc_customer c1 JOIN oc_cus	No duplicate names should exist.	As expected	Passed	Low	Mohamed
DB_CUS_05	Check orders linked to deleted customers	Database open	DB	oc_order, oc_customer	Run: SELECT o.order_id,o.customer_id FROM oc_order o LEFT JOIN oc_customer c ON o.customer_id	No broken links between orders and cu	As expected	Passed	High	Mohamed
DB_ORD_01	Verify no order exists without customer	Database open	DB	oc_order	Run: SELECT * FROM oc_order WHERE customer_id=0 OR customer_id IS NULL;	All orders should have valid customer I	As expected	Passed	High	Mohamed
DB_ORD_02	Verify orders linked to deleted customers	Database open	DB	oc_order, oc_customer	Run: SELECT o.order_id FROM oc_order o LEFT JOIN oc_customer c ON o.customer_id=c.customer_i	No broken links.	As expected	Passed	High	Mohamed
DB_ORD_03	Check orders without order products	Database open	DB	oc_order, oc_order_product	Run: SELECT o.order_id FROM oc_order o LEFT JOIN oc_order_product op ON o.order_id=op.order_i	Every order should have at least one pr	As expected	Passed	Medium	Mohamed
DB_ORD_04	Verify shipping details not empty	Database open	DB	oc_order	Run: SELECT order_id FROM oc_order WHERE shipping_firstname='' OR shipping_address_1='';	All shipping details should be filled.	As expected	Passed	Medium	Mohamed
DB_ORD_05	Verify order totals > 0	Database open	DB	oc_order	Run: SELECT order_id,total FROM oc_order WHERE total IS NULL OR total=0;	Orders must have positive total value.	As expected	Passed	High	Mohamed
DB_PRO_01	Verify products have names	Database open	DB	oc_product, oc_product_des	Run: SELECT p.product_id FROM oc_product p LEFT JOIN oc_product_description pd ON p.product_i	Every product should have a name.	As expected	Passed	High	Mohamed
DB_PRO_02	Verify products have prices	Database open	DB	oc_product	Run: SELECT product_id FROM oc_product WHERE price IS NULL OR price=0;	No product should have empty or zero	As expected	Passed	High	Mohamed
DB_PRO_03	Check products without category	Database open	DB	oc_product, oc_product_to_	Run: SELECT p.product_id FROM oc_product p LEFT JOIN oc_product_to_category pc ON p.product_i	All products must belong to at least one	As expected	Passed	Medium	Mohamed
DB_PRO_04	Check products deleted but linked in order	Database open	DB	oc_product, oc_order_produ	Run: SELECT op.order_id,op.product_id FROM oc_order_product op LEFT JOIN oc_product p ON op.p	No broken links between order and pro	As expected	Passed	High	Mohamed
DB_PRO_05	Verify product stock quantity	Database open	DB	oc_product	Run: SELECT product_id,quantity FROM oc_product WHERE quantity<=0;	All products should have stock > 0.	Product id 49 is zer	Failed	Medium	Mohamed
DB_ADD_01	Verify address has valid customer	Database open	DB	oc_address, oc_customer	Run: SELECT a.address_id,a.customer_id FROM oc_address a LEFT JOIN oc_customer c ON a.custome	Every address should belong to an exist	As expected	Passed	High	Mohamed
DB_ADD_02	Verify address has names filled	Database open	DB	oc_address	Run: SELECT address_id FROM oc_address WHERE firstname='' OR lastname='';	All addresses should include customer	As expected	Passed	Medium	Mohamed
DB_ADD_03	Verify address has city and country	Database open	DB	oc_address	Run: SELECT address_id FROM oc_address WHERE city='' OR country_id=0;	City and country fields must be filled.	As expected	Passed	Medium	Mohamed
DB_ADD_04	Check duplicate addresses for same custor	Database open	DB	oc_address	Run: SELECT a1.customer_id,a1.firstname,a1.lastname FROM oc_address a1, oc_address a2 WHERE	No duplicate addresses should exist.	As expected	Passed	Low	Mohamed
DB_ADD_05	Verify orders not linked to deleted address	Database open	DB	oc_order, oc_address	Run: SELECT o.order_id,o.customer_id FROM oc_order o LEFT JOIN oc_address a ON o.customer_id=	All orders should have valid customer a	As expected	Passed	High	Mohamed

Database Testing Pass Rates



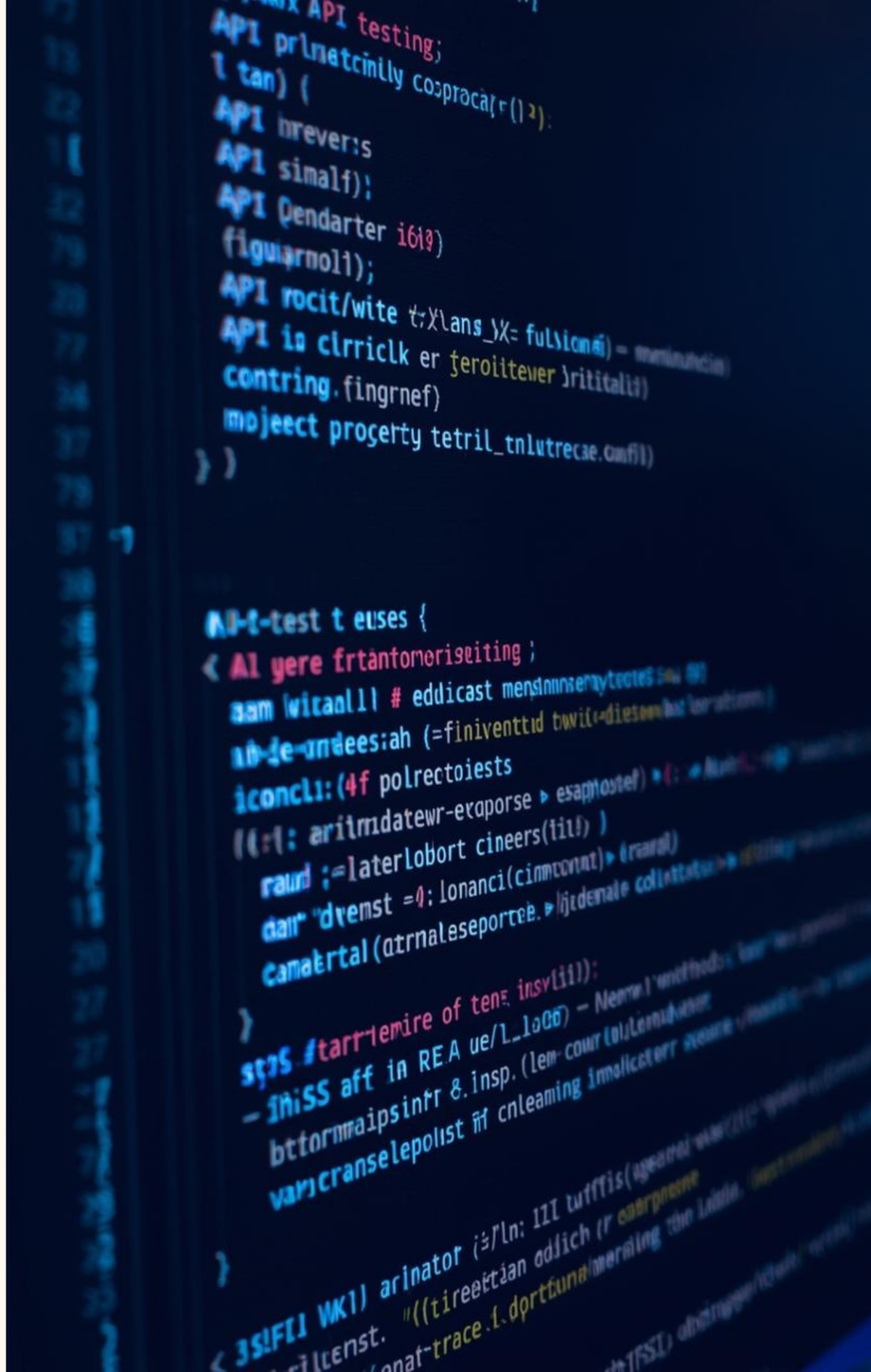


API Testing

API Testing Approach

GoREST: GraphQL and REST API Demo

- Used **Postman** to test API on GoREST
- Covered modules: **Users**, **Posts**, **Comments**, **Todos**
- Tested key HTTP methods: **GET**, **POST**, **PUT, DELETE**
- Verified response codes, payload structure, and data accuracy
- Checked authentication handling and error responses
- Ensured API behavior aligned with expected system functionality



Overview of API test cases: Users, Posts

Test ID	Test Case Description	Pre-Condition	Roles	Method	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created By	Type
API_POST_01	Verify that all posts can be retrieved successfully	Valid API token available	API Tester	GET	Endpoint: https://gorest.co.in/public/v2/posts Token: {{token}}	1. Open Postman 2. Set method to GET 3. URL: https://gorest.co.in/public/v2/posts4. Add header Authorization: Bearer {{token}}5. Send request	Status 200 OK. Response returns list of posts with id, user_id, title, body.	As expected	Pass	High	Randa	API Test
API_POST_02	Verify that unauthorized user cannot retrieve posts	No token provided	API Tester				Status 401					
API_POST_03	Verify that a new post can be created successfully	Valid token and existing user_id	API Tester									
API_POST_04	Verify that creating a post with missing fields fails	Valid token	API Tester									
API_POST_05	Verify that an existing post can be updated successfully	Valid token and existing post_id	API Tester									
API_POST_06	Verify that updating non-existing post returns error	Valid token	API Tester									
API_POST_07	Verify Post Deletion	Valid API token available	API Tester									
API_POST_08	Verify deleted Post cannot be retrieved	Valid API token available	API Tester									

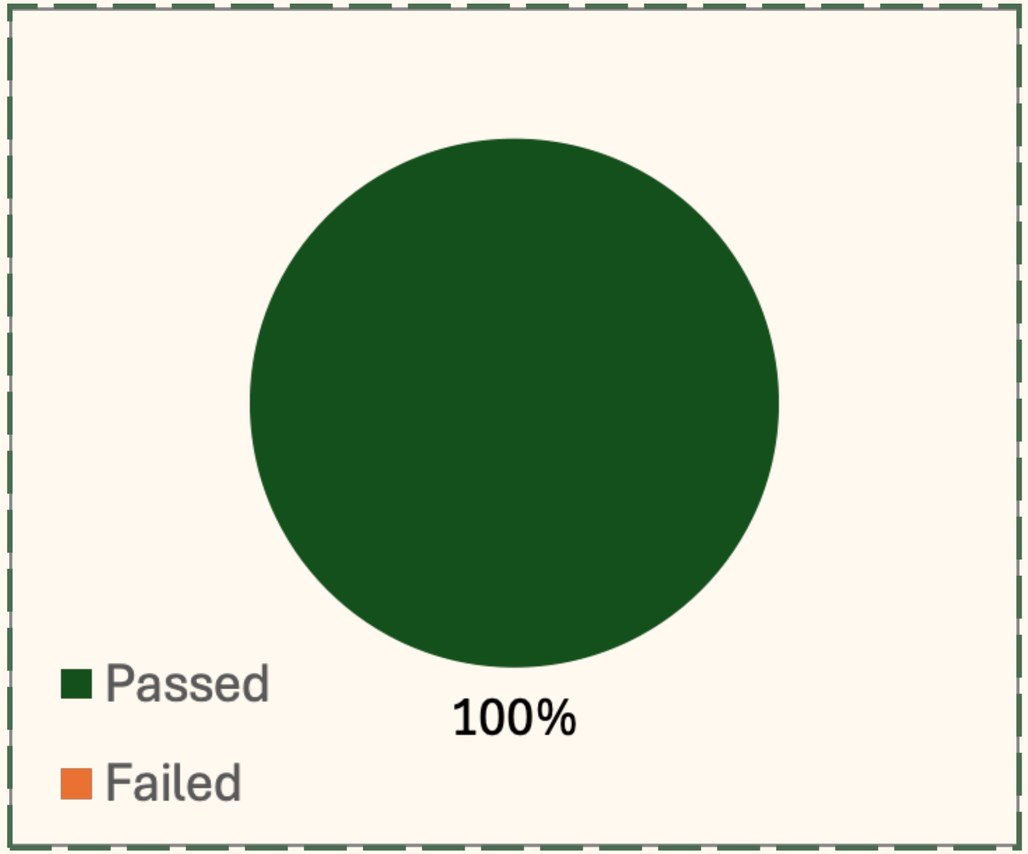
Test ID	Test Case Description	Pre-Condition	Roles	Method	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created By	Type
API_USER_01	Verify that the list of users is returned successfully	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	GET	Endpoint : https://gorest.co.in/public/v2/users	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users 4. Click Send to execute the API request.	200 OK, and a JSON array of users is returned	As expected	Pass	High	Menna	API Test
API_USER_02	Verify that request with invalid user ID returns 404	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	GET	Endpoint : https://gorest.co.in/public/v2/users ID : 99999999999	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/99999999 4. Click Send to execute the API request.	404 Not Found with error message " "Resource not found""	As expected	Pass	Medium	Menna	API Test
API_USER_03	Verify creating a new user with valid data	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	POST	Endpoint : https://gorest.co.in/public/v2/users {"name": "Test User", "gender": "female", "email": "test.user@email.com ", "status": "active"}	1. Open Postman Tool 2. Set the request type to POST. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users 4.Enter valid body 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	201 Created, returns created user data	As expected	Pass	Highest	Menna	API Test
API_USER_04	Verify error when creating user with invalid email format	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	POST	Endpoint : https://gorest.co.in/public/v2/users {"name": "Bad Email", "gender": "male", "email": "bademail", "status": "active"}	1. Open Postman Tool 2. Set the request type to POST. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users 4.Enter valid body but invalid email 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	422 Unprocessable Entity, error for invalid email	As expected	Pass	High	Menna	API Test
API_USER_05	Verify updating an existing user's name	The API server https://gorest.co.in/ Vaild: token Postman tool userID:	API Tester	PUT	Endpoint : https://gorest.co.in/public/v2/users {"name": "Updated Name"} UserID: 8234667	1. Open Postman Tool 2. Set the request type to PUT. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8234667 4.update the name in the body 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	200 OK, user data updated	As expected	Pass	High	Menna	API Test
API_USER_06	Verify error when updating user with invalid status value	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	PUT	Endpoint : https://gorest.co.in/public/v2/users {"name": "Updated Name"} UserID: 8234667 invalid status value : {"status": "unknown"}	1. Open Postman Tool 2. Set the request type to PUT. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8234667 4.Enter invalid status value {"status": "unknown"} 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	422 error, with validation message	As expected	Pass	Medium	Menna	API Test
API_USER_07	Verify User Deletion	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	DELETE	Endpoint: https://gorest.co.in/public/v2/users/8229349	1. Open Postman Tool 2. Set the request type to DELETE. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8229349 4.Click Send to execute the API request.	204 No Content	As expected	Pass	High	Menna	API Test
API_USER_08	Verify deleted User cannot be retrieved	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	GET	Endpoint: https://gorest.co.in/public/v2/users/8229349	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8229349 4.Click Send to execute the API request.	404 Not Found: resource doesn't exist.	As expected	Pass	High	Menna	API Test

Overview of API test cases: Comments, ToDos

Test ID	Test Case Description	Pre-Condition	Roles	Method	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created By	Type
API_DO_01	Verify that the GET API for the To-Do List returns the correct response and status code (200 OK).	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	GET	Endpoint: https://gorest.co.in/public/v2/todos Headers: Authorization: Bearer <valid_token>	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/todos. 4. Add Vaild token in authorization header.	API should respond with HTTP 200 OK. Response body should contain a list of To-Do items in JSON format.	As expected	Pass	Medium	Mohamed	API Test
API_DO_02	Verify that the GET API for the To-Do List returns appropriate error responses when unauthorized requests are made	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	GET	Endpoint: https Head <Inva	1. Open Postman Tool						
API_DO_03	Verify that the POST API creates a new To-Do item successfully when valid data is provided	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	POST	Endp https: Meth Head <valid Body json {							
API_DO_04	Verify that the POST API returns appropriate error messages when invalid data .	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	POST	Endp https Meth Head <valid Body							
API_DO_05	Verify that the PUT API successfully updates an existing To-Do item with valid data.	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	PUT	Endp https Meth Head <valid id:93 Body							
API_DO_06	Verify that the PUT API returns appropriate errors when invalid ID	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	PUT	Endp https Meth Head <valid invail Body json {							
API_DO_07	Verify Todo Deletion	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	DELETE	Endp https 51							
API_DO_08	Verify deleted Todo cannot be retrieved	The API server https://gorest.co.in/ Vaild token Postman tool	API Tester	GET	Endp https 51							

Test ID	Test Case Description	Pre-Condition	Roles	Method	Test Data	Steps/Actions	Expected Result	Actual Result	Status	Priority	Created By	Type
API_USER_01	Verify that the list of users is returned successfully	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	GET	Endpoint : https://gorest.co.in/public/v2/users	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users 4. Click Send to execute the API request.	200 OK, and a JSON array of users is returned	As expected	Pass	High	Menna	API Test
API_USER_02	Verify that request with invalid user ID returns 404	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	GET	Endpoint : https://gorest.co.in/public/v2/users ID : 99999999999	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/99999999 4. Click Send to execute the API request.	404 Not Found with error message " "Resource not found""	As expected	Pass	Medium	Menna	API Test
API_USER_03	Verify creating a new user with valid data	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	POST	Endpoint : https://gorest.co.in/public/v2/users {"name": "Test User", "gender": "female", "email": "test.user@email.com ", "status": "active"}	1. Open Postman Tool 2. Set the request type to POST. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users 4.Enter valid body 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	201 Created, returns created user data	As expected	Pass	Highest	Menna	API Test
API_USER_04	Verify error when creating user with invalid email format	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	POST	Endpoint : https://gorest.co.in/public/v2/users {"name": "Bad Email", "gender": "male", "email": "bademail", "status": "active"}	1. Open Postman Tool 2. Set the request type to POST. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users 4.Enter valid body but invalid email 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	422 Unprocessable Entity, error for invalid email	As expected	Pass	High	Menna	API Test
API_USER_05	Verify updating an existing user's name	The API server https://gorest.co.in/ Vaild: token Postman tool userID:	API Tester	PUT	Endpoint : https://gorest.co.in/public/v2/users {"name": "Updated Name"} UserID: 8234667	1. Open Postman Tool 2. Set the request type to PUT. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8234667 4.update the name in the body 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	200 OK, user data updated	As expected	Pass	High	Menna	API Test
API_USER_06	Verify error when updating user with invalid status value	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	PUT	Endpoint : https://gorest.co.in/public/v2/users {"name": "Updated Name"} UserID: 8234667 invalid status value : {"status": "unknown"}	1. Open Postman Tool 2. Set the request type to PUT. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8234667 4.Enter invalid status value {"status": "unknown"} 5.Enter Authorization: Bearer TOKEN 6.Click Send to execute the API request.	422 error, with validation message	As expected	Pass	Medium	Menna	API Test
API_USER_07	Verify User Deletion	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	DELETE	Endpoint: https://gorest.co.in/public/v2/users/8229349	1. Open Postman Tool 2. Set the request type to DELETE. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8229349 4.Click Send to execute the API request.	204 No Content	As expected	Pass	High	Menna	API Test
API_USER_08	Verify deleted User cannot be retrieved	The API server https://gorest.co.in/ Vaild: token Postman tool	API Tester	GET	Endpoint: https://gorest.co.in/public/v2/users/8229349	1. Open Postman Tool 2. Set the request type to GET. 3. Enter the endpoint URL: https://gorest.co.in/public/v2/users/8229349 4.Click Send to execute the API request.	404 Not Found: resource doesn't exist.	As expected	Pass	High	Menna	API Test

API Module Pass Rates



Each of the four modules assessed consisted of 8 test cases, with a pass rate of 100%

Automation Testing

Automation Testing Approach

Demoblaze ECommerce Demo

- Automated key test cases using **Selenium WebDriver** + **Java**
- Focused on high-priority flows: Login, Registration, Product Search, Cart, and Checkout
- Built a reusable framework with Page Object Model (**POM**)
- Used **TestNG** for execution, reporting, and test organization

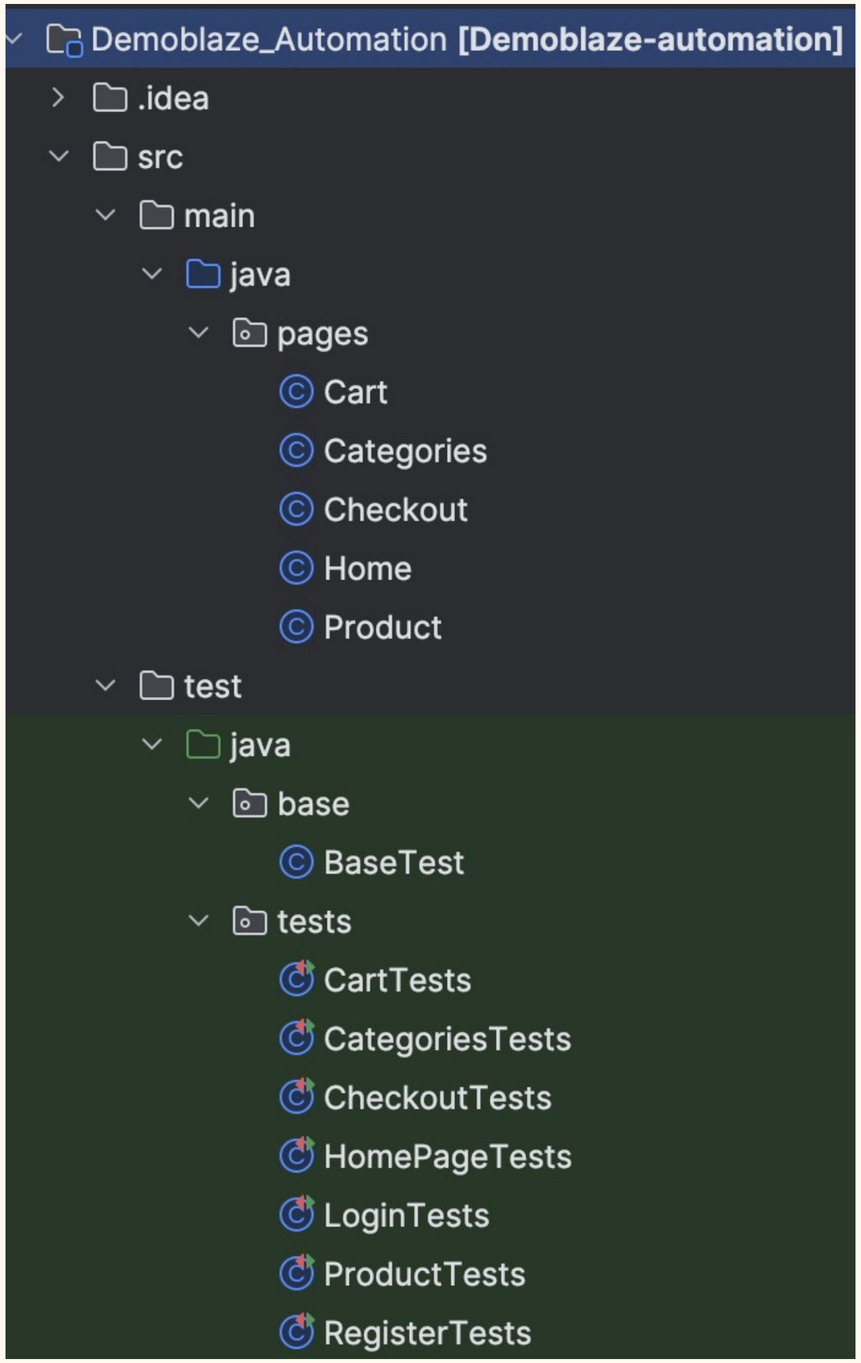
TestNG Pages Overview

Main **Pages** created included:

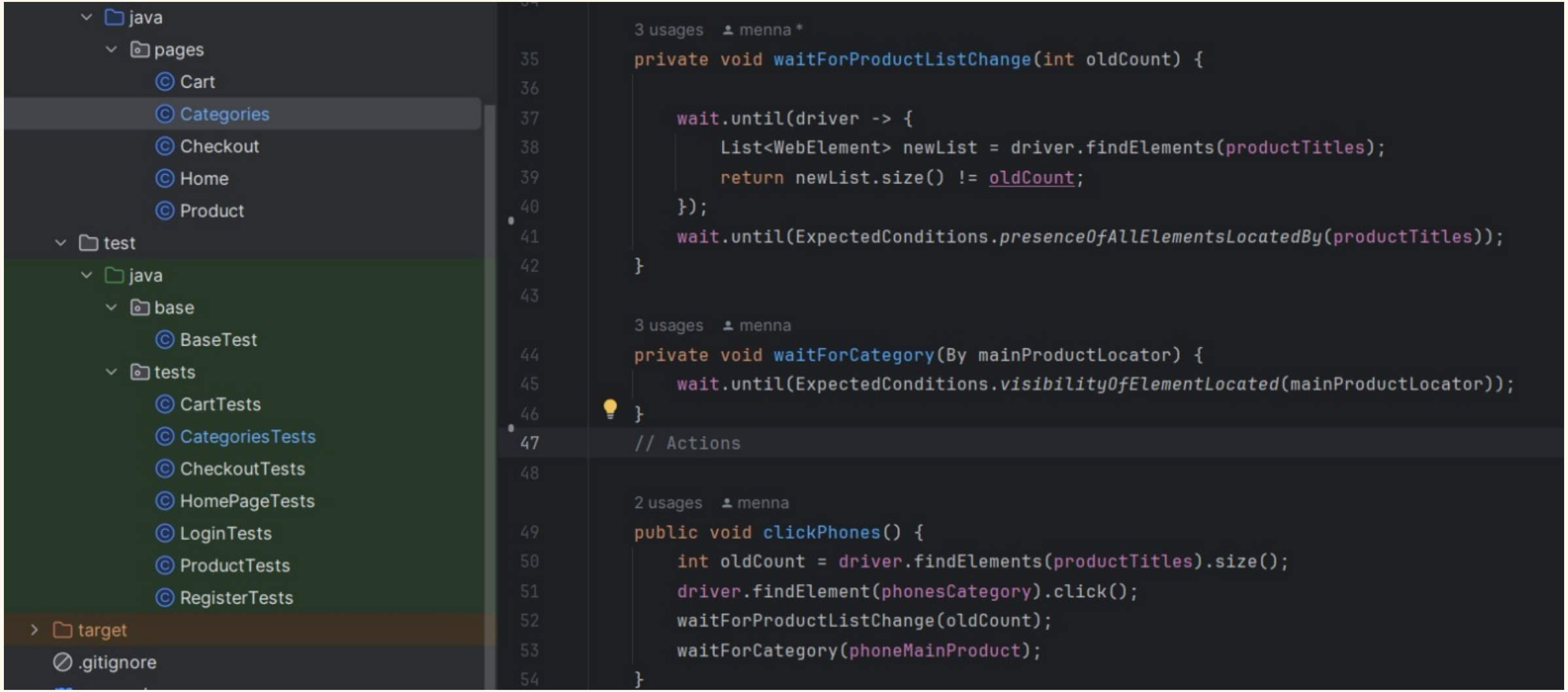
- Home, Product, Categories, Cart and Checkout

The **Tests** were divided into:

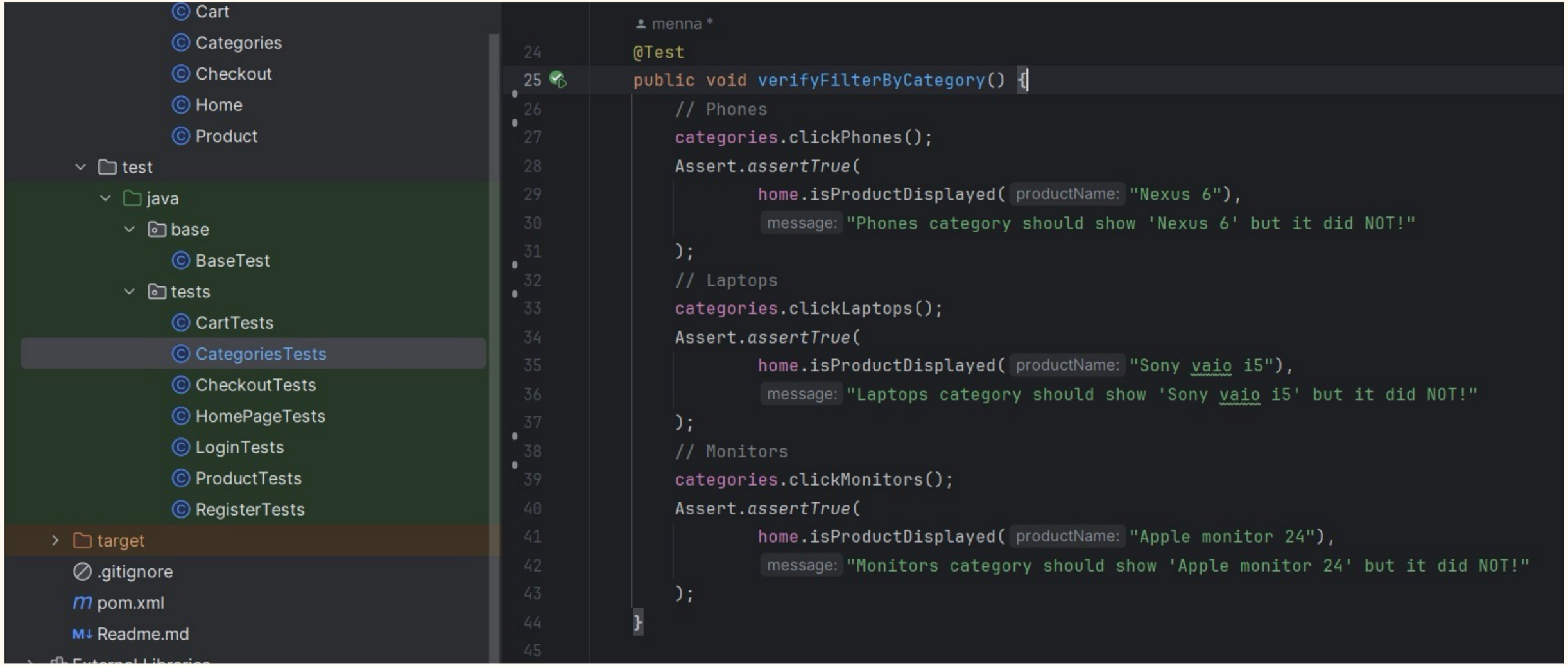
- Register, Login, Homepage, Categories, Product, Cart and Checkout tests



TestNG Page Example



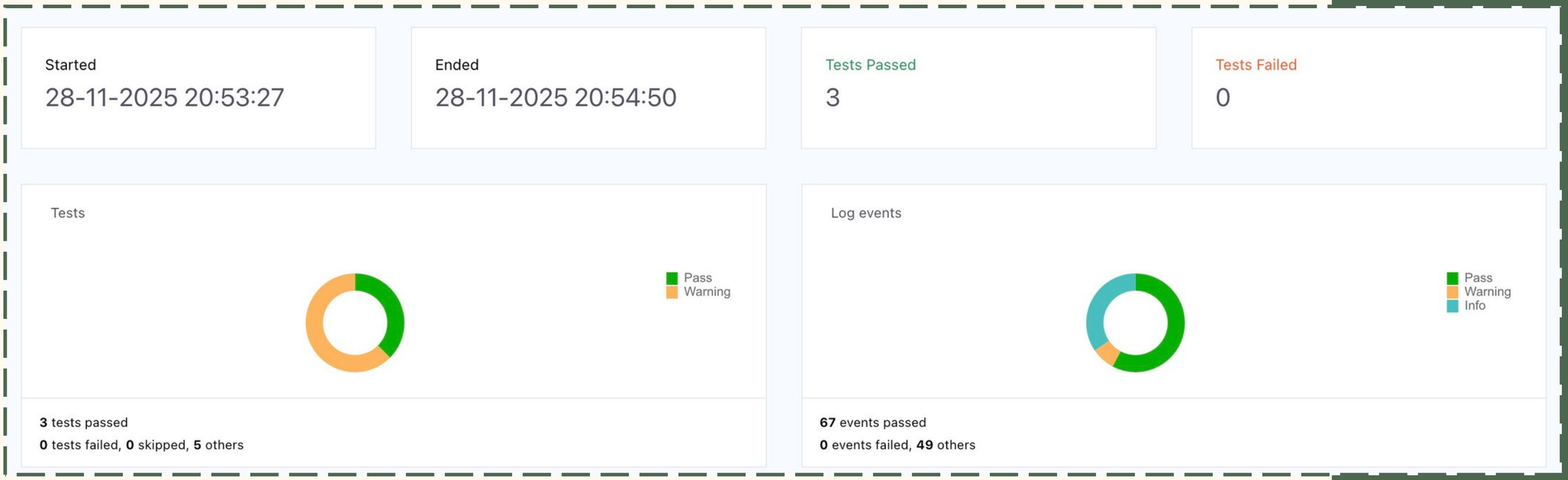
TestNG Test Class Example



Cucumber Hooks

- Implemented **BDD** (Behavior-Driven Development) using Cucumber
- Wrote test scenarios in **Gherkin** for clear, human-readable steps
- Mapped steps to **Java step definitions** for executable automation
- Organized features by modules: **Home, Product, Categories, Cart and Checkout**
- Used **Cucumber + TestNG** for reporting and structured execution
- Enhanced collaboration by aligning tests with real user behaviors

Cucumber Report



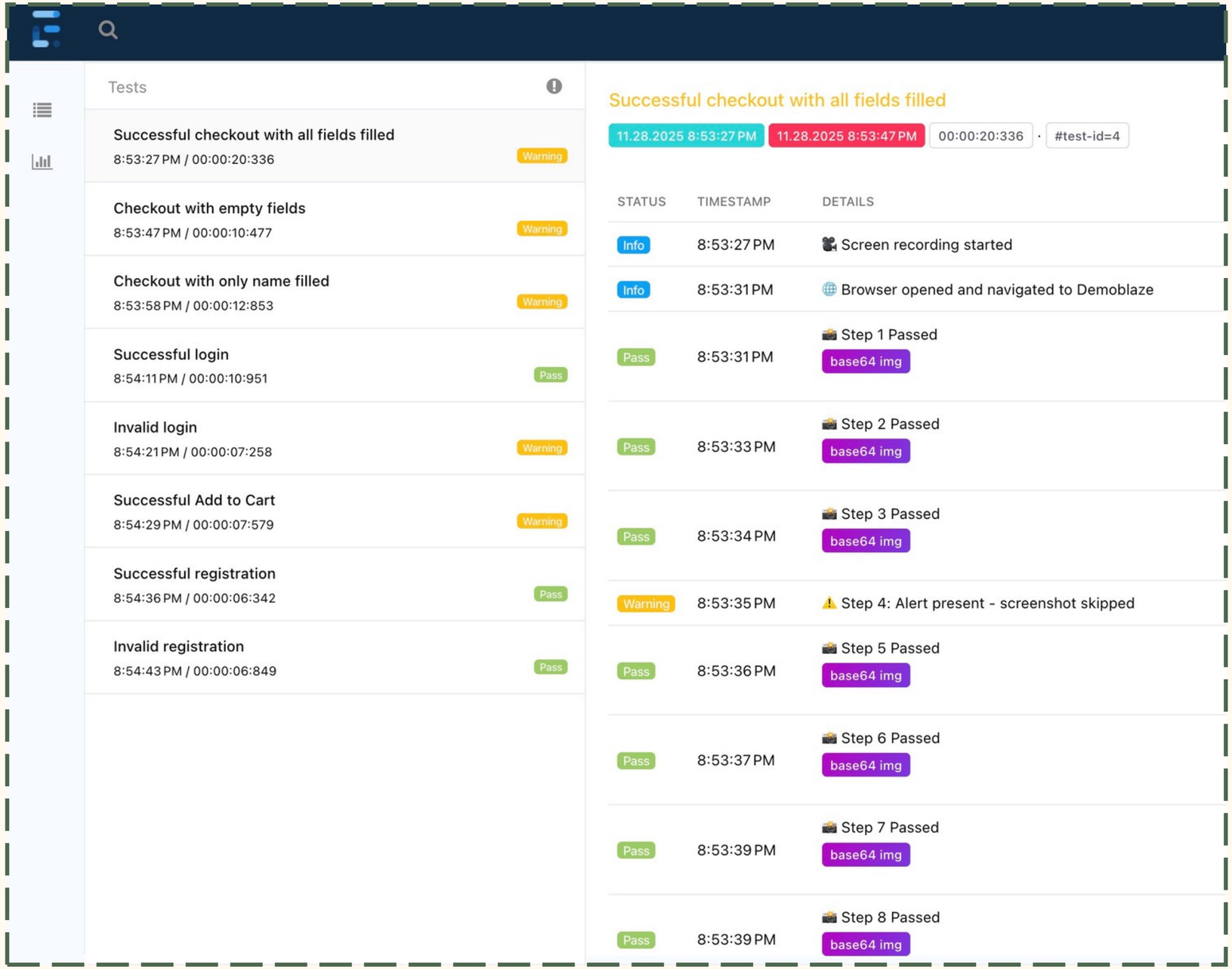
Cucumber Report

<u>Total:</u> <u>8 tests</u>

Passed/Warning: **8**

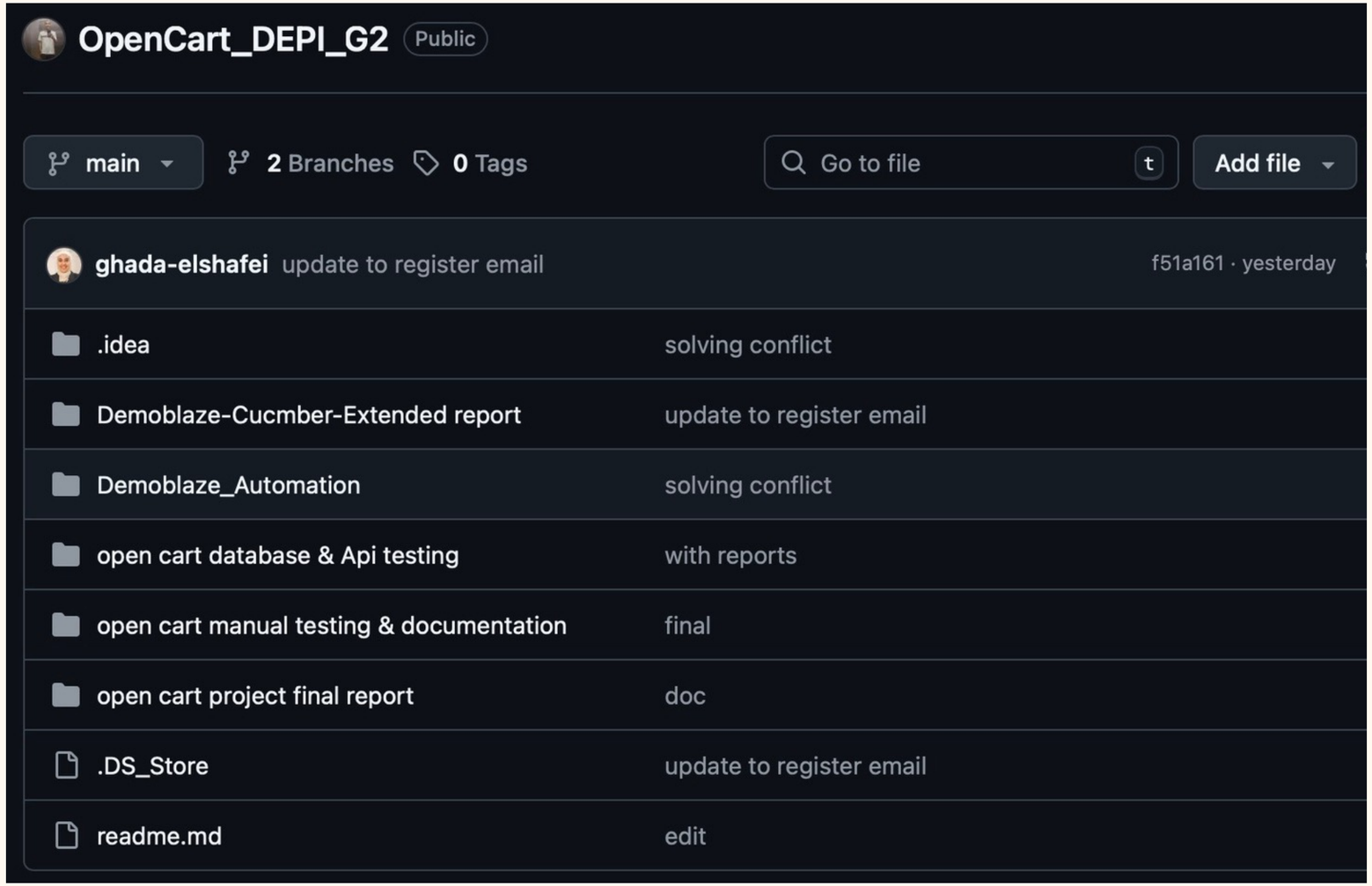
Failed: **0**

The warning was related to errors regarding the skipping of the screenshots



GitHub Collaboration

- **Repo** for source code, TCs, documentation, and reports
- **Collaborated** through version control and branch management
- **Stored** manual TCs, automation scripts, and project artifacts
- **Tracked** updates, commits, and review history



Performance Testing

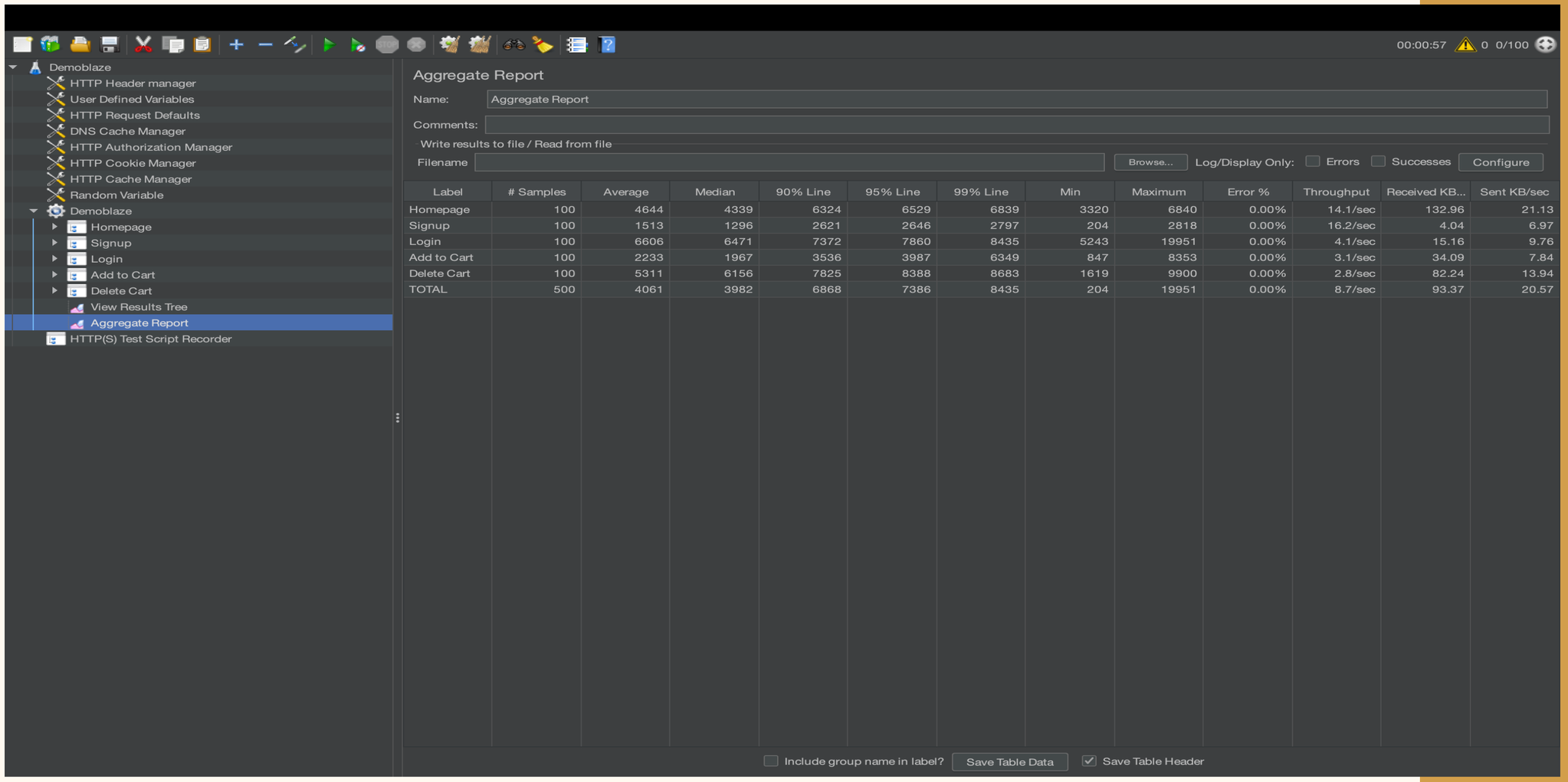
Performance Testing

Demoblaze ECommerce Demo

- Used **Apache JMeter** to assess the system performance
- **Tested key flows:** signup, login, product browsing, add-to-cart, delete cart, checkout (attempted)

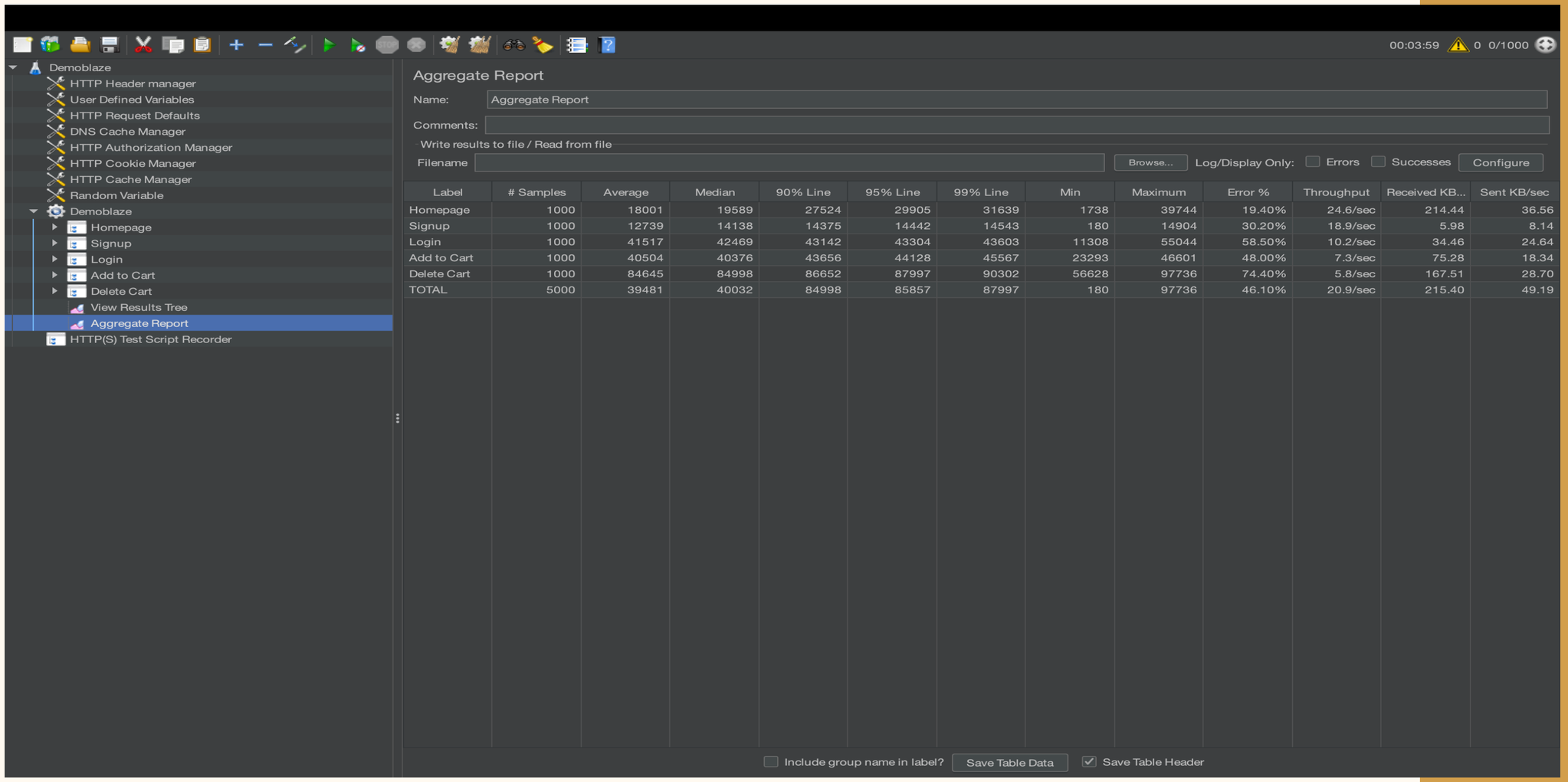
- After identification of the **breakpoint** (970 users), **load** testing using 100 users, and **stress** testing using 1000 users were performed

Aggregate Report – Load Test (Users = 100)



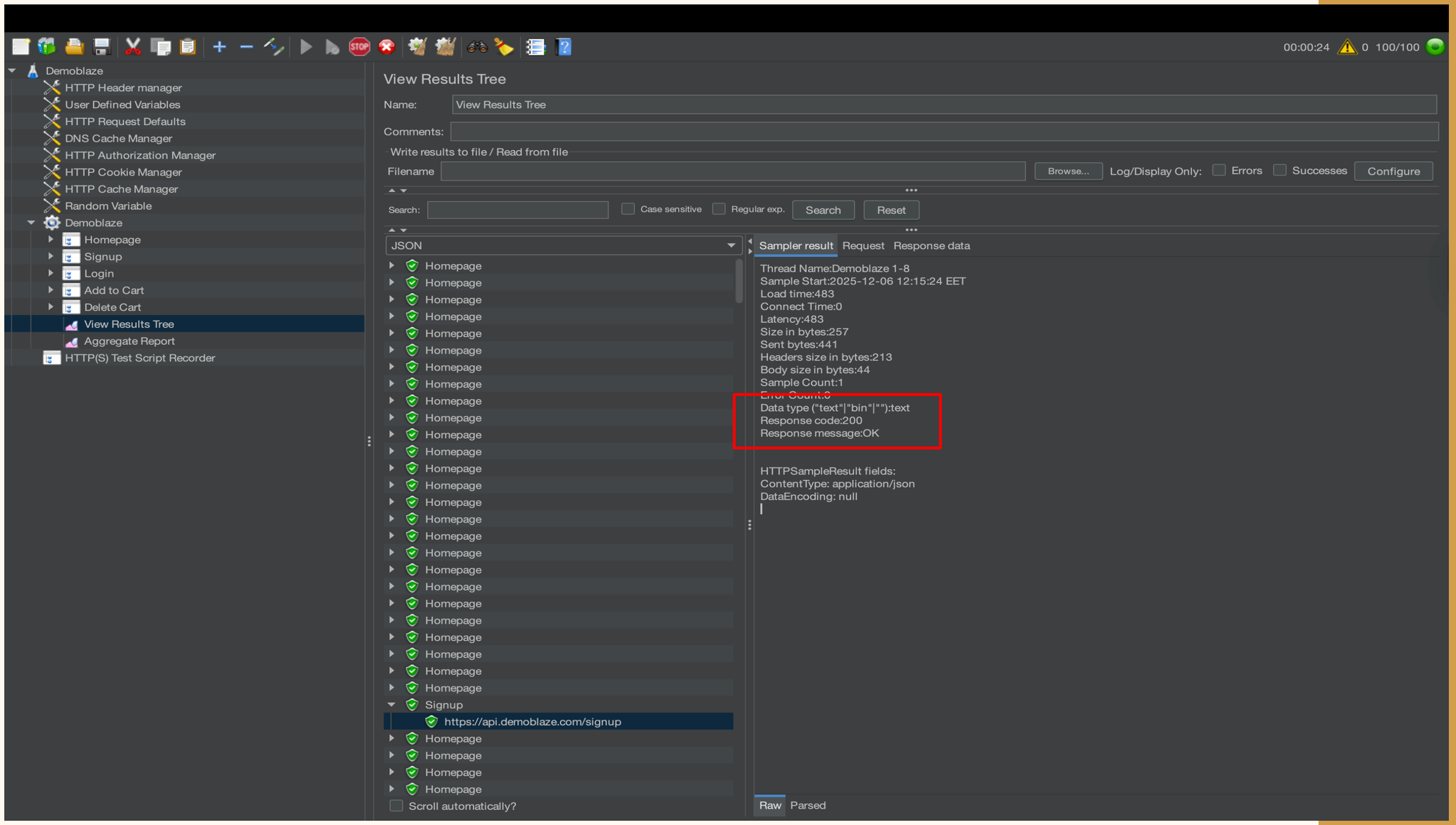
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB...	Sent KB/sec
Homepage	100	4644	4339	6324	6529	6839	3320	6840	0.00%	14.1/sec	132.96	21.13
Signup	100	1513	1296	2621	2646	2797	204	2818	0.00%	16.2/sec	4.04	6.97
Login	100	6606	6471	7372	7860	8435	5243	19951	0.00%	4.1/sec	15.16	9.76
Add to Cart	100	2233	1967	3536	3987	6349	847	8353	0.00%	3.1/sec	34.09	7.84
Delete Cart	100	5311	6156	7825	8388	8683	1619	9900	0.00%	2.8/sec	82.24	13.94
TOTAL	500	4061	3982	6868	7386	8435	204	19951	0.00%	8.7/sec	93.37	20.57

Aggregate Report – Stress Test (Users = 1000)

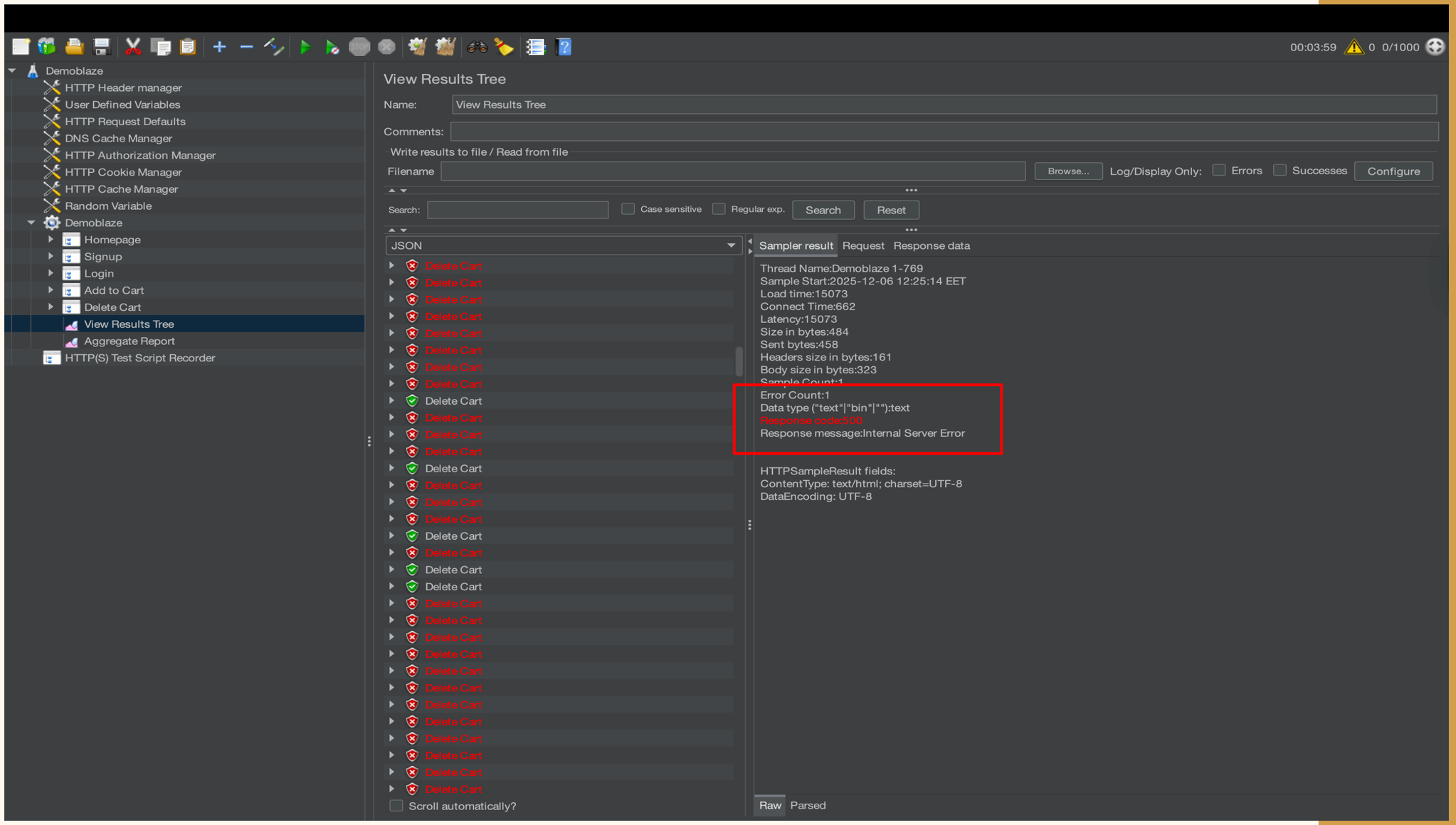


Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB...	Sent KB/sec
Homepage	1000	18001	19589	27524	29905	31639	1738	39744	19.40%	24.6/sec	214.44	36.56
Signup	1000	12739	14138	14375	14442	14543	180	14904	30.20%	18.9/sec	5.98	8.14
Login	1000	41517	42469	43142	43304	43603	11308	55044	58.50%	10.2/sec	34.46	24.64
Add to Cart	1000	40504	40376	43656	44128	45567	23293	46601	48.00%	7.3/sec	75.28	18.34
Delete Cart	1000	84645	84998	86652	87997	90302	56628	97736	74.40%	5.8/sec	167.51	28.70
TOTAL	5000	39481	40032	84998	85857	87997	180	97736	46.10%	20.9/sec	215.40	49.19

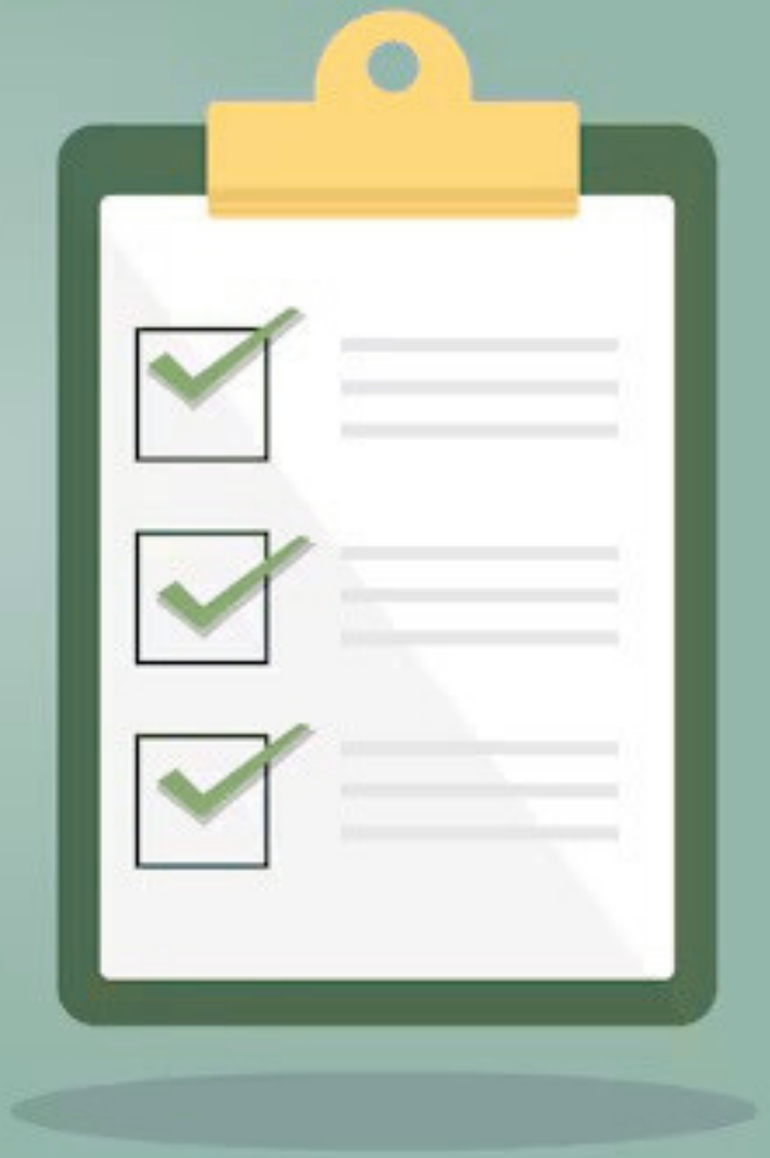
Results Tree (Users = 100)



Results Tree (Users = 1000)



Summary



Summary

Manual Testing (OpenCart)

- Conducted comprehensive functional **exploratory** testing across major modules (registration, login, user account, product browsing, product comparison, product pages, cart, checkout flow, and the admin interface).

Database Testing using MySQL (OpenCart)

- Validated backend data integrity by checking customer records, addresses, orders, and product details.

Summary

API Testing using Postman (GoREST)

- Performed structured API testing on Users, Posts, Comments, and Todos modules using GET, POST, PUT and DELETE methods.

UI Automation Testing using Selenium + TestNG (Demoblaze)

- Built an automation framework to execute repeatable tests on Demoblaze, verifying core functionalities and improving test coverage and reliability.

Summary

BDD Automation with Cucumber (Demoblaze)

- Developed behavior-driven automated tests for critical modules such as Signup, Login, Products, Cart and Checkout, producing human-readable scenarios and detailed reports.

Performance Testing using Apache JMeter (Demoblaze)

- Assessed system behavior by measuring response times, behavior and scalability for key user journeys by performing load and stress tests.

thank you