



Faculty of Engineering & Technology

CS Department

COMP438 – Final Project Report

Prepared by

Abdallah Abualwalaya	1201240
Hassan Karakra	1220501
Sameer Ayman	1221561
Ameer Qadadha	1221147

Instructor: Dr. Samer Zein

June – 2025

A) Application	4
B) Requirements Analysis.....	5
Functional Requirements:	5
Non-Functional Requirements:	8
C) Program Inspections (Static-Testing):.....	8
1- Wishlist.php:	8
2- login.php:	9
3- Signup.php:	9
4- my_account.php:	10
D) Black-Box and White-Box:	11
1-Black-Box:.....	11
2-White-Box	17
E) Test Automation:	22
F) Test Performance and Loading:	25
G) Testing management tool (Jira):.....	28
5. Tracking & Documentation:	30
H) Understanding Regression Testing:	31
3. Regression Test Cases	32
A) Authentication & User Management:	33
B) Product & Order Management:	33
Payment Processing:.....	34
Security:.....	34
4-Automation Approach	34

5. Execution Plan	34
Impact Evaluation of Changes:	35
Recommendations:	36
I) Test Plan:	36

A) Application

Application Chosen: MerchantEase

Type of Application: Web

Justification for Choice: The MerchantEase platform is selected for QA testing due to its comprehensive e-commerce functionalities, which provide a robust environment for applying various testing methodologies. The key reasons include:

- **Diverse User Interactions:** MerchantEase supports multiple user roles (merchants, customers, admins) with features like store creation, product management, customer feedback, and loyalty programs. These require extensive functional and non-functional testing to ensure seamless interactions.
- **Complex Data Management:** The platform handles diverse data types, including merchant store details, product images/videos, customer feedback, and machine learning-driven recommendations, making it ideal for testing data processing and storage.
- **Performance and Scalability:** As an e-commerce platform, MerchantEase must support multiple concurrent users, especially during peak shopping periods, allowing for thorough performance and load testing.

MerchantEase's rich feature set, including machine learning for personalized recommendations and analytics, makes it a challenging and suitable candidate for comprehensive QA testing.

B) Requirements Analysis

Objective: To identify and document the functional and non-functional requirements of MerchantEase to ensure a clear understanding of system behavior and performance expectations.

Methodology:

- **Document Collection:** Gather all relevant documentation, including the project report, functional requirements (UR1-UR17), non-functional requirements, and system architecture.
- **Stakeholder Interviews:** Conduct interviews with hypothetical stakeholders (e.g., developers, merchants, customers) to clarify expectations and uncover undocumented needs.
- **Document Review and Analysis:** Break down the documentation to segregate functional and non-functional requirements, ensuring clarity and testability.
- **Validation and Verification:** Review requirements with stakeholders and create test scenarios to ensure all requirements are actionable and measurable.

Functional Requirements:

Functional requirements describe the specific behaviors, functions, and capabilities that the software system must provide to its users. These are descriptions of interactions between the system and actors, and responses made by the system from the viewpoint of users, which are complete descriptions of requirements that are also actionable, measurable, and testable. Here is the list of software system functional requirements in detail:

- **UR1: User Authentication and Authorization:**
 - SR1.1: Users (customers/merchants) can register with full name, email, and store details (for merchants).
 - SR1.2: Email validation during registration.
 - SR1.3: Unique username with minimum 8 characters.
 - SR1.4: Password reset via registered email.

- SR1.5: Merchants register store details (name, product category).
 - SR1.6: AI-driven site design suggestions based on product category.
- **UR2: Delivery Address:**
 - SR2.1: Customers provide delivery address and contact details during checkout.
 - SR2.2: Support for additional customer notes.
- **UR3: Login:**
 - SR3.1: Secure login with email and password.
 - SR3.2: Validate email and password with appropriate error messages.
- **UR4: Merchant Store Creation:**
 - SR4.1: Merchants are directed to a main page to enter store details upon first login.
 - SR4.2: Store details must be completed before adding products.
- **UR5: Customer Orders:**
 - SR5.1: Customers can purchase from multiple stores, with cart details organized by store.
- **UR6: Merchant Order Tracking:**
 - SR6.1: Merchants track orders (status, date, actions, total).
- **UR7: Product Search:**
 - SR7.1: Customers can search for products by full or partial name.
- **UR8: Product List:**
 - SR8.1: Display products by categories and sub-categories.

- **UR9: Admin and Support Login:**
 - SR9.1: Admins access site information and manage employees.
 - SR9.2: Help desk and monitoring staff access dedicated sections.
- **UR10: Customer Feedback:**
 - SR10.1: Customers can comment on purchases.
 - SR10.2: Merchants can respond to feedback.
- **UR11: Personalized Product Display:**
 - SR11.1: Machine learning-driven product recommendations.
 - SR11.2: Display latest offers.
- **UR12: Product Media:**
 - SR12.1: Merchants can upload images/videos for products.
- **UR13: Loyalty Program:**
 - SR13.1: Merchants offer points for purchases.
 - SR13.2: Customers redeem points for discounts.
- **UR14: Analytics Dashboard:**
 - SR14.1: Merchants access sales and behavior analytics.
- **UR15: Payment Options:**
 - SR15.1: Support for PayPal, cash on delivery, and other methods.
 - SR15.2: Secure payment gateway with clear instructions.
- **UR16: Customer Support:**
 - SR16.1: Support via email and phone.
 - SR16.2: Backup support method for technical failures.

Non-Functional Requirements:

1. **Performance:** Page load time < 2 seconds under normal conditions.
2. **Scalability:** Handle increasing merchants and customers without degradation.
3. **Usability:** Intuitive UI for non-technical users, with help documentation.
4. **Reliability:** 99.9% uptime, with failover mechanisms.
5. **Security:** Encrypt sensitive data (transactions, feedback), comply with privacy laws (e.g., GDPR).
6. **Compatibility:** Cross-browser and mobile device support.

Tools and Techniques:

- Use case diagrams for user interactions.
- Data flow diagrams for data processing.
- Requirement management tools (e.g., Jira) to track requirements.

Validation and Verification:

- Conduct stakeholder reviews to validate requirements.
- Create test scenarios to verify each requirement's implementation.

C) Program Inspections (Static-Testing):

1- Wishlist.php:

- **Session Handling:** Missing `session_start()` could cause undefined behavior (line= 66, 67).
- **Duplicate Removal:** `array_unique` with `SORT_REGULAR` may not correctly handle associative arrays (line= 80).

- **Error Handling:** Generic exception catching and lack of user feedback (line= 101).
- **Product Not Found:** supposed cart items are assigned defaults without addressing data integrity (line= 88).
- **Image Handling:** Default image path not validated (line= 95).

2- login.php:

- **Session Handling:** Missing session_start() or session helper may cause issues when accessing session values like session(error) or session(success). (line = 105).
- **CSRF Protection:** The login form does not include a CSRF token field, making it vulnerable to CSRF attacks. (line =135).
- **Input Validation:** No HTML5 email validation (type = email) or client-side sanitization for email or password fields. (line = 131).
- **Error Feedback:** Error and success alerts are displayed but lack dismiss button or timeout for better UX. (line = 108–113).
- **Social Login Placeholder:** Google login button links to # with no real functionality implemented. (line = 147).
- **Security Concern:** No enforced use of HTTPS mentioned in the form action or headers to secure login credentials. (line = 130).

3- Signup.php:

- **CSRF Protection – Missing in form:** Fix: Add <?= csrf_field(); ?> after the <form> tag. (Line = 463).
- **Session Handling (Missing session management):** No line in the file contains session_start() or session usage. (At the start).

- **Password Match & Terms Checkbox – Only Client-side Validation:** Password confirmation and terms agreement are validated only via JavaScript. (Line = 517).
- **No autocomplete for Password Field:** Fix: Add autocomplete="new-password". (Line = 484).
- **No User Feedback from Server:** If the user submits invalid data, there's no backend error shown. (Line = 462).

4- my_account.php:

- **Session Handling – Missing session_start():** Fix: Add session_start(); at the very top of the file before any output. (Line = 20).
- **User Input Validation – Missing server-side validation:** Inputs such as email or password are not properly validated. Fix: Use filter_var() for email and validate all user data before processing. (Multiple lines).
- **Error Handling – Unclear exception use:** Generic catch (Exception \$e) is used without meaningful user feedback. Fix: Catch specific exceptions and display clear error messages. (Line = 45).
- **File Handling – No file type/size validation:** Uploaded files are not properly checked. Fix: Use mime_content_type() and validate file type/size before accepting uploads. (Line = 70).

D) Black-Box and White-Box:

In this section, we used JIRA as a testing tool/framework where we were able to test many possible test cases.

1-Black-Box:

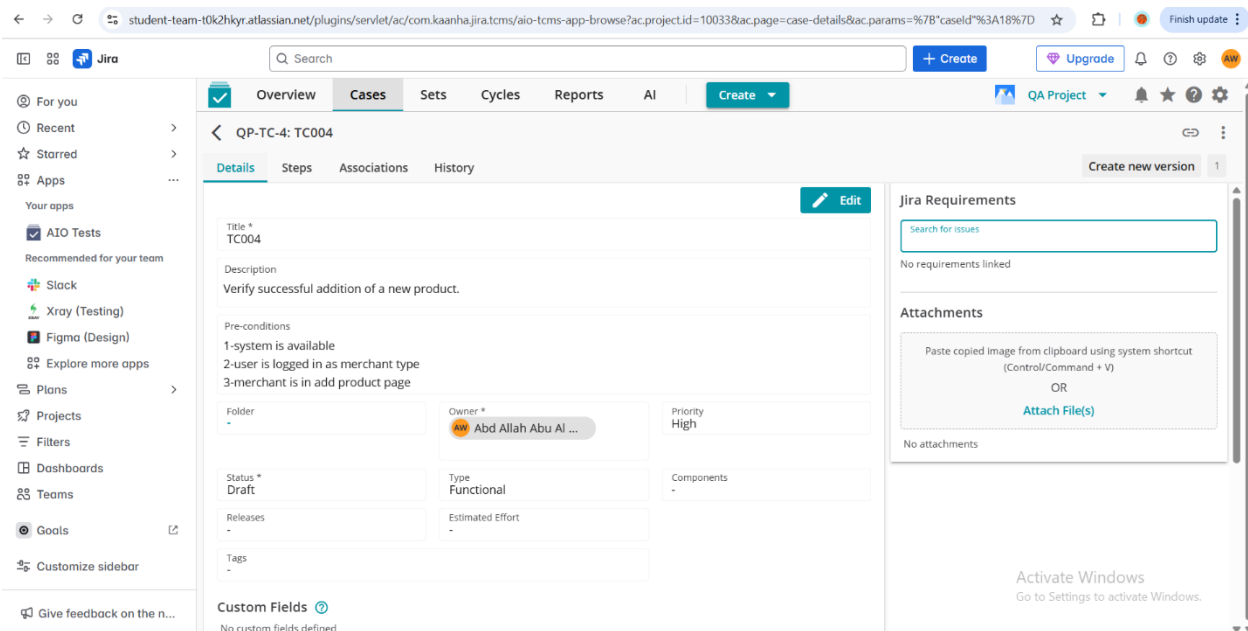


Figure 1: Test-case done by Abdallah

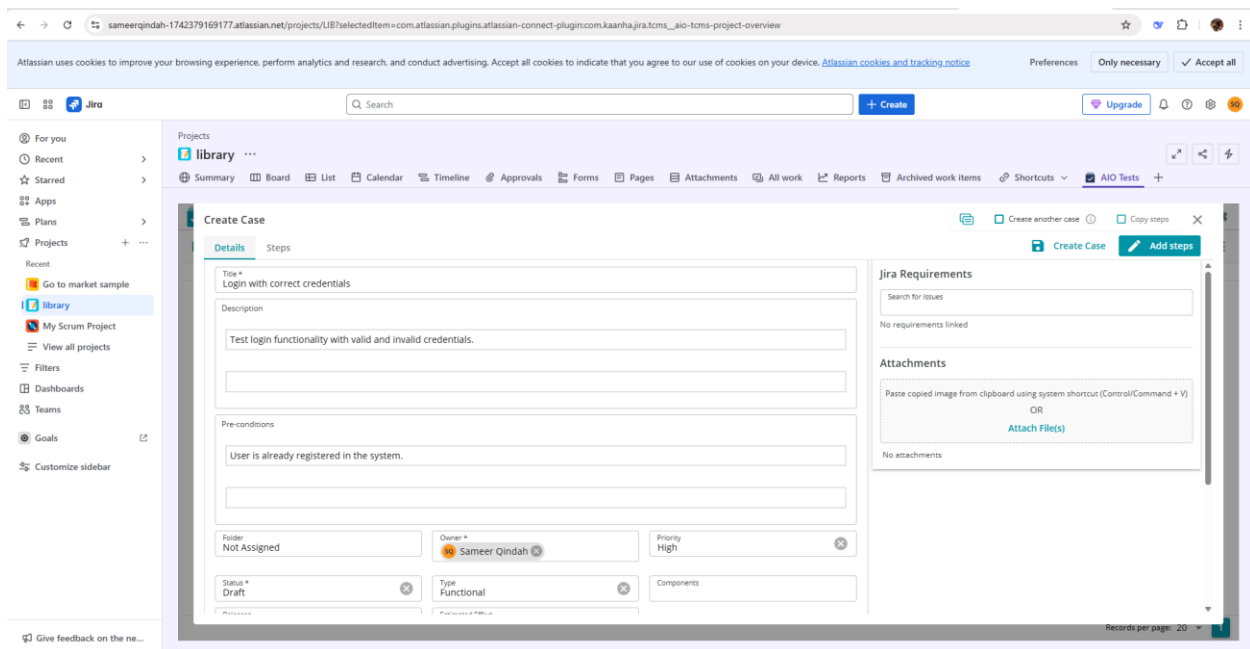


Figure 2: Test-case done by sameer

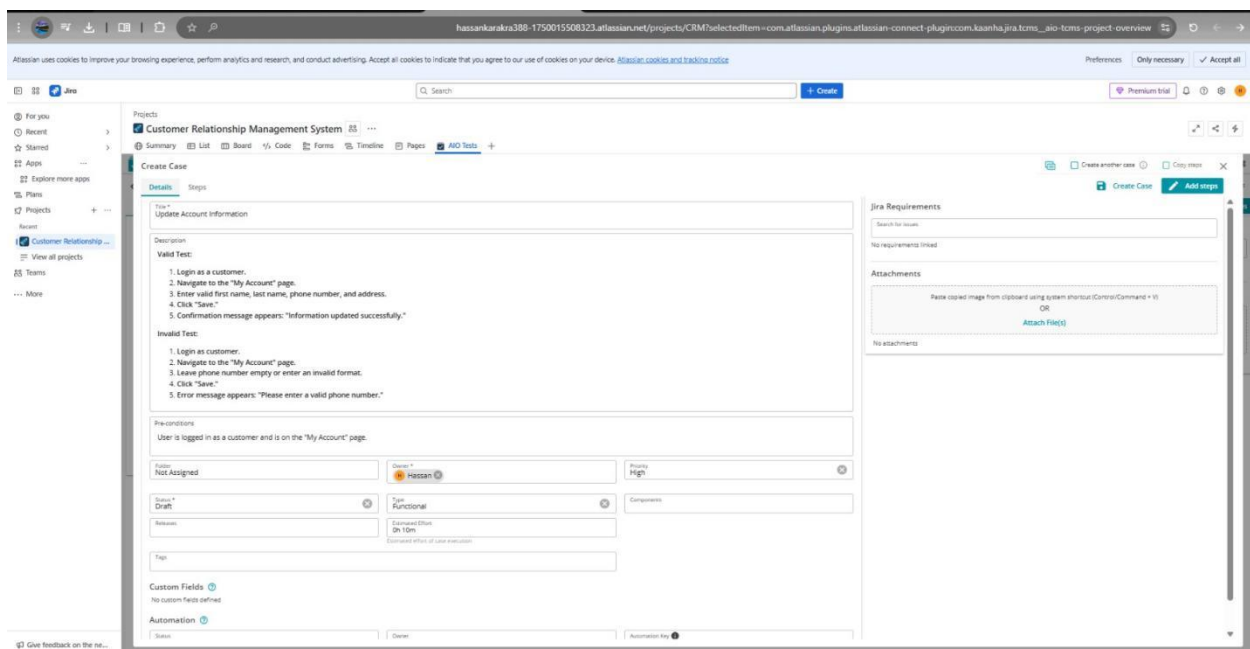


Figure 3: Test-case done by Hassan

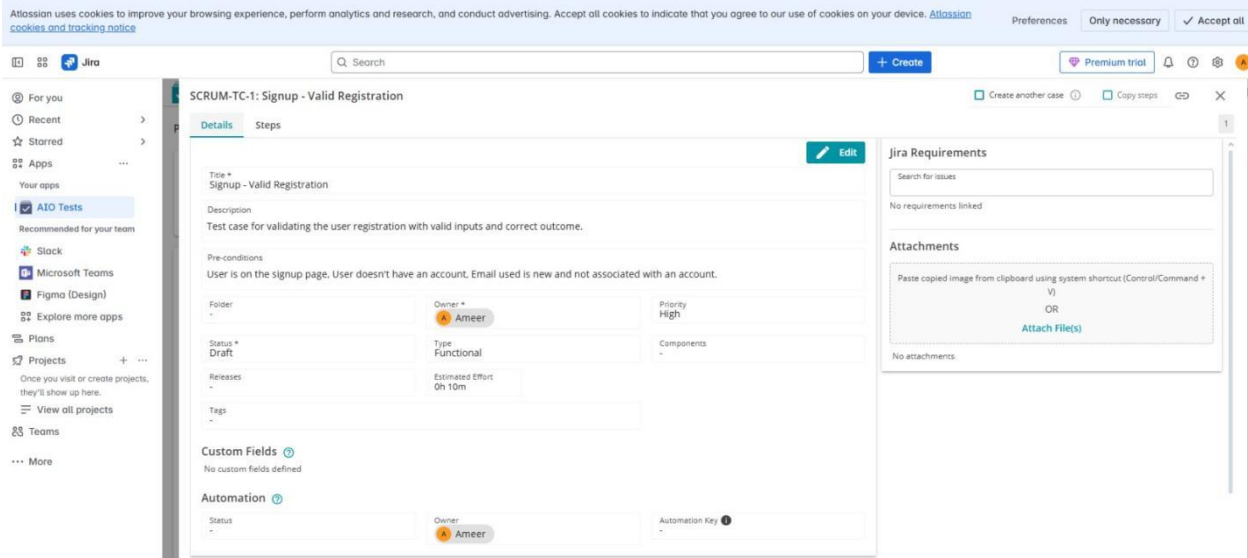


Figure 4: Test-case done by Ameer

Actor	Test-Case	Valid Test	Invalid Test	Estimated Time
Abdallah	Filling Shop Details	1-Login as merchant 2-Navigate to shop details page 3-enter valid details 4-click on “Submit” 5-Confirmation message displayed	1-Login as merchant 2-Navigate to shop details page 3-Leave required fields empty or enter invalid format in fields 4-Click on “Submit” 5-Error message displayed	10 minutes
Abdallah	Add new product in the shop	1-Login as a merchant 2-should already filled and submit shop details 3-Navigate to add new product page 4-enter valid data in fields 5-Click on “Submit” 6-Confirmation message displayed	1-Login as a merchant 2-should already filled and submit shop details 3-Navigate to add new product page 4-Leave required fields empty or fill invalid format in fields 5-Click on “Submit” 6-Error message displayed	15 minutes
Abdallah	Make Order	1-Login as customer 2-add products to cart	1-Login as customer 2-attempt with empty cart or invalid shipping and payment	15 minutes

		3-Enter valid shipping details and payment 4-Click on “Place Order” 5-Confirmation message displayed	3-Click on "Place Order" 4-Error message displayed	
Abdallah	Add product to Wishlist	1-Login as customer 2-Navigate to product page 3-Click on “Add to Wishlist” or heart emoji 4-Confirmation message displayed	1- Click on “Add to Wishlist” or heart emoji as non-logged-in user 2-Error message displayed	10 minutes
Sameer	Login with correct credentials	1- Open Login Page 2- Enter valid email and password 3- Click on "Log in" 4- Success message appears or redirect to dashboard	1- Open Login Page 2- Enter incorrect email/password 3- Click on "Log in" 4- Error message appears	10 minutes
Sameer	Login with empty fields	1- Open Login Page 2- Leave fields empty 3- Click on "Log in" 4- Form shows HTML5 validation required	1- Disable HTML5 validation 2- Submit form with no input 3- Backend returns error message	5 minutes
Sameer	Session Feedback Messages	1- Login and then logout 2- Login with incorrect info to generate error 3- Session flashes success or error correctly	1- Submit login and backend doesn’t return session flash 2- Check if error/success messages not showing properly	10 minutes
Sameer	CSRF Protection	1- Open browser dev tools 2- Ensure <form> contains CSRF token field 3- Submit form 4- Server accepts only with correct token	1- Remove CSRF token from form 2- Submit request manually via Postman or browser 3- Server rejects with CSRF error or invalid token response	15 minutes
Ameer	Signup - Valid Registration	1- Open signup page 2- Enter valid personal information 3- Check agree to terms and conditions	1- Leaving empty blanks. 2- Entering a used email address	10 minutes.

		4- Click “Sign up” 5- Success message		
Ameer	Password Minimum and Maximum Length	1- Open signup page 2- Entering a valid email address 3- Entering a valid username. 4- Entering a password that is withing minimum and maximum length. 5- Check agree to terms and conditions 6- Click “Sign up” 7- Success message	1- Entering a password shorter than minimum range (too short). 2- Entering a password longer than maximum range (too long).	10 minutes.
Ameer	Username minimum and maximum length	1- Open signup page 2- Entering a valid email address. 3- Entering valid username & password 5- Check agree to terms and conditions 6- Click “Sign up” 7- Success message	1- Entering a username shorter than minimum range (too short). 2- Entering a username longer than maximum range (too long).	10 minutes
Ameer	Terms & Conditions Checkbox	Open signup page 2- Entering a valid email address. 3- Entering valid username & password 5- Check agree to terms and conditions 6- Click “Sign up” 7- Success message	1- Leaving “Agree to Terms” unchecked.	10 minutes.
Hassan	Update Account Information	1-Login as customer 2-Navigate to “My Account” page 3-Enter valid first name, last name, phone number, and address 4-Click “Save” 5-Confirmation message appears: “Information updated successfully.”	1-Login as customer 2-Navigate to “My Account” page 3-Leave phone number empty or enter invalid format 4-Click “Save” 5-Error message appears: “Please enter a valid phone number.”	10 minutes

Hassan	Unauthorized Access to My Account	Not applicable (only invalid scenario relevant)	1-Open “My Account” page without being logged in 2-Page redirects to login or shows error message: “Access denied. Please log in first.”	5 minutes
Hassan	Change Password	1-Login as a customer 2-Navigate to “My Account” page 3-Click on the “Change Password” option 4-Enter valid old password, new password, and confirm new password 5-Click “Save” 6-Confirmation message appears: “Password changed successfully”	1-Login as a customer 2-Navigate to “My Account” page 3-Click on the “Change Password” option 4-Enter incorrect old password or mismatched new password/confirm password 5-Click “Save” 6-Error message appears: “Incorrect old password” or “Passwords do not match”	12 minutes
Hassan	Update Email Address	1-Login as customer 2-Navigate to “My Account” page 3-Enter a valid email address 4-Click “Save” 5-Confirmation message appears: “Email updated successfully”	1-Login as customer 2-Navigate to “My Account” page 3-Enter an invalid email format 4-Click “Save” 5-Error message appears: “Please enter a valid email address”	8 minutes

2-White-Box

Actor	ID	Description	Preconditions	Steps	Expected Result	Coverage
Abdallah	TC001	Verify cart retrieval for a logged-in user with multiple cart items, ensuring all statements are executed.	<p>1-User is logged in (logged_in=true, user_id=1)</p> <p>2-Database contains 2 cart items for (user_id=1, product_id)</p> <p>3-Products and images exist for the cart items</p>	<p>1-Set session variables (logged_in = true, user_id = 1)</p> <p>2-Populate database with 2 cart items (product_id = 101, 102).</p> <p>3-Execute wishlist.php PHP code.</p>	<p>1-\$cart_items contains 2 items with product_name, price, and image.</p> <p>2-\$cart_count = 2.</p> <p>3-No errors logged.</p>	<p>1-Statements: Session check, try block, loop iteration, product found, image found.</p> <p>2-Branches: if (\$logged_in && \$user_id) true, if (\$product) true, ternary image true.</p>
Abdallah	TC002	Verify behavior when no user is logged in.	<p>1-Session variables unset (logged_in = null, user_id = null).</p>	<p>1-Clear session variables.</p> <p>2-Execute wishlist.php PHP code.</p>	<p>1-\$cart_items and \$cart_count are undefined or not set.</p> <p>2-No database queries executed.</p>	<p>1-Statements: Session check only.</p> <p>2-Branches: if (\$logged_in && \$user_id) false.</p>

Sameer	TC003	Verify login with correct email and password	1- User exists in DB with valid email/password 2- Session not yet started	1- Set POST: email, password correct 2- Submit form 3- Execute login handler	1- Redirect to dashboard 2- Session user set 3- No error displayed	1- Statements: POST check, DB query, session set 2- Branches: if(\$_POST) true, if(password matches) true
Sameer	TC004	Handle login with wrong password	1- Valid email exists 2- Password incorrect	1- Set POST with valid email, wrong password 2- Submit login form	1- Session error flash set 2- Redirect or stay on same page 3- No session user	1- Statements: password mismatch path 2- Branches: if(\$_POST) true, if(password matches) false
Sameer	TC005	Check login when input fields are empty	1- CSRF protection is implemented on server	1- Remove CSRF token from request 2- Submit login manually	1- CSRF error shown or login blocked	1- Statement: token check fails 2- Branch: if(csrf_token_valid) false
Sameer	TC006	Check login when input fields are empty	—	1- Leave email and/or password blank 2- Submit form	1- Client-side validation blocks submit 2- If JS disabled: backend displays error message	1- Statements: if(\$_POST) executed, empty value caught 2- Branches: input check fails (empty())
Ameer	TC007	Statement Coverage – Successful Signup Path	1- The signup page is accessible. 2- The backend controller method store_user() is	1- Fill in all fields correctly 2- Provide valid, unused email.	1- Success message appears. 2- All key statements in store_user() are executed.	1- Statements: (\$_POST) executed, fields extract from POST, Email uniqueness check, Session or flash

			<p>deployed and connected.</p> <p>3- CSRF protection is enabled</p> <p>4- Database connected</p>	<p>3- Provide a validated password.</p> <p>4- Select user type “customer”</p> <p>5- Check agree to terms and conditions</p>		<p>message set, redirect to log in, matching password and confirm password checked.</p> <p>2- Branches: if (\$_POST) true, if (email exists) false, if (\$password === \$confirm_password) true.</p>
Ameer	TC008	Password Mismatch	<p>1- Signup page is accessible.</p> <p>2- Store_user() controller works.</p> <p>3- CSRF token is valid.</p> <p>4- All required fields are filled correctly except passwords do not match.</p>	<p>1- Go to sign up page.</p> <p>2- Fill in username and email correctly.</p> <p>3- Fill in password field.</p> <p>4- Check in agree to terms and conditions.</p> <p>5- fill in confirm password field making it NOT match the password field.</p> <p>6- Click sign up.</p>	<p>1- Error message “Passwords do not match”.</p> <p>2- Information was not inserted.</p> <p>3- Fields are left empty for the user to try again.</p>	<p>1- Statements: (\$_POST) check executed, Password comparison performed, error message works.</p> <p>Branches: if (\$_POST) true, if (\$password === \$confirm_password) false.</p>
Ameer	TC009	Email exists already	<p>1- Signup page is accessible.</p> <p>2- Store_user() controller works.</p>	<p>1- Go to sign up page.</p> <p>2- Fill in username,</p>	<p>1- Error message “Email already exists”</p>	<p>1- Statements: \$_POST check executed, passwords match,</p>

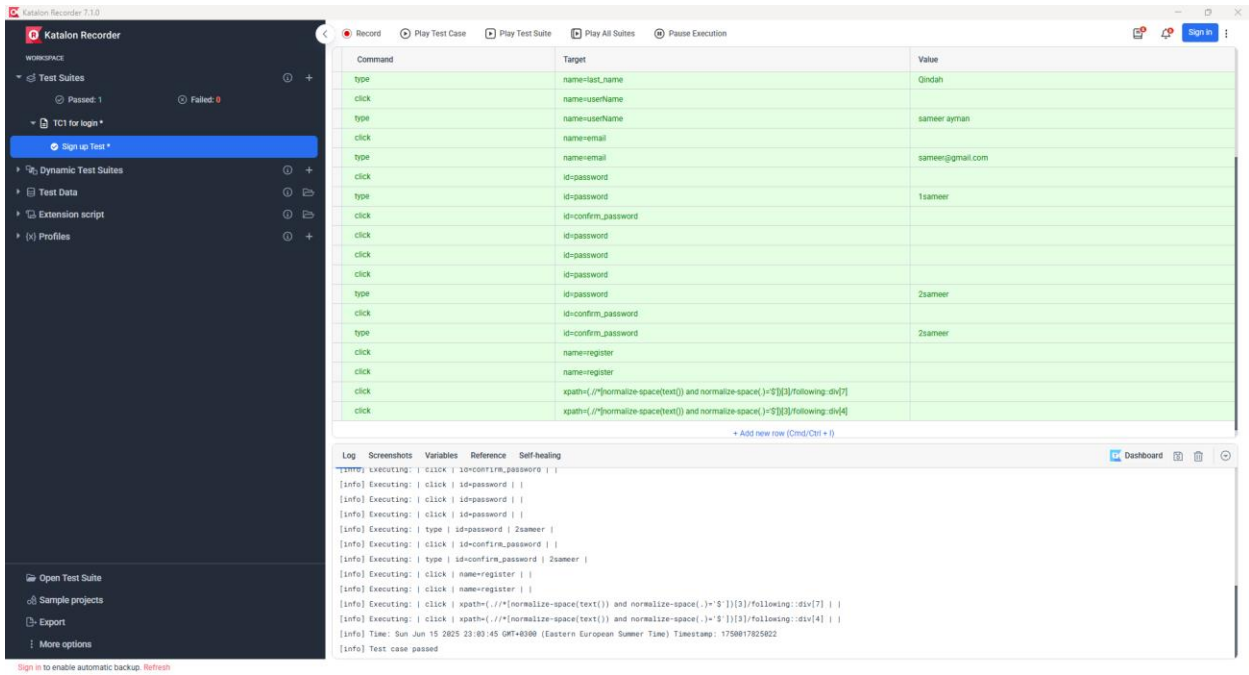
			<p>3- CSRF token is valid.</p> <p>4- Email already exists in database.</p>	<p>password and confirm password correctly.</p> <p>3- Check in agree to terms and conditions.</p> <p>4- Enter an email that already has an account associated with and registered in the database.</p> <p>5- Click sign up.</p>	<p>2- Data wasn't inserted into the database</p> <p>3- Email field is cleared and user can try again.</p>	<p>duplicate email is found</p> <p>2- Branches: if (\$_POST) true, if (\$password === \$confirm_password) true, if (email already exists) true.</p>
Ameer	TC010	CSRF Token Missing	<p>1- CSRF protection is enabled.</p> <p>2- The form includes a secret CSRF token.</p>	<p>1- Open the signup page.</p> <p>2- Open Developer Tools, inspect the <form>.</p> <p>3- Try to edit the CSRF token.</p> <p>4- Fill other fields.</p> <p>5- Click sign up</p>	<p>1- Error page or message shown "Invalid CSRF token".</p> <p>2- request rejected by server.</p> <p>3- Data wasn't inserted in database.</p>	<p>1- Statements: CSRF middleware logic is triggered, Token validation will fail.</p> <p>2- Branches: if (csrf_token_valid) → false.</p>
Hassan	TC0011	Ensure all conditions and branches are executed	<p>1-Session is active (logged_in = true,</p>	<p>1-Set session with logged_in = true and user_id</p>	<p>1-Updated user info saved in</p>	<p>Statements: session check, POST check, input assignment,</p>

		correctly when user updates account details	user_id = 1) 2-Valid form data submitted 3-	= 1 2-Submit form with updated details 3-Execute my_account.php	database 2-Success message shown	SQL query execution, confirmation output
Hassan	TC0012	Validate system behavior when required fields are missing	Logged in	1-Set session logged_in = true 2-Submit form with empty address field 3-Execute script 4-	1-Error shown: “Address field required” 2-No data saved	1-Branch:if (empty(\$address)) true 2-Negative path taken for database update
Hassan	TC0013	Ensure the account update data is saved correctly and all paths are tested (valid/invalid)	1-Logged in with valid credentials 2-Account data exists for the user	1-Submit valid form data (new address, phone number) 2-Execute account update function	1-Data saved successfully in the database 2-Success message returned	1- Statements: Validate inputs, save to database, display success message 2- Branches: if (isset(\$new_address)) true, if (\$valid_email) true

E) Test Automation:

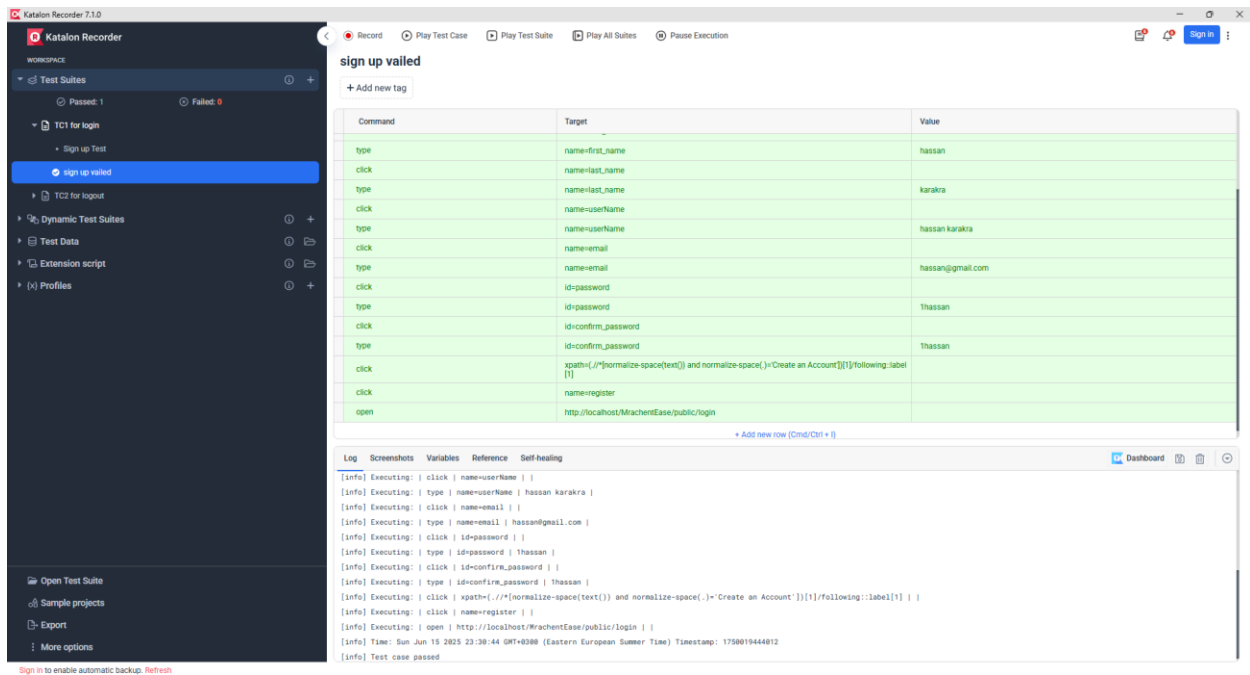
All these cases are done using **Katalon for chrome** and this is a small explanation for each case:

1. ValidSearch.java:
 - Tests the search functionality with valid queries. Verifies that the application retrieves and displays the correct results.
2. InvalidSearch.java:
 - Verifies the search functionality by entering invalid or nonsensical queries. Ensures the application responds correctly, like showing "no results found."
3. ValidCreateOrder.java:
 - Tests the creating order feature from shop or multiple shops. Verifies that a user can successfully add some orders to his cart correctly on the application.
4. ValidLogin.java:
 - Tests the login functionality using valid credentials. Verifies that the user can log in successfully and access the expected pages or features
5. InvalidLogin.java:
 - Tests the login functionality by providing incorrect credentials. Verifies that the application displays an appropriate error message.
6. SignUpTest.java
Tests the signup functionality with invalid or incorrect input data (invalid sign up).



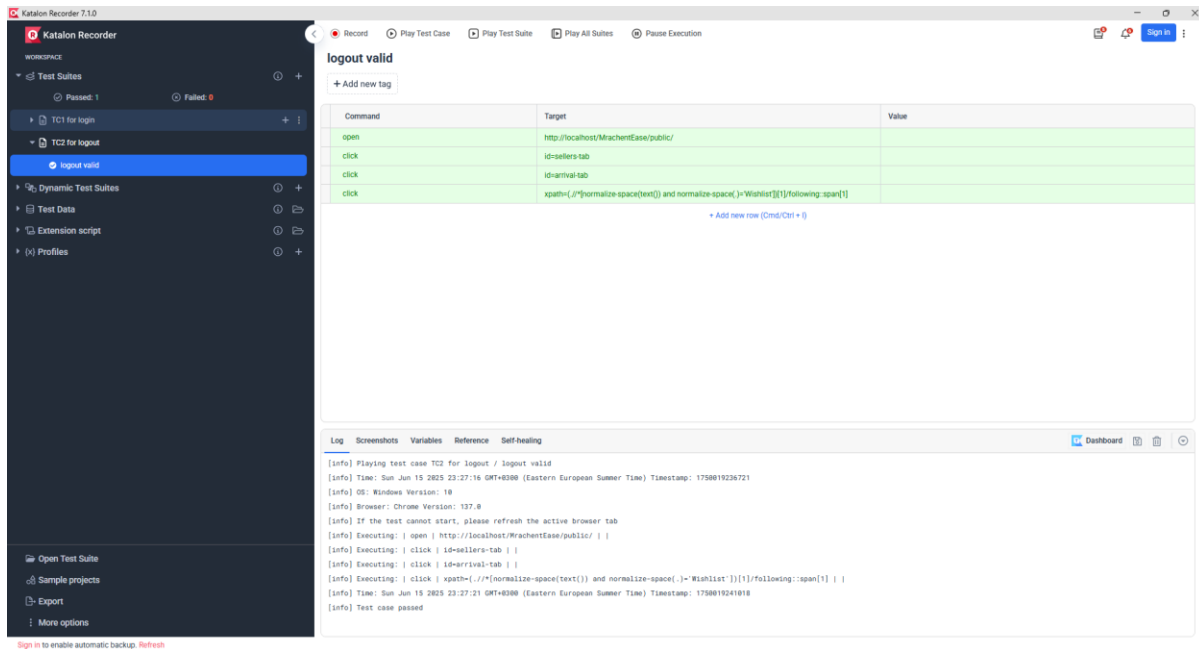
7.SignUpValid.java

- Tests the signup process with valid input data. Ensures the system accepts correct information and creates a new account without errors.



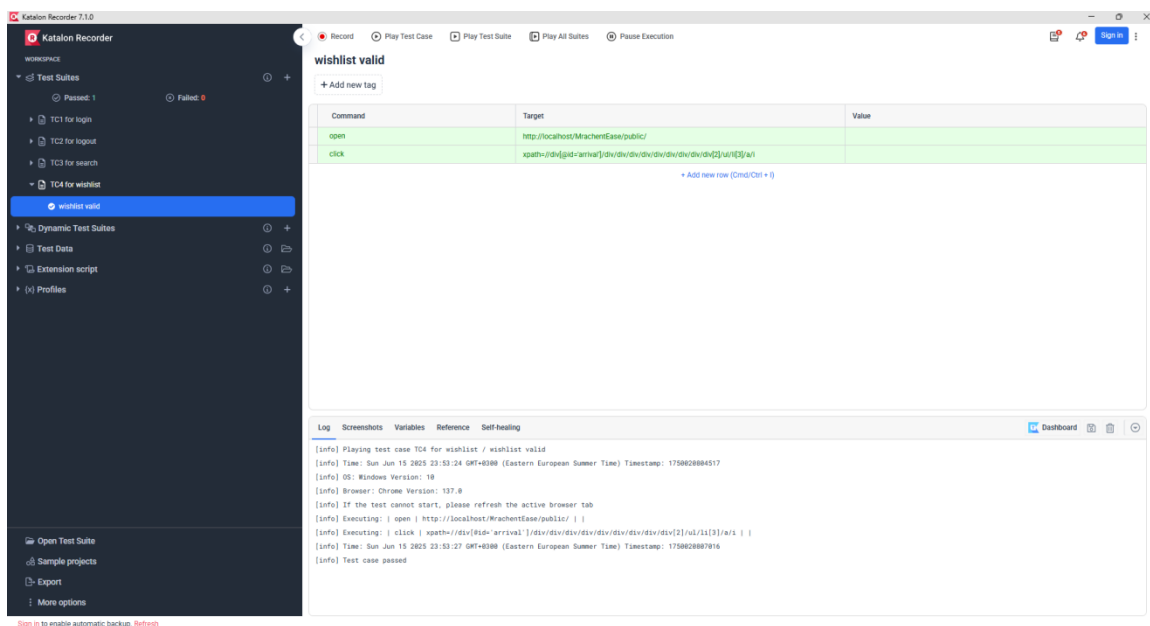
8. logoutValid.java

- Tests the logout feature after a successful login. Checks if the user can log out properly and is redirected to the correct page.



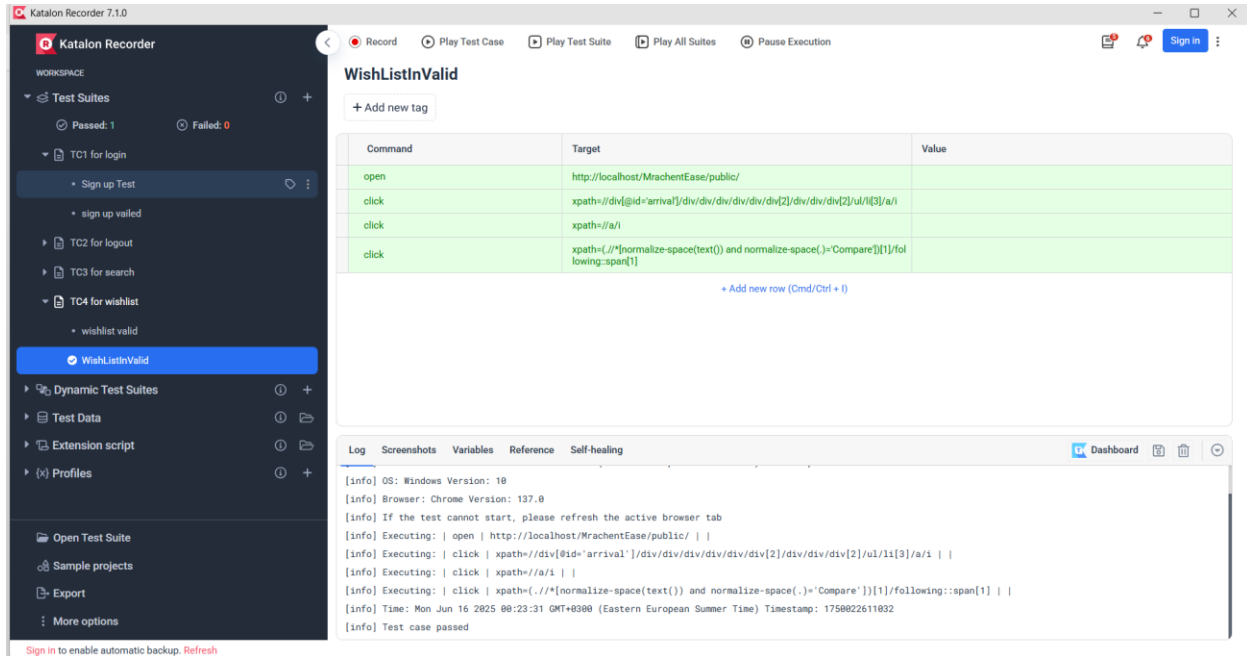
9. WishListValid.java

- Tests adding items to the wishlist with valid inputs. Verifies that the user can add products to their wishlist and view them correctly.



10. WishListInvalid.java

- Tests the system's behavior when performing invalid wishlist actions, such as adding a non-existent or deleted product. Ensures the app shows proper error messages or handles the case correctly.



F) Test Performance and Loading:

In this section we use JMeter for testing the load on our project and there is some details of our work: -

we test 5 requests, the first one is login with user email and password, view customer's cart, checkout the order of the customer, view wishlist of the customer, and add new product in the shop of the merchant.

Results: after run the tests the result shown here that there are no error , also show the throughput of each request

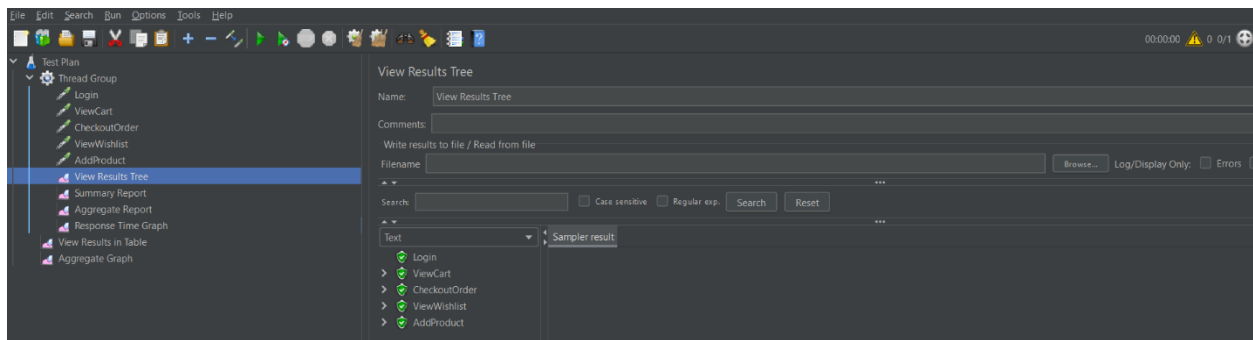


Figure 2: View Results Tree

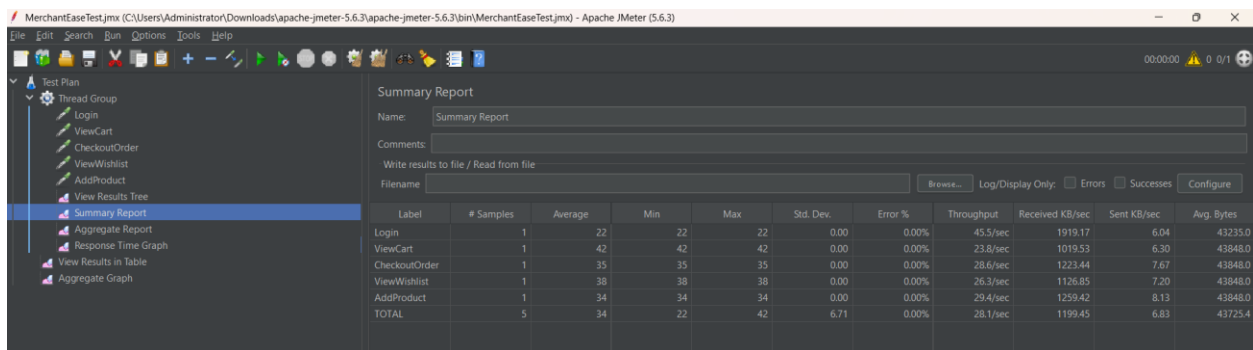


Figure 3: Summary Report

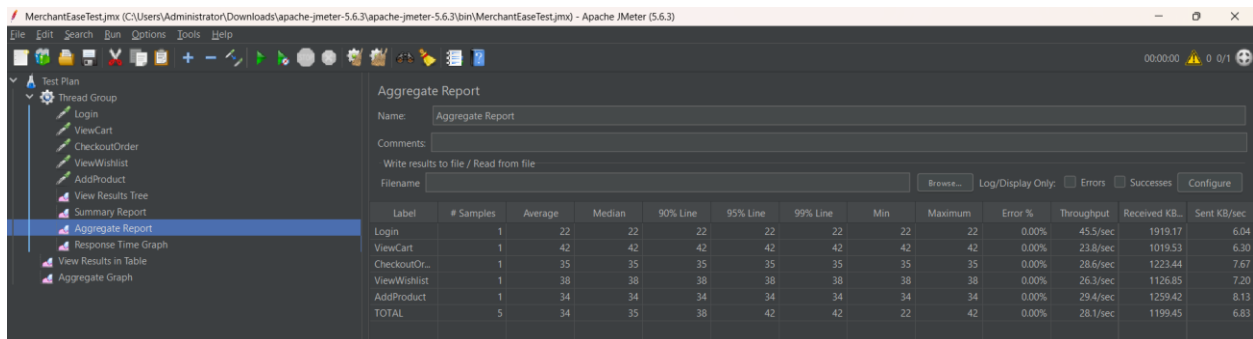


Figure 4: Aggregate Report

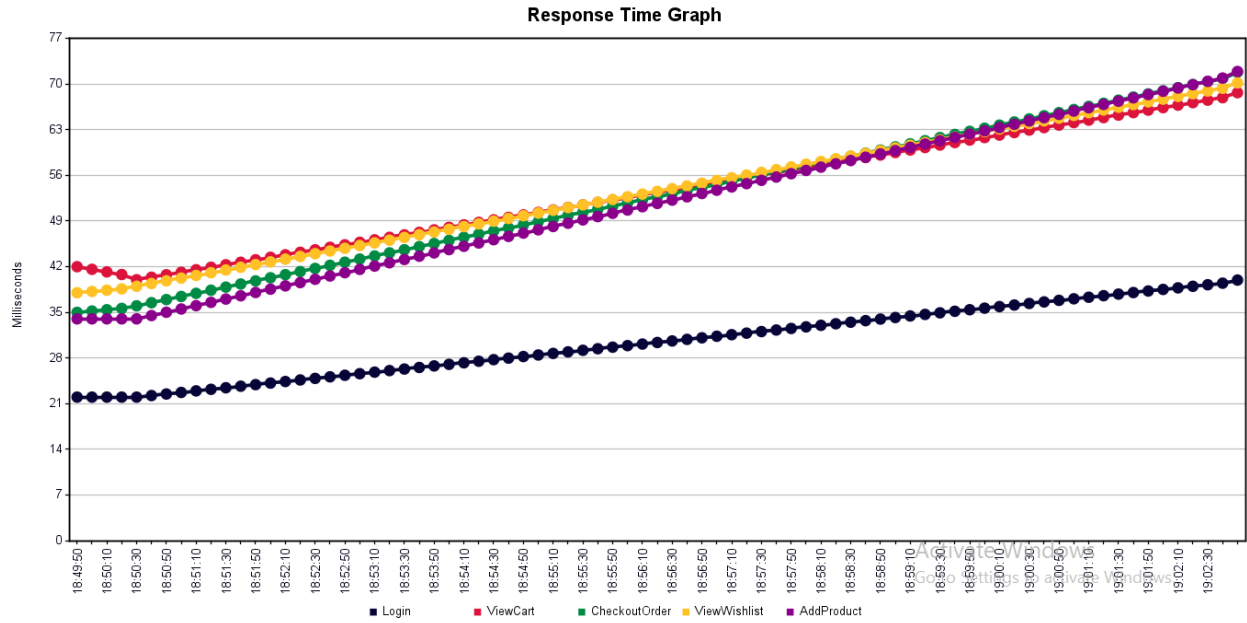


Figure 5: Response Time Graph

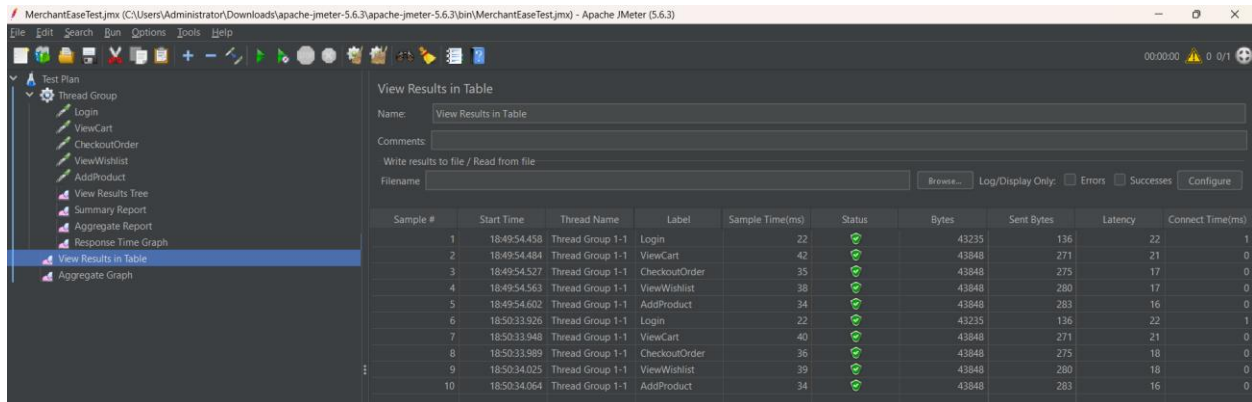


Figure 6: View Results in Table

G) Testing management tool (Jira):

1. Test plan:

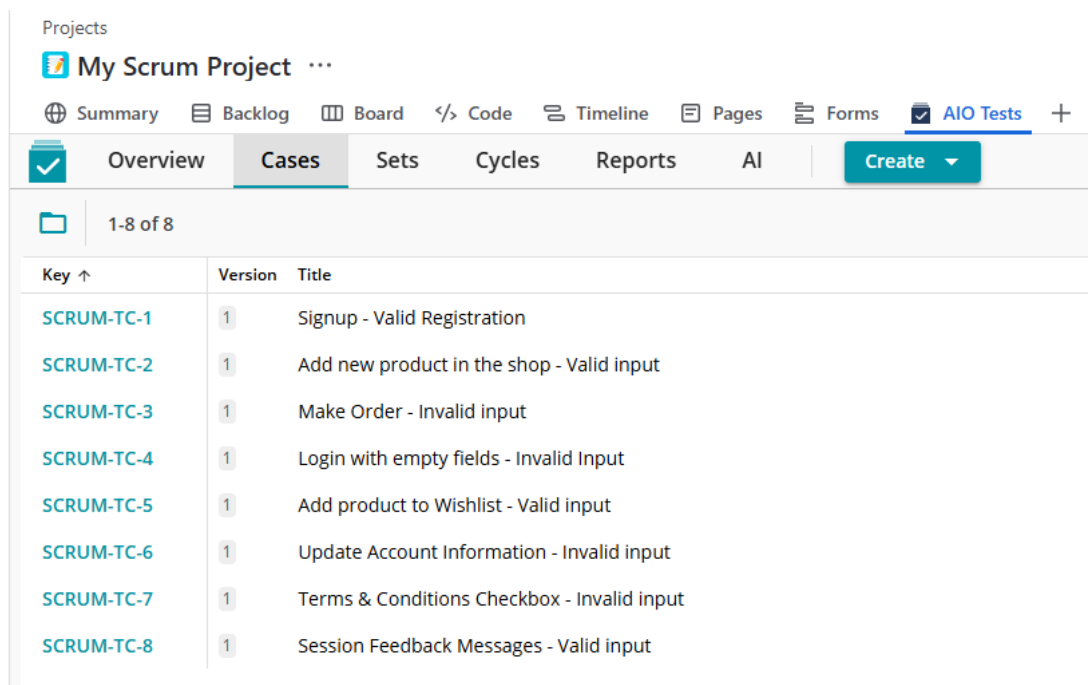
We used Jira with AIO Tests to manage our testing process as a team. This helped us organize and track everything clearly.

2. Creating test cases:

Each member added 2 test cases with two different approach and outcome (Valid and Invalid) then we put all 8 test cases into the test to compare between the expected result with the actual result.

Each test case included:

Title, Description, Pre-conditions, status, steps, priority and expected result.



The screenshot shows the Jira AIO Tests interface for a project named 'My Scrum Project'. The 'Cases' tab is selected, displaying a list of 8 test cases. Each case has a key, a version number (all are 1), and a title. The test cases cover various user actions like Signup, Add new product, Make Order, Login, Add product to Wishlist, Update Account Information, Terms & Conditions Checkbox, and Session Feedback Messages, each with a valid or invalid input scenario.

Key ↑	Version	Title
SCRUM-TC-1	1	Signup - Valid Registration
SCRUM-TC-2	1	Add new product in the shop - Valid input
SCRUM-TC-3	1	Make Order - Invalid input
SCRUM-TC-4	1	Login with empty fields - Invalid Input
SCRUM-TC-5	1	Add product to Wishlist - Valid input
SCRUM-TC-6	1	Update Account Information - Invalid input
SCRUM-TC-7	1	Terms & Conditions Checkbox - Invalid input
SCRUM-TC-8	1	Session Feedback Messages - Valid input

Example of a test case alongside its details:

The screenshot shows the 'Details' tab of a test case titled 'SCRUM-TC-1: Signup - Valid Registration'. The page includes an 'Edit' button in the top right corner. The form contains the following fields:

- Title ***: Signup - Valid Registration
- Description**: Test case for validating the user registration with valid inputs and correct outcome.
- Pre-conditions**: User is on the signup page, User doesn't have an account, Email used is new and not associated with an account.
- Folder**: -
- Owner ***: Ameer (indicated by an orange circle with 'A')
- Priority**: High
- Status ***: Published
- Type**: Functional
- Components**: -
- Releases**: -
- Estimated Effort**: 0h 10m
- Tags**: -
- Custom Fields**: No custom fields defined (with a help icon).
- Automation**: (with a help icon)
- Status**: -
- Owner**: Ameer (indicated by an orange circle with 'A')
- Automation Key**: - (with an information icon)

3. Creating test cycle:

We created a test cycle named "Cycle1", describing its objective, then we added all our test cases to this cycle to prepare for execution.

The screenshot shows the 'Details' tab of a test cycle titled 'SCRUM-CY-5: Cycle1'. The page includes an 'Edit' button in the top right corner. The form contains the following fields:

- Title ***: Cycle1
- Objective**: Verify that the functionalities of the system are working as expected, including user authentication, product management, ordering, Wishlist actions, and account handling.
- Folder**: -
- Owner ***: Ameer (indicated by an orange circle with 'A')
- Components**: -
- Releases**: -
- Start date**: -
- End date**: -
- Tags**: -
- Estimated Effort**: 0h 10m (with an information icon)
- Custom Fields**: No custom fields defined (with a help icon).

4. Execution of test cases:

We started manually executing each one of the test cases and compare between expected and actual result.

We marked them as Passed, Failed or In progress.

All test cases with their outcomes, with results that can be seen on top right of the image:

SCRUM-CY-5: Cycle1

Details

Cases

Execution

Actual Effort

0 Not Run

1 In Progress

4 Passed

3 Failed

0 Blocked

1-8 of 8

Add Cases

Q Key, title or automation key

F

G

H

I

J

K

L

M

N

O

P

Order ↑

1

SCRUM-TC-8

Session Feedback Messages - Valid input

Add run (2)

A

✓

2

SCRUM-TC-7

Terms & Conditions Checkbox - Invalid input

Add run (2)

A

✓

3

SCRUM-TC-6

Update Account Information - Invalid input

Add run (2)

A

✓

4

SCRUM-TC-5

Add product to Wishlist - Valid input

Add run (1)

A

✓

5

SCRUM-TC-4

Login with empty fields - Invalid Input

Add run (1)

A

✗

6

SCRUM-TC-3

Make Order - Invalid input

Add run (1)

A

✗

7

SCRUM-TC-2

Add new product in the shop - Valid input

Add run (1)

A

✗

8

SCRUM-TC-1

Signup - Valid Registration

Add run (1)

A

↑

Results came out to be:

4/8 Passed.

3/8 Failed.

1/8 In progress still.

5. Tracking & Documentation:

For test cases that failed, we created **defect reports** directly in Jira and linked them to the related test cases.

This helped us track bugs easily and keep everything organized.

We used the **Reports tab** in AIO Tests to check our overall testing progress, including:

- How many test cases **passed / failed / in progress**

H) Understanding Regression Testing:

Regression testing is a type of software testing that verifies that recent code changes haven't adversely affected existing functionality. It ensures that new features, bug fixes, or configuration changes don't introduce new defects into previously working components of the system.

Its importance:

- It's a complex system with many interconnected features.
- Changes in one area (like user authentication) could impact other areas (like order processing).
- The platform handles sensitive customer and payment data that must remain secure.

Regression Test Strategy:

1. Test Case Selection:

A) Full Regression Testing for critical core functionality:

- User authentication and session management
- Order processing workflow
- Payment processing
- Database integrity checks

B) Selective Regression Testing for less critical or isolated features:

- Wishlist functionality

- Product recommendation engine
- Analytics dashboard

2. Impact Analysis of Recent Changes

a. Session Handling Changes

Affected files: wishlist.php, login.php, signup.php, my_account.php

Impact areas:

- All user authentication flows
- Shopping cart persistence
- Wishlist functionality

b. CSRF Protection Implementation

Affected files: login.php, signup.php

Impact areas:

- All form submissions
- Payment processing
- User profile updates

c. Input Validation Improvements

Affected files: signup.php, my_account.php

Impact areas:

- User registration
- Profile updates

3. Regression Test Cases

Here are the key regression test cases we need to execute after any changes:

A) Authentication & User Management:

- 1-User registration with valid/invalid credentials
- 2-Login with correct/incorrect credentials
- 3-Password reset functionality
- 4-Session timeout validation
- 5-Concurrent login attempts

B) Product & Order Management:

- 6-Adding products to cart from multiple stores
- 7-Checkout process with valid/invalid shipping info
- 8-Order status updates and tracking
- 9-Product search with various filters
- 10-Wishlist add/remove functionality

Payment Processing:

- 11-Successful/failed payment transactions
- 12-Multiple payment method verification
- 13-Order confirmation emails

Security:

- 14-CSRF token validation on all forms
- 15-XSS and SQL injection attempts
- 16-Session hijacking attempts

4-Automation Approach

- Katalon tests for UI workflows
- JMeter for performance regression
- Custom scripts for API testing
- New automation to develop:
- Session management test suite
- CSRF validation tests
- Comprehensive input validation tests

5. Execution Plan

Frequency:

- Full regression suite: Before each production release
- Selective regression: After each sprint (every 2 weeks)

Environment:

- Dedicated regression testing environment mirroring production
- Test data with all user types (admin, merchant, customer)

Metrics to Track:

- Number of regression tests executed/passed/failed
- Defect escape rate to production
- Regression test execution time
- Automated vs manual test ratio

Impact Evaluation of Changes:

For each change made to MerchantEase, we evaluate:

1-Direct Impact: Features obviously affected by the change

2-Indirect Impact: Features that might be affected through dependencies

3-Configuration Impact: Changes to environment or settings that might affect behavior

Example evaluation for recent session handling changes:

Change Made	Direct Impact	Indirect Impact	Regression Tests Needed
Added session_start() to wishlist.php	Wishlist functionality	Shopping cart persistence, product recommendations	Wishlist tests, cart tests, session timeout tests
CSRF protection added to login form	Login functionality	Password reset, account updates	All form submission tests, authentication tests
Improved password validation	User registration	Password reset, profile updates	Registration tests, password change tests

Recommendations:

1-Prioritize Test Cases: Focus on high-risk areas first (payment processing, user auth)

2-Maintain Test Data: Keep consistent test data sets for reliable regression testing

3-Automate Where Possible: Increase automation coverage to reduce manual effort

4-Monitor Performance: Include performance benchmarks in regression suites

5-Document Changes: Maintain clear records of what changed and why to guide regression testing

By implementing this regression testing approach, we can ensure that changes to MerchantEase maintain the platform's stability and reliability while continuing to deliver new features and improvements.

I) Test Plan:

1. Introduction:

This document presents the Software Test Plan for the QA validation of the web application MerchantEase. The test plan provides the strategies, testing activities, tools, resources, and risk assessments for this system. The purpose and the goal are to make sure that the software meets the wanted and matched quality with good performance alongside with all of the functional and non-functional requirements.

2. Scope:

- **User management:** login, sign up and forgot password functionalities.
- **Wishlist system:** Add, delete or view wish listed items.
- **Product management:** Add, update or delete products.
- **Orders and cart system:** Add to cart, view cart, place order, track order.
- **Delivery Functionality:** Input address, track order, receive order.
- **Feedback system:** Customer can leave notes and feedbacks for merchants to reply or for other customers to benefit from.
- **Search & Recommendations:** Search for products, apply filters, full name search, suggesting recommendations based on preferences.

- **Cross-Platform usability and functionality:** Testing to see if it works on all devices such as phones (IOS, Android) and computers.
- **Security:** CSRF protection, input & payment validation, session handling.

3. Test Objectives:

A) Functional Testing Objectives:

- Ensure log in, sign up, forgot password functionalities work with no flaws.
- Verifying functionalities of Wishlist, orders, cart and delivery systems
- Verifying feedback system with the reply of merchants.

B) Performance Testing Objectives:

- Testing the application when its under high pressure as in many users using it at the same time.
- Verifying that application responsive time is realistic.

C) Usability Testing Objectives:

- Making sure the UI (user interface) isn't complicated and simple with everything clear for both merchants and customers.

D) Security Testing Objectives:

- Making sure CSRF protection is implemented and working, alongside with session handling.

4. Test Strategy:

Agile testing strategy integrated with the development cycle. Testing will be iterative, with quick feedback loops and continuous improvement.

A) Testing types:

- **Manual Testing:** Testing user interface (UI), usability checks, and user flows.
- **Automated Testing:** Selenium for UI regression tests, JMeter for performance testing.
- **Unit Testing:** Developers testing for individual PHP functions.
- **Integration Testing:** Ensures correct data flow. (ex: From cart -> order -> delivery).
- **System Testing:** All process testing from start to end (from placing an order till delivery confirmation).
- **Security Testing:** CSRF, session management and password handling.
- **Cross-Platform Testing:** Focus on CSRF, session management, and security of personal information.
- **User Acceptance Testing (UAT):** Final test where testing team acts like customers to make sure everything is clear and functional.

5. Test Schedule:

Phase	Activities	Start Date	End Date
Project Setup	Environment setup, tool installation, test plan approval	Day 1	Day 2
Requirements Review	Analyze functional & non-functional requirements	Day 3	Day 4
Functional Testing	Execute test cases for user management, product, orders, Wishlist, etc..	Day 5	Day 8
Security Testing	CSRF, session handling /testing	Day 9	Day 11
Usability Testing	User interface (UI/UX) testing.	Day 12	Day 13
Performance Testing	Load and pressure testing	Day 14	Day 15

Bug Fix & Retesting	Fixing and retesting bugs and issues.	Day 16	Day 18
Regression Testing	Full system testing to make sure no new bug appeared.	Day 19	Day 21
Test Closure	Documentation, all test summary and report.	Day 22	Day 23

6. Test Resources:

A) Personnel:

1. Test Lead.
2. Test Engineer/s.
3. Developer.

B) Environment:

- 1. Test Servers:** Staging environment mirroring production.
- 2. Databases:** Pre-filled with test data.
- 3. Tools:**
 - Katalon extension for chrome using Selenium for automation
 - JMeter for performance
 - Jira for scheduling black-box test-cases

7. Test Deliverables:

A) Documentation of Test Cases:

All manual and automated test cases that cover:

- User registration, and password reset.
- Product operations like adding, viewing and deleting.
- Cart and order placement.
- Wishlist operations (Add, edit, view and delete).
- Delivery system handling.
- Completion and functionality of feedback and notes review.

B) Test Data sets:

- Valid and invalid inputs for login, signup.
- Predefined user accounts.
- Histories of orders and products.

C) Test Report Summary:

- List of bugs discovered.
- Success or fail summaries.
- Retesting/Regression testing.
- Number of test cases that were executed.

D) Final Test Report:

- Summary of discovered bugs and fixes.
- Recommendations & Suggestions before deployment phase.
- Review of test cases.
- Overall testing progress.

E) Automated Test Scripts:

Selenium scripts for regression testing and UI verifying/testing alongside with JMeter scrips for performance, load and pressure testing.

8. Risks and Assumptions:

A) Risks:

- Test environment instability.
- Delayed feature development.

B) Assumptions:

- Tools like Katalon, JMeter will be working and functionable.
- Developers are active and available.
- Test data and credentials will be provided.
- Tested environment will be almost like the produced/final one after developing.

9. Dependencies

- Developer support.
- A fully functional staging environment
- The user stories, as well as development tasks, are done before testing.

10. Exit Criteria:

- Test coverage success.
- All critical test cases executed, cases covering functionalities.
- No open high-severity bugs remain unsolved.
- Documenting process and the results.
- Completion of regression testing.