Manual Testing Report

Project Name:	User Registration and Login System
Test Date:	2025/12/3
Tester:	SWE2309509 He Bolin SWE2309520 Li Ruibing SWE2309540 Wu Yi

1. Test Summary

1.1 Objective

To conduct a comprehensive manual test of the system, in **Black Box testing** we separate **functional testing** and **non-functional testing** as two layers, which can ensure more reliable quality assurance.

- **Functional Testing Layer (REG / LGN):** Focuses on validating the **Business Logic** and **Input Verification** of the Registration (REG) and Login (LGN) modules. The goal is to ensure that all user interactions align with functional requirements.
- **Non-Functional Testing Layer (DAT / SEC / EXP):** Extends beyond basic functionality to verify the system's reliability and integrity:
  - **Data Integrity (DAT):** Validate the consistency between the UI and the Backend Database.

- **Security (SEC):** Verify defenses against vulnerabilities like SQLi and XSS, and enforces strict authentication policies.

- **Experience-based Testing (EXP):** Assesses stability under edge cases based on Testers' past experience.

In **White Box testing (Static Code Review)**, by inspecting the source code (React Components, Hooks, API Logic) directly, our verification focus on logic flaws, security vulnerability, usability gaps and error handling coverage, also including code maintainability and standards. These details cannot be fully identified via Black Box testing.

1.2 Test Techniques

To ensure efficient defect detection, this execution primarily applies Boundary Value Analysis supported by fundamental Equivalence Partitioning concepts.

- **Boundary Value Analysis:** We strictly tested input limits (minimum/maximum lengths) to validate system stability and data constraints. This technique is important in identifying critical edge-case defects, such as the system crash caused by extreme email length.
- **Rationale:** While methods like Decision Table Testing and State Transition Testing were also considered during planning phase, we mainly focus on the validating data and logic. Therefore, BVA was selected as the most effective technique for this testing to expose input validation errors.

1.3 Tools Used

- **Test Case Management:** Microsoft Excel → Link: Manual Test Case Sharing Excel
- **Browser Developer Tools:** Chrome DevTools (Used for inspecting Network requests and simulating slow network conditions)

1.4 Test Environment

- **Operating System:** Windows 11
- **Browser:** Google Chrome & Microsoft Edge
- **Application URL:** Localhost Environment

2. Black Box Test Cases

2.1 Functional Test Cases (REG / LGN)

2.1.1 REG Test Cases

- **REG_1.x: Successful Register**

  Test Objective: Register success with all valid inputs

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_1.1	Register with all valid inputs	Username=newuser; Email=newuser@example.com; Password=Abc123!@; Confirm=Abc123!@	1. Fill Username, Email, Password, Confirm 2. Click Register	Registration success; redirect to Login or auto-login	**TEST PASS** Registration Successful

- **REG_2.x: Mandatory Field Validation**

Test Objective: Register Failure with empty input/ all fields invalid (Username, Email, Password, Confirm)

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_2.1	Register with Empty Username	Username= (empty); Email=a@b.com; Password=Abc123!@; Confirm=Abc123!@	1. Leave Username empty 2. Click Register	Show “Username is required” error	**TEST PASS** Show “Username must be at least 3 characters”
REG_2.2	Register with Empty Email	Username=user; Email= (empty); Password=Abc123!@; Confirm=Abc123!@	1. Leave Email empty 2. Click Register	Show “Email is required” error	**TEST PASS** Show “Please enter a valid email address” error

REG_2.3	Register with Empty Password	Username=user;
		Email=a@b.com;
		Password= (empty);
		Confirm= (empty)
			1. Leave Password empty
			2. Click Register
				Show 'Password is required' error	**TEST PASS**
					Show “Password must be at least 8 characters” error

REG_2.4	Register with Empty Confirm Password	Username=user;
		Email=a@b.com;
		Password= (empty);
		Confirm= (empty)
			1. Leave Confirm empty
			2. Click Register
				Show 'Please confirm your password' error	Meaning of warning is not clear
					Show “Password must be at least 8 characters” error

REG_2.5	Register with invalid inputs in all fields	Username= (empty);
		Email= 1;
		Password= 123;
		Confirm= 456
			1. Leave Username empty
			2. Enter other invalid inputs
			3. Click Register
				All the invalid inputs should be highlighted	Warning is not adequate
					Only “Invalid Email” is highlighted

- **REG_3.x: Email Validation**

  Test Objectives: Register failure with invalid-format/ out-of-bound email

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_3.1	Register with invalid email format with missing @	Email=abc.com	1. Enter invalid email 2. Click Register	Show “Invalid email format” error	**TEST PASS** Show “Please add a ‘@’. ‘abc.com’ does not have a ‘@’.”
REG_3.2	Register with invalid email format with missing domain	Email=abc@	1. Enter invalid email 2. Click Register	Show “Invalid email format” error	**TEST PASS** Show “Please enter a part following ‘@’. ‘abc@’ is incomplete.”
REG_3.3	Disposable /temporary register email blocked	Email= temp@mailinator.com	1. Enter disposable email 2. Click Register	System may block or flag; show message	**TEST PASS** Show “N/A”
REG_3.4	Register email with plus-addressing	Email= user+tag@example.com	1. Enter plus-addressed email 2. Click Register	Should accept as valid format	**TEST PASS** Show “Registration successful”; Jump back to login page.

REG_3.5	Register email case-insensitive uniqueness	Email= EXIST@example.com	1. Enter email with different case 2. Click Register	Show “Username already exists” error (Treat as same email)	**TEST PASS** Show “Registration failed: Email already registered”
REG_3.6	Register email length boundary test	Email= (254 characters)@test.com	1. Enter a prepared extremely long email 2. If the system did not warn us, continue to add more characters 3. Click Register	System sanitizes input; Script does NOT execute	No any email length constraints System page crashes

- **REG_4.x: Password Validation (Policy & UX)**

  Test Objectives:

  - REG_4.1 ~ 4.5: Failure to comply with complexity rules (Upper, Lower, Digit, Special Char, whitespace-only) will cause register failure.
  - REG_4.6: Length boundary tests (MIN boundary value: 8; MAX boundary value: 32)

BVA – Password Length		
Invalid (min -1)	**Valid** (min, min +1, max -1, max)	**Invalid** (max +1)
7	**8, 9, 127, 128**	**129**

- REG_4.7: Visual inline hints verification (UX)

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_4.1	Register password lacks uppercase	Password=abc123!@; Confirm=abc123!@	1. Enter password without uppercase 2. Click Register	Show “Password must contain at least one uppercase letter” error	**TEST PASS** As expect
REG_4.2	Register password lacks lowercase	Password=ABC123!@; Confirm=ABC123!@	1. Enter password without lowercase 2. Click Register	Show “Password must contain at least one lowercase letter” error	**TEST PASS** As expect
REG_4.3	Register password lacks digit	Password=Abcdef!@; Confirm=Abcdef!@	1. Enter password without digit 2. Click Register	Show “Password must contain at least one digit” error	**TEST PASS** As expect

REG_4.4	Register password lacks special character	Password=Abc12345; Confirm=Abc12345	1. Enter password without special char 2. Click Register	Show “Password must contain at least one special character” error	**TEST PASS** As expect
REG_4.5	Register with whitespace-only password	Password= ‘ ’	1. Enter spaces for passwords 2. Click Register	Reject as invalid; show error	**TEST PASS** Show “Password contains illegal characters.”
REG_4.6	Register password length boundary Value test cases	*BV 1:* Enter length 7	1. Enter passwords with *BV 1* / *BV 2* / *BV 3* / *BV 4* / *BV 5* / *BV 6* 2. Click Register	Invalid (Rejected)	**TEST PASS** As expect
		BV 2: Enter length 8		Valid (Accepted)	
		BV 3: Enter length 9		Valid (Accepted)	
		BV 4: Enter length 127		Valid (Accepted)	
		BV 5: Enter length 128		Valid (Accepted)	
		BV 6: Enter length 129		Invalid (Rejected)	
REG_4.7	Presence of visual inline password rule hints	Inspect UI	1. Focus password field 2. Observe hints	Password rules displayed and update as user types	1. No real-time password UI inspector at first time register; 2. No password up limitation hint before submit.

- **REG_5.x: Confirm password Consistency Validation**

  Test Objectives: Register failure when confirm password and password mismatch

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_5.1	Confirm password not match with password	Password=Abc123!@; Confirm=Abc123!@1	1. Enter mismatched confirm 2. Click Register	Show “Passwords do not match” error	**TEST PASS** As expect
REG_5.2	Confirm password auto-validate on blur	Inspect UI (Mismatch then blur)	1. Enter Password 2. Enter different Confirm 3. Inspect blur field	Inline message “Passwords do not match” appears	**TEST PASS** Show “Passwords do not match” error

- **REG_6.x: Username Specifications**

  Test Objectives:

  - REG_6.1: System auto trim spaces if register with leading/trailing spaces in username
  - REG_6.2: Length boundary tests (MIN boundary value: 3; MAX boundary value: 50)

BVA – Username Length		
Invalid (min -1)	**Valid** (min, min +1, max -1, max)	**Invalid** (max +1)
2	**3, 4, 49, 50**	**51**

- REG_6.6: Internationalization support (Unicode/Chinese characters)

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_6.1	Register Username has Leading/trailing spaces	Username= ‘ newuser ’	1. Enter username with spaces 2. Click Register	Trim spaces and register OR reject as invalid and show error	**TEST PASS** Show “Username can only contain letters, numbers, and underscores” error
REG_6.2	Register Username length boundary Value test cases	*BV 1:* Enter length 2 *BV 2:* Enter length 3 *BV 3:* Enter length 4 *BV 4:* Enter length 49 *BV 5:* Enter length 50 *BV 6:* Enter length 51	1. Enter username with *BV 1* / *BV 2* / *BV 3* / *BV 4* / *BV 5* / *BV 6* 2. Click Register	Invalid (Rejected) Valid (Accepted) Valid (Accepted) Valid (Accepted) Valid (Accepted) Invalid (Rejected)	**TEST PASS (BV)** No “Username must be at most 50 characters” hint right after the user enters the username (but after submit)
REG_6.3	Register with common special chars in username (.-_)	Username= ‘john.doe_jr-1’	1. Enter username with . _ - 2. Click Register	Accept or reject based on spec; should not crash	**TEST PASS** Show “Username can only contain letters, numbers, and underscores” error

REG_6.4	Register with Unicode char in username	Username= ‘hh←’	1. Enter Unicode username 2. Click Register	Accept or reject based on spec; must not crash	**TEST PASS** Show “Username can only contain letters, numbers, and underscores” error

- **REG_7.x: Duplicate Registration**

Test Objectives: Correctly handle existing data (Username taken, Email already registered), duplicate registration is illegal

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_7.1	Register with already taken username	Username=existingUser;	1. Enter exist username 2. Click Register	Show “Username already taken” error	**TEST PASS** Show “Register failed: Username already registered”
REG_7.2	Register with already registered email	Email=existing@example.com	1. Enter registered email 2. Click Register	Show “email already registered' error	**TEST PASS** Show “Register failed. Email already registered”

- **REG_8.x: Injection Defense**

Test Objectives: XSS (Cross-Site Scripting) must be prevented

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_8.1	XSS attempt in register username or email	Username= <script>alert(1)</script>	1. Enter malicious payload 2. Click Register	System sanitizes input; Script does NOT execute	**TEST PASS** Show “Username can only contain letters, numbers, and underscores” error

- **REG_9.x: Network & System Stability**

Test Objectives: Correct network timeout handling, graceful failure on slow connections

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
REG_9.1	Register with slow network	Valid data	1. Use Browser tool (F12) to control network interrupt 2. Recover network after 1 min	Show loading indicator and friendly timeout/error message Or register successfully after network recover	**TEST PASS** Registration Successful after network recovered

2.1.2 LGN Test Cases

- **LGN_1.x: Successful Login & Normal Session**

  Test Objective: Login/Logout success with all valid credentials, and navigation link to Register page works

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
LGN_1.1	Login with valid credentials	Username=testuser; Password=Password1!	1. Enter Username 2. Enter Password 3. Click Login	Login successful; redirect to Dashboard	**TEST PASS** As expect
LGN_1.2	Logout after enter dashboard page	Inspect UI after click "Log out" button	1. Click Logout button	Logout and navigate to login page.	**TEST PASS** As expect
LGN_1.3	Login link to Register navigates correctly	Inspect UI after click "Register now"	1. Click Register link	Page navigates to Register page	**TEST PASS** As expect

- **LGN_2.x: Mandatory Field Validation**

  Test Objective: Login failure with empty input (Username, Password)

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
LGN_2.1	Login with username and password empty	Username=(empty), Password=(empty)	1. Click “Login” button directly	Error “Username is required”; Error “Password is required”; stay on Login page	**TEST PASS** As expect
LGN_2.2	Login with username empty	Username= (empty); Password=Password1!	1. Leave Username empty 2. Enter Password 3. Click Login	Error “Username is required”; stay on Login page	**TEST PASS** As expect
LGN_2.3	Login with password empty	Username=testuser; Password= (empty)	1. Enter Username 2. Leave Password empty 3. Click Login	Error “Password is required”; stay on Login page	**TEST PASS** As expect

- **LGN_3.x: Credentials Validation (Auth logic)**

  Test Objective: Login failure with invalid Credentials (Username, Password)

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
LGN_3.1	Login with non-existent username	Username=no_user; Password=AnyPass1!	1. Enter unknown Username 2. Enter Password 3. Click Login	Error "User not found" or generic auth error	**TEST PASS** Show "Login failed: Incorrect username or password"; Show Warning: Multiple failed attempts may lock your account temporarily
LGN_3.2	Login with incorrect password	Username=testuser; Password=WrongPass1!	1. Enter Username 2. Enter wrong Password 3. Click Login	Error "Incorrect password" and deny access	**TEST PASS** Same as above

- **LGN_4.x: Data Formatting & Boundaries**

  Test Objectives:

  - LGN_4.1 ~ 4.3: Verify input normalization→ normal input should be valid; Behavior consistent with spec (case-sensitive)
  - LGN_4.4: Username length boundary tests (MIN boundary value: 3; MAX boundary value: 50 (same as register))

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
LGN_4.1	Login with leading/trailing spaces in username	Username=' testuser ' ; Password=Password1!	1. Enter Username with spaces 2. Enter Password 3. Click Login	Trim spaces then login success OR show invalid username if spaces considered	**TEST PASS** Show “Login failed: Incorrect username or password”;
LGN_4.2	Login with case-sensitive username handling	Username=TestUser vs testuser; Password correct	1. Try with different casing 2. Click Login	Behavior matches spec (either case-sensitive or not); consistent	**TEST PASS** Same as above
LGN_4.3	Login with special characters in username	Username=user!@#; Password=Password1!	1. Enter Username with special chars 2. Enter Password 3. Click Login	Accept or reject according to spec; should not crash	**TEST PASS** Same as above
LGN_4.4	Login Username length boundary Value test cases	*BV 1:* Enter length 2	1. Enter username with *BV 1 / BV 2 / BV 3 / BV 4 / BV 5 / BV 6* 2. Click Login	Invalid	**TEST PASS**
		BV 2: Enter length 3		Valid	
		BV 3: Enter length 4		Valid	
		BV 4: Enter length 49		Valid	
		BV 5: Enter length 50		Valid	
		BV 6: Enter length 51		Invalid	

- **LGN_5.x: UI Experience & Password Visibility**

Test Objectives: Validate UI usability regarding password privacy (masking) and visibility toggle functionality.

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
LGN_5.1	Login shows password masked	Password=Abc123!@	1. Type into Password field 2. Try to unmask the password	Password is masked in default, only show when user click “show password” button, and mask again when user click “hide password”	Usability defect Password is masked in default. But this system has no design of “Show/Hide password” button
LGN_5.2	Password visibility toggle after blur and refocus	Password=Abc123!@	1. Input password and reveal it. 2. Blur and refocus the input field. 3. Check if toggle still works	Toggle remains functional after blur/refocus; icon state syncs with password visibility	Usability defect Icon is unresponsive after blur/refocus. Password remains hidden and cannot be toggled.

- **LGN_6.x: Account Protection & Integrity**

  Test Objectives: Verify the system's resilience against abuse attempts and malicious input patterns.

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
LGN_6.1	Login with SQL injection attempt in username field	Username= ' OR '1'='1; Password=anything	1. Enter SQL payload in Username 2. Enter Password 3. Click Login	Input sanitized; no SQL injection; login fails	**TEST PASS** As expect
LGN_6.2	Lockout and unlock when login fail	Invalid username and password	1. Enter invalid username and password 2. Click login 3. If the system warn "Login failed: Multiple failed attempts may lock your account temporarily", click login again 4. Continue to log in until the system is locked	The system is locked after several times of login failure, tell the user to wait	**TEST PASS** System is locked after five times of login failure; user is unable to login for the next 1 minutes

LGN_6.3	Use another IP to attempt login after one IP is locked	Invalid username and password; Two different IP: “1.1.1.1” and “2.2.2.2”	1. Enter invalid username and password 2. In one IP, click login until the system is locked 3. Switch to another IP and try login with same username and password	In the other IP, the system should still be locked and not allow further login attempts	Raised security problem Further login attempt is allowed in a new IP
LGN_6.4	Account lockout policy bypass	Invalid username and password; Valid username and password	1. Enter invalid username and password for four times 2. Enter valid username and password for one time 3. Logout 4. Try to enter invalid username and password again for several times	The system should lock when total number of login failure reaches 5 times	Failed Login Counter Resets After Successful Login Successful login resets the failure count to zero. Lockout requires 5 new consecutive failures post-logout, ignoring previous attempts.

2.2 Non-functional Test Cases (DAT / SEC / EXP)

2.2.1 DAT Test Cases

- **DAT _1.x: Database Verification**

  Test Objectives:

  - DAT_1.1 ~ 1.2: Verify data consistency between UI and Database for creation and immediate retrieval.
  - DAT_1.3 ~ 1.4: Validate access denial and session termination logic upon user deletion.
  - DAT_1.5 ~ 1.7: Ensure authentication mechanisms correctly reflect backend data updates (Deactivation & Credential changes).
  - DAT_1.8: Verify database constraints regarding username case sensitivity.

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
DAT _1.1	Registration Data Integrity check (UI to DB)	1. Username = “user01” 2. Email= email01@test.com 3. Password = Abc123!@	1. Enter register page 2. Input valid username, email, password, confirm password 3. Click Register Button 4. Check database for user info	The user info in the database is as same as what we used in registration	**TEST PASS** As expect

DAT _1.2	Login After Registration	1. Username = “user02” 2. Email= email02@test.com 3. Password = Abc123!@	1. Register a new account with valid inputs 2. Back to Login Page 3. Login using the newly registered credentials 4. Click Login Button	User login successfully and is navigated to dashboard page	**TEST PASS** As expect
DAT _1.3	Login After user data is deleted	1. Username = “user03” 2. Email= email03@test.com 3. Password = Abc123!@	1. Register a new account and ensure logged out 2. Delete user record from DB 3. Try login with the deleted credentials	For the first time the user can login successfully, after user is deleted, the login fails	**TEST PASS** As expect
DAT _1.4	Delete user while user is still in dashboard page	1. Username = “user01” 2. Email= email01@test.com 3. Password = Abc123!@	1. Login with a valid account 2. Enter dashboard page 3. Check DB → delete user data and observe interface 4. Check DB → refresh the page multiple times, observe interface	System invalidates the session, log the user out, and redirect them to Login Page	Session persists after DB deletion Deleted user remains logged in; Refreshing the page continues to display user info.

DAT _1.5	Login after deactivate account	1. Username = “user01” 2. Email= email02@test.com 3. Password = Abc123!@ 4. New value of ‘is_active’ = False	1. Login with a valid account then manually deactivate account in DB (is_active= ‘false’) 2. Verify access is denied upon page refresh and re-login attempts 3. Reactivate account in DB (is_active= ‘true’) and confirm login works again	Logging failed; the system shows “account is banned” or “wrong username/password”	**TEST PASS** As expect
DAT _1.6	Login After username modification	1. Username = “user01” 2. Email= email03@test.com 3. Password = Abc123!@ 4. New value of “username” = user12345	1. Register “user01” and ensure login works 2. Update “user01” to “user12345” in DB then logout 3. Try login with “user01” 4. Try login with “user12345”	User cannot login with “user01” after modification, can successfully log in with “user12345”	**TEST PASS** As expect

DAT _1.7	Login After password modification	1. Username = “user01” 2. Email= email03@test.com 3. Password = Abc123!@ 4. New value of “username” = user12345	1. Register an account with password ‘Abc123!@’ 2. Modify the password to ‘123Abc!@’ in DB 3. Try login with new password 4. Try login with old password 5. Try login with previous password token	New password should success and old password fail, previous token login should fail.	Auth Broken; Token Valid Both password login failed (password is hashed), but previous token success
DAT _1.8	Case sensitivity chaos	1. Username1 = “user01”, email and password valid 2. Username2 = “USER01”, email and password valid	1. Register two accounts with usernames that differ in case 2. Delete ‘user01’ in DB 3. Log in with ‘user01’ then logout 4. Log in with ‘USER01’	The ‘user01’ should not be able to log in while ‘USER01’ can	**TEST PASS** As expect

2.2.2 SEC Test Cases

- **SEC _1.x: Security Validation**

  Test Objectives:

  - SEC_1.1 ~ 1.2: Validate input sanitization mechanisms against common injection attacks
  - SEC_1.3 ~ 1.4: Verify secure transport protocols to ensure credentials are not exposed in URLs or GET requests.

- SEC_1.5 ~ 1.7: Ensure secure session management via JWT implementation (Headers, Storage, and Secret Keys).

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result
SEC _1.1	Login with SQL injection pattern	Username:' OR '1'= '1; Password=Abc123!@	1. Enter SQL injection payload in username field 2. Enter random password 3. Click Login	- Login request sent normally (POST) - Toast shows: "Login failed" - No JS errors in Console - No SQL stack trace exposed - User not logged in	**TEST PASS** 1. Requests sent normally; 2. Server refused but not crash; 3. No frontend error JS Syntax Error/Type Error; 4. No SQL error stack; 5. UI toast has "Login failed"
SEC _1.2	Login with XSS injection in Username	Username: <script>alert(1)</script> Password=Abc123!@	1. Enter script tag in username 2. Submit form 3. Inspect page + console → Check if script executes	- No browser alert popup - Toast displays plain text error (no script execution) - DOM does NOT contain injected <script> tag - Console contains no JS injection errors	**TEST PASS** 1. No alert popup 2. toast shows normal "Login failed" 3. Elements search found 0 user-injected <script> tags 5. No XSS executed

SEC _1.3	Login form submit via HTTP POST	Valid/Invalid Credentials	1. Open DevTools (Network) 2. Perform Login 3. Check Request Method	- Login must use HTTP POST method - No GET request made to /api/auth/token - URL remains clean (/login)	**TEST PASS** 1. Login is via POST; 2. URL remains “/request”
SEC _1.4	Presence of sensitive data in URLs after Login	Valid Credentials	1. Perform Login 2. Observe Browser Address Bar (URL and Network Request URL)	- URL must not contain query parameters like password= or username=; - All sensitive data is only in Request Body - No query params containing credentials	**TEST PASS** 1. URL contains no sensitive data; 2. All sensitive data like username, password are in the request body; 3. No query params containing credentials
SEC _1.5	JWT token passed via Authorization header	Valid Credentials	1. Observe /api/auth/users/me request 2. Check Authorization: Bearer xxx header	- Token ONLY in Authorization header - Token not in URL - Token not in cookies unless HttpOnly	**TEST PASS** As expect

SEC _1.6	JWT stored in local Storage	Valid Credentials; Inspect local storage	1. Login successfully 2. Open browser →DevTools→Application →Local Storage 3. Inspect access_token/ jwt	Token should be stored securely, no sensitive tokens found in localStorage	RAW JWT token is clearly visible in localStorage
SEC _1.7	Hardcoded Fallback Secret Key	Source Code /Config	1. Manually analyze the code	No hardcoded secret keys in source code; Active SECRET_KEY is random or loaded from env vars	Code explicitly falls back to a hardcoded default value: “09d25e094faa6ca2556c81816 6b7a9563b93f7099f6f0f4caa6 cf63b88e8d3e7”

2.2.3 EXP Test Cases

- **EXP _1.x: Experienced Based Validation**

Test Objectives: Validate stability and usability under edge cases based on Tester experience.

TC ID	TC Name	Test Data	Steps to Execute	Expected Result	Actual Result

EXP_1.1	Password Masking	Username= user;
		Email=a@b.com;
		Password=Abc123!@;
		Confirm=Abc123!@
			1. Enter valid inputs in all fields
			2. Try to unmask the password and confirm password
				The input password is masked in default, user can click on 'Show/Hide password' button to show/ hide the password.	Usability defect
					The system has no design of 'Hide/Show password'

EXP_1.2	Page Refresh Retention	All inputs are valid; Register button unclicked	1. Enter valid inputs in all fields 2. Refresh Page	All the inputs should be cleared	**TEST PASS** As expect

EXP _1.3	Interrupted Edge Case (Submit + refresh)	Username= user;
		Email=a@b.com;
		Password=Abc123!@;
		Confirm=Abc123!@
			1. Enter valid input in all fields
			2. Click 'Register' and refresh page at the same time
				The account should be set up correctly after refreshing is over	Lack of notification & navigation
					The account is recorded correctly with no notification of 'registration success' on page, and system did not navigate back to login page

3. White Box Test Cases

3.1 Business Logic & Data Consistency (WB_LOG)

Test Objectives: Verify that state management and form validation logic align with data integrity requirements.

Tc Id	Target Component	Code Inspection Logic	Expected Implementation	Actual Code Finding	Status
WB_LOG_01	`loginMutation` (onSuccess)	Inspect async flow of Token storage vs User fetching.	Rollback: If `getCurrentUser()` fails, the previously stored token must be cleared.	Code executes `setToken(token)` then awaits user info. If fetch fails (catch block), token remains in store while UI shows error. User ends in "half-logged-in" state.	**FAIL**
WB_LOG_02	`useForm` (Zod Resolver)	Check dependency validation between `password` and `confirmPassword`.	Changing the main password should trigger re-validation of the confirm field immediately.	`confirmPassword` does not automatically re-validate when `password` changes. Error only appears upon form submission.	**FAIL**
WB_LOG_03	`RegisterSchema`.ts (Zod)	Inspect Password Complexity Regex patterns.	Schema must enforce: 1 Upper, 1 Lower, 1 Digit, 1 Special Char, Min 8 chars.	Zod schema correctly implements regex checks (`.regex(/[A-Z]/, ...)`). Backend validation is aligned.	**PASS**

3.2 User Experience & Usability (WB_UX)

Test Objectives: Verify the UI implementation provides necessary feedback mechanisms and state persistence for a good user experience.

Tc Id	Target Component	Code Inspection Logic	Expected Implementation	Actual Code Finding	Status
WB_UX_01	`Login.tsx` (State)	Inspect `isLockedOut` state lifecycle.	Lockout state must persist across page refreshes (e.g., stored in `localStorage` or determined by Backend timestamp).	State is local: `useState(false).` Refreshing the page resets `isLockedOut` to false, misleading the user that they can retry immediately.	**FAIL**
WB_UX_02	**PasswordInput** (Component)	Check for visibility toggle implementation.	Password field should include an "Eye Icon" to toggle `type="text"` / `type="password"`.	Standard `<Input type="password"/>` used without any toggle wrapper. Users cannot verify input before submitting.	**FAIL**
WB_UX_03	`Register.tsx` (UI Text)	Inspect Password Rule Feedback mechanism.	Rule list should be dynamic (highlight/check off items as user types).	Rules are hardcoded static HTML (`<p>MUST contain...</p>`). Text remains static regardless of input, providing poor feedback loop.	**FAIL**
WB_UX_04	`Layout.tsx` (Responsiveness)	Inspect UI responsiveness on mobile breakpoints.	Layout should adapt to mobile screens. For example, `flex-col` on small screens	Tailwind classes `(md:flex-row, w-full)` are correctly applied for responsive design.	**PASS**

3.3 Error Handling & Security (WB_ERR / WB_SEC)

Test Objectives: Ensure the system handles edge cases gracefully and adheres to basic security protocols.

Tc Id	**Target Component**	**Code Inspection Logic**	**Expected Implementation**	**Actual Code Finding**	**Status**
WB_ERR_01	**`authApi`** (onError)	Check HTTP Status Code coverage in error handler	Should handle specific codes: 429 (Lockout), 500 (Server Error), and `ERR_NETWORK`.	Code only checks `status === 429`. Generic fallback misses 500 or Network Errors, potentially showing undefined/ugly toasts.	**FAIL**
WB_SEC_01	**`authApi.ts`** (Methods)	Inspect HTTP Methods for Auth endpoints	Login/Register must use `doPost()` method.	Axios calls use `axios.post('/login')` and `axios.post('/register')`.	**PASS**
WB_SEC_02	**Input** (Component)	Inspect Output Rendering	React automatically escapes values in `{}` to prevent XSS.	Usage of `{...register('field')}` and standard React binding ensures basic XSS protection.	**PASS**

4. Manual Testing Result Table

Method	Module	Total	Pass	Fail	Pass Rate
Black Box	**REG**	30	26	**4**	86.7%
	LGN	19	14	**5**	73.7%
	EXP	3	1	**2**	33.3%
	DAT	8	6	**2**	75.0%
	SEC	7	5	**2**	71.4%
White Box	CODE	10	4	**6**	40.0%