WalletWatch – User Acceptance Testing (UAT) Plan

Team: Group 013-4
Semester: Fall 2025
Environment: Localhost (Docker), Chrome Browser
Testing Framework: Mocha, Chai, Chai-HTTP
Testers: Team members + 1 external volunteer

Feature 1: User Registration Functionality

Feature Description

Users can create a new account by providing a username, email, and password. The system validates the input, hashes the password securely, and returns a JWT token upon successful registration.

Test Data

Field	Example Value
Username	testuser2025
Email	testuser@example.com
Password	SecurePass123!

Acceptance Criteria

1. User cannot register unless all mandatory fields (username, email, password) are completed.
2. Successful registration creates a new user row in the database with hashed password.
3. System returns a 201 status code with user info and JWT token upon success.
4. Form provides clear error messages for invalid or missing fields.
5. Duplicate usernames or emails are rejected with appropriate error messages.

UAT Test Cases

✅ Test Case 1: Successful User Registration (Positive)

Steps:

1. Navigate to registration page.
2. Enter valid username, email, and password.
3. Click Register/Submit.

Expected Result:

- Status code 201 returned
- Response body includes: id, username, email, and token
- User record exists in `users` table with hashed password
- User sees success confirmation

Pass Criteria:

- ✓ User saved to database
- ✓ Password properly hashed
- ✓ Valid JWT token generated
- ✓ No errors displayed

✅ Test Case 2: Missing Required Fields (Negative)

Steps:

1. Navigate to registration page.
2. Leave one or more fields blank (e.g., only enter email).
3. Click Register/Submit.

Expected Result:

- Status code 400 returned
- Error message displays: "fill all the fields" or similar user-friendly message
- No database insertion occurs
- No JWT token generated

Pass Criteria:

- ✓ No user created
- ✓ Correct error messages shown
- ✓ Backend validation prevents incomplete registration

✅ Test Case 3: Invalid Email Format

Steps:

1. Enter valid username and password.
2. Enter invalid email (e.g., `testuser@com` or `notanemail`).
3. Click Register/Submit.

Expected Result:

- System blocks the submission
- Error: "Invalid email format"
- Status code 400 returned
- No DB insertion

Pass Criteria:

- ✓ Email validation prevents bad data
- ✓ No user created
- ✓ Clear error feedback provided

✅ Test Case 4: Weak Password Validation

Steps:

1. Enter valid username and email.
2. Enter weak password (e.g., "123").
3. Submit form.

Expected Result:

- Error: "Password must be 8+ characters with letters & numbers"
- Status code 400 returned
- No DB update

Pass Criteria:

- ✓ Password validation enforced
- ✓ No user created with weak password

Feature 2: User Login Functionality

Feature Description

Registered users can log into their account using their email and password. The system verifies credentials, compares the hashed password, and returns a JWT token for authenticated sessions.

Test Data

Field	Example Value
Email	kishore@example.com
Password	Test1234!

Acceptance Criteria

1. Login form requires both email and password fields.
2. System validates credentials against database records.
3. Successful login returns status 200 with JWT token.
4. Invalid credentials return status 400 with descriptive error.
5. Password comparison uses secure hashing (bcrypt).

UAT Test Cases

✅ Test Case 1: Successful Login (Positive)

Steps:

1. Register a user with valid credentials (if not already registered).
2. Navigate to login page.
3. Enter correct email and password.
4. Click Login/Submit.

Expected Result:

- Status code 200 returned
- Response body includes: id, username, email, and token
- Valid JWT token generated
- User redirected to dashboard/home page
- Success message displayed

Pass Criteria:

- ✓ JWT token generated correctly
- ✓ Password hashing verification works
- ✓ User authenticated successfully
- ✓ No errors

✅ Test Case 2: Invalid Credentials (Negative)

Steps:

1. Navigate to login page.
2. Enter email for non-existent user or wrong password.
3. Click Login/Submit.

Expected Result:

- Status code 400 returned
- Error message: "Invalid credentials" or similar
- No JWT token issued
- User remains on login page

Pass Criteria:

- ✓ Invalid users denied access
- ✓ Secure authentication flow maintained
- ✓ No token generated for invalid login
- ✓ Proper error handling

✅ Test Case 3: Missing Login Fields (Negative)

Steps:

1. Navigate to login page.
2. Leave email or password field blank.
3. Click Login/Submit.

Expected Result:

- Status code 400 returned
- Error: "fill all the fields" or similar message
- No authentication attempt made
- No token issued

Pass Criteria:

- ✓ Form validation prevents incomplete submission
- ✓ Clear error feedback
- ✓ Backend validation active

Feature 3: Server Health Check (Welcome Endpoint)

Feature Description

A basic endpoint to verify that the Express server is running and responding to requests correctly.

Acceptance Criteria

1. Endpoint responds with status 200.
2. No authentication required.
3. Response confirms server initialization.

UAT Test Cases

✅ Test Case 1: Server Running Verification (GET /welcome)

Steps:

1. Send GET request to `/welcome` endpoint.
2. Observe response.

Expected Result:

- Status code 200 returned
- Response body contains welcome message
- No errors or timeouts

Pass Criteria:

- ✓ Server responds successfully
- ✓ Express server initialized correctly
- ✓ Endpoint accessible

✅ Test Environment

- **Browser:** Chrome v120+
- **Backend:** Node.js 18 (Dockerized)
- **Database:** PostgreSQL 15 (Dockerized)
- **Testing Framework:** Mocha, Chai, Chai-HTTP
- **Testing Location:** Localhost (127.0.0.1:3000)
- **Tools:** Docker Compose, VS Code, Postman (optional)

✅ UAT Execution Summary (Week 4 – to be filled later)

Feature	Tester	Pass/Fail	Notes
User Registration	—	—	—
User Login	—	—	—
Server Health Check	—	—	—

✅ Changes Made After Testing

- Adjusted error messages for missing fields to be more user-friendly (e.g., "fill all the fields" instead of generic errors)
- Verified JWT generation and hashing consistency between test and production environment
- Confirmed password hashing implementation uses bcrypt consistently
- Enhanced error handling for invalid credentials to prevent information leakage

---

✅ Risks

Organizational Risks

- Team members unavailable during testing
- Limited access to external user testers
- Coordination challenges for simultaneous testing

Technical Risks

- Untested edge cases (e.g., SQL injection, XSS attacks)
- Unexpected database connection failures
- Docker environment differences across team member machines
- JWT token expiration and refresh logic not yet tested
- Rate limiting not implemented for authentication endpoints
-

Business Risks

- Users frustrated by unclear error messages
- Poor authentication flow leading to abandoned registrations
- Slow response times affecting user experience during high load