Authentication System - QA Test Cases

Overview

This document contains comprehensive test cases for the JWT-based authentication system with refresh tokens, session management, and security features.

---

1. Web Login (Session-Based)

Happy Path

TC-WL-001: Successful Login with Valid Credentials

- **Preconditions**: Valid admin user exists
- **Steps**:
  1. Navigate to login page
  2. Enter valid email and password
  3. Click "Login"
- **Expected Results**:
  - Redirects to dashboard
  - `refresh_token` cookie set (HttpOnly, 30 days)
  - `api_csrf_token` cookie set (30 days)
  - AuthSession created in database
  - Laravel session established
  - Success message displayed

TC-WL-002: Login with "Remember Me"

- **Preconditions**: Valid admin user exists
- **Steps**:
  1. Navigate to login page
  2. Enter valid credentials
  3. Check "Remember me" checkbox
  4. Click "Login"
- **Expected Results**:
  - Session persists after browser close
  - User remains logged in

TC-WL-003: Login Redirects to Intended URL

- **Preconditions**: User tries to access protected page while logged out
- **Steps**:
  1. Access protected route (e.g., `/admin/dashboard`)
  2. Get redirected to login
  3. Enter valid credentials
  4. Click "Login"
- **Expected Results**:
  - Redirects to originally requested URL (`/admin/dashboard`)
  - Not just to default dashboard

Validation Errors

TC-WL-004: Login with Missing Email

- **Steps**:
  1. Navigate to login page
  2. Enter password only
  3. Click "Login"
- **Expected Results**:
  - Validation error: "Email is required"
  - No login attempt logged

TC-WL-005: Login with Missing Password

- **Steps**:
  1. Navigate to login page
  2. Enter email only
  3. Click "Login"
- **Expected Results**:
  - Validation error: "Password is required"
  - No login attempt logged

TC-WL-006: Login with Invalid Email Format

- **Steps**:
  1. Navigate to login page
  2. Enter invalid email (e.g., "notanemail")
  3. Enter password
  4. Click "Login"
- **Expected Results**:
  - Validation error: "Email must be a valid email address"
  - No login attempt logged

TC-WL-007: Login with Invalid Credentials

- **Steps**:
  1. Navigate to login page
  2. Enter valid email with wrong password
  3. Click "Login"
- **Expected Results**:
  - Error message: "The provided credentials do not match our records"
  - Failed login attempt logged in `login_attempts` table
  - No session created
  - No cookies set

Rate Limiting

TC-WL-008: IP-Based Rate Limiting

- **Preconditions**: 5 failed login attempts from same IP in last 15 minutes
- **Steps**:
  1. Make 5 failed login attempts from same IP

    2. Attempt 6th login (even with valid credentials)

- **Expected Results**:
    - HTTP 429 status code
    - Error message: "Too many login attempts. Please try again later."
    - Login blocked

TC-WL-009: Email-Based Rate Limiting

- **Preconditions**: 5 failed login attempts for same email in last 15 minutes
- **Steps**:
    1. Make 5 failed login attempts for same email
    2. Attempt 6th login (even with valid credentials)
- **Expected Results**:
    - Error message: "Too many login attempts for this account. Please try again later."
    - Login blocked

TC-WL-010: Rate Limit Reset After Time Window

- **Preconditions**: IP is rate limited
- **Steps**:
    1. Wait 15+ minutes
    2. Attempt login with valid credentials
- **Expected Results**:
    - Login succeeds
    - Rate limit counter reset

---

2. API Login (JWT-Based)

Happy Path

TC-AL-001: Successful API Login

- **Preconditions**: Valid admin user exists
- **Steps**:
    1. POST to `/api/admin/login`
    2. Send JSON: `{"email": "admin@example.com", "password": "password"}`
- **Expected Results**:
    - HTTP 200 status
    - Response contains:
        - `access_token` (JWT string)
        - `token_type` : "Bearer"
        - `expires_in` : 900 (15 minutes in seconds)
        - `user` object with id, name, email
    - `refresh_token` cookie set (HttpOnly, SameSite=Strict, 30 days)

TC-AL-002: Access Token is Valid JWT

- **Preconditions**: Successful API login
- **Steps**:
    1. Decode the returned access_token

- **Expected Results**:
  - Token is valid JWT format
  - Payload contains:
    - `iss` : app URL
    - `sub` : user ID
    - `type` : "Admin" or "OrganizationUser"
    - `session_id` : UUID
    - `iat` : current timestamp
    - `exp` : 15 minutes from `iat`

TC-AL-003: Access Token Expiration Time

- **Preconditions**: Successful API login
- **Steps**:
  1. Decode access_token
  2. Check `exp` claim
- **Expected Results**:
  - `exp` = `iat` + 900 seconds (15 minutes)
  - `expires_in` in response matches 900

Validation Errors

TC-AL-004: API Login with Missing Email

- **Steps**:
  1. POST to `/api/admin/login`
  2. Send JSON: `{"password": "password"}`
- **Expected Results**:
  - HTTP 422 validation error
  - Error message indicates email is required

TC-AL-005: API Login with Missing Password

- **Steps**:
  1. POST to `/api/admin/login`
  2. Send JSON: `{"email": "admin@example.com"}`
- **Expected Results**:
  - HTTP 422 validation error
  - Error message indicates password is required

TC-AL-006: API Login with Invalid Email Format

- **Steps**:
  1. POST to `/api/admin/login`
  2. Send JSON: `{"email": "notanemail", "password": "password"}`
- **Expected Results**:
  - HTTP 422 validation error
  - Error message indicates invalid email format

TC-AL-007: API Login with Invalid Credentials

- **Steps**:

1. POST to `/api/admin/login`
2. Send JSON: `{"email": "admin@example.com", "password": "wrongpassword"}`

- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Invalid credentials."}`
  - Failed login attempt logged

Rate Limiting

TC-AL-008: API Rate Limiting by IP

- **Preconditions**: 5 failed API login attempts from same IP
- **Steps**:
  1. Make 5 failed login attempts via API
  2. Attempt 6th login (even with valid credentials)
- **Expected Results**:
  - HTTP 429 status
  - Response: `{"message": "Too many login attempts. Please try again later."}`

TC-AL-009: Rate Limiting Applies to API Endpoint

- **Preconditions**: IP rate limited from web login attempts
- **Steps**:
  1. Attempt API login from same IP
- **Expected Results**:
  - HTTP 429 status
  - Rate limit applies across both web and API endpoints

Security

TC-AL-010: Refresh Token Cookie is HttpOnly

- **Preconditions**: Successful API login
- **Steps**:
  1. Check `refresh_token` cookie properties
- **Expected Results**:
  - Cookie has `HttpOnly` flag set
  - Cookie not accessible via JavaScript `document.cookie`

TC-AL-011: Refresh Token Cookie SameSite Policy

- **Preconditions**: Successful API login
- **Steps**:
  1. Check `refresh_token` cookie properties
- **Expected Results**:
  - Cookie has `SameSite=Strict`
  - CSRF protection enabled

TC-AL-012: Refresh Token Format

- **Preconditions**: Successful API login
- **Steps**:

    1. Extract refresh_token from cookie (via server-side inspection)
- **Expected Results**:
    - Token is exactly 128 hexadecimal characters
    - Matches pattern: `^[0-9a-f]{128}$`

---

3. Token Refresh

Happy Path

TC-TR-001: Successful Token Refresh

- **Preconditions**: Valid refresh token cookie exists
- **Steps**:
    1. POST to `/api/admin/refresh`
    2. Include `refresh_token` cookie
- **Expected Results**:
    - HTTP 200 status
    - New `access_token` in response
    - New `refresh_token` cookie set
    - Old refresh token cannot be reused
    - `expires_in`: 900 seconds

TC-TR-002: Token Rotation Works

- **Preconditions**: Valid refresh token cookie exists
- **Steps**:
    1. Refresh token (get new refresh_token)
    2. Attempt to use old refresh_token again
- **Expected Results**:
    - First refresh succeeds
    - Second refresh with old token fails (401)
    - Old token marked as reused in database

TC-TR-003: New Access Token is Valid

- **Preconditions**: Token refresh successful
- **Steps**:
    1. Use new access_token in Authorization header
    2. Access protected endpoint
- **Expected Results**:
    - Request succeeds
    - User authenticated correctly

TC-TR-004: Session Activity Updated on Refresh

- **Preconditions**: Valid refresh token exists
- **Steps**:
    1. Note `last_activity_at` timestamp
    2. Refresh token
    3. Check `last_activity_at` in database

- **Expected Results**:
    - `last_activity_at` updated to current time
    - Sliding expiry reset

Error Cases

TC-TR-005: Refresh Without Token Cookie

- **Steps**:
    1. POST to `/api/admin/refresh`
    2. Do not include `refresh_token` cookie
- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "No refresh token provided."}`

TC-TR-006: Refresh with Invalid Token

- **Steps**:
    1. POST to `/api/admin/refresh`
    2. Include invalid/random refresh_token cookie
- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "Invalid or expired session."}`
    - Cookie cleared from response

TC-TR-007: Refresh with Expired Token (Absolute Expiry)

- **Preconditions**: Refresh token older than 30 days
- **Steps**:
    1. Manually set session `expires_at` to past date in database
    2. Attempt refresh
- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "Invalid or expired session."}`

TC-TR-008: Refresh with Inactive Token

- **Preconditions**: Session inactive for 60+ minutes
- **Steps**:
    1. Manually set `last_activity_at` to 61 minutes ago in database
    2. Attempt refresh
- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "Invalid or expired session."}`

TC-TR-009: Refresh with Revoked Token

- **Preconditions**: Session revoked in database
- **Steps**:
    1. Set `is_revoked = true` for session
    2. Attempt refresh with associated refresh_token
- **Expected Results**:

    - HTTP 401 status
    - Response: `{"message": "Invalid or expired session."}`

Security

TC-TR-010: Token Reuse Detection

- **Preconditions**: Valid refresh token exists
- **Steps**:
    1. Refresh token (get new token)
    2. Attempt to use old refresh_token again
- **Expected Results**:
    - First refresh succeeds
    - Second refresh fails (401)
    - Session marked as revoked
    - `revoke_reason` = "token_reuse_detected"

TC-TR-011: Rotation Count Increments

- **Preconditions**: Valid refresh token exists
- **Steps**:
    1. Check initial `rotation_count` in database
    2. Refresh token multiple times
    3. Check `rotation_count` after each refresh
- **Expected Results**:
    - `rotation_count` increments by 1 each time
    - `last_rotated_at` updated each time

---

4. Access Token Validation (Middleware)

Happy Path

TC-AT-001: Valid Access Token Allows Request

- **Preconditions**: Valid access token from API login
- **Steps**:
    1. Make request to protected endpoint
    2. Include header: `Authorization: Bearer <access_token>`
- **Expected Results**:
    - Request succeeds (HTTP 200)
    - User attached to request
    - Can access `$request->get('jwt_user')` in controller

TC-AT-002: Bearer Token Format Accepted

- **Preconditions**: Valid access token
- **Steps**:
    1. Make request with `Authorization: Bearer <token>`
- **Expected Results**:
    - Token extracted correctly

    - Request proceeds

TC-AT-003: User Accessible in Controller

- **Preconditions**: Valid access token
- **Steps**:
    1. Access protected route with valid token
    2. In controller, access `$request->get('jwt_user')`
- **Expected Results**:
    - Returns authenticated user model
    - User properties accessible (id, name, email, etc.)

TC-AT-004: Session Accessible in Controller

- **Preconditions**: Valid access token
- **Steps**:
    1. Access protected route with valid token
    2. In controller, access `$request->get('jwt_session')`
- **Expected Results**:
    - Returns AuthSession model
    - Session properties accessible

TC-AT-005: Sliding Expiry Updates Activity

- **Preconditions**: Valid access token
- **Steps**:
    1. Note `last_activity_at` timestamp
    2. Make request with valid token
    3. Check `last_activity_at` in database
- **Expected Results**:
    - `last_activity_at` updated to current time
    - Inactivity timeout reset

Error Cases

TC-AT-006: Missing Authorization Header

- **Steps**:
    1. Make request to protected endpoint
    2. Do not include Authorization header
- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "Access token required.", "error": "missing_token"}`

TC-AT-007: Invalid Token Format

- **Steps**:
    1. Make request with `Authorization: Bearer invalidtoken123`
- **Expected Results**:
    - HTTP 401 status

- Response: `{"message": "Invalid or expired access token.", "error": "invalid_token"}`

TC-AT-008: Expired Access Token

- **Preconditions**: Access token older than 15 minutes
- **Steps**:
  1. Wait 16 minutes after login
  2. Use old access_token in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Invalid or expired access token.", "error": "invalid_token"}`

TC-AT-009: Tampered Token

- **Preconditions**: Valid access token
- **Steps**:
  1. Modify token string (change characters)
  2. Use tampered token in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Invalid or expired access token.", "error": "invalid_token"}`

TC-AT-010: Revoked Session

- **Preconditions**: Valid access token, but session revoked in database
- **Steps**:
  1. Set `is_revoked = true` for session in database
  2. Use access_token (from before revocation) in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Session has been revoked or expired.", "error": "session_invalid"}`

TC-AT-011: Expired Session (Absolute)

- **Preconditions**: Session `expires_at` in past
- **Steps**:
  1. Manually set `expires_at` to past date
  2. Use access_token in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Session has been revoked or expired.", "error": "session_invalid"}`

TC-AT-012: Inactive Session

- **Preconditions**: Session inactive for 60+ minutes
- **Steps**:
  1. Set `last_activity_at` to 61 minutes ago

2. Use access_token in request

- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Session has been revoked or expired.", "error": "session_invalid"}`

TC-AT-013: User Deleted After Token Issued

- **Preconditions**: Valid access token issued
- **Steps**:
  1. Delete user from database
  2. Use access_token in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "User not found.", "error": "user_not_found"}`

TC-AT-014: Wrong Guard Type

- **Preconditions**: Admin access token, route requires different guard
- **Steps**:
  1. Use admin token on route with `jwt.auth:organization` middleware
- **Expected Results**:
  - HTTP 403 status
  - Response: `{"message": "Unauthorized for this resource.", "error": "wrong_guard"}`

TC-AT-015: Unknown User Type in Token

- **Preconditions**: Token with invalid `type` claim
- **Steps**:
  1. Manually create token with `type: "InvalidType"`
  2. Use token in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Unknown user type.", "error": "unknown_type"}`

Guard Testing

TC-AT-016: Admin Token on Admin Route

- **Preconditions**: Admin access token
- **Steps**:
  1. Use admin token on route with `jwt.auth:admin` middleware
- **Expected Results**:
  - Request succeeds
  - User authenticated

TC-AT-017: Admin Token on Organization Route

- **Preconditions**: Admin access token
- **Steps**:
  1. Use admin token on route with `jwt.auth:organization` middleware
- **Expected Results**:

  - HTTP 403 status
  - Access denied

TC-AT-018: OrganizationUser Token on Admin Route

- **Preconditions**: OrganizationUser access token
- **Steps**:
  1. Use organization token on route with `jwt.auth:admin` middleware
- **Expected Results**:
  - HTTP 403 status
  - Access denied

---

5. Logout

Web Logout

TC-LO-001: Logout Revokes Session

- **Preconditions**: User logged in
- **Steps**:
  1. Click logout button
- **Expected Results**:
  - AuthSession marked as `is_revoked = true`
  - `revoked_at` timestamp set
  - `revoke_reason` = "logout"

TC-LO-002: Logout Clears Cookies

- **Preconditions**: User logged in
- **Steps**:
  1. Click logout
  2. Check cookies in browser
- **Expected Results**:
  - `refresh_token` cookie removed
  - `api_csrf_token` cookie removed

TC-LO-003: Logout Invalidates Laravel Session

- **Preconditions**: User logged in
- **Steps**:
  1. Click logout
  2. Try to access protected route
- **Expected Results**:
  - Redirected to login page
  - Cannot access protected routes

TC-LO-004: Logout Redirects to Login

- **Preconditions**: User logged in
- **Steps**:
  1. Click logout

- **Expected Results**:
    - Redirects to login page
    - Success message: "Logged out successfully."

API Logout

TC-LO-005: API Logout

- **Preconditions**: User logged in via API
- **Steps**:
    1. POST to logout endpoint (if exists)
    2. Or manually revoke session
- **Expected Results**:
    - Session revoked
    - Cookies cleared
    - HTTP 200 or appropriate response

TC-LO-006: Access Token Invalidated After Logout

- **Preconditions**: User logged in, has access token
- **Steps**:
    1. Logout
    2. Use access token in request
- **Expected Results**:
    - HTTP 401 status
    - Token no longer valid

TC-LO-007: Refresh Token Invalidated After Logout

- **Preconditions**: User logged in
- **Steps**:
    1. Logout
    2. Attempt to refresh token
- **Expected Results**:
    - HTTP 401 status
    - Refresh fails

Edge Cases

TC-LO-008: Logout Without Active Session

- **Preconditions**: No active session
- **Steps**:
    1. Attempt logout
- **Expected Results**:
    - No errors thrown
    - Redirects to login page

TC-LO-009: Logout with Multiple Sessions

- **Preconditions**: User logged in on multiple devices
- **Steps**:

    1. Logout from one device

- **Expected Results**:
  - Only current session revoked
  - Other sessions remain active

---

6. Session Management

List Sessions

TC-SM-001: Get Active Sessions

- **Preconditions**: User logged in, multiple active sessions
- **Steps**:
  1. GET `/api/admin/sessions` (or equivalent endpoint)
- **Expected Results**:
  - Returns JSON array of sessions
  - Each session contains:
    - `id` : session UUID
    - `device_name` : parsed device name
    - `ip_address` : IP address
    - `last_activity` : human-readable time
    - `created_at` : human-readable time
    - `is_current` : boolean

TC-SM-002: Current Session Marked

- **Preconditions**: User logged in on multiple devices
- **Steps**:
  1. Get sessions list
  2. Check `is_current` flag
- **Expected Results**:
  - Current session has `is_current: true`
  - Other sessions have `is_current: false`

TC-SM-003: Only Active Sessions Returned

- **Preconditions**: User has revoked/expired sessions
- **Steps**:
  1. Get sessions list
- **Expected Results**:
  - Only non-revoked, non-expired sessions returned
  - Revoked sessions excluded

Revoke Sessions

TC-SM-004: Revoke Specific Session

- **Preconditions**: User has multiple active sessions
- **Steps**:
  1. Get session ID from sessions list

    2. POST to revoke endpoint with session ID

- **Expected Results**:
    - Session marked as `is_revoked = true`
    - `revoke_reason` = "user_revoked"
    - HTTP 200 with success message

TC-SM-005: Revoke All Other Sessions

- **Preconditions**: User logged in on multiple devices
- **Steps**:
    1. Call "revoke other sessions" endpoint
- **Expected Results**:
    - All other sessions revoked
    - Current session remains active
    - Success message displayed

TC-SM-006: Revoke Non-Existent Session

- **Steps**:
    1. Attempt to revoke invalid session ID
- **Expected Results**:
    - HTTP 404 status
    - Response: `{"message": "Session not found."}`

TC-SM-007: Revoke Another User's Session

- **Preconditions**: Two users logged in
- **Steps**:
    1. User A attempts to revoke User B's session ID
- **Expected Results**:
    - HTTP 404 status
    - Cannot access other user's sessions

Security

TC-SM-008: Revoked Session Cannot Be Used

- **Preconditions**: Valid access token, session then revoked
- **Steps**:
    1. Revoke session
    2. Use access token in request
- **Expected Results**:
    - HTTP 401 status
    - Token rejected

TC-SM-009: Revoked Session Cannot Refresh

- **Preconditions**: Valid refresh token, session then revoked
- **Steps**:
    1. Revoke session
    2. Attempt to refresh token
- **Expected Results**:

- HTTP 401 status
- Refresh fails

---

7. Token Expiration and Timeouts

Access Token Expiration

TC-EX-001: Access Token Expires After 15 Minutes

- **Preconditions**: Valid access token issued
- **Steps**:
  1. Wait 16 minutes
  2. Use access token in request
- **Expected Results**:
  - HTTP 401 status
  - Token expired

TC-EX-002: Refresh Before Expiration

- **Preconditions**: Valid access token, 10 minutes old
- **Steps**:
  1. Refresh token before 15-minute expiry
- **Expected Results**:
  - New access token issued
  - New refresh token issued
  - Old tokens invalidated

TC-EX-003: Use Expired Token

- **Preconditions**: Access token expired
- **Steps**:
  1. Use expired token in request
- **Expected Results**:
  - HTTP 401 status
  - Response: `{"message": "Invalid or expired access token.", "error": "invalid_token"}`

Session Expiration

TC-EX-004: Absolute Expiry (30 Days)

- **Preconditions**: Session created 30+ days ago
- **Steps**:
  1. Manually set `expires_at` to 31 days ago
  2. Attempt to use access token or refresh token
- **Expected Results**:
  - HTTP 401 status
  - Session expired

TC-EX-005: Inactivity Timeout (60 Minutes)

- **Preconditions**: Session inactive for 60+ minutes

- **Steps**:
  1. Set `last_activity_at` to 61 minutes ago
  2. Attempt to use access token or refresh token
- **Expected Results**:
  - HTTP 401 status
  - Session expired due to inactivity

TC-EX-006: Activity Resets Inactivity Timer

- **Preconditions**: Session with 50 minutes of inactivity
- **Steps**:
  1. Use access token in request
  2. Check `last_activity_at` in database
- **Expected Results**:
  - `last_activity_at` updated to current time
  - Inactivity timer reset
  - Session remains valid

---

8. Security Features

CSRF Protection

TC-SF-001: CSRF Token Generated

- **Preconditions**: User logged in
- **Steps**:
  1. Check cookies after login
- **Expected Results**:
  - `api_csrf_token` cookie set
  - Cookie NOT HttpOnly (JavaScript can read it)
  - Token is 64 hex characters (32 bytes)

TC-SF-002: CSRF Token Validation

- **Preconditions**: User logged in, CSRF middleware enabled
- **Steps**:
  1. Make API request without CSRF token
  2. Make API request with invalid CSRF token
  3. Make API request with valid CSRF token
- **Expected Results**:
  - Without token: Request rejected (if middleware applied)
  - Invalid token: Request rejected
  - Valid token: Request succeeds

TC-SF-003: Missing CSRF Token

- **Preconditions**: CSRF middleware enabled on route
- **Steps**:
  1. Make request without CSRF token header/cookie
- **Expected Results**:
  - Request rejected

    - Appropriate error response

Token Security

TC-SF-004: Refresh Token Hashed in Database

- **Preconditions**: User logged in
- **Steps**:
  1. Check `auth_sessions` table
  2. Inspect `refresh_token_hash` column
- **Expected Results**:
  - Plain refresh token NOT stored
  - Only hash stored (64 characters, SHA-256 HMAC)
  - Cannot reverse-engineer original token

TC-SF-005: Token Reuse Detection

- **Preconditions**: Valid refresh token
- **Steps**:
  1. Refresh token (get new one)
  2. Attempt to use old refresh token again
- **Expected Results**:
  - Old token reuse detected
  - Session revoked
  - `revoke_reason` = "token_reuse_detected"
  - HTTP 401 response

TC-SF-006: Session Tracking

- **Preconditions**: User logged in
- **Steps**:
  1. Check `auth_sessions` table
- **Expected Results**:
  - `ip_address` stored
  - `user_agent` stored
  - `device_name` parsed and stored
  - `last_activity_at` tracked

Rate Limiting

TC-SF-007: IP-Based Rate Limiting

- **Steps**:
  1. Make 5 failed login attempts from same IP
  2. Attempt 6th login
- **Expected Results**:
  - 6th attempt blocked
  - HTTP 429 status
  - Rate limit: 5 attempts per 15 minutes per IP

TC-SF-008: Email-Based Rate Limiting

- **Steps**:

    1. Make 5 failed login attempts for same email
    2. Attempt 6th login
- **Expected Results**:
    - 6th attempt blocked
    - Rate limit: 5 attempts per 15 minutes per email

TC-SF-009: Successful Login Resets Counter

- **Preconditions**: 4 failed attempts from IP
- **Steps**:
    1. Make successful login
    2. Make 5 more failed attempts
- **Expected Results**:
    - Successful login resets rate limit counter
    - Can make 5 more attempts before rate limit

---

9. Edge Cases and Error Scenarios

Invalid Inputs

TC-EC-001: SQL Injection in Email

- **Steps**:
    1. Attempt login with email: `admin@example.com' OR '1'='1`
- **Expected Results**:
    - Input sanitized
    - No SQL execution
    - Login fails (invalid credentials)

TC-EC-002: XSS in Email

- **Steps**:
    1. Attempt login with email: `admin@example.com<script>alert('xss')</script>`
- **Expected Results**:
    - Input sanitized
    - No script execution
    - Login fails (invalid email format or credentials)

TC-EC-003: Very Long Email

- **Steps**:
    1. Attempt login with email: 300+ characters
- **Expected Results**:
    - Validation error
    - Email length validation enforced

TC-EC-004: Special Characters in Password

- **Steps**:
    1. Attempt login with password containing special characters
- **Expected Results**:

    - Handled correctly
    - Password validation works

Network and Timing

TC-EC-005: Concurrent Logins

- **Preconditions**: Same user account
- **Steps**:
    1. Login from Device A
    2. Simultaneously login from Device B
- **Expected Results**:
    - Both logins succeed
    - Two separate sessions created
    - Both sessions valid

TC-EC-006: Rapid Token Refresh

- **Preconditions**: Valid refresh token
- **Steps**:
    1. Refresh token multiple times rapidly (within seconds)
- **Expected Results**:
    - Each refresh succeeds
    - Token rotation works correctly
    - No race conditions

TC-EC-007: Clock Skew

- **Preconditions**: Server and client clocks differ
- **Steps**:
    1. Use token with slight time difference
- **Expected Results**:
    - Token validation handles reasonable clock skew
    - Or rejects if skew too large

Data Integrity

TC-EC-008: Session Deleted from Database

- **Preconditions**: Valid access token
- **Steps**:
    1. Manually delete session from database
    2. Use access token in request
- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "Session has been revoked or expired.", "error": "session_invalid"}`

TC-EC-009: User Deleted After Login

- **Preconditions**: Valid access token
- **Steps**:
    1. Delete user from database

    2. Use access token in request

- **Expected Results**:
    - HTTP 401 status
    - Response: `{"message": "User not found.", "error": "user_not_found"}`

TC-EC-010: Session ID Mismatch

- **Preconditions**: Valid access token
- **Steps**:
    1. Manually change `session_id` in token payload (tamper)
    2. Use token in request
- **Expected Results**:
    - Token signature invalid
    - HTTP 401 status

---

10. Integration Tests

Multi-Device Scenarios

TC-IT-001: Login from Multiple Devices

- **Preconditions**: Same user account
- **Steps**:
    1. Login from Device A (browser)
    2. Login from Device B (mobile app)
    3. Login from Device C (API client)
- **Expected Results**:
    - All three logins succeed
    - Three separate sessions created
    - All sessions active and valid

TC-IT-002: Revoke All Other Sessions

- **Preconditions**: User logged in on 3 devices
- **Steps**:
    1. From Device A, revoke all other sessions
    2. Check sessions on Device B and C
- **Expected Results**:
    - Device A session remains active
    - Device B and C sessions revoked
    - Cannot use tokens from B and C

TC-IT-003: Logout from One Device

- **Preconditions**: User logged in on 3 devices
- **Steps**:
    1. Logout from Device A
    2. Check sessions on Device B and C
- **Expected Results**:
    - Device A session revoked
    - Device B and C sessions remain active

- Can still use tokens from B and C

Cookie Behavior

TC-IT-004: Cookies Set Correctly

- **Preconditions**: User logged in
- **Steps**:
  1. Inspect cookies in browser dev tools
- **Expected Results**:
  - `refresh_token` : HttpOnly, Secure (if HTTPS), SameSite=Strict
  - `api_csrf_token` : NOT HttpOnly, Secure (if HTTPS), SameSite=Lax
  - Correct expiration times set

TC-IT-005: Cookie Expiration

- **Preconditions**: User logged in
- **Steps**:
  1. Check cookie expiration dates
- **Expected Results**:
  - `refresh_token` expires in 30 days
  - `api_csrf_token` expires in 30 days

TC-IT-006: Cookie Domain/Path

- **Preconditions**: User logged in
- **Steps**:
  1. Check cookie domain and path attributes
- **Expected Results**:
  - Cookies set for correct domain
  - Path set correctly (usually `/`)

Test Data Requirements

Test Users

- **Valid Admin User**
  - Email: `admin@test.com`
  - Password: `password123`
  - Status: Active
- **Invalid Credentials**
  - Email: `admin@test.com`
  - Password: `wrongpassword`
- **Non-Existent User**
  - Email: `nonexistent@test.com`
  - Password: `anypassword`

Test IPs

- Multiple test IP addresses for rate limiting tests
- IP: `192.168.1.100` (for rate limit testing)
- IP: `192.168.1.101` (for concurrent login testing)

Test Devices

- **Browser**: Chrome, Firefox, Safari (different user agents)
- **Mobile**: iOS Safari, Android Chrome
- **API Client**: Postman, cURL, custom client

Test Environment Setup

Prerequisites

1. **Database**
   - Test database with migrations run
   - Test users created
   - Clean state before each test run
2. **Configuration**
   - `.env` file configured
   - `APP_KEY` set (for JWT signing)
   - Database connection configured
3. **Tools Required**
   - **API Testing**: Postman, Insomnia, or cURL
   - **Cookie Inspector**: Browser DevTools
   - **Database Tool**: phpMyAdmin, TablePlus, or Laravel Tinker
   - **JWT Decoder**: jwt.io or similar
   - **Network Proxy**: Burp Suite or OWASP ZAP (optional)
4. **Clock Manipulation** (for expiration tests)
   - Database time manipulation
   - Or wait for actual time (not recommended for automated tests)

Priority Levels

P0 - Critical (Must Test Before Release)

- TC-AL-001: Successful API Login
- TC-AL-007: API Login with Invalid Credentials
- TC-AT-001: Valid Access Token Allows Request
- TC-AT-006: Missing Authorization Header
- TC-AT-007: Invalid Token Format
- TC-TR-001: Successful Token Refresh
- TC-TR-005: Refresh Without Token Cookie
- TC-LO-001: Logout Revokes Session

P1 - High (Should Test Before Release)

- TC-WL-001: Successful Login with Valid Credentials

- TC-WL-007: Login with Invalid Credentials
- TC-AL-002: Access Token is Valid JWT
- TC-AT-008: Expired Access Token
- TC-AT-010: Revoked Session
- TC-TR-006: Refresh with Invalid Token
- TC-SF-007: IP-Based Rate Limiting
- TC-SF-004: Refresh Token Hashed in Database

P2 - Medium (Test During QA Cycle)

- All other test cases

Test Execution Checklist

Pre-Test

- [ ] Database reset/cleanup
- [ ] Test users created
- [ ] Environment variables configured
- [ ] API endpoints accessible
- [ ] Tools ready (Postman, browser, etc.)

During Test

- [ ] Execute test cases in priority order
- [ ] Document actual results vs expected
- [ ] Capture screenshots/requests for bugs
- [ ] Note any deviations from expected behavior

Post-Test

- [ ] Compile test results
- [ ] Log bugs/issues found
- [ ] Verify fixes in retest
- [ ] Update test documentation if needed

Notes

- **Access Token TTL**: 15 minutes (900 seconds)
- **Refresh Token TTL**: 30 days absolute, 60 minutes inactivity
- **Rate Limit**: 5 attempts per 15 minutes (IP and email-based)
- **JWT Algorithm**: HS256
- **Refresh Token Format**: 128 hex characters (64 random bytes)
- **CSRF Token Format**: 64 hex characters (32 random bytes)

Document Version: 1.0

Last Updated: 2024-12-14

Maintained By: QA Team