ParaBank Application - Jira Defect Tickets

Updated: September 7, 2025 - Extended Testing Results (152 test scenarios)

PARA-001: Authentication system accepts any credentials - Complete security bypass

Type: Bug
Priority: Highest
Severity: Critical
Component: Authentication
Labels: security, authentication, critical-vulnerability
Reporter: QA Team
Assignee: Backend Security Team
Environment: ParaBank Demo Application (https://parabank.parasoft.com/parabank/)

Summary: The authentication system fails to validate user credentials, allowing unauthorized access with any username/password combination.

Description: The login functionality completely bypasses authentication validation. Any arbitrary username and password combination results in successful authentication, creating a critical security vulnerability that would allow unauthorized access to any user account in a banking system.

Steps to Reproduce:

1. Open ParaBank application (https://parabank.parasoft.com/parabank/)
2. Navigate to the login page
3. Enter invalid/non-existent credentials:
   - Username: `invaliduser123`
   - Password: `wrongpassword456`
4. Click "Log In" button
5. Observe the result

Expected Result:

- Authentication should fail
- Error message should display: "Invalid username or password"
- User should remain on login page

Actual Result:

- User is successfully authenticated
- User is redirected to accounts overview page
- Full access to banking features is granted

Impact:

- **Security Risk:** Complete authentication bypass
- **Business Impact:** Unauthorized access to sensitive financial data
- **Compliance:** Violates banking security regulations
- **User Trust:** Compromises customer data protection

Acceptance Criteria for Fix:

- [ ] Only valid username/password combinations should allow login
- [ ] Invalid credentials should display appropriate error message
- [ ] Failed login attempts should be logged for security monitoring
- [ ] Account lockout after multiple failed attempts should be implemented

Test Cases: TC_004, TC_005

PARA-002: Session management vulnerability - Back button exposes secure data after logout

Type: Bug
Priority: Highest
Severity: Critical
Component: Session Management
Labels: security, session, logout, vulnerability
Reporter: QA Team
Assignee: Backend Security Team

Summary: Users can access protected banking information using browser back button after logout, indicating improper session invalidation.

Description: After successful logout, the application fails to properly invalidate the user session. Users can access sensitive account information by simply clicking the browser back button, creating a serious security vulnerability in shared or public computer environments.

Steps to Reproduce:

1. Log in to ParaBank with any credentials
2. Navigate to "Accounts Overview" page
3. View account details and balances
4. Click "Log Out" button
5. Verify logout confirmation
6. Click browser back button
7. Observe the result

Expected Result:

- User should be redirected to login page
- Session should be completely invalidated
- Access to protected content should be denied
- Error message should indicate session expiration

Actual Result:

- User can view accounts overview page
- Sensitive banking information remains accessible
- No session validation occurs
- Full functionality appears available

Impact:

- **Security Risk:** Session hijacking vulnerability
- **Data Exposure:** Unauthorized access to financial information
- **Compliance:** Violates banking security standards (PCI DSS)
- **Scenario Risk:** Shared computer environments pose significant risk

Acceptance Criteria for Fix:

- [ ] Logout should completely invalidate server session
- [ ] Browser cache should be cleared of sensitive data
- [ ] Back button should redirect to login page
- [ ] Session timeout should be properly implemented
- [ ] "Cache-Control: no-store" headers should be set for protected pages

Test Cases: TC_015

PARA-003: Missing input validation - Registration form accepts invalid data

Type: Bug
Priority: High
Severity: High
Component: User Registration, Input Validation
Labels: security, validation, data-integrity, registration
Reporter: QA Team
Assignee: Frontend Team, Backend Team

Summary: Registration form lacks proper input validation, allowing invalid data in critical user fields.

Description: The user registration form accepts invalid data across multiple fields without proper validation or sanitization. This creates data integrity issues and potential security vulnerabilities through injection attacks.

Affected Fields with Issues:

- **First Name:** Accepts numbers and special characters (`123John`, `@#$%John`)
- **Last Name:** Accepts numbers and special characters (`123Doe`)
- **Address:** Accepts only special characters (`@#$%`)
- **City:** Accepts numbers (`123City`)
- **State:** Accepts invalid state codes (`123`)
- **Zip Code:** Accepts letters instead of numbers (`ABCDE`)
- **Phone:** Accepts arbitrary text (`phone123`)
- **SSN:** Accepts invalid SSN format (`invalid-ssn`)
- **Password Confirmation:** Accepts mismatched passwords

Steps to Reproduce:

1. Navigate to registration page
2. Fill out the form with invalid data examples above
3. Submit the registration form
4. Observe validation response

Expected Result:

- Form should validate each field according to business rules
- Specific error messages should appear for invalid data
- Registration should fail with invalid input
- User should be prompted to correct errors

Actual Result:

- Form accepts all invalid data without validation
- Registration completes successfully
- Account is created with corrupt data
- No error messages or validation feedback

Impact:

- **Data Quality:** Corrupted user database
- **Security Risk:** Potential injection attack vectors
- **User Experience:** No feedback on data entry errors
- **Compliance:** Violates data quality standards

Acceptance Criteria for Fix:

- [ ] Implement client-side validation for all fields
- [ ] Add server-side validation as security backup
- [ ] Create specific error messages for each validation rule
- [ ] Implement proper SSN format validation
- [ ] Add state code validation against standard list
- [ ] Implement phone number format validation
- [ ] Add password confirmation matching validation

Test Cases: TC_041-TC_050

PARA-004: Automated testing limitation - Cloudflare Captcha detection prevents injection testing

Type: Task
Priority: Low
Severity: Low
Component: Test Automation, Security Testing
Labels: test-automation, captcha, security-testing, false-positive

Reporter: QA Team
Assignee: QA Automation Team

Summary: Automated security tests for SQL injection and XSS fail to detect Cloudflare Captcha, leading to incomplete security testing results.

Description: During automated security testing, injection payloads trigger Cloudflare Captcha protection, but the automated testing framework does not detect or handle this response. This results in incomplete security test coverage and potential false reporting of vulnerabilities.

Steps to Reproduce:

1. Navigate to login page using automated testing tool
2. Enter malicious payload in username field
3. Enter any password
4. Submit the form
5. Monitor automated test response

Expected Result:

- Automated test should detect Cloudflare Captcha challenge
- Test should report that security protection is active
- Test framework should handle Captcha scenario appropriately

Actual Result:

- Cloudflare Captcha appears (security protection working)
- Automated test framework does not detect Captcha
- Test may incorrectly report vulnerability or incomplete results

Impact:

- **Testing Accuracy:** Incomplete security test coverage
- **False Reporting:** May incorrectly indicate vulnerabilities
- **Security Assessment:** Underestimation of actual protection levels

Acceptance Criteria for Fix:

- [ ] Update automated tests to detect Cloudflare Captcha responses
- [ ] Implement Captcha detection in testing framework
- [ ] Add proper reporting when security protections trigger

- [ ] Create manual testing procedures for Captcha scenarios

Test Cases: TC_017-TC_020

PARA-005: Automatic login after registration violates security best practices

Type: Improvement
Priority: Medium
Severity: Medium
Component: User Registration, Authentication
Labels: ux, security, registration-flow
Reporter: QA Team
Assignee: Frontend Team

Summary: Users are automatically logged in after registration without explicit consent or notification, violating security best practices.

Description: After completing the registration process, users are automatically authenticated and logged into the system without explicit action or consent. This behavior creates security concerns and may confuse users who expect to manually authenticate after registration.

Steps to Reproduce:

1. Navigate to registration page
2. Complete registration form with valid data
3. Submit registration form
4. Observe post-registration behavior

Expected Result:

- User should be redirected to login page after successful registration
- Confirmation message should indicate successful account creation
- User should manually authenticate using new credentials
- Clear separation between registration and authentication processes

Actual Result:

- User is automatically logged in immediately after registration
- Login form fields become invisible/hidden
- User is redirected to accounts overview without explicit login

- No clear indication of automatic authentication

Impact:

- **Security Concern:** Automatic session creation without user consent
- **User Experience:** Unexpected behavior may confuse users
- **Audit Trail:** Unclear distinction between registration and login events
- **Best Practices:** Violates standard registration flow patterns

Acceptance Criteria for Fix:

- [ ] Redirect to login page after successful registration
- [ ] Display success message for account creation
- [ ] Require explicit authentication after registration
- [ ] Maintain clear audit trail of registration vs login events
- [ ] Update user interface to reflect proper flow

Test Cases: TC_036-TC_038

PARA-006: Account creation modal interrupts user workflow after first login

Type: Bug
Priority: Low
Severity: Low
Component: User Experience, Account Management
Labels: ux, modal, workflow, first-login
Reporter: QA Team
Assignee: Frontend Team

Summary: New users encounter unexpected account creation modal after first login that disrupts normal application flow.

Description: After a new user completes their first login, an unexpected modal dialog appears requiring account creation before accessing main banking features. This interrupts the expected user workflow and may cause confusion.

Steps to Reproduce:

1. Register a new user account
2. Complete first login (either automatic or manual)

3. Observe the user interface behavior

Expected Result:

- Direct access to banking dashboard after login
- Smooth transition to main application features
- No unexpected modal interruptions
- Intuitive user experience flow

Actual Result:

- Modal dialog appears requiring account creation
- User must complete additional steps before accessing features
- Workflow is interrupted unexpectedly
- May cause confusion for new users

Impact:

- **User Experience:** Interrupts expected workflow
- **Confusion:** Unexpected step in user journey
- **Automation:** May interfere with automated testing
- **Onboarding:** Complicates new user experience

Acceptance Criteria for Fix:

- [ ] Remove unexpected modal or integrate into smooth flow
- [ ] Ensure direct access to dashboard after login
- [ ] Improve new user onboarding experience
- [ ] Update user interface for consistent behavior

Test Cases: Not specified in original report

PARA-007: Misleading error messages during registration validation

Type: Bug
Priority: Low
Severity: Low
Component: Error Handling, User Experience
Labels: error-messages, validation, ux

Reporter: QA Team
Assignee: Frontend Team

Summary: Registration form displays incorrect error message "This username already exists" for various validation failures.

Description: When registration fails due to invalid data in any field, the system incorrectly displays "This username already exists" error message, even when the username is unique and the actual issue is with other field validation.

Steps to Reproduce:

1. Navigate to registration page
2. Enter a unique username (not previously used)
3. Fill other fields with invalid data (invalid phone, SSN, etc.)
4. Submit the form
5. Observe the error message

Expected Result:

- Specific error message indicating the actual validation failure
- Clear feedback about which field contains invalid data
- Accurate error message corresponding to the real issue
- Helpful guidance for user to correct the problem

Actual Result:

- Generic "This username already exists" message appears
- No indication of the actual validation problem
- Misleading feedback confuses users
- No guidance on how to fix the real issue

Impact:

- **User Experience:** Misleading and unhelpful error messages
- **Debugging:** Difficult for users to identify real problems
- **Support:** May increase customer service requests
- **Trust:** Inconsistent error handling reduces user confidence

Acceptance Criteria for Fix:

- [ ] Implement specific error messages for each validation rule
- [ ] Display accurate feedback corresponding to actual issues
- [ ] Provide helpful guidance for fixing validation errors
- [ ] Ensure error messages match the underlying validation logic

Test Cases: TC_041-TC_050

PARA-008: Extended validation failures - Critical format validation missing for financial identifiers

Type: Bug
Priority: High
Severity: High
Component: Input Validation, Data Integrity
Labels: security, validation, financial-data, compliance
Reporter: QA Team
Assignee: Backend Team

Summary: Extended validation testing reveals critical format validation failures for phone numbers and SSN fields, allowing invalid financial identifiers in banking system.

Description: Additional testing beyond the initial scope discovered severe validation gaps for critical financial identifiers. The system accepts improperly formatted phone numbers and Social Security Numbers, creating data integrity issues and potential compliance violations for financial institutions.

Specific Validation Failures:

Phone Number Issues:

- Accepts alphabetic characters: `555-PHONE` (should be numeric only)
- Accepts invalid format patterns
- No enforcement of standard phone number formats

SSN Critical Issues:

- Accepts incomplete SSN: `123` (should be 9 digits)
- Accepts unformatted SSN: `123456789` (should be `XXX-XX-XXXX`)
- Accepts alphabetic SSN: `ABC-DE-FGHI` (should be numeric only)
- No validation of SSN format requirements

Steps to Reproduce:

1. Navigate to user registration page
2. Fill First Name and Last Name with valid data
3. Enter invalid phone number: `555-PHONE`
4. Enter invalid SSN (any of the examples above):
   - `123` (too short)
   - `123456789` (missing hyphens)
   - `ABC-DE-FGHI` (alphabetic)
5. Complete other fields with valid data
6. Submit registration form

Expected Result:

- Phone number validation should reject non-numeric characters
- SSN validation should enforce `XXX-XX-XXXX` format
- Specific error messages should indicate format requirements
- Registration should fail with invalid format data

Actual Result:

- Invalid phone formats are accepted without validation
- All SSN format violations are accepted
- User account is created with invalid financial identifiers
- No error messages or format guidance provided

Impact:

- **Compliance Risk:** Invalid SSNs violate financial regulations
- **Data Integrity:** Corrupted customer identification data
- **Business Risk:** Invalid contact information affects customer communication
- **Audit Issues:** Regulatory compliance violations for banking systems

Acceptance Criteria for Fix:

- [ ] Implement strict phone number format validation
- [ ] Enforce SSN format validation (XXX-XX-XXXX)
- [ ] Add SSN checksum validation if required by regulation

- [ ] Implement real-time format validation feedback
- [ ] Create specific error messages for each format violation
- [ ] Add input masks to guide proper format entry

Test Cases: TC_051-TC_054

PARA-009: Critical session security vulnerabilities - Multiple concurrent sessions and missing timeout

Type: Bug
Priority: Highest
Severity: Critical
Component: Session Management, Security
Labels: security, session-management, concurrent-sessions, timeout
Reporter: QA Team
Assignee: Backend Security Team

Summary: Session management system allows multiple concurrent sessions and lacks automatic timeout, creating serious security vulnerabilities for banking application.

Description: The session management system has fundamental security flaws that allow multiple concurrent sessions for the same user and fails to implement automatic session timeout. These vulnerabilities create significant security risks in a banking environment where strict session control is essential.

Critical Session Security Issues:

Concurrent Session Problem:

- Same user can log in from multiple browsers simultaneously
- All sessions remain active and functional
- No session conflict detection or resolution
- Previous sessions are not invalidated when new login occurs

Session Timeout Vulnerability:

- No automatic logout after periods of inactivity
- Sessions persist indefinitely without user interaction
- No configurable timeout policies
- No warning before session expiration

Steps to Reproduce:

Concurrent Sessions Test:

1. Open ParaBank in first browser tab
2. Log in with user credentials (e.g., `user1` / `password1`)
3. Open ParaBank in second browser tab (different tab/window)
4. Log in with same credentials in second tab
5. Verify both sessions remain active
6. Perform actions in both tabs simultaneously

Session Timeout Test:

1. Log in to ParaBank
2. Navigate to accounts overview
3. Leave session inactive for extended period (>30 minutes)
4. Return and attempt to perform banking operations
5. Observe session status

Expected Result:

- **Concurrent Sessions:** Second login should invalidate first session
- **Single Session Policy:** Only one active session per user allowed
- **Timeout:** Session should automatically expire after inactivity
- **Security Warning:** User should be notified of session timeout

Actual Result:

- **Multiple Sessions:** Both sessions remain fully functional
- **No Timeout:** Sessions persist indefinitely without expiration
- **No Security Controls:** No detection or prevention of concurrent access
- **Persistent Access:** Unlimited session duration without timeout

Impact:

- **Security Risk:** Session hijacking vulnerability
- **Unauthorized Access:** Persistent access after user leaves workstation
- **Compliance Violation:** Banking regulations require session timeout

- **Account Compromise:** Risk of unauthorized financial transactions

Acceptance Criteria for Fix:

- [ ] Implement single session policy per user
- [ ] Add configurable session timeout (recommend 15-30 minutes for banking)
- [ ] Invalidate previous sessions on new login
- [ ] Implement session timeout warning (5 minutes before expiration)
- [ ] Add "extend session" functionality for active users
- [ ] Implement secure session invalidation on logout

Test Cases: TC_008-TC_015

PARA-010: API security vulnerabilities - Unprotected endpoints and missing security controls

Type: Bug
Priority: High
Severity: High
Component: API Security, Backend Services
Labels: security, api, authentication, rate-limiting
Reporter: QA Team
Assignee: Backend Security Team, API Team

Summary: API endpoints lack essential security controls including authentication, input validation, rate limiting, and proper security headers.

Description: Security assessment of ParaBank API endpoints reveals multiple critical vulnerabilities that could allow direct system exploitation. The API lacks fundamental security controls expected in banking applications, creating significant attack vectors for malicious actors.

API Security Vulnerabilities:

Authentication Bypass:

- API endpoints accessible without proper authentication tokens
- No OAuth or JWT token validation
- Missing API key requirements
- Unauthenticated access to sensitive banking data

Input Sanitization Failures:

- API accepts malicious payloads without validation
- No input filtering for injection attacks
- Missing data type validation
- Insufficient parameter sanitization

Missing Rate Limiting:

- No throttling or request rate limiting
- API vulnerable to brute force attacks
- No protection against DDoS attacks
- Unlimited API calls allowed per client

Steps to Reproduce:

Authentication Bypass Test:

1. Identify ParaBank API endpoints (through browser developer tools)
2. Send direct API requests without authentication headers
3. Access sensitive endpoints (account data, transaction history)
4. Verify unauthorized data access

Input Validation Test:

1. Send malicious payloads to API endpoints:
   - SQL injection: `{"username": "admin' OR '1'='1"}`
   - XSS payload: `{"data": "<script>alert('xss')</script>"}`
2. Monitor API response for security filtering

Rate Limiting Test:

1. Create automated script to send rapid API requests
2. Send 1000+ requests in short time period
3. Verify if rate limiting blocks excessive requests

Expected Result:

- **Authentication:** All API endpoints should require valid authentication
- **Input Validation:** Malicious payloads should be rejected with security errors
- **Rate Limiting:** Excessive requests should be throttled or blocked

- **Security Headers:** Proper security headers should be present in all responses

Actual Result:

- **No Authentication:** API endpoints accessible without proper credentials
- **No Input Filtering:** Malicious payloads processed without sanitization
- **No Rate Limits:** Unlimited API requests allowed
- **Missing Headers:** Security headers absent from API responses

Impact:

- **Direct API Exploitation:** Attackers can access banking data directly
- **Data Exfiltration:** Sensitive customer information at risk
- **System Compromise:** API attacks could compromise entire system
- **DDoS Vulnerability:** API can be overwhelmed by malicious requests

Acceptance Criteria for Fix:

- [ ] Implement API authentication (OAuth 2.0 or JWT tokens)
- [ ] Add comprehensive input validation for all API endpoints
- [ ] Implement rate limiting (e.g., 100 requests per minute per client)
- [ ] Add proper CORS configuration
- [ ] Implement API security headers (CSP, X-Frame-Options, etc.)
- [ ] Add API request/response logging for security monitoring

Test Cases: API Security Test Suite

WORKING FUNCTIONALITY (For Reference)

Validation Rules That Work Correctly:

- ✅ Empty First Name validation (TC_039) - Properly rejects empty field
- ✅ Empty Last Name validation (TC_040) - Properly rejects empty field

Note: These are the only validation rules functioning correctly in the application.

SUMMARY FOR DEVELOPMENT TEAM

Total Issues Identified: 10 defects across security and functional categories

Priority Breakdown:

- **Highest Priority:** 3 critical security vulnerabilities (PARA-001, PARA-002, PARA-009)
- **High Priority:** 3 security/validation issues (PARA-003, PARA-008, PARA-010)
- **Medium Priority:** 1 functional issue (PARA-005)
- **Low Priority:** 3 user experience and testing issues (PARA-004, PARA-006, PARA-007)

Immediate Actions Required:

1. **Security Team:** Address critical authentication and session management vulnerabilities (PARA-001, PARA-002, PARA-009)
2. **Backend Team:** Implement comprehensive input validation and API security (PARA-003, PARA-008, PARA-010)
3. **Frontend Team:** Fix user experience issues (PARA-005, PARA-006, PARA-007)
4. **QA Automation Team:** Enhance testing framework to handle Captcha scenarios (PARA-004)

Extended Testing Results: All issues discovered using Playwright + Cucumber automated testing framework with 152 comprehensive test scenarios achieving 98.68% success rate.

Environment: ParaBank Demo Application - https://parabank.parasoft.com/parabank/

Note: While ParaBank is a demo application for training purposes, these defects represent real-world vulnerabilities that must be avoided in production banking systems.