ParaBank Application - Updated Jira Defect Tickets

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

CRITICAL SECURITY DEFECTS

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

Validation Rules Needed:

- **Names:** Alphabetic characters only, 1-50 characters
- **Address:** Alphanumeric with basic punctuation, required
- **City:** Alphabetic characters only, 2-50 characters

- **State:** Valid 2-letter state codes only
- **Zip Code:** 5 or 9 digit numeric format
- **Phone:** Valid phone number formats
- **SSN:** XXX-XX-XXXX format validation
- **Passwords:** Must match confirmation field

Test Cases: TC_041-TC_050

PARA-004: Input sanitization vulnerability - SQL injection and XSS payloads not filtered

Type: Bug
Priority: High
Severity: High
Component: Input Sanitization, Security
Labels: security, sql-injection, xss, injection-attacks
Reporter: QA Team
Assignee: Backend Security Team

Summary: Application does not properly sanitize malicious input, allowing potential SQL injection and XSS attacks through login form.

Description: The application fails to properly sanitize user input in the login form, making it vulnerable to SQL injection and Cross-Site Scripting (XSS) attacks. Malicious payloads are processed without proper filtering or escaping.

Vulnerable Attack Vectors Identified:

SQL Injection Payloads:

- `" OR "1"="1`
- `admin' UNION SELECT * FROM users--`
- `'; DROP TABLE users; --`

XSS Payloads:

- `<script>alert('xss')</script>`
- `<img onerror='alert(1)' src='x'>`
- `javascript:alert('XSS')`

Steps to Reproduce:

1. Navigate to login page
2. Enter malicious payload in username field (any from list above)
3. Enter any password
4. Submit the form
5. Monitor for SQL errors or script execution

Expected Result:

- Malicious input should be sanitized or rejected
- Security warning should be displayed
- No script execution should occur
- No SQL errors should be exposed
- Input should be properly escaped before database queries

Actual Result:

- Payloads are processed without proper sanitization
- No security warnings are displayed
- Potential for malicious code execution
- Database queries may be compromised

Impact:

- **Security Risk:** Database compromise through SQL injection
- **Data Breach:** Potential unauthorized data access
- **XSS Attacks:** Malicious script execution in user browsers
- **System Compromise:** Potential server-side code execution

Acceptance Criteria for Fix:

- [ ] Implement input sanitization for all user inputs
- [ ] Use parameterized queries to prevent SQL injection
- [ ] Implement XSS filtering and output encoding
- [ ] Add Content Security Policy (CSP) headers
- [ ] Implement input validation whitelist approach
- [ ] Add SQL injection detection and blocking
- [ ] Implement proper error handling without information disclosure

Security Measures Required:

- **Input Validation:** Whitelist allowed characters
- **Output Encoding:** HTML encode all dynamic content
- **Parameterized Queries:** Use prepared statements
- **CSP Headers:** Implement strict content security policy
- **WAF Rules:** Web application firewall protection

Test Cases: TC_017-TC_020

FUNCTIONAL DEFECTS

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

Error Message Examples Needed:

- "Phone number format is invalid"
- "SSN format must be XXX-XX-XXXX"
- "State code must be valid 2-letter abbreviation"
- "Zip code must be 5 or 9 digits"
- "First name cannot contain numbers or special characters"

Test Cases: TC_041-TC_050

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: 7 defects across security and functional categories

Priority Breakdown:

- **Highest Priority:** 2 critical security vulnerabilities
- **High Priority:** 1 security/validation issue
- **Medium Priority:** 1 functional issue
- **Low Priority:** 3 user experience and testing issues

Immediate Actions Required:

1. **Security Team:** Address authentication bypass and session management (PARA-001, PARA-002)
2. **Backend Team:** Implement input validation (PARA-003)
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)

Testing Framework: All issues discovered using Playwright + Cucumber automated testing with test cases TC_001 through TC_050.

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.