
CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

# CEBU INSTITUTE OF TECHNOLOGY UNIVERSITY

COLLEGE OF COMPUTER STUDIES


## Software Test Document

*for*

Darknet Duel

# Change History

| Version | Date | Description | Author |
|---|---|---|---|
| 1.0 | 20 September, 2025 | Initial Document | Entirety of Team 27 of Capstone 1 |

# Table of Contents


**Change History** ..... **2**
**Table of Contents** ..... **3**
**1. Introduction** ..... **6**
1.1. System Overview ..... 6
1.2. Test Approach ..... 6
1.3. Definitions and Acronyms ..... 6
**2. Test Plan** ..... **8**
2.1 Features to be Tested ..... 8
2.2 Features not to be Tested ..... 9
2.3 Testing Tools and Environment ..... 9
**3. Test Cases** ..... **10**
3.1 (DD-01-001) User Registration ..... 10
*3.1.1 Purpose* ..... *10*
*3.1.2 Inputs* ..... *10*
*3.1.3 Expected Outputs & Pass/Fail Criteria* ..... *10*
*3.1.4 Test Procedure* ..... *11*
3.2 (DD-01-002) User Login ..... 12
*3.2.1 Purpose* ..... *12*
*3.2.2 Inputs* ..... *12*
*3.2.3 Expected Outputs & Pass/Fail Criteria* ..... *12*
*3.2.4 Test Procedure* ..... *13*
3.3 (DD-01-003) Profile Management ..... 14
*3.3.1 Purpose* ..... *14*
*3.3.2 Inputs* ..... *14*
*3.3.3 Expected Outputs & Pass/Fail Criteria* ..... *15*
*3.3.4 Test Procedure* ..... *15*
3.4 (DD-01-004) Role-based Access ..... 17
*3.4.1 Purpose* ..... *17*
*3.4.2 Inputs* ..... *17*
*3.4.3 Expected Outputs & Pass/Fail Criteria* ..... *17*
*3.4.4 Test Procedure* ..... *17*
3.5 (DD-02-001) Lobby Browser ..... 18
*3.5.1 Purpose* ..... *18*
*3.5.2 Inputs* ..... *18*
*3.5.3 Expected Outputs & Pass/Fail Criteria* ..... *19*
*3.5.4 Test Procedure* ..... *19*
3.6 (DD-02-002) Create/Join/Leave Lobbies ..... 21
*3.6.1 Purpose* ..... *21*
*3.6.2 Inputs* ..... *21*
*3.6.3 Expected Outputs & Pass/Fail Criteria* ..... *21*
*3.6.4 Test Procedure* ..... *22*
3.7 (DD-02-003) Real-Time Lobby Updates ..... 23
*3.7.1 Purpose* ..... *23*
*3.7.2 Inputs* ..... *24*
*3.7.3 Expected Outputs & Pass/Fail Criteria* ..... *24*
*3.7.4 Test Procedure* ..... *24*
3.8 (DD-02-004) Lobby Chat ..... 26
*3.8.1 Purpose* ..... *26*
*3.8.2 Inputs* ..... *26*
*3.8.3 Expected Outputs & Pass/Fail Criteria* ..... *26*
*3.8.4 Test Procedure* ..... *26*
3.9 (DD-03-001) Game Creation and Initialization ..... 28
*3.9.1 Purpose* ..... *28*
*3.9.2 Inputs* ..... *28*
*3.9.3 Expected Outputs & Pass/Fail Criteria* ..... *28*
*3.9.4 Test Procedure* ..... *29*
3.10 (DD-03-002) Turned-Based Gameplay, AP Allocation, Card Play, and Targeting ..... 30
*3.10.1 Purpose* ..... *30*
*3.10.2 Inputs* ..... *30*
*3.10.3 Expected Outputs & Pass/Fail Criteria* ..... *31*

*3.10.4 Test Procedure* ... *31*
3.11 (DD-03-003) Real-Time State Synchronization ... 33
*3.11.1 Purpose* ... *33*
*3.11.2 Inputs* ... *33*
*3.11.3 Expected Outputs & Pass/Fail Criteria* ... *33*
*3.11.4 Test Procedure* ... *33*
3.12 (DD-03-004) Disconnection/Reconnection Handling ... 35
*3.12.1 Purpose* ... *35*
*3.12.2 Inputs* ... *35*
*3.12.3 Expected Outputs & Pass/Fail Criteria* ... *35*
*3.12.4 Test Procedure* ... *35*
3.13 (DD-03-005) Game State Persistence and Recovery ... 37
*3.13.1 Purpose* ... *37*
*3.13.2 Inputs* ... *37*
*3.13.3 Expected Outputs & Pass/Fail Criteria* ... *37*
*3.13.4 Test Procedure* ... *38*
3.14 (DD-04-001) Card Play, Targeting, and Effect Resolution ... 39
*3.14.1 Purpose* ... *39*
*3.14.2 Inputs* ... *39*
*3.14.3 Expected Outputs & Pass/Fail Criteria* ... *39*
*3.14.4 Test Procedure* ... *40*
3.15 (DD-04-002) Infrastructure State Tracking ... 41
*3.15.1 Purpose* ... *41*
*3.15.2 Inputs* ... *41*
*3.15.3 Expected Outputs & Pass/Fail Criteria* ... *42*
*3.15.4 Test Procedure* ... *42*
3.16 (DD-04-003) Game Rules Enforcement ... 43
*3.16.1 Purpose* ... *43*
*3.16.2 Inputs* ... *44*
*3.16.3 Expected Outputs & Pass/Fail Criteria* ... *44*
*3.16.4 Test Procedure* ... *44*
3.17 (DD-04-004) Game State Visualization ... 46
*3.17.1 Purpose* ... *46*
*3.17.2 Inputs* ... *46*
*3.17.3 Expected Outputs & Pass/Fail Criteria* ... *46*
*3.17.4 Test Procedure* ... *46*
3.18 (DD-05-001) Match Result Display ... 48
*3.18.1 Purpose* ... *48*
*3.18.2 Inputs* ... *48*
*3.18.3 Expected Outputs & Pass/Fail Criteria* ... *48*
*3.18.4 Test Procedure* ... *49*
3.19 (DD-05-002) Player Performance Statistics ... 50
*3.19.1 Purpose* ... *50*
*3.19.2 Inputs* ... *50*
*3.19.3 Expected Outputs & Pass/Fail Criteria* ... *51*
*3.19.4 Test Procedure* ... *51*
3.20 (DD-05-003) Match History Storage and Browsing ... 52
*3.20.1 Purpose* ... *52*
*3.20.2 Inputs* ... *52*
*3.20.3 Expected Outputs & Pass/Fail Criteria* ... *53*
*3.20.4 Test Procedure* ... *53*
3.21 (DD-06-001) In-Game Currency Management ... 55
*3.21.1 Purpose* ... *55*
*3.21.2 Inputs* ... *55*
*3.21.3 Expected Outputs & Pass/Fail Criteria* ... *55*
*3.21.4 Test Procedure* ... *56*
3.22 (DD-06-002) Store Browsing, Item Purchase, and Application of Decoration ... 57
*3.22.1 Purpose* ... *57*
*3.22.2 Inputs* ... *57*
*3.22.3 Expected Outputs & Pass/Fail Criteria* ... *57*
*3.22.4 Test Procedure* ... *58*
3.23 (DD-06-003) Payment Integration ... 59
*3.23.1 Purpose* ... *59*
*3.23.2 Inputs* ... *59*
*3.23.3 Expected Outputs & Pass/Fail Criteria* ... *60*

*3.23.4 Test Procedure* ... *60*
3.24 (DD-07-001) User Search, Ban, and Moderation ... 62
*3.24.1 Purpose* ... *62*
*3.24.2 Inputs* ... *62*
*3.24.3 Expected Outputs & Pass/Fail Criteria* ... *62*
*3.24.4 Test Procedure* ... *62*
3.25 (DD-07-002) Report Management ... 64
*3.25.1 Purpose* ... *64*
*3.25.2 Inputs* ... *64*
*3.25.3 Expected Outputs & Pass/Fail Criteria* ... *65*
*3.25.4 Test Procedure* ... *65*
3.26 (DD-07-003) System Logs and Audit Trails ... 67
*3.26.1 Purpose* ... *67*
*3.26.2 Inputs* ... *67*
*3.26.3 Expected Outputs & Pass/Fail Criteria* ... *67*
*3.26.4 Test Procedure* ... *67*
3.27 (DD-07-004) User Modification ... 69
*3.27.1 Purpose* ... *69*
*3.27.2 Inputs* ... *69*
*3.27.3 Expected Outputs & Pass/Fail Criteria* ... *70*
*3.27.4 Test Procedure* ... *70*
3.28 (DD-08-001) Lore Video Playback ... 72
*3.28.1 Purpose* ... *72*
*3.28.2 Inputs* ... *72*
*3.28.3 Expected Outputs & Pass/Fail Criteria* ... *72*
*3.28.4 Test Procedure* ... *72*
3.29 (DD-08-002) Interactive Tutorial ... 74
*3.29.1 Purpose* ... *74*
*3.29.2 Inputs* ... *74*
*3.29.3 Expected Outputs & Pass/Fail Criteria* ... *74*
*3.29.4 Test Procedure* ... *75*
**Appendix (Test Logs) ... 78**
A.1 Test Logs ... 78
A.2 Test Results ... 78
A.3 Incident Report ... 78

# 1. Introduction

## *1.1. System Overview*

The Darknet Duel system is a cybersecurity-themed web-based card game that supports real-time multiplayer gameplay. The system consists of three main components:

- **Frontend Application**: React/TypeScript-based single-page application providing the user interface
- **Backend Server**: Node.js/Express REST API handling authentication, user management, store, and admin functions
- **Game Server**: boardgame.io-powered server managing real-time multiplayer game logic

The testing scope focuses on the frontend application as it is the primary interface through which testers will interact with the system. The frontend provides all user-facing functionality including authentication, lobby management, gameplay, store operations, and administrative tools.

## *1.2. Test Approach*

This test plan follows a **manual testing approach** using black-box testing techniques. The testing strategy includes:

- **Functional Testing**: Verification that each module's transactions work as specified in the SRS
- **User Interface Testing**: Validation of UI components, navigation, and user experience
- **Integration Testing**: Testing the interaction between frontend and backend services
- **Boundary Value Analysis (BVA)**: Testing edge cases and boundary conditions
- **Equivalence Partitioning**: Testing representative values from different input classes
- **Error Handling Testing**: Verification of error messages and recovery mechanisms

**Test Management**: All test cases will be tracked using Trello boards, with each module having its own test case file for detailed tracking and documentation.

## *1.3. Definitions and Acronyms*

| Term | Definition |
|---|---|
| STD | Software Test Definition – this document |
| SRS | Software Requirements Specification |
| SDD | Software Design Description |

| Term | Definition |
|---|---|
| BVA | Boundary Value Analysis – testing technique focusing on boundary conditions |
| UI | User Interface |
| API | Application Programming Interface |
| JWT | JSON Web Token – authentication mechanism |
| WebSocket | Real-time communication protocol |
| E2E | End-to-End testing |
| Tester | Person executing the test cases |
| DUT | Device Under Test – the frontend application |
| Pass/Fail | Test result indicating whether the test case passed or failed |

# 2. Test Plan

## *2.1 Features to be Tested*

The following modules and their transactions will be tested:

**Module 1: User Management and Authentication**
- Transaction 1.1: User Registration
- Transaction 1.2: User Login
- Transaction 1.3: Profile Management
- Transaction 1.4: Role-based Access

**Module 2: Lobby and Matchmaking**
- Transaction 2.1: Lobby Browser
- Transaction 2.2: Create/Join/Leave Lobbies
- Transaction 2.3: Real-Time Lobby Updates
- Transaction 2.4: Lobby Chat

**Module 3: Real-Time Multiplayer Game**
- Transaction 3.1: Game Creation and Initialization
- Transaction 3.2: Turn-Based Gameplay, AP Allocation, Card Play, and Targeting
- Transaction 3.3: Real-Time State Synchronization
- Transaction 3.4: Disconnection/Reconnection Handling
- Transaction 3.5: Game State Persistence and Recovery

**Module 4: Card System and Game Logic**
- Transaction 4.1: Card Play, Targeting, and Effect Resolution
- Transaction 4.2: Infrastructure State Tracking
- Transaction 4.3: Game Rules Enforcement
- Transaction 4.4: Game State Visualization

**Module 5: Game Statistics and Result Tracking**
- Transaction 5.1: Match Result Display
- Transaction 5.2: Player Performance Statistics
- Transaction 5.3: Match History Storage and Browsing

**Module 6: Store and Currency**
- Transaction 6.1: In-Game Currency Management
- Transaction 6.2: Store Browsing, Item Purchase, and Application of Decoration
- Transaction 6.3: Payment Integration

**Module 7: Admin and Moderation Tools**
- Transaction 7.1: User Search, Ban, and Moderation
- Transaction 7.2: Report Management
- Transaction 7.3: System Logs and Audit Trails
- Transaction 7.4: User Modification

**Module 8: First-Time Experience**

- Transaction 8.1: Lore Video Playback
- Transaction 8.2: Interactive Tutorial

## *2.2 Features not to be Tested*

The following are explicitly out of scope for this testing phase:

- Backend server internal logic and database operations
- Game server internal game logic and state management
- Network infrastructure and server configuration
- Performance testing and load testing
- Security penetration testing (beyond basic input validation)
- Cross-browser compatibility testing
- Mobile device testing
- Third-party service integration testing (payment processing, external APIs)
- Code-level unit testing and integration testing
- Automated test script development

## *2.3 Testing Tools and Environment*

**Testing Environment:**

- **Browser**: Chrome (latest stable version)
- **Operating System**: Windows 10/11
- **Screen Resolution**: 1920x1080 (minimum)
- **Network**: Stable internet connection
- **Test Data**: Pre-configured test accounts with different user roles

**Testing Tools:**

- **Trello**: Test case tracking and management
- **Browser Developer Tools**: For debugging and inspection
- **Screenshot Tool**: For capturing test evidence
- **Test Data Management**: Pre-created test accounts and game data

**Test Environment Setup:**

1. Access the Darknet Duel frontend application
2. Ensure backend and game servers are running
3. Clear browser cache and cookies before each test session
4. Use incognito/private browsing mode for clean test sessions

# 3. Test Cases

## 3.1 (DD-01-001) User Registration

### 3.1.1 Purpose

To verify that new users can successfully create accounts with valid information and that the system properly validates input data and handles errors.

### 3.1.2 Inputs

**Valid Inputs (Equivalence Partitioning):**

- Email: Valid email format (e.g., "test@example.com")
- Username: 3-50 characters (e.g., "testuser123")
- Password: 8+ characters with mixed case and numbers (e.g., "Password123")
- Confirm Password: Matches password exactly

**Invalid Inputs (Boundary Value Analysis):**

- Email: Invalid formats ("invalid-email", "@domain.com", "user@", "user@domain")
- Username: Too short (1-2 chars), too long (51+ chars), special characters
- Password: Too short (1-7 chars), no numbers, no uppercase, no lowercase
- Confirm Password: Mismatched passwords

### 3.1.3 Expected Outputs & Pass/Fail Criteria

**Pass Criteria:**

- Registration form displays with proper styling and cyberpunk theme
- Form validation works in real-time as user types
- Valid registration creates account and auto-logs in user
- Success message displays and redirects to dashboard
- User data is properly stored and accessible

**Fail Criteria:**

- Form accepts invalid data
- Validation errors not displayed properly
- Registration fails with valid data
- User not logged in after successful registration
- UI/UX issues or styling problems

### *3.1.4 Test Procedure*

1. **Navigate to Registration**
    - Open browser and go to Darknet Duel application
    - Click "CREATE_NEW_IDENTITY" link on login form
    - Verify registration form loads with cyberpunk styling
    - Verify form fields: Email, Username, Password, Confirm Password
2. **Test Valid Registration**
    - Enter valid email: "testuser@example.com"
    - Enter valid username: "testuser123"
    - Enter valid password: "Password123"
    - Enter matching confirm password: "Password123"
    - Click "AUTHENTICATE" button
    - Verify loading animation appears
    - Verify success message and redirect to dashboard
    - Verify user is logged in and profile data is accessible
3. **Test Email Validation (BVA)**
    - Test invalid email formats:
        - "invalid-email" (no @)
        - "@domain.com" (no username)
        - "user@" (no domain)
        - "user@domain" (no TLD)
    - Verify real-time validation shows error messages
    - Verify form submission is blocked
4. **Test Username Validation (BVA)**
    - Test boundary values:
        - 1 character: "a" (too short)
        - 2 characters: "ab" (too short)
        - 3 characters: "abc" (valid minimum)
        - 50 characters: "a".repeat(50) (valid maximum)
        - 51 characters: "a".repeat(51) (too long)
    - Test special characters: "user@123" (invalid)
    - Verify appropriate error messages
5. **Test Password Validation (BVA)**
    - Test boundary values:
        - 7 characters: "Pass123" (too short)
        - 8 characters: "Password123" (valid minimum)
    - Test missing requirements:

        - No uppercase: "password123"
        - No lowercase: "PASSWORD123"
        - No numbers: "Password"
    - Verify real-time validation feedback
6. **Test Password Confirmation**
    - Enter password: "Password123"
    - Enter different confirm password: "Password456"
    - Verify mismatch error message
    - Verify form submission blocked
7. **Test Duplicate Registration**
    - Attempt to register with existing email/username
    - Verify appropriate error message
    - Verify form remains on registration page
8. **Test Form Reset and Navigation**
    - Fill form partially, then switch to login form
    - Switch back to registration form
    - Verify form is cleared/reset
    - Test "NEW USER REGISTRATION" link functionality

## *3.2 (DD-01-002) User Login*

### *3.2.1 Purpose*

To verify that existing users can successfully authenticate with valid credentials and that the system properly handles invalid login attempts.

### *3.2.2 Inputs*

**Valid Inputs:**

- Email: Existing user email (e.g., "testuser@example.com")
- Password: Correct password for the account

**Invalid Inputs (BVA):**

- Email: Non-existent email, invalid format
- Password: Wrong password, empty password
- Both fields empty

### *3.2.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Login form displays with proper cyberpunk styling
- Valid login authenticates user and redirects to dashboard

- Loading animation appears during authentication
- Success message and smooth transition to dashboard
- User session is properly established

**Fail Criteria:**

- Invalid credentials are accepted
- Error messages not displayed for invalid attempts
- Valid login fails
- UI/UX issues or poor error handling

### *3.2.4 Test Procedure*

1. **Navigate to Login**
    - Open browser and go to Darknet Duel application
    - Verify login form is displayed by default
    - Verify cyberpunk styling and terminal-style status message
    - Verify form fields: Email, Password
2. **Test Valid Login**
    - Enter valid email: "testuser@example.com"
    - Enter correct password: "Password123"
    - Click "AUTHENTICATE" button
    - Verify loading animation with "AUTHENTICATING" text
    - Verify success transition and redirect to dashboard
    - Verify user is properly authenticated
3. **Test Invalid Email**
    - Enter non-existent email: "nonexistent@example.com"
    - Enter any password
    - Click "AUTHENTICATE" button
    - Verify error message displays
    - Verify error animation (red flash, pulse effect)
    - Verify form remains on login page
4. **Test Wrong Password**
    - Enter valid email: "testuser@example.com"
    - Enter wrong password: "WrongPassword"
    - Click "AUTHENTICATE" button
    - Verify error message displays
    - Verify error animation and sound effects
    - Verify form remains on login page

5. **Test Empty Fields (BVA)**
    - Leave both fields empty
    - Click "AUTHENTICATE" button
    - Verify validation errors for both fields
    - Test with only email filled
    - Test with only password filled
6. **Test Email Format Validation**
    - Enter invalid email format: "invalid-email"
    - Enter any password
    - Verify real-time validation error
    - Verify form submission is blocked
7. **Test Form Toggle**
    - Click "CREATE_NEW_IDENTITY" link
    - Verify switch to registration form
    - Click back to login form
    - Verify form is properly reset
8. **Test Error Clearing**
    - Trigger login error
    - Start typing in email field
    - Verify error message clears
    - Verify error animation stops

## *3.3 (DD-01-003) Profile Management*

### *3.3.1 Purpose*

To verify that users can view and edit their profile information, including avatar upload and bio updates.

### *3.3.2 Inputs*

**Valid Inputs:**

- Username: 3-50 characters, alphanumeric
- Email: Valid email format
- Bio: Text up to 500 characters
- Avatar: Valid image file (JPG, PNG, GIF) under 2MB

**Invalid Inputs (BVA):**

- Username: Too short/long, special characters
- Email: Invalid format, duplicate email
- Bio: Over 500 characters

- Avatar: Invalid file type, oversized file

### *3.3.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Profile page displays user information correctly
- Edit modal opens with current user data
- Valid updates save successfully
- Avatar upload works with preview
- Success messages display
- Changes reflect immediately in UI

**Fail Criteria:**

- Profile data not displayed correctly
- Edit functionality not working
- Invalid data accepted
- Avatar upload fails
- UI/UX issues

### *3.3.4 Test Procedure*

1. **Access Profile Page**
    - Login with valid credentials
    - Navigate to profile page (via AppBar or direct URL)
    - Verify profile information displays correctly
    - Verify cyberpunk styling and layout
2. **Test Profile Display**
    - Verify username, email, bio display correctly
    - Verify avatar displays (if uploaded)
    - Verify user stats and information
    - Verify role/type display (user/moderator/admin)
3. **Test Edit Profile Modal**
    - Click edit profile button/modal trigger
    - Verify modal opens with current user data pre-filled
    - Verify all editable fields are present
    - Verify modal styling matches cyberpunk theme
4. **Test Username Update**
    - Change username to valid new name: "newusername123"
    - Click save/update button

- Verify success message
- Verify username updates in profile and throughout app
- Test with invalid usernames (too short, special chars)

5. **Test Email Update**
   - Change email to valid new email: "newemail@example.com"
   - Click save/update button
   - Verify success message
   - Verify email updates in profile
   - Test with invalid email format
6. **Test Bio Update**
   - Change bio to valid text: "This is my new bio"
   - Click save/update button
   - Verify success message
   - Verify bio updates in profile
   - Test with very long bio (over 500 chars)
7. **Test Avatar Upload**
   - Click avatar upload button
   - Select valid image file (JPG, PNG, GIF under 2MB)
   - Verify image preview appears
   - Click save/update button
   - Verify avatar updates in profile and throughout app
   - Test with invalid file types and oversized files
8. **Test Password Change**
   - Access password change section
   - Enter current password
   - Enter new password: "NewPassword123"
   - Enter confirm password: "NewPassword123"
   - Click save/update button
   - Verify success message
   - Test with mismatched passwords
9. **Test Validation and Error Handling**
   - Test all invalid inputs
   - Verify appropriate error messages
   - Verify form submission blocked for invalid data
   - Test duplicate email/username scenarios

# 3.4 (DD-01-004) Role-based Access

## 3.4.1 Purpose

To verify that the system properly enforces role-based access control and displays appropriate UI elements based on user roles.

## 3.4.2 Inputs

**Test Accounts:**

- Regular User: Standard player account
- Moderator: Account with moderator privileges
- Admin: Account with admin privileges

## 3.4.3 Expected Outputs & Pass/Fail Criteria

**Pass Criteria:**

- UI elements display based on user role
- Admin/moderator features only visible to appropriate roles
- Role-based navigation works correctly
- Access restrictions properly enforced
- Role indicators display correctly

**Fail Criteria:**

- Regular users can access admin features
- Role indicators not displayed
- Navigation issues based on role
- Access control bypassed

## 3.4.4 Test Procedure

1. **Test Regular User Access**
    - Login with regular user account
    - Verify standard user interface elements
    - Verify admin/moderator features are hidden
    - Verify role indicator shows "USER" or similar
    - Test navigation to restricted pages (should redirect)
2. **Test Moderator Access**
    - Login with moderator account
    - Verify moderator-specific UI elements

- Verify access to moderation tools
- Verify role indicator shows "MODERATOR"
- Verify cannot access admin-only features

3. **Test Admin Access**
    - Login with admin account
    - Verify admin-specific UI elements
    - Verify access to all admin tools
    - Verify role indicator shows "ADMIN"
    - Verify full system access
4. **Test Role-based Navigation**
    - Test navigation to admin pages with different roles
    - Verify appropriate redirects or access
    - Test direct URL access to restricted pages
    - Verify proper error messages for unauthorized access
5. **Test Role Indicators**
    - Verify role tags/badges display correctly
    - Test in profile pages, user lists, and other locations
    - Verify styling and visibility
6. **Test Feature Visibility**
    - Compare UI elements across different role accounts
    - Verify admin tools only visible to admins
    - Verify moderation tools only visible to moderators/admins
    - Verify standard features visible to all roles
7. **Test Access Control Enforcement**
    - Attempt to access restricted features with regular user
    - Verify proper error handling and redirects
    - Test API calls that should be restricted
    - Verify frontend properly handles access denied responses

## *3.5 (DD-02-001) Lobby Browser*

### *3.5.1 Purpose*

To verify that users can view available game lobbies, refresh the lobby list, and navigate to lobby details.

### *3.5.2 Inputs*

**Valid Inputs:**

- Refresh action (button click)
- Lobby selection (clicking on lobby)
- Private lobby code entry (valid format)

**Invalid Inputs (BVA):**

- Empty private lobby code
- Invalid lobby code format
- Non-existent lobby code

### *3.5.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Lobby list displays with proper cyberpunk styling
- Real-time updates show current lobby status
- Refresh functionality works correctly
- Lobby details accessible via click
- Private lobby entry works with valid codes
- Loading states and error handling work properly

**Fail Criteria:**

- Lobby list not displayed or styled incorrectly
- Real-time updates not working
- Refresh functionality broken
- Navigation to lobby details fails
- Private lobby entry not working
- Poor error handling

### *3.5.4 Test Procedure*

1. **Access Lobby Browser**
    - Login with valid credentials
    - Navigate to lobby page
    - Verify lobby browser loads with cyberpunk styling
    - Verify "DARKNET_LOBBIES" header and status indicators
    - Verify connection status shows "SECURE CONNECTION ESTABLISHED"
2. **Test Lobby List Display**
    - Verify lobby list displays available games
    - Verify each lobby shows:
        - Lobby name/ID
        - Player count and capacity
        - Lobby status (waiting, ready, in progress)
        - Host information

        - Join button (if available)
    - Verify cyberpunk styling and visual effects
3. **Test Lobby Status Indicators**
    - Verify different lobby statuses display correctly:
        - Waiting lobbies (can join)
        - Ready lobbies (can join)
        - In-progress lobbies (cannot join)
        - Full lobbies (cannot join)
    - Verify appropriate visual indicators and button states
4. **Test Refresh Functionality**
    - Click refresh button
    - Verify loading indicator appears
    - Verify lobby list updates with current data
    - Test multiple refresh operations
    - Verify refresh works when no lobbies exist
5. **Test Lobby Selection**
    - Click on an available lobby
    - Verify navigation to lobby detail page
    - Test with different lobby types (public/private)
    - Test with different lobby statuses
6. **Test Private Lobby Entry**
    - Enter valid private lobby code
    - Click join button
    - Verify navigation to private lobby
    - Test with invalid lobby codes
    - Test with empty lobby code
    - Verify appropriate error messages
7. **Test Empty State**
    - Test when no lobbies are available
    - Verify appropriate empty state message
    - Verify "Create Lobby" button is available
    - Verify refresh functionality still works
8. **Test Real-time Updates**
    - Open lobby browser in one browser tab
    - Create/join/leave lobbies in another tab
    - Verify lobby list updates automatically
    - Test lobby status changes

- Test player count updates

9. **Test Error Handling**
    - Test with network disconnected
    - Verify appropriate error messages
    - Test reconnection functionality
    - Verify graceful degradation

# *3.6 (DD-02-002) Create/Join/Leave Lobbies*

## *3.6.1 Purpose*

To verify that users can create new lobbies, join existing lobbies, and leave lobbies with proper validation and error handling.

## *3.6.2 Inputs*

**Valid Inputs:**

- Lobby name: 3-50 characters
- Privacy setting: Public or Private
- Game mode: Standard or Custom
- Join action: Clicking join button on available lobby
- Leave action: Clicking leave button

**Invalid Inputs (BVA):**

- Lobby name: Too short (1-2 chars), too long (51+ chars)
- Empty lobby name
- Special characters in lobby name

## *3.6.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Create lobby form displays with proper validation
- Lobby creation succeeds with valid data
- Join functionality works for available lobbies
- Leave functionality works properly
- Real-time updates reflect changes
- Proper error handling for invalid inputs

**Fail Criteria:**

- Lobby creation fails with valid data
- Join functionality not working
- Leave functionality broken

- Validation not working properly
- Real-time updates not reflecting changes

### *3.6.4 Test Procedure*

1. **Test Create Lobby Form**
    - Click "Create Lobby" button
    - Verify create lobby form opens
    - Verify form fields: Lobby Name, Privacy, Game Mode
    - Verify cyberpunk styling and validation
2. **Test Valid Lobby Creation**
    - Enter valid lobby name: "Test Lobby"
    - Select privacy setting: Public
    - Select game mode: Standard
    - Click "Create Lobby" button
    - Verify lobby creation success
    - Verify redirect to lobby detail page
    - Verify user is set as host
3. **Test Lobby Name Validation (BVA)**
    - Test boundary values:
        - 1 character: "a" (too short)
        - 2 characters: "ab" (too short)
        - 3 characters: "abc" (valid minimum)
        - 50 characters: "a".repeat(50) (valid maximum)
        - 51 characters: "a".repeat(51) (too long)
    - Test empty lobby name
    - Test special characters: "Test@Lobby#123"
    - Verify appropriate error messages
4. **Test Private Lobby Creation**
    - Create lobby with privacy set to Private
    - Verify lobby code is generated
    - Verify lobby is not visible in public lobby list
    - Verify lobby code can be shared for joining
5. **Test Join Public Lobby**
    - Find available public lobby
    - Click "Join" button
    - Verify successful join
    - Verify redirect to lobby detail page
    - Verify user appears in lobby player list

6. **Test Join Private Lobby**
    - Get private lobby code
    - Enter code in private lobby entry field
    - Click "Join" button
    - Verify successful join
    - Verify redirect to lobby detail page
7. **Test Join Restrictions**
    - Attempt to join full lobby
    - Attempt to join in-progress lobby
    - Attempt to join lobby user is already in
    - Verify appropriate error messages
    - Verify join buttons are disabled appropriately
8. **Test Leave Lobby**
    - Join a lobby
    - Click "Leave Lobby" button
    - Verify confirmation dialog (if applicable)
    - Verify successful leave
    - Verify redirect to lobby browser
    - Verify user removed from lobby player list
9. **Test Host Leave Lobby**
    - Create a lobby as host
    - Have other players join
    - Leave the lobby as host
    - Verify lobby is closed or host transferred
    - Verify other players are notified
10. **Test Real-time Updates**
    - Open lobby in multiple browser tabs
    - Create/join/leave lobbies
    - Verify all tabs update in real-time
    - Test lobby status changes
    - Test player list updates

## *3.7 (DD-02-003) Real-Time Lobby Updates*

### *3.7.1 Purpose*

To verify that lobby information updates in real-time across all connected clients without requiring manual refresh.

### *3.7.2 Inputs*

**Real-time Events:**

- New lobby creation
- Player joins lobby
- Player leaves lobby
- Lobby status changes
- Lobby deletion/closure

### *3.7.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Lobby list updates automatically without refresh
- Player counts update in real-time
- Lobby status changes reflect immediately
- New lobbies appear automatically
- Closed lobbies disappear automatically
- All connected clients see updates simultaneously

**Fail Criteria:**

- Updates require manual refresh
- Delayed or missing updates
- Inconsistent updates across clients
- Real-time functionality not working

### *3.7.4 Test Procedure*

1. **Test Lobby Creation Updates**
    - Open lobby browser in two browser tabs
    - Create lobby in one tab
    - Verify lobby appears in other tab automatically
    - Verify no refresh required
    - Test with different lobby types
2. **Test Player Join Updates**
    - Create lobby in one tab
    - Join lobby from another tab
    - Verify player count updates in both tabs
    - Verify player list updates in both tabs
    - Test with multiple players joining
3. **Test Player Leave Updates**
    - Have multiple players in lobby
    - Have one player leave

- Verify player count updates in all tabs
- Verify player list updates in all tabs
- Test host leaving scenario

4. **Test Lobby Status Changes**
    - Create lobby and have players join
    - Start the game (change status to in-progress)
    - Verify status updates in lobby browser
    - Verify join buttons become disabled
    - Test status changes back to waiting
5. **Test Lobby Closure Updates**
    - Create lobby with players
    - Close/delete lobby
    - Verify lobby disappears from browser list
    - Verify all connected clients see update
    - Test with different closure scenarios
6. **Test Multiple Simultaneous Updates**
    - Open multiple browser tabs
    - Perform multiple actions simultaneously:
        - Create multiple lobbies
        - Join/leave multiple lobbies
        - Change lobby statuses
    - Verify all updates appear correctly
    - Verify no updates are missed
7. **Test Update Timing**
    - Measure time between action and update
    - Verify updates appear within reasonable time (< 2 seconds)
    - Test with different network conditions
    - Verify updates are consistent across clients
8. **Test Error Recovery**
    - Disconnect network during updates
    - Reconnect network
    - Verify updates resume working
    - Verify missed updates are caught up
    - Test with intermittent connectivity
    -

# *3.8 (DD-02-004) Lobby Chat*

## *3.8.1 Purpose*

To verify that users can send and receive chat messages in lobbies with proper real-time functionality and message handling.

## *3.8.2 Inputs*

**Valid Inputs:**

- Chat message: Text up to 500 characters
- Send action: Click send button or press Enter
- Message types: Player messages, system messages

**Invalid Inputs (BVA):**

- Empty messages
- Messages over 500 characters
- Special characters and formatting

## *3.8.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Chat interface displays with proper styling
- Messages send and receive in real-time
- Message history displays correctly
- System messages appear appropriately
- Message validation works properly
- Chat scrolls automatically to latest messages

**Fail Criteria:**

- Chat interface not displaying
- Messages not sending or receiving
- Real-time updates not working
- Message validation not working
- UI/UX issues with chat

## *3.8.4 Test Procedure*

1. **Access Lobby Chat**
    - Join a lobby
    - Verify chat interface appears
    - Verify cyberpunk styling and layout
    - Verify input field and send button
2. **Test Message Sending**

- Type message: "Hello everyone!"
- Click send button
- Verify message appears in chat
- Verify message shows sender name and timestamp
- Test sending with Enter key

3. **Test Message Receiving**
   - Have another user send message
   - Verify message appears in chat
   - Verify real-time delivery
   - Verify message formatting and styling
4. **Test Message Validation (BVA)**
   - Test empty message (should not send)
   - Test very long message (500+ characters)
   - Test message with special characters
   - Verify appropriate validation messages
   - Verify send button disabled for invalid messages
5. **Test Message History**
   - Send multiple messages
   - Refresh page or reconnect
   - Verify message history is preserved
   - Verify messages display in correct order
   - Test with long message history
6. **Test System Messages**
   - Have player join lobby
   - Verify system message appears: "Player joined"
   - Have player leave lobby
   - Verify system message appears: "Player left"
   - Test other system events
7. **Test Chat Scrolling**
   - Send many messages to fill chat area
   - Verify chat scrolls to latest message
   - Test manual scrolling
   - Verify auto-scroll behavior
8. **Test Multiple Users Chat**
   - Have multiple users in lobby
   - Test messages from different users
   - Verify all messages appear for all users

- Verify user identification in messages
- Test concurrent messaging

9. **Test Chat Error Handling**
    - Test with network disconnected
    - Verify appropriate error messages
    - Test reconnection and message recovery
    - Test with invalid message formats
10. **Test Chat Performance**
    - Send many messages rapidly
    - Verify chat performance remains good
    - Test with long message history
    - Verify memory usage is reasonable

## *3.9 (DD-03-001) Game Creation and Initialization*

### *3.9.1 Purpose*

To verify that games can be created and initialized properly, with correct game state setup and player assignment.

### *3.9.2 Inputs*

**Valid Inputs:**

- Game creation from lobby
- Player joining existing game
- Game configuration settings

**Invalid Inputs (BVA):**

- Invalid game ID
- Missing players
- Corrupted game state

### *3.9.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Game initializes with proper state
- Players are assigned correct roles (attacker/defender)
- Game board displays correctly
- Initial resources and cards are distributed
- Game rules are properly enforced
- Loading screens and transitions work smoothly

**Fail Criteria:**

- Game fails to initialize
- Players not assigned correct roles
- Game board not displaying
- Initial state incorrect
- Loading issues or crashes

### *3.9.4 Test Procedure*

1. **Test Game Creation from Lobby**
    - Create lobby with 2 players
    - Click "Start Game" button
    - Verify loading screen appears
    - Verify game initializes successfully
    - Verify redirect to game board
2. **Test Game State Initialization**
    - Verify game board loads with proper layout
    - Verify both players are present
    - Verify player roles are assigned (attacker/defender)
    - Verify initial action points (AP) are set
    - Verify initial hand size is correct
3. **Test Initial Card Distribution**
    - Verify each player receives starting cards
    - Verify hand size limits are enforced
    - Verify cards are properly shuffled
    - Verify no duplicate cards in hands
4. **Test Infrastructure Setup**
    - Verify infrastructure cards are placed on board
    - Verify initial infrastructure states (secure/vulnerable)
    - Verify proper positioning and layout
    - Verify infrastructure cards are visible to both players
5. **Test Game Rules Initialization**
    - Verify turn order is established
    - Verify game phases are set correctly
    - Verify win conditions are clear
    - Verify game timer starts (if applicable)
6. **Test Player Assignment**
    - Verify attacker is assigned correctly
    - Verify defender is assigned correctly

- Verify role indicators display properly
- Verify role-based UI elements

7. **Test Game Board Layout**
    - Verify cyberpunk styling and theme
    - Verify all UI elements are present
    - Verify responsive layout works
    - Verify game controls are accessible
8. **Test Loading and Transitions**
    - Verify smooth transition from lobby to game
    - Verify loading animations work properly
    - Verify no UI glitches during transition
    - Verify game loads within reasonable time
9. **Test Error Handling**
    - Test with invalid game ID
    - Test with missing players
    - Test with network issues during initialization
    - Verify appropriate error messages
    - Verify graceful error recovery

## *3.10 (DD-03-002) Turned-Based Gameplay, AP Allocation, Card Play, and Targeting*

### *3.10.1 Purpose*

To verify that turn-based gameplay works correctly with proper AP allocation, card playing mechanics, and targeting systems.

### *3.10.2 Inputs*

**Valid Inputs:**

- Card selection from hand
- Target selection for targeted cards
- End turn action
- Card play actions

**Invalid Inputs (BVA):**

- Insufficient AP for card play
- Invalid target selection
- Playing cards out of turn
- Invalid card combinations

### *3.10.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Turn indicators display correctly
- AP allocation works properly
- Card play mechanics function correctly
- Targeting system works for targeted cards
- Turn progression works smoothly
- Game rules are enforced properly

**Fail Criteria:**

- Turn system not working
- AP allocation incorrect
- Card play not working
- Targeting system broken
- Game rules not enforced
- UI/UX issues

### *3.10.4 Test Procedure*

1. **Test Turn Indicator**
    - Verify current player turn is displayed
    - Verify turn indicator updates when turn changes
    - Verify visual indicators for active player
    - Verify turn counter displays correctly
2. **Test AP Allocation**
    - Verify initial AP is set correctly
    - Verify AP updates at start of each turn
    - Verify AP decreases when playing cards
    - Verify AP cannot go below zero
    - Verify AP display updates in real-time
3. **Test Card Selection**
    - Verify cards in hand are clickable
    - Verify playable cards are highlighted
    - Verify unplayable cards are disabled
    - Verify card selection feedback
4. **Test Card Play Mechanics**
    - Select playable card from hand
    - Click play button

- Verify card is played successfully
- Verify AP is deducted correctly
- Verify card is removed from hand
- Verify card effects are applied

5. **Test Targeting System**
    - Play card that requires targeting
    - Verify targeting mode is activated
    - Verify valid targets are highlighted
    - Verify invalid targets are disabled
    - Select valid target
    - Verify target is selected correctly
    - Verify card effect is applied to target
6. **Test Invalid Card Play**
    - Attempt to play card with insufficient AP
    - Verify error message appears
    - Verify card is not played
    - Verify AP is not deducted
    - Test with invalid targets
7. **Test Turn Progression**
    - Complete turn with valid actions
    - Click "End Turn" button
    - Verify turn passes to opponent
    - Verify turn indicator updates
    - Verify AP is allocated to new player
    - Verify hand is drawn (if applicable)
8. **Test Card Effects**
    - Play different types of cards
    - Verify effects are applied correctly
    - Verify visual feedback for effects
    - Verify effect duration and persistence
    - Test with multiple card effects
9. **Test Game Rules Enforcement**
    - Verify hand size limits are enforced
    - Verify turn time limits (if applicable)
    - Verify card play restrictions
    - Verify targeting restrictions
    - Test edge cases and boundary conditions

10. **Test Real-time Updates**
    - Verify all actions update in real-time
    - Verify both players see changes immediately
    - Verify game state synchronization
    - Test with network interruptions

## *3.11 (DD-03-003) Real-Time State Synchronization*

### *3.11.1 Purpose*

To verify that game state updates are synchronized in real-time across all connected clients.

### *3.11.2 Inputs*

**Real-time Events:**

- Card plays
- Turn changes
- AP updates
- Infrastructure state changes
- Game state modifications

### *3.11.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- All game state changes sync in real-time
- Both players see updates simultaneously
- No state inconsistencies between clients
- Smooth synchronization without delays
- Proper handling of concurrent actions

**Fail Criteria:**

- Delayed or missing updates
- State inconsistencies
- Synchronization failures
- Poor real-time performance

### *3.11.4 Test Procedure*

1. **Test Card Play Synchronization**
    - Play card in one client
    - Verify card play appears in other client
    - Verify AP updates in both clients
    - Verify hand updates in both clients

- Test with multiple card plays

2. **Test Turn Change Synchronization**
    - End turn in one client
    - Verify turn indicator updates in both clients
    - Verify AP allocation in both clients
    - Verify UI updates in both clients
    - Test turn progression
3. **Test Infrastructure State Sync**
    - Apply effect to infrastructure card
    - Verify state change appears in both clients
    - Verify visual indicators update
    - Verify state persistence
    - Test multiple infrastructure changes
4. **Test Concurrent Actions**
    - Have both players perform actions simultaneously
    - Verify proper handling of concurrent actions
    - Verify no state conflicts
    - Verify proper turn order enforcement
    - Test with rapid actions
5. **Test State Consistency**
    - Compare game state between clients
    - Verify all values match
    - Verify no discrepancies
    - Test after various actions
    - Verify state remains consistent
6. **Test Update Timing**
    - Measure time between action and update
    - Verify updates appear within 1 second
    - Test with different network conditions
    - Verify consistent timing across clients
7. **Test State Recovery**
    - Disconnect one client temporarily
    - Perform actions in other client
    - Reconnect first client
    - Verify state is synchronized
    - Verify no missed updates
8. **Test Large State Updates**

- Perform multiple actions rapidly
- Verify all updates are received
- Verify no updates are lost
- Test with complex game states
- Verify performance remains good

# 3.12 (DD-03-004) Disconnection/Reconnection Handling

## 3.12.1 Purpose

To verify that the game handles player disconnections and reconnections gracefully without losing game state.

## 3.12.2 Inputs

**Disconnection Scenarios:**

- Network interruption
- Browser crash/close
- Server disconnection
- Temporary connectivity loss

## 3.12.3 Expected Outputs & Pass/Fail Criteria

**Pass Criteria:**

- Game state is preserved during disconnection
- Reconnection restores game state correctly
- Other players are notified of disconnection
- Game continues smoothly after reconnection
- No data loss or corruption

**Fail Criteria:**

- Game state lost during disconnection
- Reconnection fails or corrupts state
- Other players not notified
- Game becomes unplayable
- Data loss or corruption

## 3.12.4 Test Procedure

1. **Test Temporary Disconnection**
    - Disconnect network for 10-30 seconds
    - Verify game state is preserved

- Reconnect network
- Verify game state is restored
- Verify no data loss

2. **Test Browser Refresh**
    - Refresh browser during game
    - Verify game state is restored
    - Verify player reconnects automatically
    - Verify game continues normally
    - Test multiple refreshes
3. **Test Browser Close/Reopen**
    - Close browser during game
    - Reopen browser and navigate to game
    - Verify game state is restored
    - Verify player reconnects
    - Verify game continues
4. **Test Server Disconnection**
    - Simulate server disconnection
    - Verify appropriate error messages
    - Verify reconnection attempts
    - Verify game state recovery
    - Test with different disconnection durations
5. **Test Multiple Disconnections**
    - Have multiple players disconnect
    - Verify game state is preserved
    - Have players reconnect
    - Verify all players rejoin successfully
    - Verify game continues normally
6. **Test Disconnection Notifications**
    - Have one player disconnect
    - Verify other players are notified
    - Verify appropriate UI indicators
    - Verify reconnection notifications
    - Test with different notification types
7. **Test Game Pause/Resume**
    - Verify game pauses during disconnection
    - Verify game resumes after reconnection
    - Verify turn order is maintained

- Verify no actions are lost
- Test with different pause durations

8. **Test State Recovery**
    - Perform various actions before disconnection
    - Disconnect and reconnect
    - Verify all actions are preserved
    - Verify game state is identical
    - Test with complex game states

# *3.13 (DD-03-005) Game State Persistence and Recovery*

## *3.13.1 Purpose*

To verify that game state is properly persisted and can be recovered after interruptions or page refreshes.

## *3.13.2 Inputs*

**Recovery Scenarios:**

- Page refresh
- Browser close/reopen
- Network interruption
- Server restart

## *3.13.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Game state is persisted on server
- State recovery works after interruptions
- No data loss during recovery
- Game continues from correct state
- All players can rejoin successfully

**Fail Criteria:**

- Game state not persisted
- Recovery fails or corrupts state
- Data loss during recovery
- Game cannot be resumed
- Players cannot rejoin

### *3.13.4 Test Procedure*

1. **Test Page Refresh Recovery**
    - Play game for several turns
    - Refresh browser page
    - Verify game state is restored
    - Verify all actions are preserved
    - Verify game continues normally
2. **Test Browser Close/Reopen Recovery**
    - Play game for several turns
    - Close browser completely
    - Reopen browser and navigate to game
    - Verify game state is restored
    - Verify all progress is preserved
3. **Test Network Interruption Recovery**
    - Play game for several turns
    - Disconnect network for extended period
    - Reconnect network
    - Verify game state is restored
    - Verify no data loss
4. **Test Server Restart Recovery**
    - Play game for several turns
    - Restart game server
    - Verify game state is restored
    - Verify all players can rejoin
    - Verify game continues normally
5. **Test Complex State Recovery**
    - Play game with complex state (many cards played, effects active)
    - Perform various recovery scenarios
    - Verify all state is preserved
    - Verify game continues correctly
    - Test with multiple players
6. **Test State Consistency**
    - Compare game state before and after recovery
    - Verify all values match exactly
    - Verify no data corruption
    - Verify all players see same state
    - Test with different game phases

7. **Test Recovery Performance**
    - Measure time to recover game state
    - Verify recovery completes within reasonable time
    - Test with large game states
    - Verify performance is acceptable
8. **Test Edge Case Recovery**
    - Test recovery during critical game moments
    - Test recovery with pending actions
    - Test recovery with active effects
    - Verify all edge cases are handled properly

# *3.14 (DD-04-001) Card Play, Targeting, and Effect Resolution*

## *3.14.1 Purpose*

To verify that card playing mechanics work correctly with proper targeting systems and effect resolution.

## *3.14.2 Inputs*

**Valid Inputs:**

- Playable cards from hand
- Valid target selections
- Card effect parameters

**Invalid Inputs (BVA):**

- Unplayable cards (insufficient AP, wrong phase)
- Invalid targets (wrong type, already targeted)
- Invalid effect parameters

## *3.14.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Cards play correctly with proper validation
- Targeting system works for targeted cards
- Card effects are applied correctly
- Visual feedback is provided for all actions
- Game state updates properly after card play

**Fail Criteria:**

- Cards play without proper validation
- Targeting system not working

- Card effects not applied correctly
- Poor visual feedback
- Game state not updated properly

### *3.14.4 Test Procedure*

1. **Test Basic Card Play**
    - Select playable card from hand
    - Click play button
    - Verify card is played successfully
    - Verify AP is deducted correctly
    - Verify card is removed from hand
    - Verify visual feedback is provided
2. **Test Card Validation**
    - Attempt to play card with insufficient AP
    - Verify error message appears
    - Verify card is not played
    - Verify AP is not deducted
    - Test with different validation rules
3. **Test Targeting System**
    - Play card that requires targeting
    - Verify targeting mode is activated
    - Verify valid targets are highlighted
    - Verify invalid targets are disabled
    - Select valid target
    - Verify target is selected correctly
    - Verify card effect is applied to target
4. **Test Invalid Targeting**
    - Play card that requires targeting
    - Attempt to select invalid target
    - Verify error message appears
    - Verify targeting mode remains active
    - Verify card is not played
    - Test with different invalid targets
5. **Test Card Effects**
    - Play different types of cards
    - Verify effects are applied correctly
    - Verify visual feedback for effects
    - Verify effect duration and persistence

- Test with multiple card effects

6. **Test Effect Resolution**
    - Play card with immediate effect
    - Verify effect is resolved immediately
    - Verify game state is updated
    - Verify visual indicators are updated
    - Test with delayed effects
7. **Test Card Combinations**
    - Play multiple cards in sequence
    - Verify effects stack correctly
    - Verify no conflicts between effects
    - Verify proper effect resolution order
    - Test with complex combinations
8. **Test Card Play Feedback**
    - Verify visual feedback for card play
    - Verify sound effects (if applicable)
    - Verify animation effects
    - Verify UI updates
    - Test with different card types
9. **Test Error Handling**
    - Test with invalid card data
    - Test with corrupted game state
    - Test with network issues
    - Verify appropriate error messages
    - Verify graceful error recovery

## *3.15 (DD-04-002) Infrastructure State Tracking*

### *3.15.1 Purpose*

To verify that infrastructure cards maintain correct states and update properly based on card effects and game events.

### *3.15.2 Inputs*

**Valid Inputs:**

- Card effects that modify infrastructure
- Game events that change infrastructure state

- State transition triggers

**Invalid Inputs (BVA):**

- Invalid state transitions
- Corrupted infrastructure data
- Missing infrastructure cards

### *3.15.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Infrastructure states are tracked correctly
- State changes are applied properly
- Visual indicators update correctly
- State persistence works across turns
- All infrastructure cards maintain correct states

**Fail Criteria:**

- Infrastructure states not tracked
- State changes not applied
- Visual indicators not updating
- State persistence issues
- Infrastructure cards in wrong states

### *3.15.4 Test Procedure*

1. **Test Initial Infrastructure State**
    - Verify all infrastructure cards start in correct state
    - Verify visual indicators are displayed
    - Verify state information is accessible
    - Verify no corrupted states
2. **Test State Transitions**
    - Apply card effect that changes infrastructure state
    - Verify state transition occurs correctly
    - Verify visual indicators update
    - Verify state is persisted
    - Test with different state transitions
3. **Test State Persistence**
    - Change infrastructure state
    - End turn and have opponent play
    - Verify state is maintained
    - Verify no state reversion
    - Test across multiple turns

4. **Test Visual State Indicators**
    - Verify different states have different visual indicators
    - Verify state changes are visually clear
    - Verify indicators are consistent
    - Verify accessibility of state information
    - Test with different visual themes
5. **Test State Validation**
    - Attempt invalid state transitions
    - Verify appropriate error handling
    - Verify state remains valid
    - Verify no state corruption
    - Test with edge cases
6. **Test Multiple Infrastructure Cards**
    - Test state tracking for multiple cards
    - Verify each card maintains independent state
    - Verify no state interference
    - Verify proper state updates
    - Test with complex state combinations
7. **Test State Recovery**
    - Change infrastructure states
    - Disconnect and reconnect
    - Verify states are restored correctly
    - Verify no state loss
    - Test with different recovery scenarios
8. **Test State Synchronization**
    - Change infrastructure state
    - Verify both players see state change
    - Verify real-time updates
    - Verify state consistency
    - Test with concurrent changes

## *3.16 (DD-04-003) Game Rules Enforcement*

### *3.16.1 Purpose*

To verify that game rules are properly enforced throughout gameplay, including AP limits, hand size limits, and turn restrictions.

### *3.16.2 Inputs*

**Valid Inputs:**

- Actions within game rules
- Valid card plays
- Proper turn progression

**Invalid Inputs (BVA):**

- Actions that violate game rules
- Invalid card plays
- Out-of-turn actions

### *3.16.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Game rules are enforced consistently
- Invalid actions are prevented
- Appropriate error messages are shown
- Game state remains valid
- Turn progression works correctly

**Fail Criteria:**

- Game rules not enforced
- Invalid actions allowed
- Poor error handling
- Game state becomes invalid
- Turn progression issues

### *3.16.4 Test Procedure*

1. **Test AP Limit Enforcement**
    - Attempt to play card with insufficient AP
    - Verify action is blocked
    - Verify appropriate error message
    - Verify AP is not deducted
    - Test with different AP amounts
2. **Test Hand Size Enforcement**
    - Attempt to draw card when hand is full
    - Verify action is blocked
    - Verify appropriate error message
    - Verify hand size limit is enforced
    - Test with different hand sizes

3. **Test Turn Order Enforcement**
    - Attempt to play card out of turn
    - Verify action is blocked
    - Verify appropriate error message
    - Verify turn order is maintained
    - Test with different turn scenarios
4. **Test Card Play Restrictions**
    - Attempt to play restricted cards
    - Verify restrictions are enforced
    - Verify appropriate error messages
    - Verify game state remains valid
    - Test with different restrictions
5. **Test Phase Restrictions**
    - Attempt actions in wrong phase
    - Verify phase restrictions are enforced
    - Verify appropriate error messages
    - Verify phase transitions work correctly
    - Test with different phases
6. **Test Win Condition Enforcement**
    - Achieve win condition
    - Verify win is detected correctly
    - Verify game ends appropriately
    - Verify winner is determined correctly
    - Test with different win conditions
7. **Test Rule Validation**
    - Test with invalid game state
    - Verify rules are enforced
    - Verify state is corrected
    - Verify no rule violations
    - Test with edge cases
8. **Test Error Recovery**
    - Trigger rule violation
    - Verify appropriate error handling
    - Verify game state is corrected
    - Verify game continues normally
    - Test with different violations

# *3.17 (DD-04-004) Game State Visualization*

## *3.17.1 Purpose*

To verify that the game state is properly visualized with clear, accurate, and responsive UI elements.

## *3.17.2 Inputs*

**Visualization Requirements:**

- Game board layout
- Card displays
- State indicators
- Player information
- Game progress

## *3.17.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Game state is clearly visualized
- All UI elements are accurate
- Visual updates are responsive
- Information is easily accessible
- UI is consistent and intuitive

**Fail Criteria:**

- Game state not clearly visualized
- UI elements inaccurate
- Poor visual responsiveness
- Information not accessible
- UI inconsistencies

## *3.17.4 Test Procedure*

1. **Test Game Board Layout**
    - Verify game board displays correctly
    - Verify all areas are visible
    - Verify layout is responsive
    - Verify cyberpunk styling
    - Test with different screen sizes
2. **Test Card Displays**
    - Verify cards display correctly
    - Verify card information is readable

- Verify card states are clear
- Verify card interactions work
- Test with different card types

3. **Test State Indicators**
    - Verify current turn indicator
    - Verify AP display
    - Verify hand size display
    - Verify game phase indicator
    - Verify all indicators are accurate
4. **Test Player Information**
    - Verify player names display
    - Verify player roles display
    - Verify player stats display
    - Verify player status indicators
    - Test with different player types
5. **Test Visual Updates**
    - Perform game actions
    - Verify visual updates are immediate
    - Verify animations work smoothly
    - Verify no visual glitches
    - Test with rapid actions
6. **Test Information Accessibility**
    - Verify all information is visible
    - Verify tooltips work correctly
    - Verify help information is available
    - Verify information is organized logically
    - Test with different user preferences
7. **Test UI Consistency**
    - Verify consistent styling throughout
    - Verify consistent behavior
    - Verify consistent layout
    - Verify consistent interactions
    - Test with different UI elements
8. **Test Performance**
    - Test with complex game states
    - Verify UI remains responsive
    - Verify no performance issues

- Verify smooth animations
- Test with different devices

9. **Test Error Visualization**
    - Trigger various errors
    - Verify error messages are clear
    - Verify error indicators are visible
    - Verify error recovery is clear
    - Test with different error types
10. **Test Accessibility**
    - Test with different accessibility settings
    - Verify keyboard navigation works
    - Verify screen reader compatibility
    - Verify color contrast is adequate
    - Test with different user needs

# *3.18 (DD-05-001) Match Result Display*

## *3.18.1 Purpose*

To verify that match results are displayed correctly with proper winner determination, statistics, and post-game options.

## *3.18.2 Inputs*

**Valid Inputs:**

- Completed game matches
- Win conditions met
- Game statistics data

**Invalid Inputs (BVA):**

- Incomplete game data
- Corrupted statistics
- Missing result information

## *3.18.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Match results display correctly
- Winner is determined accurately
- Statistics are accurate and complete

- Post-game options work properly
- Visual presentation is clear and engaging

**Fail Criteria:**

- Match results not displayed
- Winner determination incorrect
- Statistics inaccurate or missing
- Post-game options not working
- Poor visual presentation

### *3.18.4 Test Procedure*

1. **Test Match Completion**
    - Play game to completion
    - Verify match result screen appears
    - Verify winner is displayed correctly
    - Verify win condition is shown
    - Verify game statistics are displayed
2. **Test Winner Display**
    - Verify winner announcement is clear
    - Verify winner team/player is highlighted
    - Verify win reason is displayed
    - Verify appropriate visual effects
    - Test with different win conditions
3. **Test Statistics Display**
    - Verify game duration is shown
    - Verify cards played count is accurate
    - Verify infrastructure changes are shown
    - Verify other relevant statistics
    - Test with different game lengths
4. **Test Post-Game Options**
    - Verify rematch button works
    - Verify return to lobby button works
    - Verify post-game chat is available
    - Verify all options are functional
    - Test with different user roles
5. **Test Result Screen Layout**
    - Verify cyberpunk styling is maintained
    - Verify layout is responsive

- Verify all elements are visible
- Verify information is organized clearly
- Test with different screen sizes

6. **Test ELO Rating Updates**
    - Verify ELO changes are displayed
    - Verify rating updates are accurate
    - Verify rating history is shown
    - Verify rating calculations are correct
    - Test with different rating scenarios
7. **Test Spectator View**
    - Have spectator view match result
    - Verify spectator sees appropriate information
    - Verify spectator cannot access player-only options
    - Verify spectator view is complete
    - Test with different spectator scenarios
8. **Test Error Handling**
    - Test with incomplete game data
    - Test with corrupted statistics
    - Verify appropriate error messages
    - Verify graceful error handling
    - Test with different error scenarios

## *3.19 (DD-05-002) Player Performance Statistics*

### *3.19.1 Purpose*

To verify that player performance statistics are tracked, displayed, and updated correctly.

### *3.19.2 Inputs*

**Valid Inputs:**

- Completed matches
- Player actions and decisions
- Game outcomes

**Invalid Inputs (BVA):**

- Incomplete match data
- Corrupted statistics
- Missing player information

### *3.19.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Statistics are tracked accurately
- Statistics display correctly
- Statistics update in real-time
- Historical data is preserved
- Statistics are accessible and useful

**Fail Criteria:**

- Statistics not tracked
- Statistics display incorrectly
- Statistics not updating
- Historical data lost
- Statistics not accessible

### *3.19.4 Test Procedure*

1. **Test ELO Rating Tracking**
    - Play multiple matches
    - Verify ELO rating updates after each match
    - Verify rating changes are calculated correctly
    - Verify rating history is maintained
    - Test with different win/loss scenarios
2. **Test Win/Loss Record**
    - Play multiple matches
    - Verify win/loss record updates
    - Verify record is accurate
    - Verify record is displayed correctly
    - Test with different match outcomes
3. **Test Match History**
    - Play multiple matches
    - Verify match history is recorded
    - Verify history includes all relevant information
    - Verify history is accessible
    - Test with different match types
4. **Test Performance Charts**
    - Play multiple matches over time
    - Verify charts display correctly
    - Verify chart data is accurate

- Verify charts update in real-time
- Test with different chart types

5. **Test Statistics Display**
    - Access profile page
    - Verify statistics are displayed
    - Verify statistics are organized clearly
    - Verify statistics are readable
    - Test with different user roles
6. **Test Real-time Updates**
    - Play match while viewing statistics
    - Verify statistics update in real-time
    - Verify updates are accurate
    - Verify no data loss
    - Test with different update scenarios
7. **Test Historical Data**
    - Play matches over extended period
    - Verify historical data is preserved
    - Verify data is accessible
    - Verify data is accurate
    - Test with different time periods
8. **Test Statistics Accuracy**
    - Compare displayed statistics with actual game data
    - Verify all calculations are correct
    - Verify no data corruption
    - Verify consistency across different views
    - Test with different data scenarios

## *3.20 (DD-05-003) Match History Storage and Browsing*

### *3.20.1 Purpose*

To verify that match history is properly stored and can be browsed with appropriate filtering and pagination.

### *3.20.2 Inputs*

**Valid Inputs:**

- Match history queries

- Filter parameters
- Pagination controls

**Invalid Inputs (BVA):**

- Invalid filter parameters
- Out-of-range pagination
- Corrupted history data

### *3.20.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Match history is stored correctly
- History can be browsed easily
- Filtering works properly
- Pagination works correctly
- History data is complete and accurate

**Fail Criteria:**

- Match history not stored
- History browsing not working
- Filtering not working
- Pagination issues
- History data incomplete or inaccurate

### *3.20.4 Test Procedure*

1. **Test Match History Storage**
    - Play multiple matches
    - Verify each match is stored
    - Verify all match data is preserved
    - Verify data is accessible
    - Test with different match types
2. **Test History Display**
    - Access match history page
    - Verify history is displayed correctly
    - Verify all relevant information is shown
    - Verify layout is clear and organized
    - Test with different history lengths
3. **Test History Filtering**
    - Apply different filters (date, opponent, result)
    - Verify filters work correctly
    - Verify filtered results are accurate

- Verify filter combinations work
- Test with different filter scenarios

4. **Test History Pagination**
    - Navigate through multiple pages
    - Verify pagination controls work
    - Verify page numbers are correct
    - Verify navigation is smooth
    - Test with different page sizes
5. **Test History Search**
    - Search for specific matches
    - Verify search results are accurate
    - Verify search works with different criteria
    - Verify search performance is good
    - Test with different search terms
6. **Test History Details**
    - Click on specific match
    - Verify detailed information is shown
    - Verify details are accurate
    - Verify details are complete
    - Test with different match types
7. **Test History Performance**
    - Test with large history datasets
    - Verify performance remains good
    - Verify loading times are acceptable
    - Verify no memory issues
    - Test with different data sizes
8. **Test History Data Integrity**
    - Verify history data is not corrupted
    - Verify data consistency
    - Verify no data loss
    - Verify data accuracy
    - Test with different data scenarios
9. **Test History Access Control**
    - Test with different user roles
    - Verify appropriate access levels
    - Verify privacy is maintained
    - Verify security is enforced

- Test with different access scenarios

10. **Test History Export**
    - Test history export functionality
    - Verify exported data is complete
    - Verify export format is correct
    - Verify export performance is good
    - Test with different export options

# *3.21 (DD-06-001) In-Game Currency Management*

## *3.21.1 Purpose*

To verify that in-game currency (Creds and Crypts) is properly managed, displayed, and updated throughout the application.

## *3.21.2 Inputs*

**Valid Inputs:**

- Currency transactions
- Currency transfers
- Currency displays

**Invalid Inputs (BVA):**

- Negative currency amounts
- Invalid transfer amounts
- Corrupted currency data

## *3.21.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Currency balances display correctly
- Currency transactions work properly
- Currency transfers function correctly
- Currency updates in real-time
- Currency validation works properly

**Fail Criteria:**

- Currency balances not displayed
- Currency transactions not working
- Currency transfers failing
- Currency not updating

- Currency validation not working

### *3.21.4 Test Procedure*

1. **Test Currency Display**
    - Login to application
    - Verify Creds and Crypts balances are displayed
    - Verify balances are shown in appropriate locations
    - Verify currency icons and styling
    - Test with different balance amounts
2. **Test Currency Updates**
    - Perform actions that change currency
    - Verify balances update in real-time
    - Verify updates are accurate
    - Verify updates appear in all relevant locations
    - Test with different update scenarios
3. **Test Currency Validation**
    - Attempt to spend more currency than available
    - Verify transaction is blocked
    - Verify appropriate error message
    - Verify balance is not changed
    - Test with different validation scenarios
4. **Test Currency Transfers**
    - Transfer currency to another user
    - Verify transfer is processed correctly
    - Verify both users' balances are updated
    - Verify transfer history is recorded
    - Test with different transfer amounts
5. **Test Currency Earning**
    - Play matches to earn Creds
    - Verify Creds are awarded correctly
    - Verify earning amounts are accurate
    - Verify earnings are displayed
    - Test with different earning scenarios
6. **Test Currency Spending**
    - Purchase items from store
    - Verify currency is deducted correctly
    - Verify transaction is processed
    - Verify balance is updated

- Test with different purchase scenarios

7. **Test Currency Persistence**
    - Change currency balances
    - Refresh page or restart application
    - Verify balances are preserved
    - Verify no currency loss
    - Test with different persistence scenarios
8. **Test Currency Error Handling**
    - Test with invalid currency data
    - Test with network issues
    - Verify appropriate error messages
    - Verify graceful error handling
    - Test with different error scenarios

## *3.22 (DD-06-002) Store Browsing, Item Purchase, and Application of Decoration*

### *3.22.1 Purpose*

To verify that the store system works correctly for browsing items, making purchases, and applying decorations.

### *3.22.2 Inputs*

**Valid Inputs:**

- Store item selection
- Purchase actions
- Decoration application

**Invalid Inputs (BVA):**

- Insufficient currency
- Invalid item selection
- Corrupted store data

### *3.22.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Store items display correctly
- Purchase process works smoothly
- Decorations apply correctly
- Currency validation works
- User experience is intuitive

**Fail Criteria:**

- Store items not displaying
- Purchase process not working
- Decorations not applying
- Currency validation failing
- Poor user experience

### *3.22.4 Test Procedure*

1. **Test Store Display**
    - Access store page
    - Verify store items are displayed
    - Verify item information is complete
    - Verify pricing is shown correctly
    - Verify cyberpunk styling is maintained
2. **Test Item Browsing**
    - Browse different item categories
    - Verify items are organized logically
    - Verify item details are accessible
    - Verify search functionality works
    - Test with different browsing scenarios
3. **Test Item Information**
    - Click on specific items
    - Verify detailed information is shown
    - Verify item previews work
    - Verify pricing is accurate
    - Test with different item types
4. **Test Purchase Process**
    - Select item for purchase
    - Verify purchase confirmation dialog
    - Verify currency validation
    - Complete purchase
    - Verify purchase is processed correctly
5. **Test Purchase Validation**
    - Attempt to purchase item with insufficient currency
    - Verify purchase is blocked
    - Verify appropriate error message
    - Verify currency is not deducted

- Test with different validation scenarios

6. **Test Decoration Application**
    - Purchase decoration item
    - Apply decoration to profile
    - Verify decoration is applied correctly
    - Verify decoration is visible in profile
    - Test with different decoration types
7. **Test Purchase History**
    - Make multiple purchases
    - Verify purchase history is recorded
    - Verify history is accessible
    - Verify history is accurate
    - Test with different purchase scenarios
8. **Test Store Performance**
    - Test with large number of items
    - Verify store loads quickly
    - Verify browsing is smooth
    - Verify no performance issues
    - Test with different store configurations
9. **Test Store Error Handling**
    - Test with network issues
    - Test with invalid store data
    - Verify appropriate error messages
    - Verify graceful error handling
    - Test with different error scenarios

## *3.23 (DD-06-003) Payment Integration*

### *3.23.1 Purpose*

To verify that payment integration works correctly for purchasing Crypts with real money.

### *3.23.2 Inputs*

**Valid Inputs:**

- Payment package selection
- Payment method selection
- Payment completion

**Invalid Inputs (BVA):**

- Invalid payment methods
- Corrupted payment data
- Failed payment transactions

### *3.23.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Payment process works smoothly
- Payment packages display correctly
- Payment completion is handled properly
- Crypts are credited correctly
- Payment security is maintained

**Fail Criteria:**

- Payment process not working
- Payment packages not displaying
- Payment completion failing
- Crypts not credited
- Payment security issues

### *3.23.4 Test Procedure*

1. **Test Payment Package Display**
    - Access top-up page
    - Verify payment packages are displayed
    - Verify package information is complete
    - Verify pricing is accurate
    - Test with different package types
2. **Test Payment Package Selection**
    - Select different payment packages
    - Verify selection is recorded
    - Verify package details are shown
    - Verify pricing is calculated correctly
    - Test with different selection scenarios
3. **Test Payment Method Selection**
    - Select payment method
    - Verify payment method is recorded
    - Verify payment options are available
    - Verify payment security is maintained
    - Test with different payment methods
4. **Test Payment Process**

- Initiate payment process
- Verify payment window opens
- Verify payment form is displayed
- Verify payment security is maintained
- Test with different payment scenarios

5. **Test Payment Completion**
    - Complete payment successfully
    - Verify payment is processed
    - Verify Crypts are credited
    - Verify balance is updated
    - Test with different completion scenarios
6. **Test Payment Failure**
    - Simulate payment failure
    - Verify appropriate error message
    - Verify payment is not processed
    - Verify Crypts are not credited
    - Test with different failure scenarios
7. **Test Payment Security**
    - Verify payment data is encrypted
    - Verify payment information is secure
    - Verify no sensitive data is exposed
    - Verify payment compliance
    - Test with different security scenarios
8. **Test Payment History**
    - Complete multiple payments
    - Verify payment history is recorded
    - Verify history is accessible
    - Verify history is accurate
    - Test with different payment scenarios
9. **Test Payment Error Handling**
    - Test with network issues
    - Test with invalid payment data
    - Verify appropriate error messages
    - Verify graceful error handling
    - Test with different error scenarios
10. **Test Payment Performance**
    - Test payment process performance

- Verify payment loads quickly
- Verify payment processes efficiently
- Verify no performance issues
- Test with different payment scenarios

# 3.24 (DD-07-001) User Search, Ban, and Moderation

## 3.24.1 Purpose

To verify that admin and moderator tools work correctly for user management, including search, ban, and moderation functions.

## 3.24.2 Inputs

**Valid Inputs:**

- User search queries
- Ban/unban actions
- Moderation actions

**Invalid Inputs (BVA):**

- Invalid search queries
- Invalid user IDs
- Unauthorized actions

## 3.24.3 Expected Outputs & Pass/Fail Criteria

**Pass Criteria:**

- User search works correctly
- Ban/unban functions work properly
- Moderation actions are effective
- Access control is enforced
- Audit logging works correctly

**Fail Criteria:**

- User search not working
- Ban/unban functions failing
- Moderation actions not working
- Access control bypassed
- Audit logging not working

## 3.24.4 Test Procedure

1. **Test User Search**

- Login as admin/moderator
- Access user management page
- Search for users by username
- Search for users by email
- Verify search results are accurate

2. **Test User Search Filters**
   - Apply different search filters
   - Filter by user role (user/moderator/admin)
   - Filter by user status (active/banned)
   - Filter by registration date
   - Verify filters work correctly
3. **Test User List Display**
   - Verify user list displays correctly
   - Verify user information is complete
   - Verify user status indicators
   - Verify user role indicators
   - Test with different user types
4. **Test User Ban Function**
   - Select user to ban
   - Enter ban reason
   - Confirm ban action
   - Verify user is banned
   - Verify ban reason is recorded
5. **Test User Unban Function**
   - Select banned user
   - Confirm unban action
   - Verify user is unbanned
   - Verify user can login again
   - Verify unban is recorded
6. **Test Ban Validation**
   - Attempt to ban admin user
   - Verify ban is blocked
   - Verify appropriate error message
   - Verify admin protection works
   - Test with different protection scenarios
7. **Test Moderation Actions**
   - Issue warning to user

- Verify warning is recorded
- Verify user is notified
- Verify warning appears in user history
- Test with different warning types

8. **Test Access Control**
    - Test with regular user account
    - Verify admin functions are hidden
    - Verify access is denied
    - Verify appropriate error messages
    - Test with different user roles
9. **Test Audit Logging**
    - Perform admin actions
    - Verify actions are logged
    - Verify log entries are complete
    - Verify log entries are accurate
    - Test with different action types
10. **Test User Management Performance**
    - Test with large number of users
    - Verify search performance is good
    - Verify list loading is fast
    - Verify no performance issues
    - Test with different data sizes

# *3.25 (DD-07-002) Report Management*

## *3.25.1 Purpose*

To verify that the report management system works correctly for submitting, reviewing, and resolving reports.

## *3.25.2 Inputs*

**Valid Inputs:**

- Report submissions
- Report reviews
- Report resolutions

**Invalid Inputs (BVA):**

- Invalid report data
- Missing report information

- Unauthorized report access

### *3.25.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Reports can be submitted correctly
- Reports can be reviewed properly
- Reports can be resolved effectively
- Report data is accurate and complete
- Report workflow functions correctly

**Fail Criteria:**

- Report submission not working
- Report review not working
- Report resolution not working
- Report data inaccurate
- Report workflow broken

### *3.25.4 Test Procedure*

1. **Test Report Submission**
    - Access report modal
    - Fill out report form
    - Select report type and reason
    - Submit report
    - Verify report is submitted successfully
2. **Test Report Form Validation**
    - Submit report with missing information
    - Verify validation errors appear
    - Verify form submission is blocked
    - Test with different validation scenarios
    - Verify appropriate error messages
3. **Test Report Display**
    - Access report management page
    - Verify reports are displayed correctly
    - Verify report information is complete
    - Verify report status indicators
    - Test with different report types
4. **Test Report Filtering**
    - Filter reports by status

- Filter reports by type
- Filter reports by date
- Verify filters work correctly
- Test with different filter combinations

5. **Test Report Review**
    - Select report for review
    - Verify report details are shown
    - Verify review options are available
    - Update report status
    - Verify status update is recorded
6. **Test Report Resolution**
    - Resolve report
    - Verify resolution is recorded
    - Verify report status is updated
    - Verify user is notified
    - Test with different resolution types
7. **Test Report Statistics**
    - Access report statistics
    - Verify statistics are accurate
    - Verify statistics are up-to-date
    - Verify statistics are useful
    - Test with different time periods
8. **Test Report Access Control**
    - Test with different user roles
    - Verify appropriate access levels
    - Verify privacy is maintained
    - Verify security is enforced
    - Test with different access scenarios
9. **Test Report Performance**
    - Test with large number of reports
    - Verify report loading is fast
    - Verify search performance is good
    - Verify no performance issues
    - Test with different data sizes
10. **Test Report Error Handling**
    - Test with network issues
    - Test with invalid report data

- Verify appropriate error messages
- Verify graceful error handling
- Test with different error scenarios

# *3.26 (DD-07-003) System Logs and Audit Trails*

## *3.26.1 Purpose*

To verify that system logs and audit trails are properly recorded, displayed, and accessible for administrative purposes.

## *3.26.2 Inputs*

**Valid Inputs:**

- Log queries
- Filter parameters
- Log access requests

**Invalid Inputs (BVA):**

- Invalid log queries
- Unauthorized access attempts
- Corrupted log data

## *3.26.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- System logs are recorded correctly
- Audit trails are complete
- Logs are accessible to authorized users
- Log filtering works properly
- Log data is accurate and useful

**Fail Criteria:**

- System logs not recorded
- Audit trails incomplete
- Logs not accessible
- Log filtering not working
- Log data inaccurate

## *3.26.4 Test Procedure*

1. **Test Log Recording**
    - Perform various system actions
    - Verify logs are recorded

- Verify log entries are complete
- Verify log timestamps are accurate
- Test with different action types

2. **Test Log Display**
    - Access system logs page
    - Verify logs are displayed correctly
    - Verify log information is complete
    - Verify log formatting is clear
    - Test with different log types
3. **Test Log Filtering**
    - Filter logs by date range
    - Filter logs by user
    - Filter logs by action type
    - Verify filters work correctly
    - Test with different filter combinations
4. **Test Log Search**
    - Search logs by text
    - Search logs by specific criteria
    - Verify search results are accurate
    - Verify search performance is good
    - Test with different search terms
5. **Test Log Pagination**
    - Navigate through log pages
    - Verify pagination controls work
    - Verify page numbers are correct
    - Verify navigation is smooth
    - Test with different page sizes
6. **Test Log Details**
    - Click on specific log entry
    - Verify detailed information is shown
    - Verify details are accurate
    - Verify details are complete
    - Test with different log types
7. **Test Log Access Control**
    - Test with different user roles
    - Verify appropriate access levels
    - Verify sensitive information is protected

- Verify security is enforced
- Test with different access scenarios

8. **Test Log Performance**
    - Test with large number of logs
    - Verify log loading is fast
    - Verify search performance is good
    - Verify no performance issues
    - Test with different data sizes
9. **Test Log Data Integrity**
    - Verify log data is not corrupted
    - Verify log consistency
    - Verify no log data loss
    - Verify log accuracy
    - Test with different data scenarios
10. **Test Log Export**
    - Test log export functionality
    - Verify exported data is complete
    - Verify export format is correct
    - Verify export performance is good
    - Test with different export options

## *3.27 (DD-07-004) User Modification*

### *3.27.1 Purpose*

To verify that admin and moderator tools allow proper modification of user accounts with appropriate validation and security.

### *3.27.2 Inputs*

**Valid Inputs:**

- User modification requests
- Valid user data
- Authorized modification actions

**Invalid Inputs (BVA):**

- Invalid user data
- Unauthorized modifications
- Corrupted user information

### *3.27.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- User modifications work correctly
- Data validation is enforced
- Security is maintained
- Changes are logged properly
- User experience is smooth

**Fail Criteria:**

- User modifications not working
- Data validation not enforced
- Security compromised
- Changes not logged
- Poor user experience

### *3.27.4 Test Procedure*

1. **Test User Selection**
    - Access user management page
    - Select user for modification
    - Verify user details are loaded
    - Verify modification form is displayed
    - Test with different user types
2. **Test User Data Display**
    - Verify user information is displayed correctly
    - Verify all relevant fields are shown
    - Verify data is current and accurate
    - Verify form is properly populated
    - Test with different user data
3. **Test Username Modification**
    - Change user username
    - Verify validation works correctly
    - Verify username is updated
    - Verify change is logged
    - Test with different username scenarios
4. **Test Email Modification**
    - Change user email
    - Verify email validation works
    - Verify email is updated

- Verify change is logged
- Test with different email scenarios

5. **Test Password Modification**
    - Change user password
    - Verify password validation works
    - Verify password is updated
    - Verify change is logged
    - Test with different password scenarios
6. **Test Role Modification**
    - Change user role
    - Verify role validation works
    - Verify role is updated
    - Verify change is logged
    - Test with different role scenarios
7. **Test Data Validation**
    - Attempt to enter invalid data
    - Verify validation errors appear
    - Verify form submission is blocked
    - Test with different validation scenarios
    - Verify appropriate error messages
8. **Test Security Enforcement**
    - Test with different user roles
    - Verify appropriate access levels
    - Verify sensitive data is protected
    - Verify security is enforced
    - Test with different security scenarios
9. **Test Modification Logging**
    - Perform user modifications
    - Verify changes are logged
    - Verify log entries are complete
    - Verify log entries are accurate
    - Test with different modification types
10. **Test Modification Performance**
    - Test with large number of users
    - Verify modification process is fast
    - Verify no performance issues
    - Verify system remains responsive

- Test with different data sizes

# *3.28 (DD-08-001) Lore Video Playback*

## *3.28.1 Purpose*

To verify that first-time users are properly guided through the lore video with appropriate controls and completion handling.

## *3.28.2 Inputs*

**Valid Inputs:**

- First-time user login
- Video playback controls
- Video completion/skip actions

**Invalid Inputs (BVA):**

- Corrupted video data
- Network issues during playback
- Invalid user state

## *3.28.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Lore video displays for first-time users
- Video controls work properly
- Video completion is tracked correctly
- Skip functionality works
- User progression is handled properly

**Fail Criteria:**

- Lore video not displaying
- Video controls not working
- Completion not tracked
- Skip functionality broken
- User progression issues

## *3.28.4 Test Procedure*

1. **Test First-Time User Detection**
    - Create new user account
    - Login for first time
    - Verify lore video is displayed
    - Verify video is full-screen

- Verify cyberpunk styling is maintained

2. **Test Video Playback Controls**
    - Verify play/pause button works
    - Verify video plays correctly
    - Verify video quality is good
    - Verify audio works (if applicable)
    - Test with different video controls
3. **Test Video Skip Functionality**
    - Click skip button
    - Verify confirmation dialog appears
    - Confirm skip action
    - Verify video is skipped
    - Verify user proceeds to tutorial
4. **Test Video Completion**
    - Let video play to completion
    - Verify completion is detected
    - Verify user proceeds to tutorial
    - Verify completion is tracked
    - Test with different video lengths
5. **Test Video Error Handling**
    - Test with network issues during playback
    - Verify appropriate error messages
    - Verify fallback options are available
    - Verify user can still proceed
    - Test with different error scenarios
6. **Test Video Accessibility**
    - Test with different accessibility settings
    - Verify keyboard controls work
    - Verify captions are available (if applicable)
    - Verify screen reader compatibility
    - Test with different user needs
7. **Test Video Performance**
    - Test video loading performance
    - Verify video plays smoothly
    - Verify no performance issues
    - Verify video quality is acceptable
    - Test with different devices

8. **Test User State Persistence**
    - Complete video and proceed to tutorial
    - Logout and login again
    - Verify video is not shown again
    - Verify user proceeds directly to main app
    - Test with different user states
9. **Test Video Content**
    - Verify video content is appropriate
    - Verify video length is reasonable
    - Verify video quality is good
    - Verify video is engaging
    - Test with different content scenarios

## *3.29 (DD-08-002) Interactive Tutorial*

### *3.29.1 Purpose*

To verify that the interactive tutorial guides new users through core game mechanics with proper step-by-step instructions and completion handling.

### *3.29.2 Inputs*

**Valid Inputs:**

- Tutorial step completion
- User interactions
- Tutorial navigation

**Invalid Inputs (BVA):**

- Invalid tutorial data
- Corrupted tutorial state
- Missing tutorial steps

### *3.29.3 Expected Outputs & Pass/Fail Criteria*

**Pass Criteria:**

- Tutorial displays correctly
- Step-by-step instructions are clear
- User interactions are captured properly
- Tutorial progression works smoothly
- Completion is handled correctly

**Fail Criteria:**

- Tutorial not displaying

- Instructions not clear
- Interactions not captured
- Progression not working
- Completion not handled

### *3.29.4 Test Procedure*

1. **Test Tutorial Initialization**
    - Complete lore video
    - Verify tutorial starts automatically
    - Verify tutorial overlay is displayed
    - Verify first step is shown
    - Verify tutorial styling is consistent
2. **Test Tutorial Steps**
    - Follow tutorial step-by-step
    - Verify each step is clear and understandable
    - Verify step instructions are helpful
    - Verify step progression works
    - Test with different tutorial steps
3. **Test User Interactions**
    - Perform required interactions for each step
    - Verify interactions are captured correctly
    - Verify step completion is detected
    - Verify next step is shown
    - Test with different interaction types
4. **Test Tutorial Navigation**
    - Use next/previous buttons
    - Verify navigation works correctly
    - Verify step numbers are accurate
    - Verify progress indicator works
    - Test with different navigation scenarios
5. **Test Tutorial Validation**
    - Attempt to skip required interactions
    - Verify validation prevents progression
    - Verify appropriate error messages
    - Verify user must complete step
    - Test with different validation scenarios
6. **Test Tutorial Completion**

- Complete all tutorial steps
- Verify completion is detected
- Verify tutorial is marked as complete
- Verify user proceeds to main app
- Test with different completion scenarios

7. **Test Tutorial Persistence**
   - Complete tutorial partially
   - Logout and login again
   - Verify tutorial resumes from last step
   - Verify progress is preserved
   - Test with different persistence scenarios
8. **Test Tutorial Error Handling**
   - Test with network issues during tutorial
   - Verify appropriate error messages
   - Verify tutorial can be resumed
   - Verify no data loss
   - Test with different error scenarios
9. **Test Tutorial Accessibility**
   - Test with different accessibility settings
   - Verify keyboard navigation works
   - Verify screen reader compatibility
   - Verify clear visual indicators
   - Test with different user needs
10. **Test Tutorial Performance**
    - Test tutorial loading performance
    - Verify tutorial runs smoothly
    - Verify no performance issues
    - Verify responsive design works
    - Test with different devices
11. **Test Tutorial Content**
    - Verify tutorial content is accurate
    - Verify instructions match actual gameplay
    - Verify tutorial covers all essential mechanics
    - Verify tutorial is engaging
    - Test with different content scenarios
12. **Test Tutorial Customization**
    - Test with different user preferences

- Verify tutorial adapts to user needs
- Verify tutorial can be customized
- Verify personalization works
- Test with different customization scenarios

# Appendix (Test Logs)

## *A.1 Test Logs*

Test execution logs will be maintained in Trello cards for each test case, including:

- Test execution date and time
- Tester name
- Test environment details
- Actual results vs expected results
- Screenshots and evidence
- Issues and observations

## *A.2 Test Results*

Overall test results will be summarized in a master Trello board showing:

- Total test cases executed
- Pass/fail statistics
- Critical issues found
- Test coverage analysis
- Recommendations for fixes

## *A.3 Incident Report*

Any defects or issues found during testing will be documented in Trello with:

- Issue description and severity
- Steps to reproduce
- Expected vs actual behavior
- Screenshots and evidence
- Priority for fixing
- Status tracking