Phase 4A Session 2 - Test Cases

Testing Goal: Verify enhanced sorting system integration is working correctly

Date: June 5, 2025

Status: Ready for Testing

🚀 Pre-Test Setup

1. Start Application

```bash
cd C:\Users\abaec\Development\mtg-deck-builder
npm start
```

2. Open Browser Console

- **Chrome/Edge:** F12 → Console tab
- **Firefox:** F12 → Console tab
- **Keep console open** during all tests to monitor logs

3. Expected Initial State

- ✅ Application loads without TypeScript errors
- ✅ Collection shows popular cards by default
- ✅ Console shows sorting system initialization logs
- ✅ No error messages in console

📋 Test Suite 1: Collection Server-Side Sorting

Test 1.1: Large Search + Server-Side Sort

Objective: Verify large searches trigger server-side re-search when sorting

Steps:

1. Search for `creature` in collection search box
2. Wait for results to load (should be 100+ cards)
3. Click "Sort" button in collection area
4. Click "Color" to sort by color

Expected Results:

```
Console Logs:
🔄 Collection sort changed: {criteria: "color", direction: "asc"}
🔄 Sort change analysis: {totalCards: X, loadedCards: Y, shouldUseServerSort: true}
🌐 Using server-side sorting - re-searching with new sort parameters
🔍 ENHANCED SEARCH SUCCESS: Object
```

Visual Results:

- ✅ Loading indicator appears briefly
- ✅ Cards reorder by color (White → Blue → Black → Red → Green → Colorless)
- ✅ No errors in console
- ✅ Sort button shows "Color ↑"

Pass/Fail: ☐

Test 1.2: Small Search + Client-Side Sort

Objective: Verify small searches use instant client-side sorting

Steps:

1. Search for `lightning bolt` in collection
2. Wait for results (should be <10 cards)
3. Click "Sort" → "Mana Value"

Expected Results:

```
Console Logs:
🔄 Collection sort changed: {criteria: "mana", direction: "asc"}
🔄 Sort change analysis: {totalCards: X, loadedCards: Y, shouldUseServerSort: false}
🏠 Using client-side sorting - all results already loaded
```

Visual Results:

- ✅ NO loading indicator
- ✅ Instant reorder by mana cost (0, 1, 2, 3...)
- ✅ Sort button shows "Mana Value ↑"

Pass/Fail: ☐

Test 1.3: Sort Direction Toggle

Objective: Verify sort direction toggles work properly

Steps:

1. With previous "Mana Value ↑" sort active
2. Click "Mana Value" again to toggle direction

Expected Results:

```
Console Logs:
🔄 Collection sort changed: {criteria: "mana", direction: "desc"}
```

Visual Results:

- ✅ Cards reorder by mana cost descending (high to low)
- ✅ Sort button shows "Mana Value ↓"

Pass/Fail: ☐

Test 1.4: All Collection Sort Options

Objective: Test all collection sort criteria work

Steps: Test each sort option:

1. **Name:** Click "Sort" → "Name" → Verify alphabetical order
2. **Mana Value:** Already tested above
3. **Color:** Already tested above
4. **Rarity:** Click "Sort" → "Rarity" → Verify order (Common → Uncommon → Rare → Mythic)

Expected Results:

- ✅ Each sort works correctly
- ✅ Console logs show sort changes
- ✅ Cards display in expected order
- ✅ Sort indicators update correctly

Pass/Fail: ☐

📋 Test Suite 2: Sort Persistence

Test 2.1: Page Refresh Persistence

Objective: Verify sort preferences persist across page reloads

Steps:

1. Set collection sort to "Rarity ↓"
2. Press F5 to refresh page
3. Wait for application to reload

Expected Results:

- ✅ Collection still sorted by "Rarity ↓"
- ✅ Sort button shows "Rarity ↓"
- ✅ Card order matches pre-refresh order

Pass/Fail: ☐

Test 2.2: Cross-Session Persistence

Objective: Verify sort preferences persist across browser sessions

Steps:

1. Set collection sort to "Color ↑"
2. Close browser completely
3. Reopen browser and navigate to application

Expected Results:

- ✅ Collection still sorted by "Color ↑"
- ✅ Sort preferences restored correctly

Pass/Fail: ☐

📋 Test Suite 3: Deck & Sideboard Enhanced Sorting

Test 3.1: Deck Sorting Integration

Objective: Verify deck area uses enhanced sorting correctly

Steps:

1. Add 5-10 cards to deck (mix of different mana costs and colors)
2. Switch deck to "List" view
3. Test deck sort options: "Mana Value", "Color", "Rarity", "Name", "Card Type"

Expected Results:

- ✅ All sort options work in deck
- ✅ Cards reorder correctly for each criteria
- ✅ Sort state persists when switching view modes
- ✅ No API calls triggered (deck sorting is local)

Pass/Fail: ☐

Test 3.2: Sideboard Sorting Integration

Objective: Verify sideboard area uses enhanced sorting correctly

Steps:

1. Add 5-10 cards to sideboard
2. Switch sideboard to "List" view
3. Test sideboard sort options

Expected Results:

- ✅ All sort options work in sideboard
- ✅ Cards reorder correctly
- ✅ Independent from deck sort settings

Pass/Fail: ☐

Test 3.3: Card View Sorting

Objective: Verify enhanced sorting works in card view mode

Steps:

1. Ensure deck/sideboard in "Card" view
2. Test sorting options from sort dropdown

3. Verify PileView integration

Expected Results:

- ✅ Card view reflects sort order
- ✅ PileView groups reflect sorting
- ✅ Sort persistence across view mode changes

Pass/Fail: ☐

📋 Test Suite 4: Error Handling & Edge Cases

Test 4.1: Empty Search Results

Objective: Verify sorting works with no results

Steps:

1. Search for `nonexistentcardname12345`
2. Wait for "No results found" message
3. Try changing sort options

Expected Results:

- ✅ No errors when sorting empty results
- ✅ Sort options remain functional
- ✅ System handles gracefully

Pass/Fail: ☐

Test 4.2: Network Error Handling

Objective: Verify behavior when server-side sorting fails

Steps:

1. (If possible) Disconnect internet briefly
2. Perform large search + sort change
3. Reconnect internet

Expected Results:

- ✅ Graceful fallback to client-side sorting
- ✅ Error messages (if any) are user-friendly
- ✅ System recovers when connection restored

Pass/Fail: ☐

Test 4.3: Rapid Sort Changes

Objective: Verify system handles quick consecutive sort changes

Steps:

1. Perform large search (e.g., `creature`)
2. Rapidly click different sort options: Name → Color → Mana → Rarity
3. Check for race conditions or errors

Expected Results:

- ✅ Only final sort state is applied
- ✅ No duplicate API calls
- ✅ No console errors

Pass/Fail: ☐

📋 Test Suite 5: Integration Compatibility

Test 5.1: View Mode Switching

Objective: Verify sorting persists across view mode changes

Steps:

1. Set collection sort to "Mana Value ↓"
2. Switch between Card view and List view
3. Verify sort state maintained

Expected Results:

- ✅ Sort state persists across view changes
- ✅ Cards maintain sort order in both views

Pass/Fail: ☐

Test 5.2: Filter + Sort Interaction

Objective: Verify sorting works with active filters

Steps:

1. Apply color filter (e.g., Blue cards only)
2. Test sorting options on filtered results
3. Clear filters and verify sorting still works

Expected Results:

- ✅ Sorting works correctly on filtered results
- ✅ Server-side vs client-side logic still applies
- ✅ Sort state maintained when changing filters

Pass/Fail: ☐

Test 5.3: Drag & Drop Compatibility

Objective: Verify sorting doesn't interfere with drag & drop

Steps:

1. Set collection sort to any option
2. Drag cards from collection to deck
3. Verify deck/sideboard functionality unchanged

Expected Results:

- ✅ Drag & drop works normally
- ✅ Cards move correctly between areas
- ✅ Sort states independent per area

Pass/Fail: ☐

📋 Test Suite 6: Performance Verification

Test 6.1: Large Dataset Performance

Objective: Verify system handles large result sets efficiently

Steps:

1. Search for `*` (all cards) or broad term like `the`
2. Wait for results to load
3. Test sort changes and measure response time

Expected Results:

- ✅ Server-side sorting used for large datasets
- ✅ Reasonable response times (<2 seconds)
- ✅ UI remains responsive during sorting

Pass/Fail: ☐

Test 6.2: Memory Usage

Objective: Verify no memory leaks from sorting system

Steps:

1. Open browser developer tools → Performance/Memory tab
2. Perform 20+ sort operations
3. Check for memory growth patterns

Expected Results:

- ✅ No significant memory leaks
- ✅ Stable memory usage patterns

Pass/Fail: ☐

🎯 Test Results Summary

Overall Integration Status

- **Collection Server-Side Sorting:** ☐ Pass / ☐ Fail
- **Collection Client-Side Sorting:** ☐ Pass / ☐ Fail
- **Sort Persistence:** ☐ Pass / ☐ Fail
- **Deck/Sideboard Enhanced Sorting:** ☐ Pass / ☐ Fail
- **Error Handling:** ☐ Pass / ☐ Fail

- **Integration Compatibility:** ☐ Pass / ☐ Fail
- **Performance:** ☐ Pass / ☐ Fail

Critical Issues Found

1. ☐ Issue 1: *[Description]*
2. ☐ Issue 2: *[Description]*
3. ☐ Issue 3: *[Description]*

Non-Critical Issues Found

1. ☐ Issue 1: *[Description]*
2. ☐ Issue 2: *[Description]*

Session 2 Integration Status

☐ **COMPLETE** - All tests passing, enhanced sorting fully integrated
☐ **NEEDS FIXES** - Some issues found, requires additional work
☐ **MAJOR ISSUES** - Significant problems, integration incomplete

---

🔧 Debug Information Collection

Console Log Examples

```
[Copy relevant console logs here during testing]
```

Network Tab Observations

```
[Copy API request details here if relevant]
```

Specific Error Messages

```
[Copy any error messages encountered]
```

✅ Post-Test Actions

If All Tests Pass:

1. **Update project_status.md** - Mark Phase 4A Session 2 as complete
2. **Create completion document** - Archive Session 2 technical details

3. **Plan next development -** Phase 4B or other enhancements

If Issues Found:

1. **Document specific failures** in detail
2. **Prioritize fixes** by impact and complexity
3. **Create targeted fix plan** for remaining issues
4. **Retest after fixes** using subset of test cases

Testing Prepared By: Claude (Phase 4A Session 2 Integration)
Test Environment: Windows laptop, React TypeScript, Enhanced Sorting System
Expected Duration: 30-45 minutes for complete test suite