Text Visibility Audit Report

Date: November 23, 2025
Branch: feature/app-grid-launcher-enhancements

Executive Summary

This audit identifies text visibility issues throughout the loomOS application where hardcoded colors prevent proper adaptation between light and dark modes. While the application has a comprehensive design system (`webos-design-system.css`) with both light and dark mode support, many components still use hardcoded color values.

Current State

Design System

- ✅ Comprehensive CSS variables defined in `styles/webos-design-system.css`
- ✅ Dark mode support via `.dark` class selector (Tailwind-compatible)
- ✅ Media query support for `prefers-color-scheme: dark`
- ✅ Well-defined semantic color tokens

Issues Found

High Priority - Status Bar Colors (dashboard/layout.tsx)

The SystemStatusBar component uses hardcoded colors that don't adapt to theme:

```
// Lines 67-69
backgroundColor: '#1a1a1a',
borderBottom: '1px solid rgba(255, 255, 255, 0.1)',
boxShadow: '0 2px 8px rgba(0, 0, 0, 0.15)'
```

Recommended Fix:

```
backgroundColor: 'var(--chrome-darker)',
borderBottom: '1px solid var(--glass-border-overlay)',
boxShadow: 'var(--shadow-md)'
```

Medium Priority - Icon Colors

Many icons use hardcoded white/gray colors:
- Lines 83, 95, 113, 120, 127, 134, 141, 148: Icon colors
- Should use: `var(--chrome-text)` or `var(--text-inverse)`

Medium Priority - Dropdown Menus

Profile dropdown and mobile menu use hardcoded backgrounds:
- Lines 197-199, 232-234: Dropdown backgrounds
- Should use: `var(--bg-elevated)` or `var(--chrome-dark)`

Low Priority - App Colors

App icon colors (lines 303-312) are hardcoded but are intentionally brand colors. These could be:
1. Kept as-is if they’re brand colors
2. Moved to design system as semantic tokens
3. Made theme-aware with adjusted opacity in dark mode

Recommendations

Short-term (Quick Wins)

1. **Fix SystemStatusBar** - Highest impact
   - Replace hardcoded colors with CSS variables
   - Affects primary navigation element
   - Estimated effort: 30 minutes

2. **Fix Modal/Dialog Overlays**
   - App launcher overlay (line 377)
   - Search overlays
   - Estimated effort: 20 minutes

Medium-term (Component Updates)

1. **Audit UI Components** (`components/ui/`)
   - Button, Card, Dialog, Input, Textarea components found with hardcoded colors
   - Systematic replacement with design tokens
   - Estimated effort: 2-4 hours

2. **Update WebOS Components** (`components/webos/`)
   - Desktop components, status bars, widgets
   - Ensure all use CSS variables
   - Estimated effort: 3-5 hours

Long-term (Architecture)

1. **Automated Testing**
   - Create automated tests to verify no hardcoded colors in new code
   - Add linter rules to catch hardcoded hex/rgba values
   - CI/CD integration

2. **Design Token Documentation**
   - Create comprehensive guide for developers
   - Examples of when to use which tokens
   - Migration guide for existing components

Automated Fix Script

A Python script (`fix-text-visibility.py`) has been created to help identify and fix hardcoded colors.
Usage:

```
# Dry run to see what would be changed
python3 fix-text-visibility.py --dry-run

# Apply fixes
python3 fix-text-visibility.py
```

Note: The script provides a starting point but manual review is recommended for:
- Brand colors that should remain consistent
- Intentional color choices for specific designs
- Complex rgba() expressions with variables

Testing Strategy

Manual Testing Checklist

- [ ] Toggle dark/light mode in system preferences
- [ ] Verify status bar is visible in both modes
- [ ] Check app launcher overlay readability
- [ ] Test all interactive elements (buttons, inputs, etc.)
- [ ] Verify text contrast meets WCAG AA standards

Browser Testing

- [ ] Chrome (latest)
- [ ] Firefox (latest)
- [ ] Safari (latest)
- [ ] Mobile browsers (iOS Safari, Chrome Mobile)

Accessibility Testing

- [ ] Run axe DevTools or similar
- [ ] Verify color contrast ratios
- [ ] Test with high contrast mode
- [ ] Test with reduced motion preferences

Implementation Plan

Phase 1: Critical Fixes (This PR)

- ✅ Enhanced app launcher with search, sorting, pagination
- 🔄 Document text visibility issues
- ⏳ Fix critical status bar colors

Phase 2: Component Library (Future PR)

- Update all UI components in `components/ui/`
- Create migration guide
- Add linter rules

Phase 3: Full Application (Future Sprint)

- Systematic component updates
- Automated testing

- Documentation

Related Files

Files with Most Issues

1. `app/dashboard/layout.tsx` - 60+ hardcoded colors
2. `app/dashboard/page.tsx` - 40+ hardcoded colors
3. `components/ui/*` - Various components with hardcoded colors
4. `components/webos/*` - WebOS-specific components

Reference Files

- `styles/webos-design-system.css` - Design token definitions
- `LIGHT_DARK_MODE_AUDIT_REPORT.md` - Previous audit findings
- `STYLING_AUDIT_AND_FIXES.md` - Previous styling fixes

Conclusion

While the application has excellent infrastructure for theme support, consistent application of design tokens throughout the codebase is needed. This audit provides a roadmap for systematic improvements while prioritizing the most visible and impactful changes.

The enhanced app launcher implemented in this PR follows best practices with:
- Semantic HTML and ARIA labels
- Keyboard navigation support
- Responsive design
- Clear visual hierarchy

Future improvements should maintain this standard while progressively adopting CSS variables for all color values.