AI-Powered Color Scheme Standardization

This document describes the AI-powered tools created to standardize the color scheme across all components in the Sports Bar AI Assistant application.

Overview

The system uses local AI (Ollama) to automatically analyze React components, identify styling inconsistencies, and suggest or apply fixes to match the standardized color scheme.

Components

1. Color Scheme Standard (`COLOR_SCHEME_STANDARD.md`)

A comprehensive style guide that documents:
- **Color palette**: Background, text, accent, and border colors
- **Component patterns**: Standardized styling for cards, buttons, badges, inputs, tabs, etc.
- **Rules to follow**: Guidelines for text contrast, component styling, and interactive elements
- **Anti-patterns**: What NOT to use
- **Accessibility standards**: Contrast ratios and focus indicators
- **Implementation checklist**: Step-by-step guide for updating components

This serves as the single source of truth for all styling decisions.

2. AI Style Analyzer (`scripts/ai-style-analyzer.js`)

An intelligent analyzer that:
- **Scans all React components** in the `src/` directory
- **Uses Ollama AI** to compare each component against the style guide
- **Identifies issues** with backgrounds, text colors, borders, badges, buttons, icons, and cards
- **Generates detailed reports** with specific suggestions for each issue
- **Calculates statistics** on total files, issues by severity, and problem areas

Usage:

```
cd /home/ubuntu/Sports-Bar-TV-Controller
node scripts/ai-style-analyzer.js
```

The analyzer will:
1. Check if Ollama is installed and running
2. Load the color scheme standard
3. Find all `.tsx` and `.jsx` files
4. Analyze each file using AI
5. Generate a detailed JSON report in `ai-style-reports/`
6. Display a summary in the console

Output:

- **JSON report**: `ai-style-reports/style-analysis-[timestamp].json`
- **Console summary**: Statistics and severity breakdown

- **Issue details**: Line numbers, current vs. suggested classes, and explanations

3. AI Style Fixer (`scripts/ai-style-fixer.js`)

An automated fixer that:
- **Loads analysis reports** from the analyzer
- **Applies suggested fixes** to component files
- **Creates backups** before making changes
- **Supports multiple modes**:
- **Interactive**: Review and approve each file individually
- **Auto-fix**: Apply all fixes automatically
- **Review-only**: Display issues without making changes

Usage:

```
# List available reports
node scripts/ai-style-fixer.js

# Apply fixes from a specific report
node scripts/ai-style-fixer.js ai-style-reports/style-analysis-[timestamp].json
```

The fixer will:
1. Load the specified report
2. Ask which mode to use (interactive, auto-fix, or review-only)
3. Process each file with issues
4. Create backups in `ai-style-backups/` before making changes
5. Apply the suggested fixes
6. Display a summary of changes

Safety Features:

- **Automatic backups**: Every modified file is backed up with timestamp
- **Interactive mode**: Review each file before changes
- **Regex-based replacement**: Precise matching to avoid unintended changes
- **Dry-run option**: Review-only mode to see what would change

Workflow

Initial Analysis

```
cd /home/ubuntu/Sports-Bar-TV-Controller
node scripts/ai-style-analyzer.js
```

This will scan all components and generate a report showing:
- Which files have styling issues
- Severity of each issue (high, medium, low)
- Specific problematic class names
- Suggested replacements

Review and Fix

After analysis, you have three options:

Option 1: Interactive Fixing

```
node scripts/ai-style-fixer.js ai-style-reports/style-analysis-[latest].json
# Choose option 1 for interactive mode
```

This lets you:
- Review each file's issues
- Decide whether to apply fixes
- Skip files you want to handle manually
- Quit at any time

Option 2: Auto-Fix All

```
node scripts/ai-style-fixer.js ai-style-reports/style-analysis-[latest].json
# Choose option 2 for auto-fix mode
```

This will:
- Apply all fixes automatically
- Create backups of all modified files
- Show a summary of changes

Option 3: Review Only

```
node scripts/ai-style-fixer.js ai-style-reports/style-analysis-[latest].json
# Choose option 3 for review-only mode
```

This will:
- Display all issues in detail
- Not modify any files
- Help you understand what needs to be changed

Testing and Verification

After applying fixes:

1. **Run the development server**:
   `bash`
   `   cd app && yarn dev`

2. **Test the application**:
   - Check all pages load correctly
   - Verify text is readable
   - Ensure buttons and interactive elements work
   - Check that the dark theme is consistent

3. **Run the analyzer again**:
   `bash`
   `   node scripts/ai-style-analyzer.js`
   This verifies that issues were fixed correctly

4. **If issues occur**, restore from backups:
   `bash`

```
# Backups are stored in ai-style-backups/
cp ai-style-backups/ComponentName.tsx.[timestamp].bak src/components/ComponentName.tsx
```

Example Analysis Output

```json
{
  "filePath": "components/SomeComponent.tsx",
  "hasIssues": true,
  "severity": "high",
  "issues": [
    {
      "line": 42,
      "type": "background",
      "current": "bg-white",
      "suggested": "bg-slate-800",
      "reason": "White background should be replaced with dark theme color"
    },
    {
      "line": 45,
      "type": "text",
      "current": "text-gray-900",
      "suggested": "text-slate-100",
      "reason": "Dark text on dark background has poor contrast"
    }
  ],
  "summary": "Component uses light theme colors inconsistent with style guide"
}
```

Key Features

AI-Powered Analysis

- Uses natural language understanding to identify styling issues
- Considers context (e.g., nested components, hover states)
- Provides human-readable explanations
- Learns from the comprehensive style guide

Safe Automated Fixing

- Creates timestamped backups
- Interactive approval process
- Precise regex-based replacements
- Rollback capability

Comprehensive Reporting

- JSON format for programmatic processing
- Human-readable console summaries
- Severity categorization
- Statistics and metrics

Customization

Changing the AI Model

Edit `scripts/ai-style-analyzer.js`:

```
const CONFIG = {
  ollamaModel: 'llama3.2',  // Change to 'mistral', 'codellama', etc.
  // ...
};
```

Adjusting Analysis Parameters

Edit the prompt in `analyzeComponent()` function to:
- Focus on specific styling aspects
- Change severity thresholds
- Add custom rules

Extending the Style Guide

Edit `COLOR_SCHEME_STANDARD.md` to:
- Add new component patterns
- Define additional color schemes
- Update accessibility requirements
- Document new anti-patterns

Best Practices

1. **Run analyzer regularly**: After adding new components or features
2. **Fix high-severity issues first**: Start with the most critical problems
3. **Test incrementally**: Fix a few files, test, then continue
4. **Keep backups**: Don't delete backup files until you're sure fixes work
5. **Update the style guide**: Document any new patterns you introduce
6. **Review AI suggestions**: The AI is smart but not perfect—use your judgment

Troubleshooting

"Ollama is not installed"

```
# Install Ollama
curl -fsSL https://ollama.ai/install.sh | sh

# Or run the local AI installation script
cd /home/ubuntu/Sports-Bar-TV-Controller
./install-local-ai.sh
```

"Model not found"

```
# Pull the model
ollama pull llama3.2
```

“Unable to parse AI response”

- The AI response might not be valid JSON
- Check `ai-style-reports/` for the raw response
- Try a different model (e.g., `codellama`)

“Fixes caused errors”

```
# Restore from backup
cp ai-style-backups/ComponentName.tsx.[timestamp].bak src/components/ComponentName.tsx

# Or restore all files
cd ai-style-backups
for file in *.bak; do
  original=$(echo $file | sed 's/\.[0-9T-]*\.bak$//')
  cp "$file" "../src/components/$original"
done
```

Future Enhancements

Potential improvements to the system:

1. **Web Interface**: Browser-based UI for reviewing and applying fixes
2. **Git Integration**: Automatic commit creation for each fix
3. **CI/CD Integration**: Run analyzer in GitHub Actions
4. **Custom Rules Engine**: Define project-specific style rules
5. **Before/After Preview**: Visual comparison of changes
6. **Learning Mode**: AI learns from manual corrections
7. **Multi-project Support**: Analyze multiple projects
8. **Export Fixes**: Generate pull requests automatically

Conclusion

This AI-powered standardization system provides:
- **Automated analysis** of thousands of lines of code
- **Intelligent suggestions** based on the style guide
- **Safe application** of fixes with backups
- **Significant time savings** compared to manual updates
- **Consistent styling** across the entire application

By leveraging local AI, you can maintain code quality and consistency at scale without manual tedium.

Created: October 1, 2025
Version: 1.0
Author: Sports Bar AI Assistant Development Team