Epic 1.2: Storybook Foundation

Epic Overview

This epic establishes Storybook as the comprehensive documentation and development platform for THE WHEEL design system, with full workspace context support and interactive component showcases. All designs and styles should follow THE WHEEL - Complete 120+ Page Brand Bible.pdf and THE WHEEL - Complete Brand Guide & Assets Package and the midnight-amber-brand.html and all logos are in the logos folder (all in the DOCS folder in the project root)

Priority: P0 (Critical)

Timeline: 2 weeks

Dependencies: Epic 1.1 (Monorepo Architecture Setup)

Story 1.2.1: Storybook Configuration

Overview

Configure Storybook 7.0+ for the monorepo with essential addons, workspace context integration, and comprehensive component documentation capabilities.

AI Developer Prompt

You are configuring Storybook for THE WHEEL design system monorepo. Building on the package structure and component inventory from Epic 1.1, you need to create a comprehensive Storybook setup that showcases all components with workspace context awareness.

Context

- Monorepo with 6 packages containing 156 components
- Existing components have sophisticated theming and real-time features
- Need to support workspace context switching (consultant, client, admin, expert, tool creator, founder)
- Components need to be testable in isolation
- Must integrate with existing TypeScript and build system

Requirements

1. **Install and configure Storybook 7.0+ with essential addons:**

-

- **addon-essentials**: Controls, actions, docs **addon-a11y**:
- Accessibility testing **addon-interactions**: Interaction testing
- **addon-viewport**: Responsive testing • **addon-docs**:

  Documentation generation

2. **Configure monorepo integration:**

- Story discovery across all packages
- Proper TypeScript integration
- Component documentation generation Asset
- and static file handling

3. **Set up workspace context system:**

- Global decorators for workspace theming
- Toolbar controls for context switching
- Theme provider integration User
- role simulation

Specific Tasks

javascript

```
// .storybook/main.ts configuration
module.exports = { stories: [
   '../packages/*/src/**/*.stories.@(js|jsx|ts|tsx|mdx)',
   '../packages/*/src/**/*.docs.mdx'
  ],
  addons: [
   '@storybook/addon-essentials',
   '@storybook/addon-a11y',
   '@storybook/addon-interactions',
   '@storybook/addon-viewport',
   '@storybook/addon-docs',
   'storybook-addon-theme-switcher'
  ],
  framework: { name:
   '@storybook/react-vite', options:
   {}
  },
  typescript: {
   check: false, reactDocgen: 'react-docgen-
   typescript', reactDocgenTypescriptOptions: {
    shouldExtractLiteralValuesFromEnum: true, propFilter:
    (prop) => {
     if (prop.parent) {
          return !prop.parent.fileName.includes('node_modules');
     } return
     true;
    }
   }
  },
  viteFinal: async (config) => { // Custom Vite
   configuration for monorepo return config;
  }
};
```

Documentation Required

- Storybook setup and configuration guide
- Story writing guidelines and templates

  Workspace context usage documentation

-
- Component documentation standards
- Addon configuration and usage guide
- Development workflow with Storybook

Testing Requirements

- Storybook build validation tests
- Story compilation tests
- Addon functionality tests
- Workspace context switching tests
- TypeScript integration tests
- Performance tests for story loading

Integration Points

- Integrate with existing theme system
- Connect to workspace context providers
- Support for real-time collaboration features
- Integration with build system from Epic 1.1
- CI/CD pipeline integration for visual testing

Deliverables

- Fully configured Storybook instance
- Global decorators and toolbar controls
- TypeScript integration and documentation
- Story templates and writing guidelines
- Workspace context demonstration
- Performance-optimized Storybook build

Configuration Requirements

- Support for all 6 packages in monorepo
- Workspace context switching in toolbar
- Theme variations for different contexts
- Responsive testing across device sizes

- Accessibility testing integration
- Auto-generated component documentation

Performance Requirements

- Storybook startup under 30 seconds
- Story hot reload under 2 seconds
- Context switching under 500ms
- Documentation generation under 1 minute
- Build size under 50MB

Story 1.2.2: Workspace Context System

Overview

Implement a comprehensive workspace context system within Storybook that allows testing components in different workspace environments with proper theming and permissions.

AI Developer Prompt

You are implementing the workspace context system for THE WHEEL Storybook. Building on the Storybook configuration from Story 1.2.1, you need to create context-aware decorators that allow testing components in different workspace environments.

Context

- Storybook 7.0+ is configured with monorepo integration
- Existing workspace context providers for consultant, client, admin, expert, tool creator, founder roles
- Sophisticated theming system with CSS variables
- Real-time collaboration features that depend on workspace context
- Need to simulate different user roles and workspace states

Requirements

1. **Create workspace context decorators:**

- **WorkspaceProvider** decorator for workspace simulation
- **ThemeProvider** decorator for theme switching
- **UserProvider** decorator for role simulation

-

  PermissionProvider decorator for access control testing

2. **Implement global toolbar controls:**

- Workspace context selector (6 contexts)
- Theme switcher with live preview
- User role selector with permission simulation Mock
- data toggle for realistic testing

3. **Set up context persistence:**

- Remember selected context across story navigation
- URL parameter support for sharing contexts
- Local storage integration for developer preferences Context
- validation and error handling

Specific Tasks

typescript

```
// .storybook/decorators/WorkspaceDecorator.tsx import {
WorkspaceProvider } from '@wheel/workspace'; import {
ThemeProvider } from '@wheel/themes';

export const WorkspaceDecorator = (Story, context) => { const {
  globals } = context; const workspaceType = globals.workspace
  || 'consultant'; const theme = globals.theme || 'light'; const
  userRole = globals.userRole || 'admin';

  const workspaceConfig = {
    type: workspaceType,
theme: theme,     user: {
      role: userRole,
      permissions: getPermissionsForRole(userRole)
    },
    features: getFeaturesForWorkspace(workspaceType)
  };

  return (
    <WorkspaceProvider workspace={workspaceConfig}>
      <ThemeProvider theme={theme}>
        <Story />
      </ThemeProvider>
    </WorkspaceProvider>
  );
};

// Toolbar configuration export const
globalTypes = {   workspace: {     name:
'Workspace Context',     description: 'Select
workspace context',     defaultValue:
'consultant',     toolbar: {       icon: 'globe',
items: [
        { value: 'consultant', title: 'Consultant' },
        { value: 'client', title: 'Client' },
        { value: 'admin', title: 'Admin' },
        { value: 'expert', title: 'Expert' },
        { value: 'toolCreator', title: 'Tool Creator' },
        { value: 'founder', title: 'Founder' }
      ],
      showName: true
    }
  }
};
```

Documentation Required

- Workspace context system architecture
- Context decorator usage guide
- Mock data and testing scenarios
- Context switching workflow documentation
- Troubleshooting guide for context issues Best
- practices for context-aware stories

Testing Requirements

- Context decorator functionality tests
- Theme switching validation tests
- User role simulation tests
- Permission system integration tests
- Context persistence tests
- Error handling and recovery tests

Integration Points

- Integration with existing workspace context providers
- Theme system CSS variable integration
- Real-time collaboration feature compatibility
- User management system integration
- Permission and access control integration

Deliverables

- Workspace context decorator system
- Global toolbar controls for context switching
- Context persistence and URL parameter support

- Mock data providers for testing
- Context validation and error handling
- Comprehensive documentation and examples

Context Features

- **6 Workspace Contexts**: Consultant, Client, Admin, Expert, Tool Creator, Founder
- **Theme Variations**: Light/Dark modes per context
- **User Roles**: Admin, Manager, User, Guest
- **Permission Sets**: Context-specific permissions
- **Mock Data**: Realistic data for each context
- **State Persistence**: Maintain context across sessions

Story 1.2.3: Brand Integration

Overview

Integrate brand assets and identity into Storybook, creating a professional, branded experience that reflects THE WHEEL's sophisticated design system.

AI Developer Prompt

You are integrating brand assets and identity into THE WHEEL Storybook. Building on the workspace context system from Story 1.2.2, you need to create a branded Storybook experience that reflects the sophisticated design system.

Context

- Storybook with workspace context system is configured
- Existing brand bible with colors, typography, and logo assets
- Sophisticated theming system with CSS variables
- Need to support different brand contexts (consultant, client, admin, expert, tool creator, founder) Must
- create professional, polished Storybook experience

Requirements

1. **Extract and integrate brand assets:**

- Brand colors from existing brand bible
- Typography system with web fonts
- Logo assets and brand imagery
- Icon system and iconography Brand
- guidelines documentation

2. Configure Storybook theming:

- Custom Storybook theme with brand colors
- Typography integration in Storybook UI
- Logo integration in Storybook sidebar
- Brand-consistent navigation and controls Professional
- color scheme and spacing

3. Create brand documentation:

- Brand guidelines page in Storybook
- Color palette documentation
- Typography system showcase
- Logo usage guidelines Brand
- asset library

Specific Tasks

javascript

```
// .storybook/theme.js import { create } from
'@storybook/theming';

export default create({
  base: 'light',

  // Brand colors
  colorPrimary: '#2563eb', // Consultant blue colorSecondary:
  '#059669', // Client green

  // UI colors appBg: '#f9fafb',
  appContentBg: '#ffffff',
  appBorderColor: '#e5e7eb',
  appBorderRadius: 8,

  // Typography fontBase: '"Inter", -apple-system, BlinkMacSystemFont,
  sans-serif', fontCode: '"JetBrains Mono", "Monaco", monospace',

  // Text colors textColor:
  '#111827',
  textInverseColor: '#ffffff', textMutedColor:
  '#6b7280',

  // Toolbar colors barTextColor:
  '#6b7280', barSelectedColor:
  '#2563eb', barBg: '#ffffff',

  // Form colors inputBg:
  '#ffffff', inputBorder:
  '#e5e7eb', inputTextColor:
  '#111827',
  inputBorderRadius: 6,

  // Brand assets brandTitle: 'THE WHEEL
  Design System', brandUrl:
  'https://thewheel.design', brandImage:
  '/assets/logo.svg', brandTarget: '_self'
});
```

Documentation Required

- Brand integration guide and asset inventory
- Color palette and typography documentation
- Logo usage guidelines and specifications
- Brand asset library and management
- Storybook theming customization guide Brand
- guidelines implementation notes

Testing Requirements

- Brand asset loading and display tests
- Color palette validation tests
- Typography rendering tests
- Logo display and scaling tests
- Brand consistency validation tests
- Cross-browser brand rendering tests

Integration Points

- Integration with existing theme system
- Workspace context brand variations
- CSS variable system integration
- Asset management and optimization
- Build system integration for brand assets

Deliverables

- Branded Storybook with professional appearance
- Brand asset library and documentation
- Color palette and typography system
- Logo integration and usage guidelines
- Brand guidelines documentation in Storybook Brand
- asset management system

Brand Features

- • **Professional Appearance**: Polished, sophisticated UI **Brand Consistency**: Unified visual language
- **Context Variations**: Brand adapts to workspace context
- **Asset Library**: Centralized brand assets
- **Documentation**: Complete brand guidelines **Responsive**:
- Brand scales across devices

Performance Requirements

- Brand asset loading under 2 seconds
-
- Image optimization for web
- CSS optimization for theming
- Cache optimization for assets

Font loading optimization

Timeline and Dependencies

Timeline

- **Week 1**: Story 1.2.1 - Storybook Configuration
- **Week 2**: Story 1.2.2 - Workspace Context System **Week 2**:
- Story 1.2.3 - Brand Integration

Dependencies

- Requires Epic 1.1 completion (Monorepo Architecture) Stories
- 1.2.2 and 1.2.3 depend on Story 1.2.1

Success Metrics

- All 156 components documented in Storybook
- Workspace context switching fully functional
- Brand integration complete and consistent
- Performance targets met
- Developer satisfaction with documentation

Risk Mitigation

- Test Storybook configuration incrementally
- Validate context system with real components
- Get brand approval before final integration
- Performance test with full component library
- Have fallback for complex integrations