Feature Specification: Dark Mode Implementation

1. Introduction

- **1.1 Feature Name:** Dark Mode
- **1.2 Feature ID:** FEAT-001 (Internal tracking ID)
- **1.3 Version:** 1.0
- **1.4 Date:** June 12, 2025
- **1.5 Author(s):** Maximo Valencia
- **1.6 Stakeholders:** Product Management, UI/UX Team, Engineering Team, QA Team, Marketing.
- **1.7 Category:** Frontend

2. Overview

This feature aims to provide users with an alternative visual theme for the application interface, specifically a "Dark Mode." Dark Mode will invert the color scheme, typically using a dark background with light text and elements, to reduce eye strain in low-light environments and offer a personalized viewing experience. This feature will be accessible via user settings.

3. Goals and Objectives

- **3.1 Primary Goal:** To improve user experience by offering a customizable visual theme that reduces eye strain.
- **3.2 Objectives:**
    - Implement a toggle in user settings to switch between Light and Dark Mode.
    - Ensure all primary UI elements (headers, footers, navigation, content areas, buttons, forms) render correctly in Dark Mode.
    - Maintain readability and usability across all components.
    - Persist the user's selected theme across sessions.
    - Support both manual selection and system preference (e.g., prefers-color-scheme).

4. User Stories

- As a user, I want to be able to switch to a Dark Mode theme so that I can reduce eye strain, especially during evening use.
- As a user, I want my chosen theme (Light or Dark) to be remembered when I close and re-open the application.
- As a user, I want the application to respect my operating system's dark mode preference if I haven't manually selected a theme.
- As a developer, I want a clear and maintainable way to manage the application's visual themes.

5. Functional Requirements

- **FR-5.1 Theme Toggle:**
    - A dedicated toggle switch for "Theme" (with options "Light" and "Dark") must be present in the User Profile/Settings section.
    - Clicking the toggle must instantly apply the selected theme across the entire application.
- **FR-5.2 Theme Persistence:**
    - The user's theme preference must be stored locally (e.g., localStorage) and applied automatically upon subsequent visits.
    - If no preference is stored, the application should default to prefers-color-scheme if supported by the browser/OS. Otherwise, default to Light Mode.
- **FR-5.3 UI Element Styling:**
    - All core application components (see Section 7) must be styled appropriately for Dark Mode, ensuring sufficient contrast and readability.
    - Interactive elements (buttons, links, form fields) must clearly indicate their states (hover, active, disabled) in both themes.
- **FR-5.4 Image/Icon Handling:**
    - Images and icons should either adapt to the dark background or be sufficiently visible without becoming distracting. Consider

using SVG icons that can be styled with CSS. For raster images, evaluate whether a darker version is necessary or if a slight desaturation is sufficient.

- **FR-5.5 Third-Party Integrations:**
    - Assess and address styling for any embedded third-party widgets or components to ensure they blend reasonably well with the chosen theme. (E.g., if using a chat widget, check its dark mode support).

6. Non-Functional Requirements

- **NFR-6.1 Performance:** Switching themes should be instant and not introduce noticeable latency or page reloads.
- **NFR-6.2 Accessibility:**
    - Ensure sufficient color contrast ratios (WCAG 2.1 AA or AAA) for both Light and Dark modes.
    - Avoid using color as the sole means of conveying information.
- **NFR-6.3 Maintainability:** The styling solution should be modular and easily extensible for future theme enhancements. (e.g., CSS variables, CSS-in-JS theming).
- **NFR-6.4 Cross-Browser Compatibility:** Support for modern browsers (Chrome, Firefox, Safari, Edge) on desktop and mobile.

7. UI/UX Specifications

- **7.1 Theme Toggle Placement:** The theme toggle will be located within the "Settings" menu, under a new "Appearance" sub-section.
- **7.2 Visual Design:** Refer to the attached Figma/Sketch designs for specific color palettes, typography, and component styling for both Light and Dark modes. (Link to design assets: [e.g., Figma Link])
- **7.3 Component-Specific Adaptations:**
    - **Navigation Bar:** Background color will be a dark shade, text will be light, active links will have a distinct accent color.

    - **Content Areas:** Background will be a dark gray, body text a light off-white.
    - **Buttons:** Primary buttons will have a dark background with light text in Dark Mode, secondary buttons will have a light background with dark text.
    - **Forms:** Input fields will have dark backgrounds and light text, with clear focus states.
    - **Tables:** Alternating row colors should maintain readability.
    - **Charts/Graphs:** Colors used in data visualization should be re-evaluated for contrast and clarity in Dark Mode.
- **7.4 Animations/Transitions:** Smooth transitions (e.g., transition: background-color 0.3s ease;) should be applied when switching themes to enhance the visual experience.

8. Technical Design & Implementation Details (High-Level)

- **8.1 Technology Stack:** React, JavaScript, TypeScript, Tailwind CSS
- **8.2 Theming Approach:**
    - Utilize CSS Variables (Custom Properties) for defining colors. This allows for easy swapping of values based on a parent class or data attribute on the <body> or <html> element.
    - Alternatively, if using a CSS-in-JS library, define theme objects that can be dynamically passed.
- **8.3 Theme State Management:**
    - Store the current theme in a React Context/Redux store for global access.
    - Persist the theme preference using localStorage (or similar client-side storage).
- **8.4 System Preference Detection:** Use window.matchMedia('(prefers-color-scheme: dark)') to detect OS dark mode preference.

9. Test Cases (Examples)

- **TC-9.1:** Verify theme toggle functionality:
    - Toggle from Light to Dark.
    - Toggle from Dark to Light.
    - Verify all UI elements update correctly.
- **TC-9.2:** Verify theme persistence:
    - Select Dark Mode, close and reopen the application. Verify Dark Mode is active.
    - Select Light Mode, close and reopen the application. Verify Light Mode is active.
- **TC-9.3:** Verify prefers-color-scheme integration:
    - No local preference set. Set OS to Dark Mode. Verify application defaults to Dark Mode.
    - No local preference set. Set OS to Light Mode. Verify application defaults to Light Mode.
- **TC-9.4:** Verify contrast ratios using a WCAG checker in both modes.
- **TC-9.5:** Verify accessibility with keyboard navigation and screen readers in both modes.
- **TC-9.6:** Check component rendering for all key pages/modules in both themes.

10. Open Questions / Dependencies

- How should embedded iframes (e.g., YouTube videos) be handled? (May require specific API calls or external libraries if no native dark mode support).
- Are there any specific third-party libraries that *do not* support dark mode, and what's our fallback strategy?
- Should we offer more than two themes in the future (e.g., "Sepia" or custom color themes)? (Out of scope for V1 but good to consider for architecture).
- What is the specific behavior if a user's prefers-color-scheme changes *while* the app is open, and they have no explicit preference set?

11. Future Enhancements (Out of Scope for V1)

- Automatic theme switching based on time of day.
- Support for multiple custom themes.
- Theme editor for users.