This document outlines the project scope, required developer skills, and detailed feature requirements for the **Hospice Outcomes and Patient Evaluation (HOPE) Assessment Application**.

Project Brief: Next.js HOPE Assessment Application

1. Project Goal and Scope

The goal is to develop a highly specialized, secure, and compliant Next.js frontend application that serves as the data entry interface for the CMS **Hospice Outcomes and Patient Evaluation (HOPE) v1.01** assessment tool. The application will be a complex form and data management system for hospice clinicians.

Key Technology Stack:

- **Frontend:** Next.js (preferably using the App Router) with React and TypeScript).
- **Backend: Fastify.js** (Node.js) API.
- **Database: PostgreSQL**

Required Source Documents:

The developer must meticulously study and implement the rules, item definitions, and structure contained within the following provided CMS documents:

1. **HOPE Guidance Manual v1.01** (hope-guidance-manualv101.pdf-0): This is the single source of truth for all business logic, instructions, and skip patterns.
2. **HOPE All Items** (hope-v101all-item508c.pdf): For the complete, sequential structure of the assessment.
3. **HOPE Admission (ADM)** (hope-v101admission508c.pdf): Defines the assessment for the admission timepoint.
4. **HOPE Update Visit (HUV)** (hope-v101hope-update-visit508.pdf): Defines the periodic assessment for the update visit timepoint.
5. **HOPE Discharge (DC)** (hope-v101discharge508c.pdf): Defines the assessment for the discharge timepoint.

2. Required Developer Skills and Experience

Skill Category	Requirement
Core Frontend	**Next.js** (App Router), **React** (Hooks, Context), and **TypeScript** (Mandatory for type safety).
Data & State	Expertise in complex form state management (e.g., react-hook-form or Formik) to handle the hundreds of interconnected data points in the HOPE assessment.
API/Backend	Proven experience integrating with **Node.js RESTful APIs** for data fetching, synchronization (auto-save), and structured

Skill Category	Requirement
	data submission.
UX & UI	Experience with responsive design and modern component libraries (e.g., MUI, Ant Design).
Compliance	Strong familiarity with building **508-compliant** and **WCAG 2.1 (A/AA)** accessible applications, as this is a federal healthcare application.

3. Core Feature Requirements

The frontend must accurately reflect the structure and logic of the HOPE tool across all timepoints.

A. Assessment Form Implementation

- **Three Distinct Forms:** The developer must build three separate, dedicated form views based on the provided documents:
    1. **Admission (ADM) Assessment**
    2. **Update Visit (HUV) Assessment**
    3. **Discharge (DC) Assessment**
- **Conditional Logic and Skip Patterns:** All skip patterns and branching logic outlined in the **HOPE Guidance Manual** must be precisely implemented. For example, if a specific item is marked "Not Applicable," subsequent, dependent questions must be hidden or disabled.
- **Item-Level Data Validation:** Robust client-side validation is required for every item (e.g., date formats, valid code ranges, required fields) to prevent invalid data submission to the backend.

B. User Experience and Clinician Workflow

- **Section Navigation:** Implement a persistent, side-panel navigation reflecting the alphabetical sections of the HOPE tool (e.g., Section A, Section C, Section J, etc.). This navigation should allow users to jump between sections while preserving form data.
- **Guidance Display:** Develop a mechanism (e.g., tooltips, popovers, or a separate panel) to display the relevant *HOPE Guidance Manual* instructions for any item currently in focus, assisting clinicians with accurate data abstraction.
- **Data Capture and Synchronization:** Implement an automatic draft-saving feature that periodically syncs data with the Node.js backend to prevent data loss.

C. Finalization and Compliance

- **Section Z (Record Administration):** Implement the electronic signature and certification blocks, including the fields for **Signature(s) of Person(s) Completing the Record (Z0400)** and **Signature of Person Verifying Record Completion (Z0500)**, as detailed in the Section Z of the assessment documents.
- **API Data Mapping:** Work collaboratively with the Node.js developer to ensure the frontend form data structures map directly to the required API submission schema, which is necessary for generating the final CMS-compliant XML output (a backend responsibility).

- **A0050. Type of Record:** The frontend must support the selection and functionality for all record types: "Add new record," "Modify existing record," and "Inactivate existing record," typically found in Section A.