

# CDSS Project Documentation

Clinical Decision Support System Team

January 24, 2026

## 1 Project Overview

### 1.1 Concept

The Clinical Decision Support System (CDSS) is a web-based platform designed to digitize, standardize, and accelerate musculoskeletal diagnosis. It shifts the clinical approach from a broad investigation ("expanse") to a focused, systematic analysis ("essence") by utilizing patient-reported data and AI-driven preliminary assessments to support licensed clinicians.

### 1.2 Clinical Purpose

The system addresses subjectivity and lack of standardization in physical therapy assessments. It acts as a bridge between patient symptoms and professional diagnosis, streamlining history taking and providing evidence-based decision support.

### 1.3 Scope Boundaries

- **What it IS:** A data gathering, preliminary assessment, and workflow management tool.
  - **What it is NOT:** A replacement for a human clinician. The AI component provides a temporal (provisional) diagnosis only. The final diagnosis and treatment plan are strictly the responsibility of the licensed therapist.
- 

## 2 User Flows

### 2.1 Patient Flow

1. **Authentication:** Login/Sign-up.
2. **Dashboard:** View "Welcome Back" screen containing:
  - Assessments: Progress bar showing completion status (e.g., "22/22 Questions Completed").
  - Appointments: Upcoming sessions (e.g., "Infrared Therapy").
  - Guided Self-Tests: Video-based tutorials (e.g., Leg Raise Test).
  - Progress Overview: Line graph tracking monthly pain levels.
3. **Assessment:**
  - Select body region (Body Map visualization).
  - Input Pain Level (Slider 1–10).

- Complete Questionnaire (History, Medications).
  - Media Upload: Upload MRI, X-Rays, or clinical reports via modal.
4. **Results:** View “AI Preliminary Diagnosis (Provisional)” screen displaying:
- Possible Condition (e.g., Lumbar Disc Herniation).
  - Confidence Score (percentage).
  - Reasoning Indicators (e.g., “Pain worsens while walking”).
  - **Disclaimer:** Explicit statement that this is non-final and requires therapist confirmation.

## 2.2 Clinician/Admin Flow

1. **Dashboard:**
  - Activity Monitor: Graphs for “Users Activity” and “Therapist Activity.”
  - Case Management: “New Cases” cards displaying patient name, gender, and status.
  - Staff Management: Lists for “New Approved Therapists” and “Pending.”
2. **Case View:**
  - Patient Profile: Demographics and contact info.
  - Diagnostic Review: View the AI-generated Temporal Diagnosis, Risk Level, and Insights.
  - Key Findings: Review structured data (Pain Triggers, Relievers, Range of Motion).
3. **Action:**
  - Session Notes: Input Objective measurements (VAS Scale, Oxford Motor Grade).
  - Treatment Planning: Create plans (Intervention Type, Duration, Goal).
  - Referral: Generate referrals or order further imaging.

## 3 System Architecture

### 3.1 Frontend Layer

- **Patient Interface:** Optimized for data entry (questionnaires) and media consumption.
- **Clinician Interface:** Dashboard-centric, data-dense views for case management.
- **Shared Components:** Dark mode UI system (no gradients), reusable input controls.

### 3.2 Backend Layer (Next.js API)

- Serves as the orchestration layer between the UI, the Database, and the AI Agent.
- Handles file upload signatures for Cloudinary.
- Manages session-based authentication cookies.

### 3.3 AI Temporal Diagnosis Module

- **Engine:** Mistral AI Agent.
- **Role:** Replaces hard-coded heuristic logic with a Knowledge-Based RAG approach.
- **Integration:** Stateless API calls receiving patient JSON data.

### 3.4 Shared Utilities

- **Zustand Stores:** Client-side state management.
- **Mongoose Models:** Data validation schemas.

## 4 AI Temporal Diagnosis Module Details

### 4.1 Purpose

To collate patient subjective data and compare it against the structured medical rules to produce a preliminary assessment.

### 4.2 Inputs & Outputs

- **Inputs:** Demographics, Body Region, Symptom Profile, History.
- **Outputs (JSON Structure):**

```
{
  "temporal_diagnosis": "Possible Condition Name",
  "confidence_score": 85,
  "reasoning": ["Point 1", "Point 2"],
  "risk_level": "Low/Moderate/High"
}
```

### 4.3 Clinical & Ethical Constraints

- **Non-Determinism:** AI must never output a diagnosis as "Final."
- **Transparency:** UI must display Reasoning Indicators.
- **Limitation:** The AI analyzes text/structured data, not raw image interpretation.

## 5 Technology Stack

- **Framework:** Next.js (App Router)
- **Language:** JavaScript
- **Database:** MongoDB (Mongoose)
- **Authentication:** Session-based (Cookies)
- **State Management:** Zustand
- **Media Storage:** Cloudinary
- **AI Agent:** Mistral AI

## 6 Conceptual Data Flow

1. **Intake:** Patient submits Assessment Form → POST /api/assessment.
2. **Processing:** Backend retrieves Medical Rules → Sends Data to Mistral AI.
3. **Analysis:** Mistral generates Temporal Diagnosis JSON.
4. **Storage:** Backend saves Data + AI Result to MongoDB.
5. **Review:** Clinician requests GET /api/cases → Fetches data.
6. **Finalization:** Clinician inputs Objective Data → Finalizes record.

## 7 Repository Structure

```
/
|-- src/
|   |-- app/
|   |   |-- (auth)/           # Login, Signup
|   |   |-- (clinician)/      # Dashboards, Case Views
```

```

|   |   |-- (patient)/      # Assessment Wizard
|   |   `-- api/           # Backend Routes
|   |-- lib/
|   |   |-- db/            # Mongoose connection
|   |   |-- ai/            # Mistral Config
|   |   `-- auth/          # Session utilities
|   |-- models/            # Mongoose Schemas
|   |-- store/             # Zustand Stores
|   `-- components/        # Shared UI Components
`-- README.md

```

## 8 Team Workflow Rules

### 8.1 Branching Strategy

- **main:** Single source of truth. Production ready. Protected.
- **[developer-name]:** Persistent personal working branch.

### 8.2 Contribution Process

1. Push code to personal [developer-name] branch.
2. Open Pull Request (PR) from personal branch to main.
3. Requirement: PR must be reviewed by lead dev.
4. Once merged, pull main back into personal branch to sync.

### 8.3 Constraints

- No direct commits to main.
- Code must be formatted (Prettier) before pushing.