**Frontend Development Plan (SehatAI)**

**Project:** SehatAI – AI-powered Healthcare Assistant  
**Module:** Frontend (Next.js)  
**Version:** 1.0  
**Date:** September 2025

**1. Introduction**

The frontend is the **user-facing layer** of SehatAI. It provides doctors, paramedics, and staff in rural clinics with a simple and intuitive interface to:

* Upload chest X-rays.
* Upload handwritten prescriptions (Urdu/English).
* Input patient demographic data.
* View AI-generated results (diagnosis, digitized prescription, risk score).
* Export/download reports.

The frontend will be implemented in **Next.js + Tailwind CSS** with REST API integration to the backend.

**2. Objectives**

* Provide a **clean, fast, and bilingual (English + Urdu)** UI.
* Minimize technical barriers for non-technical users.
* Ensure easy data input (file uploads + forms).
* Display AI results clearly and understandably.
* Support **Phase-1 MVP requirements** only (keep it lightweight).

**3. Functional Requirements (Frontend-Specific)**

1. **Authentication (basic)**
   * Login page with simple credentials (Phase-1 minimal).
2. **Dashboard (main interface)**
   * Upload X-ray (JPG/PNG).
   * Upload prescription (JPG/PNG).
   * Input patient demographic info (age, gender, weight, BP).
3. **Results Display**
   * X-ray diagnosis → Label (Normal / TB / Pneumonia).
   * Prescription OCR → Text area with extracted text.
   * Risk score → Bar or percentage display (Low/Medium/High).
4. **Reports**
   * Ability to export results as **PDF/CSV**.
5. **Language Toggle**
   * English/Urdu support (Next-intl / i18n).

**4. Non-Functional Requirements (Frontend-Specific)**

* **Usability:** Simple design, large buttons, readable fonts.
* **Performance:** Fast rendering, responsive for desktop & tablet.
* **Security:** HTTPS + token-based API calls.
* **Accessibility:** Clear contrast, support for low-literacy users.
* **Compatibility:** Runs on Chrome, Firefox, Edge (latest versions).

**5. Page/Component Structure**

/frontend

├── pages/

│ ├── index.tsx (Landing/Login Page)

│ ├── dashboard.tsx (Main dashboard after login)

│ ├── results.tsx (Results view)

│ ├── reports.tsx (Download reports)

│

├── components/

│ ├── Layout.tsx (Navbar, Sidebar)

│ ├── UploadXray.tsx (X-ray upload form)

│ ├── UploadPrescription.tsx (Prescription upload form)

│ ├── PatientForm.tsx (Demographic data form)

│ ├── ResultsCard.tsx (Display results in cards)

│ ├── LanguageSwitcher.tsx (English/Urdu toggle)

│

├── utils/

│ ├── api.ts (API service for backend calls)

│

├── styles/

│ ├── globals.css (Tailwind CSS setup)

**6. User Workflows**

**Workflow 1: X-ray Diagnosis**

1. Doctor logs in → Dashboard.
2. Uploads X-ray → API call to backend.
3. Result displayed → “Normal / TB / Pneumonia”.

**Workflow 2: Prescription OCR**

1. Upload prescription image.
2. API processes → returns extracted text.
3. Editable text box displayed for corrections.

**Workflow 3: Risk Scoring**

1. Fill patient demographics form.
2. API processes → risk score shown as % with color (green/yellow/red).

**Workflow 4: Export Report**

1. User clicks “Download Report”.
2. Generates PDF/CSV.

**7. Integration Points**

* **Backend API Endpoints:**
  + POST /api/xray → image → JSON diagnosis.
  + POST /api/ocr → image → JSON text.
  + POST /api/risk → JSON demographics → risk score.
* **Frontend Service Layer:**
  + utils/api.ts → central API handler using Axios/Fetch.

**8. Tech Stack**

* **Best Choice:**
  + **Next.js** (Frontend framework).
  + **Tailwind CSS** (Styling).
  + **Axios** (API calls).
  + **Next-intl** (i18n for English/Urdu).
* **Alternative:**
  + React CRA (if Next.js too heavy).
  + Bootstrap (instead of Tailwind).
  + Fetch API (instead of Axios).

**9. Deployment Plan**

* Containerized via **Docker**.
* Base Image: node:18-alpine.
* Build → next build && next start.
* Expose port **3000**.
* Integrated with docker-compose for orchestration with backend/models.

**10. Timeline (Frontend Tasks Only)**

* **Day 1–2:** Setup Next.js project, Tailwind, routing.
* **Day 3–4:** Build upload forms (X-ray & Prescription).
* **Day 5:** Patient demographic form + validation.
* **Day 6–7:** Results page with cards (Diagnosis, OCR, Risk).
* **Day 8:** Report export feature (PDF/CSV).
* **Day 9:** Language toggle (English/Urdu).
* **Day 10:** Integrate all with backend APIs.
* **Day 11–12:** Styling, responsiveness, usability testing.
* **Day 13:** Bug fixes, documentation updates.
* **Day 14:** Final build + Docker container.

**11. Risks & Limitations**

* If backend is delayed, frontend can only use **mock API responses** initially.
* Urdu i18n may need extra fonts and RTL (right-to-left) support.
* Phase-1 will be desktop-focused; mobile view only partially optimized.

**12. Future Enhancements**

* Mobile app (React Native wrapper).
* Multi-user accounts with roles.
* Real-time health dashboards for government officers.
* Integration with NADRA e-Health system.