Step-by-Step Implementation Plan

1. Project Setup

sehat-ai/

├── frontend/ (Next.js app)

├── backend/ (Flask/FastAPI for APIs)

├── models/ (AI/ML models code + inference scripts)

├── docs/ (SRS, SDD, model cards, etc.)

├── docker/ (Dockerfiles + docker-compose.yml)

└── README.md

2. Frontend (Next.js)

* Simple dashboard interface for Phase-1:
  + Upload X-ray (image).
  + Upload prescription (image).
  + Form for patient demographic data.
  + Display results → Diagnosis, Digital Prescription, Risk Score.
* Use TailwindCSS for fast styling.
* Connect to backend via REST API calls.

3. Backend (API Layer)

* Framework: FastAPI (better for ML inference APIs than Flask).
* Endpoints:
  + POST /xray → Input: image → Output: JSON (diagnosis).
  + POST /ocr → Input: image → Output: JSON (prescription text).
  + POST /risk → Input: JSON (patient data) → Output: JSON (risk score).
* Security: simple JWT-based auth (Phase-1 minimum).
* Data storage: SQLite (Phase-1 lightweight).

4. Models (AI Core)

* X-ray Analyzer:
  + Use PyTorch ResNet-50 pretrained on ImageNet → fine-tune with NIH Chest X-ray dataset.
  + Save weights (xray\_model.pth).
  + Inference script in /models/xray\_inference.py.
* Prescription OCR:
  + Tesseract OCR with Urdu + English language packs.
  + Wrapper script in /models/ocr\_inference.py.
* Risk Scoring:
  + Logistic Regression (Scikit-learn).
  + Train on survey data (PDHS/WHO).
  + Save as risk\_model.pkl.

5. Documentation Integration

* Store docs in /docs/ folder.
* Add mkdocs or docusaurus to generate a simple static docs site (optional).
* Include: SRS, SDD, Data Docs, Model Cards.

6. Dockerization

* Each component in its own Dockerfile:

Frontend Dockerfile

* Base: node:18-alpine
* Install Next.js, build & serve.

Backend Dockerfile

* Base: python:3.10-slim
* Install FastAPI + uvicorn + dependencies.

Models Dockerfile

* Base: pytorch/pytorch:cpu (open-source, CPU friendly).
* Install OCR + scikit-learn.

Docs Dockerfile

* Base: python:3.10-slim
* Install mkdocs.
* Orchestrate with docker-compose.yml:

version: "3.9"

services:

frontend:

build: ./frontend

ports:

- "3000:3000"

backend:

build: ./backend

ports:

- "8000:8000"

depends\_on:

- models

models:

build: ./models

docs:

build: ./docs

ports:

- "8080:8080"

7. Suggested Implementation Order

1️⃣ Backend skeleton + models first

* Build inference APIs (/xray, /ocr, /risk).
* Test with Postman / curl.

2️⃣ Frontend next

* Connect upload forms & results display to backend.
* Test full workflow (X-ray → API → result in UI).

3️⃣ Docs integration

* Polish SRS, SDD, Data Docs → serve via mkdocs.

4️⃣ Dockerize

* Add Dockerfiles + docker-compose.
* Run entire stack with one command.

8. Timeline for Implementation (Phase-1, 14 days)

* Day 1–2: Setup repos, Docker baseline, dependencies.
* Day 3–6: Implement X-ray model + API.
* Day 7–8: Add OCR prescription digitization API.
* Day 9–10: Risk scoring model + API.
* Day 11–12: Frontend Next.js integration.
* Day 13: End-to-end testing, Docker orchestration.
* Day 14: Final polish + prepare demo video.

⚡ Recommendation: Start with backend + models, because once APIs work, frontend is just an interface. Docs are already strong → just package them neatly.