**Backend & Models Development Plan (SehatAI)**

**Project:** SehatAI – AI-powered Healthcare Assistant  
**Module:** Backend (Flask REST API) & AI Models  
**Version:** 1.0  
**Date:** September 2025

**1. Introduction**

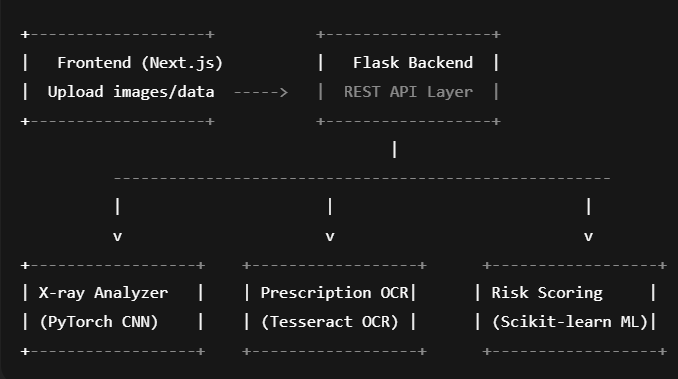
The backend and AI models are the **core intelligence** of SehatAI.

* The **Flask backend** provides REST APIs for communication between frontend and models.
* The **AI models** perform inference tasks:
  1. X-ray disease detection (TB, Pneumonia).
  2. Prescription OCR (Urdu/English).
  3. Risk scoring (diabetes/heart disease).

**2. Objectives**

* Provide a **secure, lightweight, CPU-friendly API** layer.
* Serve model predictions via REST endpoints.
* Ensure **fast response times** (<10 seconds per request).
* Use **only open-source datasets & libraries** (as per Techathon guidelines).

**3. System Architecture (Backend + Models)**



**4. Backend (Flask)**

**4.1 Framework Choice**

* **Best Choice:** Flask (lightweight, simple, Python-based).
* **Alternative:** FastAPI (faster, built-in OpenAPI docs).

**4.2 API Endpoints**

| **Endpoint** | **Method** | **Input** | **Output** |
| --- | --- | --- | --- |
| /api/xray | POST | Image (JPG/PNG, base64/form) | JSON { diagnosis } |
| /api/ocr | POST | Image (JPG/PNG) | JSON { extracted\_text } |
| /api/risk | POST | JSON (patient demographics) | JSON { risk\_score } |

**4.3 API Security**

* Simple **JWT Authentication** for Phase-1.
* HTTPS recommended.

**4.4 Response Format**

{

"status": "success",

"data": {

"diagnosis": "TB detected",

"confidence": 0.87

}

}

**5. Models (AI Core)**

**5.1 X-ray Analyzer (CNN Model)**

* **Architecture:** ResNet-50 (PyTorch).
* **Task:** Classify chest X-rays into TB, Pneumonia, Normal.
* **Dataset:** NIH Chest X-rays, Kaggle TB dataset.
* **Preprocessing:** Resize (224x224), normalize, augment.
* **Output:** { "diagnosis": "Pneumonia", "confidence": 0.92 }

**Training Setup:**

* Loss Function: CrossEntropyLoss
* Optimizer: Adam (lr=0.001)
* Epochs: 10–15 (CPU feasible with subset)

**5.2 Prescription OCR**

* **Library:** Tesseract OCR (with Urdu + English language packs).
* **Task:** Extract text from handwritten prescriptions.
* **Dataset:** UCOM Urdu handwritten dataset + synthetic prescriptions.
* **Preprocessing:** Grayscale, binarization, noise removal.
* **Output:**

{

"text": "Paracetamol 500mg, 1 tablet daily"

}

**5.3 Risk Scoring Model**

* **Algorithm:** Logistic Regression / Random Forest (Scikit-learn).
* **Task:** Predict risk for diabetes/heart disease.
* **Dataset:** Pakistan Demographic and Health Survey (PDHS).
* **Input Features:** Age, Gender, Weight, BP, lifestyle factors.
* **Output:**

{

"risk\_score": "High",

"probability": 0.81

}

**Metrics:** AUC ≥ 0.8, Precision/Recall.

**6. Integration Workflow**

1. **Frontend → Flask API:** User uploads X-ray/prescription or enters data.
2. **Flask → Models:** API forwards input to relevant model.
3. **Model → Flask:** Model inference result returned.
4. **Flask → Frontend:** API sends JSON back to frontend.
5. **Frontend:** Displays results in dashboard.

**7. Tech Stack**

* **Best Choice:**
  + **Backend:** Flask, Gunicorn, Uvicorn.
  + **Models:** PyTorch (CNN), Tesseract (OCR), Scikit-learn (Risk).
  + **Database:** SQLite (Phase-1).
* **Alternative:**
  + **Backend:** FastAPI.
  + **Models:** TensorFlow/Keras, EasyOCR, LightGBM.
  + **Database:** PostgreSQL.

**8. Deployment & Dockerization**

**Dockerfile (Backend)**

FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["gunicorn", "-b", "0.0.0.0:8000", "app:app"]

**Dockerfile (Models)**

FROM pytorch/pytorch:1.13.0-cpu

WORKDIR /models

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["python", "serve\_models.py"]

**docker-compose.yml**

version: "3.9"

services:

backend:

build: ./backend

ports:

- "8000:8000"

depends\_on:

- models

models:

build: ./models

**9. Timeline (Backend + Models)**

* **Day 1–2:** Setup Flask skeleton + endpoints.
* **Day 3–5:** Implement X-ray CNN model (training + inference).
* **Day 6–7:** Add OCR module (Tesseract).
* **Day 8–9:** Add risk scoring model.
* **Day 10:** Connect models to Flask endpoints.
* **Day 11–12:** Test with mock frontend requests.
* **Day 13:** Dockerize backend + models.
* **Day 14:** Final testing + documentation.

**10. Risks & Limitations**

* **X-ray dataset size** may be too large for CPU training → use subset or pretrained weights.
* **Messy Urdu handwriting** may reduce OCR accuracy.
* **Risk scoring** limited by dataset features (may not cover all risk factors).

**11. Future Enhancements**

* Scale backend with **FastAPI + async** for real-time.
* GPU training in **Phase-2**.
* Add support for more diseases (multi-label classification).
* Cloud-based database (PostgreSQL/AWS RDS).

✅ This document gives you a **step-by-step roadmap** for backend + models development, Dockerization, and integration with frontend.