Problem Statement 6: Product Category Creation for Healthcare Kiosks in India

Submitted by: Deepraj Bhattacharjee

Current State Challenge:

Intel Architecture-based Healthcare Kiosk exists but suffers from poor market adoption due to limited functionality that addresses only narrow market segments while ignoring broader healthcare delivery needs in India.

Core Problem Definition:

How can we transform the existing Healthcare Kiosk into a comprehensive digital health platform that integrates with India's National Digital Health Mission (NDHM) ecosystem to bridge the urban-rural healthcare accessibility gap while leveraging Intel's secure technology stack and AI capabilities?

Specific Problem Areas:

1. Limited Market Penetration

- Current product solves specific use cases but fails to address comprehensive healthcare delivery needs
- Poor uptake despite existing Intel Architecture foundation
- Lack of integration with India's digital health infrastructure

2. Healthcare Accessibility Gap

- **Urban-rural healthcare divide** remains a critical barrier to universal health coverage
- Limited specialist access in remote areas
- Fragmented healthcare data and lack of interoperability

3. Technology Integration Challenges

- Underutilized potential of Intel's secure technology assets (OneAPI, OpenVINO, Open Edge Platform)
- Missing integration with India Stack digital public infrastructure
- Absence of AI/ML capabilities for diagnostic support and federated learning

4. Regulatory and Implementation Barriers

- Complex regulatory landscape for medical devices and AI-powered diagnostics
- Need for standardized governance framework
- Certification and compliance requirements for healthcare applications

Project Structure Overview:

Folder/File	Purpose
`app.py`	Main launcher for the kiosk allowing users to navigate
`pages/`	Contains all the features developed in a separate python script
`utils/`	Helper classes for validation, encryption, and API integration
`models/`	Serialized ML artefacts with semantic version tags
`data/`	Raw, processed, and synthetic datasets
`notebooks/`	Exploratory data analysis (EDA) and feature engineering and training my CNN based model for breast cancer detection

3. System Architecture:

Core Architecture Components

1. Main Application Layer

- **Entry Point**: app.py serves as the main launcher and navigation hub
- **Framework**: Built on Streamlit for web-based interface
- **Branding**: Hospital unified interface
- **Compliance**: ABDM (Ayushman Bharat Digital Mission) compliant and HIPAA secure

2. Specialized Healthcare Modules

Diagnostic & AI Analysis

- **Breast Cancer Diagnostic Dashboard** (Breast_Cancer.py)
    - AI-powered medical imaging analysis
    - CNN-based prediction model with Open-VINO optimization
    - GradCAM visualization for explainable AI
    - Dual model support: TensorFlow (.h5) and Open-VINO (.xml)
    - Real-time confidence scoring and risk categorization

Patient Data Management

- **EHR Viewer** (EHR_Viewer.py)
    - Electronic Health Records interface
    - Mock data integration for testing
    - Patient record visualization
- **Health Assessment** (Health_Assessment.py)
    - Digital health questionnaires
    - Patient data collection and storage
    - Profile management system

Clinical Tools

- **Health Risk Calculator** (Health_Risk_Calculator.py)
    - Risk assessment algorithms
    - Lifestyle coaching recommendations
    - Multi-factor health analysis (age, BMI, blood pressure, smoking, diabetes)
- **Medication Tracker** (Medication_Tracker.py)
    - ABHA ID-based medication management
    - Active and completed medication monitoring

    - CSV-based data storage

- **Vaccination Tracking** (Vaccination_Tracking.py)
    - ABHA-integrated vaccination records
    - Dose scheduling and status tracking
    - Patient-specific vaccination history

Telemedicine Integration

- **Teleconsultation** (Teleconsultation.py)
    - Doctor booking and scheduling system
    - Real-time consultation interface
    - Emergency contact integration
    - Fee management and payment processing

Technical Architecture Stack

Frontend Layer

- **Framework**: Streamlit with custom CSS styling
- **UI Components**: Multi-column layouts, interactive widgets
- **Responsive Design**: Wide layout configuration for hospital kiosks
- **Branding**: Consistent Hospital theming across modules

AI/ML Layer

- **Deep Learning**: CNN-based models for medical imaging
- **Model Optimization**: Open-VINO for inference acceleration
- **Explainable AI**: GradCAM for model interpretability
- **Dual Runtime**: TensorFlow and Open-VINO model support

Data Layer

- **Patient Data**: CSV-based storage for medication and vaccination records
- **ABHA Integration**: Ayushman Bharat Health Account ID system
- **Mock Data**: Testing utilities with simulated patient profiles
- **Image Processing**: PIL and medical imaging utilities

Utility Layer

- **Specialized Utils**: Module-specific utility functions
    - **breast_utils**: Image preprocessing, model loading, report generation
    - **ehr_mock_data:** EHR simulation and testing
    - **data_extract:** Patient data management
    - **vaccine_utils:** Vaccination data processing

Application Features Overview

app.py - Main Application Launcher:

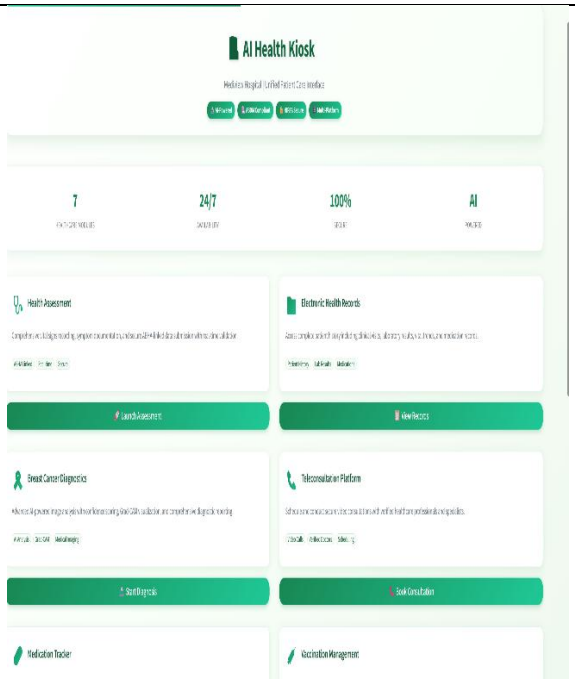
Description: The central entry point and navigation hub for the AI Health Kiosk system. Provides a unified interface for accessing all healthcare modules.

Key features include:

- ABHA-based patient identification
- Digital health screening (vitals, symptoms)
- EHR record viewer
- Teleconsultation & referral integration (future-ready)
- Clean, responsive UI for kiosk deployment

app.py

```
import streamlit as st

# Page setup
st.set_page_config(
    page_title="AI Health Kiosk Launcher",
    layout="wide",
    page_icon="🏥",
    initial_sidebar_state="collapsed"
)

# Page header
st.markdown('''
<div class="main-header">
    <h1 class="header-title">🏥 AI Health Kiosk</h1>
    <p class="header-subtitle">Mediview Hospital | Unified Patient Care
Interface</p>
    <div class="status-badges">
        <span class="status-badge">🔬 AI-Powered</span>
        <span class="status-badge">🏥 ABDM Compliant</span>
        <span class="status-badge">🔒 HIPAA Secure</span>
        <span class="status-badge">📱 Multi-Platform</span>
    </div>
</div>
''', unsafe_allow_html=True)
```

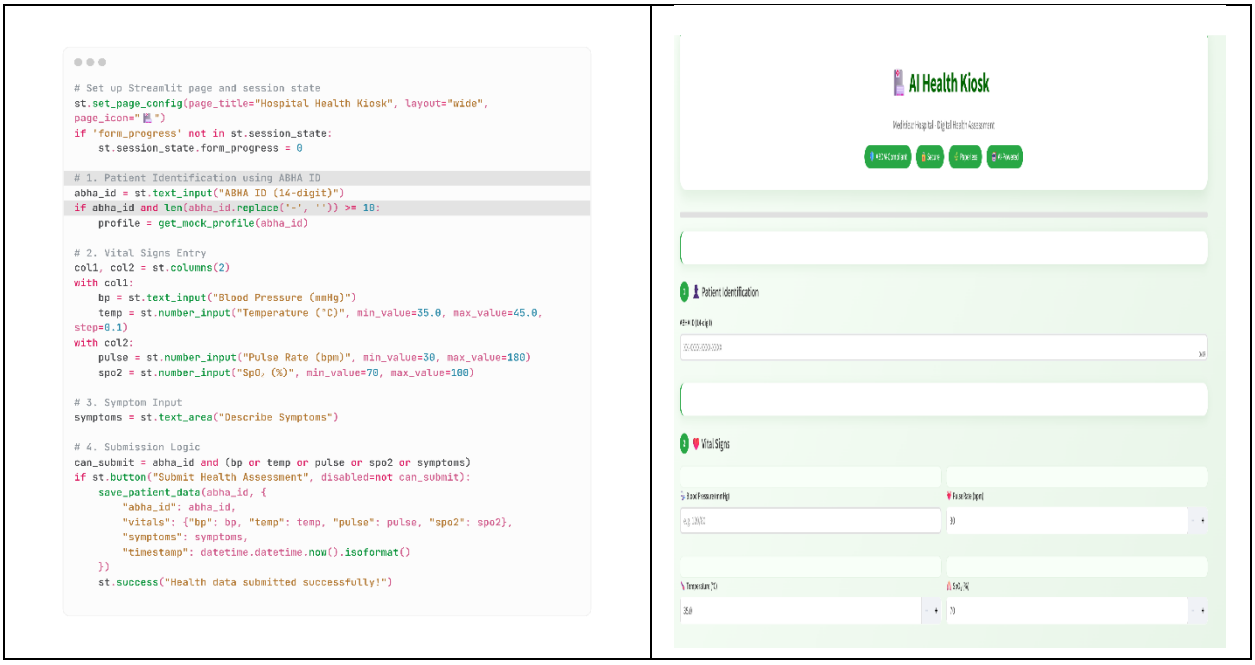


Health_Assessment.py - Digital Health Questionnaire:

Description: The **Health Assessment** page is the primary interface for capturing a patient's basic health status. It is optimized for front-desk use or kiosk deployment in clinics and hospitals.

Key sections include:

- **Patient Identification**: Uses ABHA ID to fetch pre-verified demographic data.
- **Vital Signs Input**: Collects BP, temperature, pulse, and SpO_2 levels.
- **Symptom Logging**: Patients can describe symptoms in natural language.
- **Submission Workflow:** Progress bar, validation, and secure data submission via ABDM-compatible format.
- **Next Actions**: Suggests follow-up via EHR viewing or teleconsultation.



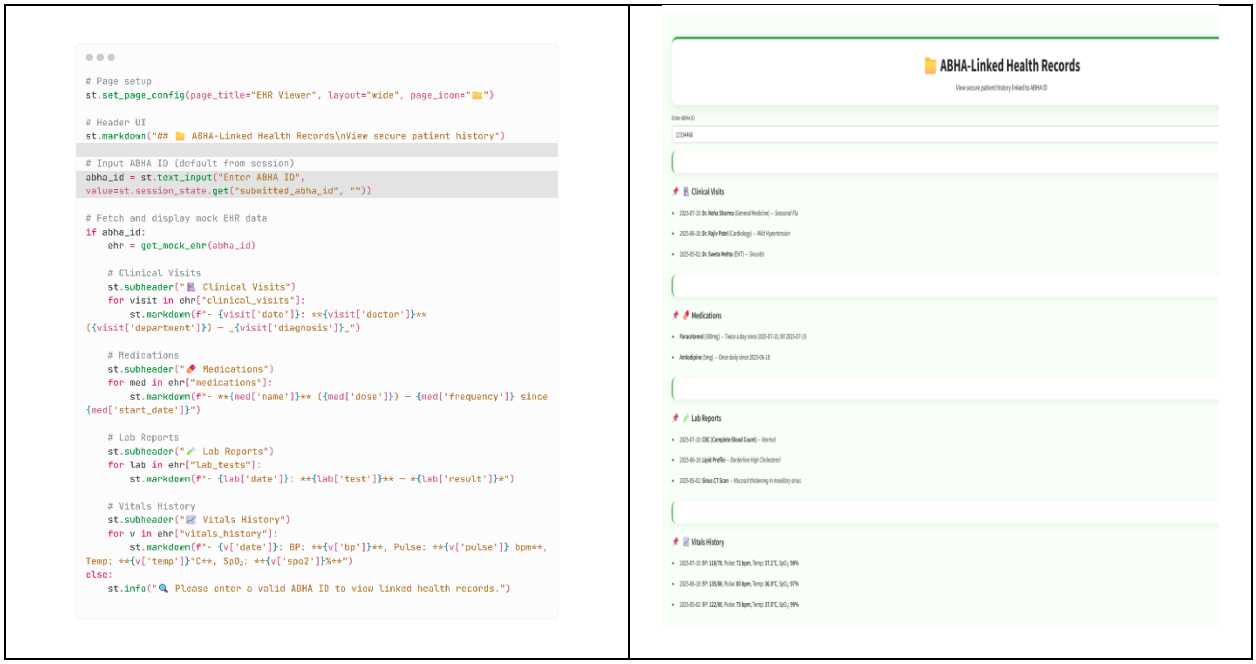
```python
# Set up Streamlit page and session state
st.set_page_config(page_title="Hospital Health Kiosk", layout="wide",
page_icon="🏥")
if 'form_progress' not in st.session_state:
    st.session_state.form_progress = 0

# 1. Patient Identification using ABHA ID
abha_id = st.text_input("ABHA ID (14-digit)")
if abha_id and len(abha_id.replace('-', '')) >= 10:
    profile = get_mock_profile(abha_id)

# 2. Vital Signs Entry
col1, col2 = st.columns(2)
with col1:
    bp = st.text_input("Blood Pressure (mmHg)")
    temp = st.number_input("Temperature (°C)", min_value=35.0, max_value=45.0,
step=0.1)
with col2:
    pulse = st.number_input("Pulse Rate (bpm)", min_value=30, max_value=180)
    spo2 = st.number_input("SpO₂ (%)", min_value=70, max_value=100)

# 3. Symptom Input
symptoms = st.text_area("Describe Symptoms")

# 4. Submission Logic
can_submit = abha_id and (bp or temp or pulse or spo2 or symptoms)
if st.button("Submit Health Assessment", disabled=not can_submit):
    save_patient_data(abha_id, {
        "abha_id": abha_id,
        "vitals": {"bp": bp, "temp": temp, "pulse": pulse, "spo2": spo2},
        "symptoms": symptoms,
        "timestamp": datetime.datetime.now().isoformat()
    })
    st.success("Health data submitted successfully!")
```

EHR_Viewer.py - Electronic Health Records Interface:

Description: Patient electronic health record viewer with mock data integration for testing and development purposes.

Displayed data includes:

- **Clinical Visit History:** Past consultations, diagnoses, and doctors seen.
- **Medication Records:** Prescriptions, dosage, frequency, and timelines.
- **Lab Reports:** Test results with dates and key indicators.
- **Vitals History:** Historical vital sign readings for trend analysis.



```
# Page setup
st.set_page_config(page_title="EHR Viewer", layout="wide", page_icon="📁")

# Header UI
st.markdown("## 📁 ABHA-Linked Health Records\nView secure patient history")

# Input ABHA ID (default from session)
abha_id = st.text_input("Enter ABHA ID",
value=st.session_state.get("submitted_abha_id", ""))

# Fetch and display mock EHR data
if abha_id:
    ehr = get_mock_ehr(abha_id)

    # Clinical Visits
    st.subheader("📋 Clinical Visits")
    for visit in ehr["clinical_visits"]:
        st.markdown(f"- {visit['date']}: **{visit['doctor']}**
({visit['department']}) — _{visit['diagnosis']}_")

    # Medications
    st.subheader("💊 Medications")
    for med in ehr["medications"]:
        st.markdown(f"- **{med['name']}** ({med['dose']}) — {med['frequency']} since
{med['start_date']}")

    # Lab Reports
    st.subheader("🧪 Lab Reports")
    for lab in ehr["lab_tests"]:
        st.markdown(f"- {lab['date']}: **{lab['test']}** — *{lab['result']}*")

    # Vitals History
    st.subheader("📈 Vitals History")
    for v in ehr["vitals_history"]:
        st.markdown(f"- {v['date']}: BP: **{v['bp']}**, Pulse: **{v['pulse']} bpm**,
Temp: **{v['temp']}°C**, SpO₂: **{v['spo2']}%**")
else:
    st.info("🔍 Please enter a valid ABHA ID to view linked health records.")
```

Teleconsultation Page (Book Virtual Consult):

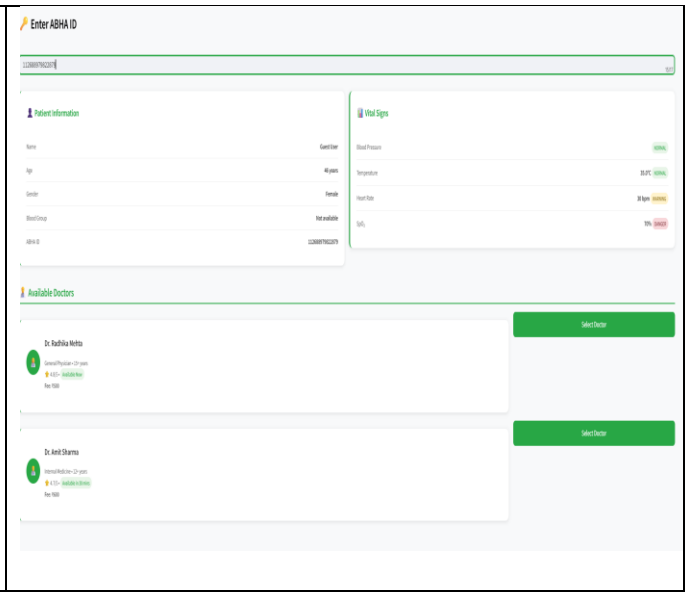
Page Purpose

- Enables booking of remote video consultations with verified doctors.
- Analyses patient's health condition using vitals + symptoms to determine urgency level.
- Provides a meeting link and consultation summary after booking.

Key Functionalities

- Patient profile + symptom display
- Dynamic doctor list with real-time status
- Booking confirmation with secure meeting ID
- Urgency triaging based on abnormal vitals
- Post-booking confirmation + reset feature

```
# Doctor selection interface
for i, doctor in enumerate(doctors):
    st.markdown(f"""
    <div class="doctor-card">
        <div class="doctor-info">
            <div class="doctor-avatar">{doctor['avatar']}</div>
            <div class="doctor-details">
                <h4>{doctor['name']}</h4>
                <p>{doctor['specialty']} • {doctor['experience']}</p>
                <p>⭐ {doctor['rating']} • <span class="tag normal">
{doctor['status']}</span></p>
                <p><strong>Fee: ₹{doctor['fee']}</strong></p>
            </div>
        </div>
    </div>
    """, unsafe_allow_html=True)
    if st.button(f"Select Doctor", key=f"select_{i}"):
        st.session_state.selected_doctor = doctor
        st.session_state.show_summary = True
        st.rerun()

# Booking confirmation + summary
if st.session_state.booking_confirmed:
    consultation_id = f"TC{datetime.datetime.now().strftime('%Y%m%d%H%M')}
{abha_id[-4:]}"
    meeting_link = f"https://mediview.consultation.secure/{consultation_id}"
    st.markdown(f"""
    <div class="success-container">
        <h3>✅ Consultation Booked Successfully!</h3>
        <div class="meeting-link"><strong>Meeting Link:</strong><br>
        <a href="{meeting_link}" target="_blank">{meeting_link}</a></div>
    </div>
    """, unsafe_allow_html=True)
```

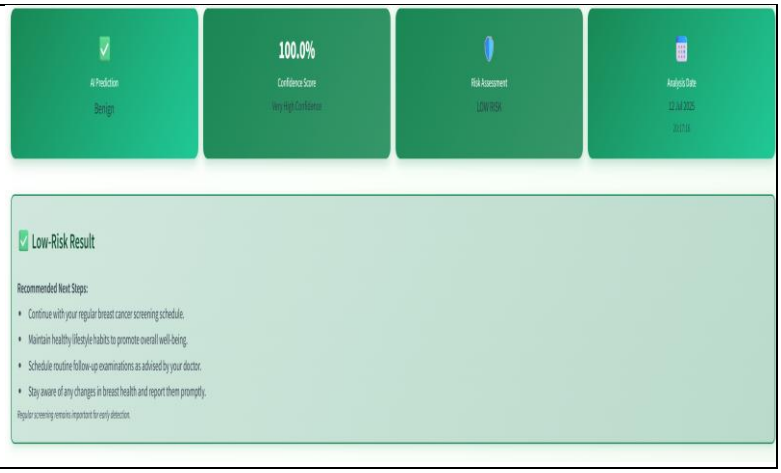


Breast Cancer AI:

Features Summary:

- Deep Learning-Powered: Uses a CNN model (OpenVINO + Keras) to predict Benign or Malignant outcomes.
- Grad-CAM Heatmaps: Visualizes AI attention areas overlaid on the uploaded scan.
- Upload & Validate: Accepts common scan image formats and validates input before processing.
- Real-Time Metrics: Displays prediction result, confidence level, and risk assessment.
- Detailed Reports: Auto-generates a downloadable diagnostic report with metadata and model info.
- ABHA Integration: Associates diagnostic records with a patient's ABHA ID.
- Patient History: Tracks and plots past scan results and confidence trends for each patient.
- Persistent Storage: Saves scan results in CSV for future references and longitudinal analysis.
- Alerts & Recommendations: Provides medical follow-up recommendations based on risk.
- Rich UI/UX: Includes a fully customized, responsive, and styled dashboard for better patient and doctor interaction

```
uploaded = st.file_uploader("Upload scan image", type=["jpg", "jpeg", "png", "bmp",
"tiff"])
abha_id = st.text_input("Enter ABHA ID", placeholder="XX-XXXX-XXXX-XXXX")

if uploaded and abha_id:
    if not validate_image(uploaded):
        st.error("Invalid image. Please try again.")
        st.stop()

    with st.spinner("Processing..."):
        img_tensor = preprocess_image(uploaded)
        prediction, confidence = predict_breast_cancer(img_tensor, compiled_model)
        heatmap = generate_gradcam(model=tf_model, img_tensor=img_tensor)

    st.image(uploaded, caption="Original Scan", use_column_width=True)
    if heatmap:
        st.image(heatmap, caption="AI Heatmap", use_column_width=True)

    st.metric("Prediction", prediction)
    st.metric("Confidence", f"{confidence:.2%}")

    report_data = format_medical_report(abha_id, prediction, confidence,
str(datetime.datetime.now()), get_image_metadata(uploaded))
    st.download_button("Download Report", "\n".join(f"{k}: {v}" for k, v in
report_data.items()), file_name="Report.txt")
```



Additional Features:

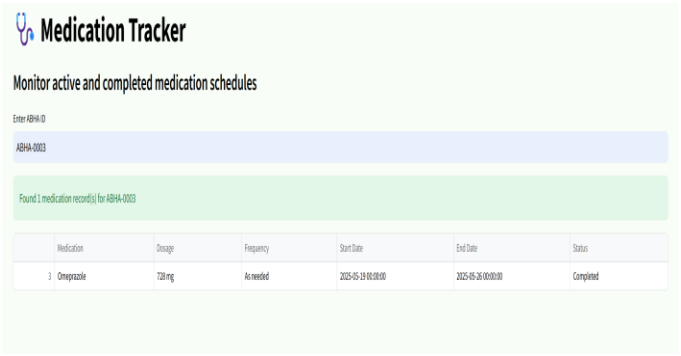
Medication Tracker:

This Stream lit dashboard allows users to enter an ABHA ID and view their medication history. It displays ongoing and completed medications with start/end dates, dosage, and frequency.

How it helps:

- Tracks ongoing and completed medications by ABHA ID.
- Helps patients follow prescriptions accurately.
- Assists doctors with quick medication history review

```
import streamlit as st
import pandas as pd

st.set_page_config(page_title="Medication Management", layout="wide")

st.title("💊 Medication Tracker")
st.subheader("Monitor active and completed medication schedules")

abha_id = st.text_input("Enter ABHA ID")

if abha_id:
    df = pd.read_csv("data/medication_data.csv", parse_dates=["Start Date", "End Date"])
    patient_data = df[df["ABHA_ID"] == abha_id]

    if patient_data.empty:
        st.warning("No records found.")
    else:
        today = pd.to_datetime("today").normalize()
        patient_data["Status"] = patient_data["End Date"].apply(lambda x: "Ongoing" if x >= today else "Completed")
        st.dataframe(patient_data[["Medication", "Dosage", "Frequency", "Start Date", "End Date", "Status"]])
```



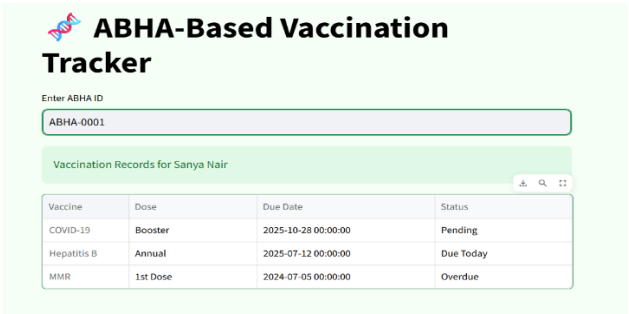
Abha Based Vaccination Tracker:

This app lets users enter an ABHA ID to view personalized vaccination records—showing dose details, due dates, and status—via a clean, user-friendly dashboard.

How It Helps?

- Tracks pending and completed vaccine doses by ABHA ID
- Helps users and doctors stay up-to-date with vaccinations
- Provides clear, formatted vaccination history instantly

```
abha_id = st.text_input("Enter ABHA ID")

if abha_id:
    df = load_vaccine_data()
    patient_data = get_patient_vaccine_status(abha_id, df)

    if not patient_data.empty:
        st.dataframe(patient_data[["Vaccine", "Dose", "Due Date", "Status"]])
    else:
        st.warning("No record found.")
```



Optional Addons:

Multilingual Voice Assistant-cum Chatbot:

The multilingual voice assistant backend is a modular speech-translation-synthesis pipeline designed to handle real-time voice input, automatic speech recognition (ASR), machine translation, and speech synthesis (TTS):

- It makes use of Google speech Api for speech recognition.
- Translates the transcribed text using Hugging Face's **Helsinki-NLP models**, enabling seamless bidirectional translation (English ↔ Hindi).
- Converts translated text back to speech using **Google Text-to-Speech (gTTS)** and plays it via local media player.

Example Use case:

Patient: "Main doctor se consult karna chahta hoon"
Assistant (translates): "I want to consult a doctor"
System: Navigates to Teleconsultation
Assistant: "Available slots are 11:00 AM, 2:00 PM. Choose one."
Patient: "11 baje"
System: Appointment booked ✅

Fronend + Backend:

Voice Assistant

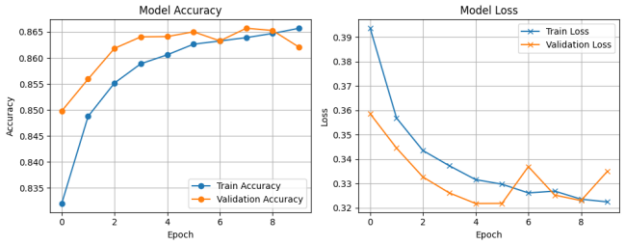
```python
import streamlit as st
from utils.voice_utils import transcribe_audio, translate_text, speak_text

st.set_page_config(page_title="Multilingual Assistant", layout="wide",
page_icon="🎙️")

st.markdown("""
  Custom CSS styles
""", unsafe_allow_html=True)

st.markdown("## 🎙️ Multilingual Voice Assistant")
st.markdown("Interact with the kiosk using your voice. Supports **English 🔤
Hindi** conversion.")

col1, col2 = st.columns([1, 1])
with col1:
    lang_direction = st.selectbox("Choose Translation Direction", ["English to
Hindi", "Hindi to English"])

with col2:
    if st.button("🎤 Speak Now"):
        input_text = transcribe_audio()
        st.markdown(f"**🎧 You said:** `{input_text}`")

        direction = "en2hi" if lang_direction == "English to Hindi" else "hi2en"
        translated = translate_text(input_text, direction=direction)

        st.markdown(f"**🌐 Translation:** `{translated}`")

        lang_code = "hi" if direction == "en2hi" else "en"
        speak_text(translated, lang=lang_code)
```

Voice Assistant

```python
import speech_recognition as sr
from gtts import gTTS
import os
from transformers import pipeline, AutoModelForSeq2SeqLM, AutoTokenizer

def transcribe_audio():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        audio = recognizer.listen(source, phrase_time_limit=5)
    try:
        return recognizer.recognize_google(audio, language="en-IN")
    except:
        return "Sorry, I couldn't hear that."

def translate_text(text, direction="en2hi"):
    if direction == "en2hi":
        model_name = "Helsinki-NLP/opus-mt-en-hi"
    else:
        model_name = "Helsinki-NLP/opus-mt-hi-en"
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
    translator = pipeline("translation", model=model, tokenizer=tokenizer)
    return translator(text)[0]['translation_text']

def speak_text(text, lang="en"):
    tts = gTTS(text=text, lang=lang)
    tts.save("temp.mp3")
    os.system("start temp.mp3")
```

Introducing - Breast Cancer. Ai:

The Convolutional Neural Network (CNN) model integrated into the kiosk system serves as a powerful tool for medical image classifications - specifically, in identifying malignant and benign breast cancer from histopathology images. This deep learning model is trained on labeled medical datasets and demonstrates a high accuracy rate in classifying images, supporting fast, reliable, and automated diagnostics.

Datasets Used	Model Architecture	Performance Metrics
The model was trained using the **Breast Histopathology Images** dataset, which contains high-resolution scanned tissue samples labeled as either **benign** or **malignant**. The dataset includes thousands of image patches, each measuring 50x50 pixels, extracted from whole-slide images. Before training, the dataset was cleaned, labeled, and augmented to ensure class balance and improve generalization.	**Input Layer**: Accepts 64 x 64 pixels RGB images **Convolutional Layers**: Three blocks with increasing filters (32, 64, 128), using 3x3 kernels and ReLU activation. **Pooling Layers**: Max pooling layers (2x2) after each convolutional block for spatial reduction **Dropout Layers**: Dropout (0.25–0.5) used to mitigate overfitting. **Fully Connected (Dense) Layers**: Dense (128) + ReLU Final Dense(1) + Sigmoid for binary classification	**Training Accuracy**: Reached **86.66%** by epoch 10. **Validation Accuracy**: Peaked at **86.57%** **Training Loss**: Reduced to **0.3221.** **Validation Loss**: Stabilized around **0.3218**, indicating good generalization. **Model Convergence**: The training and validation curves showed a steady decrease in loss without significant overfitting, which reflects a well-regularized model. 

Key Benefits:

1. **Enhanced Diagnostic Support**:
   By embedding this CNN model in the kiosk, patients can receive rapid preliminary screening results directly from the device. This makes diagnostic services accessible in remote or underserved areas where medical specialists may not be immediately available.

2. **Differentiator in the Market**:
   Most self-service medical kiosks today focus on basic biometric readings or symptom checkers. Our kiosk's integration of a deep learning-based cancer detection model sets it apart by offering real clinical-grade insights. This high-value capability can act as a unique selling point for healthcare providers or NGOs looking to deploy advanced technology in frontline health services.

3. **Scalability and Adaptability**:
   The CNN architecture used is modular and lightweight enough to be optimized for edge devices or cloud integration. This allows the kiosk to be easily scaled across different locations. Additionally, the model can be retrained on other medical datasets, allowing the same system to eventually support multiple types of disease detection—from skin conditions to other cancers—making it a flexible investment.

4. **Continuous Learning and Improvement**:
   As more images are collected through kiosk deployments, the model can be continually refined. This feedback loop ensures the system improves in accuracy and generalization over time, enhancing its clinical usefulness.

Go To Market (GTM) -Blueprint

Target Segments:

Segment	Needs Addressed	Opportunity
Government Health Initiatives (e.g., Ayushman Bharat)	Mass screening, AI-based diagnosis, rural access	Large-scale deployment in PHCs, HWCs
Private Hospitals & Diagnostics Chains	Smart OPD intake, report digitization, early screening	Clinical efficiency & differentiation
NGOs & Public Health Missions	Affordable diagnosis in underserved areas	High social impact + donor funding
Corporate CSR & Insurance Partners	Rural health camps, wellness kiosks	Risk reduction, brand value

Channel Strategy:

- Direct Sales to Hospitals/Govt via Tenders
- Partnership with Telemedicine Aggregators (eSanjeevani, Practo)
- OEM Integrations with Kiosk Manufacturers
- System Integrators (SIs): For customized deployments in NDHM-linked facilities

Pricing Strategy:

- **Hardware + AI Bundle**: One-time + subscription for AI updates
- **Leasing Model**: Monthly fee for NGOs/PHCs
- **Freemium AI Features**: Basic vitals free, advanced diagnostics via cloud = pay-per-use

Patient Journey Walkthrough: Rural Screening Scenario:

Name: Sita Devi
Age: 43
Location: Rural Bihar
Background: Mild symptoms, long distance from diagnostic centre, no recent check-ups.

Step-by-Step Flow:

🏥 **Step 1: Arrival at HWC**

- Sita enters a government-run Health & Wellness Centre.
- The health worker guides her to the **AI Health Kiosk**.

📜 **Step 2: Identification**

- Sita shares her **ABHA ID or Aadhaar**.
- The kiosk auto-loads her demographic + past health data.

❤️ **Step 3: Vitals & Symptoms**

- The kiosk records her **BP, pulse, oxygen, temp**.
- She answers questions on fatigue, pain, family history using voice assistant (Hindi).

🧬 **Step 4: Diagnostic AI**

- The worker scans her **breast biopsy slide** into the kiosk.
- **CNN model** gives a **malignancy prediction** with **confidence score** and **heatmap**.

📄 **Step 5: Report & Referral**

- AI-generated report is **auto-linked to ABHA**.
- Kiosk suggests a **tele consult** with an oncologist with the built in Teleconsultation service.
- Sita is connected via **e-Sanjeevani video call** on the spot.

📦 **Step 6: Follow-Up**

- Prescription and referral saved to her **NDHM health locker**.
- Health worker gets alerts for **next follow-up**.

Impact:

Metric	Before	After Kiosk
Diagnosis delay	Weeks	Same day
Access to specialist	Rare	Immediate (tele consult)
Record keeping	Manual	ABHA-linked digital

Conclusion:

The AI-Powered Healthcare Kiosk combines deep learning diagnostics, ABHA integration, and teleconsultation into a single, scalable platform—designed to bring accessible, affordable, and intelligent care to every corner of India.

Built with modular components and a robust Intel architecture, the kiosk addresses critical healthcare gaps—starting with breast cancer detection and extending toward broader diagnostic support. With real-time AI insights and multilingual interaction, it empowers both patients and healthcare providers.

Ready for deployment, this solution aligns with NDHM goals and offers a strong opportunity for pilot rollouts, scale-up through public-private partnerships, and further AI model expansion.

Smart. Inclusive. Scalable. This is India's next-gen digital health frontline.