**Medical Assistant Project**

**1. FastAPI Backend (server.py)**

from fastapi import FastAPI, HTTPException

from fastapi.middleware.cors import CORSMiddleware

from pydantic import BaseModel

from typing import List

from medical\_core import MedicalAssistant

app = FastAPI()

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"],

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

medical\_assistant = MedicalAssistant()

class SymptomsRequest(BaseModel):

symptoms: List[str]

@app.post("/predict")

async def predict\_disease(request: SymptomsRequest):

try:

prediction = medical\_assistant.analyze\_symptoms(request.symptoms)

return prediction

except Exception as e:

raise HTTPException(status\_code=500, detail=str(e))

@app.get("/disease-info/{disease}")

async def get\_disease\_details(disease: str):

try:

disease\_info = medical\_assistant.get\_disease\_details(disease)

return disease\_info

except Exception as e:

raise HTTPException(status\_code=500, detail=str(e))

**2. Medical Core Logic (medical\_core.py)**

import re

import json

import random

import hashlib

from typing import List, Dict, Any

from transformers import AutoModelForCausalLM, AutoTokenizerclass MedicalAssistant:

def \_\_init\_\_(self):

self.medical\_knowledge = {

"Influenza": {

"symptoms": ["Fever", "Body aches", "Fatigue", "Respiratory issues", "Headache"],

"description": "Contagious respiratory illness caused by influenza viruses."

},

"Respiratory Infection": {

"symptoms": ["Cough", "Congestion", "Sore throat", "Difficulty breathing", "Fever"],

"description": "Infection affecting respiratory system with various potential causes."

},

"Metabolic Disorder": {

"symptoms": ["Fatigue", "Weight changes", "Increased thirst", "Frequent urination", "Mood swings"],

"description": "Condition affecting body's metabolic processes and energy conversion."

}

}

self.symptom\_pool = [

"Fever", "Headache", "Fatigue", "Body ache", "Cough",

"Shortness of breath", "Nausea", "Dizziness", "Chills",

"Muscle weakness", "Joint pain", "Sweating"

]

self.model = None

self.tokenizer = None

self.\_initialize\_model()

def \_initialize\_model(self):

try:

model\_name = "google/flan-t5-small"

self.tokenizer = AutoTokenizer.from\_pretrained(model\_name)

self.model = AutoModelForCausalLM.from\_pretrained(model\_name)

except Exception as e:

print(f"Model initialization error: {e}")

def analyze\_symptoms(self, symptoms: List[str]) -> Dict[str, Any]:

matching\_diseases = self.\_find\_matching\_diseases(symptoms)

if not matching\_diseases:

return {

"predictedDisease": "Unidentified Condition",

"confidence": 30,

"description": "Unable to confidently match symptoms to a known condition.",

"recommendations": "Consult a healthcare professional for accurate diagnosis."

}

top\_match = max(matching\_diseases, key=lambda x: x['score'])

return {

"predictedDisease": top\_match['disease'],

"confidence": int(top\_match['score'] \* 100),

"description": self.medical\_knowledge[top\_match['disease']]['description'],

"matchingSymptoms": top\_match['matching\_symptoms']

}

def \_find\_matching\_diseases(self, symptoms: List[str]) -> List[Dict]:

matches = []

for disease, details in self.medical\_knowledge.items():

matching\_symptoms = [

symptom for symptom in symptoms

if any(ref.lower() in symptom.lower() for ref in details['symptoms'])

]

if matching\_symptoms:

match\_score = len(matching\_symptoms) / len(details['symptoms'])

matches.append({

'disease': disease,

'score': match\_score,

'matching\_symptoms': matching\_symptoms

})

return matches

def get\_disease\_details(self, disease\_name: str) -> Dict[str, Any]:

normalized\_name = disease\_name.lower()

for disease, details in self.medical\_knowledge.items():

if normalized\_name in disease.lower():

return {

"name": disease,

"symptoms": details['symptoms'],

"description": details['description']

}

return {

"name": disease\_name,

"symptoms": self.\_generate\_symptoms(disease\_name),

"description": "Condition details generated based on name pattern."

}

def \_generate\_symptoms(self, disease\_name: str) -> List[str]:

random.seed(hash(disease\_name))

return random.sample(self.symptom\_pool, 4)

**3. Database Schema (schemas.py)**

from pydantic import BaseModel, validator

from typing import List, Optional, Dict, Any

import datetime

class PatientBase(BaseModel):

name: str

aadhar\_number: str

date\_of\_birth: Optional[str] = None

gender: Optional[str] = None

contact\_number: Optional[str] = None

address: Optional[str] = None

@validator('aadhar\_number')

def validate\_aadhar(cls, v):

if len(v) != 12 or not v.isdigit():

raise ValueError('Aadhar number must be exactly 12 digits')

return v

class Patient(PatientBase):

id: int

created\_at: datetime.datetime

updated\_at: datetime.datetime

class DoctorBase(BaseModel):

name: str

specialty: Optional[str] = None

registration\_number: Optional[str] = None

availability: Optional[str] = "available"

rating: Optional[int] = 5

class Doctor(DoctorBase):

id: int

created\_at: datetime.datetime

updated\_at: datetime.datetime

class DiagnosisBase(BaseModel):

patient\_id: int

disease\_name: str

symptoms: List[str]

confidence: Optional[int] = 0

class Diagnosis(DiagnosisBase):

id: int

diagnosis\_date: datetime.datetime

class PrescriptionBase(BaseModel):

patient\_id: int

diagnosis\_id: int

doctor\_id: int

medicines: List[str]

dosage: List[str]

class Prescription(PrescriptionBase):

id: int

created\_at: datetime.datetime

**4. React Frontend Component (Diagnosis.tsx)**

import React, { useState, useEffect } from 'react';

import { useNavigate } from 'react-router-dom';

import { predictDiseaseFromSymptoms } from '@/services/api';

import { Button, Card, Alert } from '@/components/ui';

interface DiagnosisProps {

symptoms: string[];

}

const DiagnosisComponent: React.FC<DiagnosisProps> = ({ symptoms }) => {

const navigate = useNavigate();

const [prediction, setPrediction] = useState<any>(null);

const [error, setError] = useState<string | null>(null);

useEffect(() => {

const fetchPrediction = async () => {

try {

const result = await predictDiseaseFromSymptoms(symptoms);

setPrediction(result);

} catch (err) {

setError('Failed to analyze symptoms');

}

};

fetchPrediction();

}, [symptoms]);

const handleDoctorConsult = () => {

navigate('/doctor-consultation', {

state: {

prediction,

symptoms

}

});

};

return (

<Card>

{error && <Alert variant="destructive">{error}</Alert>}

{prediction && (

<div>

<h2>{prediction.predictedDisease}</h2>

<p>Confidence: {prediction.confidence}%</p>

<p>{prediction.description}</p>

<Button onClick={handleDoctorConsult}>

Consult Doctor

</Button>

</div>

)}

</Card>

);

};

export default DiagnosisComponent;

langchain

langchain-groq

streamlit

pypdf

faiss-cpu

openai

typing-extensions

langgraph

python-dotenv

scikit-learn

numpy

transformers

langchain\_community

fastapi

uvicorn

python-multipart

pydantic

starlette

uuid

torch

accelerate

bitsandbytes

sentencepiece

different pages of our website

import { useLocation } from "react-router-dom";

import { useEffect } from "react";

const NotFound = () => {

  const location = useLocation();

  useEffect(() => {

    console.error(

      "404 Error: User attempted to access non-existent route:",

      location.pathname

    );

  }, [location.pathname]);

  return (

    <div className="min-h-screen flex items-center justify-center bg-gray-100">

      <div className="text-center">

        <h1 className="text-4xl font-bold mb-4">404</h1>

        <p className="text-xl text-gray-600 mb-4">Oops! Page not found</p>

        <a href="/" className="text-blue-500 hover:text-blue-700 underline">

          Return to Home

        </a>

      </div>

    </div>

  );

};

export default NotFound;

import React, { useEffect } from 'react';

import { useNavigate } from 'react-router-dom';

import { useLanguage } from '@/contexts/LanguageContext';

import VoiceGuidance from '@/components/VoiceGuidance';

const LanguageSelection: React.FC = () => {

  const { setLanguage } = useLanguage();

  const navigate = useNavigate();

  const selectLanguage = (lang: 'hindi' | 'english') => {

    setLanguage(lang);

    navigate('/aadhaar-login');

  };

  return (

    <div className="kiosk-container">

      <div className="flex flex-col items-center justify-center gap-8 w-full">

        <div className="flex flex-col items-center gap-4">

          <img

            src="/india-flag.svg"

            alt="Indian Flag"

            className="w-24 h-24 rounded-full object-cover shadow-md"

            onError={(e) => {

              e.currentTarget.onerror = null;

              e.currentTarget.src = '';

            }}

          />

          <h1 className="kiosk-large-text font-hindi mb-8 text-center">

            भाषा चुनें / Choose Language

          </h1>

        </div>

        <div className="flex flex-col sm:flex-row gap-8 w-full max-w-md justify-center">

          <button

            onClick={() => selectLanguage('hindi')}

            className="kiosk-button-large kiosk-button-green"

          >

            <span className="text-3xl">🇮🇳</span>

            <span className="font-hindi">हिंदी</span>

          </button>

          <button

            onClick={() => selectLanguage('english')}

            className="kiosk-button-large kiosk-button-blue"

          >

            <span className="text-3xl">🇬🇧</span>

            <span>English</span>

          </button>

        </div>

        <VoiceGuidance

          hindiPrompt="कृपया अपनी भाषा चुनें। हिंदी के लिए हरे बटन को दबाएँ।"

          englishPrompt="Please select your language. Press the green button for Hindi."

        />

      </div>

    </div>

  );

};

export default LanguageSelection;

import React, { useState, useEffect } from 'react';

import { useNavigate, useLocation } from 'react-router-dom';

import { useLanguage } from '@/contexts/LanguageContext';

import VoiceGuidance from '@/components/VoiceGuidance';

import { Card, CardHeader, CardContent, CardFooter } from '@/components/ui/card';

import { Button } from '@/components/ui/button';

import { Alert, AlertTitle, AlertDescription } from '@/components/ui/alert';

import { predictDiseaseFromSymptoms } from '@/utils/api';

import { getDoctorsForDisease, type Doctor } from '@/utils/doctorUtils';

import DoctorCard from '@/components/DoctorCard';

import { FaSpinner } from 'react-icons/fa';

interface LocationState {

  symptoms: string[];

  diseaseName: string;

}

const Diagnosis: React.FC = () => {

  const { t } = useLanguage();

  const navigate = useNavigate();

  const location = useLocation();

  // Get data from location state

  const state = location.state as LocationState;

  const symptoms = state?.symptoms || [];

  const diseaseName = state?.diseaseName || '';

  const [prediction, setPrediction] = useState<{

    predictedDisease: string;

    confidence: number;

    description: string;

    message: string;

    matchingSymptoms?: string[];

  } | null>(null);

  const [availableDoctors, setAvailableDoctors] = useState<Doctor[]>([]);

  const [selectedDoctor, setSelectedDoctor] = useState<Doctor | null>(null);

  const [isLoading, setIsLoading] = useState(true);

  const [error, setError] = useState<string | null>(null);

  // Use the ML model to predict the disease based on symptoms

  useEffect(() => {

    if (!state || !state.symptoms || state.symptoms.length === 0) {

      setIsLoading(false);

      setError(t('लक्षण नहीं प्रदान किए गए', 'No symptoms provided. Please go back and select symptoms.'));

      return;

    }

    const fetchPrediction = async () => {

      try {

        setIsLoading(true);

        setError(null);

        console.log('Fetching prediction for symptoms:', symptoms);

        const predictionResult = await predictDiseaseFromSymptoms(symptoms);

        console.log('Received prediction:', predictionResult);

        if (predictionResult.predictedDisease === "Error") {

          throw new Error(predictionResult.message);

        }

        // Format the prediction result

        const result = {

          predictedDisease: t(predictionResult.predictedDisease, predictionResult.predictedDisease),

          confidence: predictionResult.confidence,

          description: t(predictionResult.description, predictionResult.description),

          matchingSymptoms: predictionResult.matchingSymptoms,

          message: predictionResult.message

        };

        setPrediction(result);

        // Get available doctors for the predicted disease

        const doctors = await getDoctorsForDisease(result.predictedDisease);

        setAvailableDoctors(doctors);

      } catch (err) {

        console.error('Error in fetchPrediction:', err);

        setError(err instanceof Error ? err.message : t('निदान प्राप्त करने में विफल', 'Failed to get diagnosis. Please try again.'));

      } finally {

        setIsLoading(false);

      }

    };

    fetchPrediction();

  }, [symptoms, t, state]);

  const handleDoctorSelect = (doctor: Doctor) => {

    setSelectedDoctor(doctor);

  };

  const handleCallDoctor = () => {

    if (selectedDoctor) {

      navigate('/doctor-call', {

        state: {

          doctor: selectedDoctor,

          disease: prediction?.predictedDisease,

          symptoms

        }

      });

    }

  };

  const goBack = () => {

    navigate(-1);

  };

  return (

    <div className="container mx-auto px-4 py-8">

      <VoiceGuidance pageName="diagnosis" />

      <div className="mb-4 flex items-center justify-between">

        <h1 className="text-3xl font-bold text-primary">

          {t('निदान परिणाम', 'Diagnosis Results')}

        </h1>

        <Button variant="outline" onClick={goBack}>

          {t('वापस जाएं', 'Go Back')}

        </Button>

      </div>

      {isLoading ? (

        <div className="flex justify-center items-center h-64">

          <FaSpinner className="animate-spin text-primary h-12 w-12" />

        </div>

      ) : error ? (

        <Alert variant="destructive">

          <AlertTitle>{t('त्रुटि', 'Error')}</AlertTitle>

          <AlertDescription>{error}</AlertDescription>

        </Alert>

      ) : prediction && (

        <Card className="w-full">

          <CardHeader>

            <div className="flex flex-col md:flex-row md:justify-between md:items-center gap-4">

              <div>

                <p className="text-sm text-muted-foreground">

                  {t('आपका निदान', 'Your Diagnosis')}

                </p>

                <p className="text-2xl font-bold text-primary">

                  {prediction.predictedDisease}

                </p>

              </div>

              <div className="rounded-lg bg-muted px-3 py-1 text-sm flex flex-col items-center">

                <span className="text-muted-foreground">

                  {t('विश्वास स्तर', 'Confidence')}

                </span>

                <span className="font-medium">{prediction.confidence}%</span>

              </div>

            </div>

          </CardHeader>

          <CardContent>

            <div className="space-y-4">

              <div>

                <h3 className="font-semibold">

                  {t('विवरण', 'Description')}

                </h3>

                <p className="text-muted-foreground">

                  {prediction.description}

                </p>

              </div>

              {prediction.matchingSymptoms && prediction.matchingSymptoms.length > 0 && (

                <div>

                  <h3 className="font-semibold">

                    {t('मिलान लक्षण', 'Matching Symptoms')}

                  </h3>

                  <ul className="list-disc pl-5">

                    {prediction.matchingSymptoms.map((symptom, index) => (

                      <li key={index} className="text-muted-foreground">

                        {t(symptom, symptom)}

                      </li>

                    ))}

                  </ul>

                </div>

              )}

              <div className="bg-yellow-50 p-4 rounded-md">

                <p className="text-yellow-800 text-sm">

                  {t('यह एक चिकित्सीय निदान नहीं है। कृपया स्वास्थ्य पेशेवर से परामर्श करें।',

                    'This is not a medical diagnosis. Please consult with a healthcare professional.')}

                </p>

              </div>

              <div>

                <h3 className="font-semibold mb-2">

                  {t('उपलब्ध डॉक्टर', 'Available Doctors')}

                </h3>

                <div className="grid grid-cols-1 md:grid-cols-2 gap-3">

                  {availableDoctors.map((doctor) => (

                    <DoctorCard

                      key={doctor.id}

                      doctor={doctor}

                      onSelect={handleDoctorSelect}

                      isSelected={selectedDoctor?.id === doctor.id}

                    />

                  ))}

                </div>

                <div className="flex justify-center mt-6">

                  <Button

                    onClick={handleCallDoctor}

                    className="bg-green-500 hover:bg-green-600"

                    size="lg"

                    disabled={!selectedDoctor}

                  >

                    {t('डॉक्टर से कॉल करें', 'Call Doctor')}

                  </Button>

                </div>

              </div>

            </div>

          </CardContent>

        </Card>

      )}

    </div>

  );

};

export default Diagnosis;

import React, { useState, useEffect } from 'react';

import { useNavigate } from 'react-router-dom';

import { useLanguage } from '@/contexts/LanguageContext';

import VoiceGuidance from '@/components/VoiceGuidance';

import Header from '@/components/Header';

import TouchKeyboard from '@/components/TouchKeyboard';

import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';

import { Input } from '@/components/ui/input';

import { Button } from '@/components/ui/button';

import { Alert, AlertDescription } from '@/components/ui/alert';

import { Checkbox } from '@/components/ui/checkbox';

import { Label } from '@/components/ui/label';

import { Textarea } from '@/components/ui/textarea';

import { getDiseaseSymptoms } from '@/utils/api';

interface Symptom {

  id: string;

  description: string;

}

const SymptomInput: React.FC = () => {

  const { t, language } = useLanguage();

  const navigate = useNavigate();

  const [currentScreen, setCurrentScreen] = useState<'diseaseInput' | 'symptomSelection'>('diseaseInput');

  const [diseaseInput, setDiseaseInput] = useState('');

  const [isLoading, setIsLoading] = useState(false);

  const [error, setError] = useState<string | null>(null);

  const [symptoms, setSymptoms] = useState<Symptom[]>([]);

  const [selectedSymptoms, setSelectedSymptoms] = useState<string[]>([]);

  const [customSymptom, setCustomSymptom] = useState('');

  const [activeField, setActiveField] = useState<string | null>(null);

  const [keyboardType, setKeyboardType] = useState<'numeric' | 'alphanumeric' | 'hindi'>('alphanumeric');

  const handleDiseaseSubmit = async () => {

    if (diseaseInput.trim() === '') {

      setError(t('कृपया बीमारी का नाम दर्ज करें', 'Please enter a disease name'));

      return;

    }

    try {

      setIsLoading(true);

      setError(null);

      console.log('Fetching symptoms for disease:', diseaseInput);

      // Get disease symptoms from backend

      const diseaseInfo = await getDiseaseSymptoms(diseaseInput);

      console.log('Received disease info:', diseaseInfo);

      if (!diseaseInfo || !diseaseInfo.symptoms || diseaseInfo.symptoms.length === 0) {

        setError(t('बीमारी के लक्षण प्राप्त करने में त्रुटि हुई', 'Failed to get symptoms for the disease'));

        setIsLoading(false);

        return;

      }

      // Convert symptoms to format needed for UI

      const symptomsList = diseaseInfo.symptoms.map(symptom => ({

        id: symptom.toLowerCase().replace(/\s+/g, '-'),

        description: symptom

      }));

      setSymptoms(symptomsList);

      setCurrentScreen('symptomSelection');

    } catch (err) {

      console.error('Error fetching disease symptoms:', err);

      setError(t('बीमारी के लक्षण प्राप्त करने में त्रुटि हुई', 'Failed to get symptoms for the disease'));

    } finally {

      setIsLoading(false);

    }

  };

  const handleSymptomToggle = (symptomDescription: string) => {

    setSelectedSymptoms(prev => {

      if (prev.includes(symptomDescription)) {

        return prev.filter(s => s !== symptomDescription);

      } else {

        return [...prev, symptomDescription];

      }

    });

  };

  const handleSubmit = () => {

    // Check if any symptoms are selected or custom symptom is entered

    if (selectedSymptoms.length === 0 && customSymptom.trim() === '') {

      setError(t('कृपया लक्षण चुनें या दर्ज करें', 'Please select or enter symptoms'));

      return;

    }

    // Combine selected symptoms and custom symptom if entered

    const allSymptoms = [...selectedSymptoms];

    if (customSymptom.trim() !== '') {

      allSymptoms.push(customSymptom.trim());

    }

    // Navigate to diagnosis page with symptoms

    navigate('/diagnosis', {

      state: {

        symptoms: allSymptoms,

        diseaseName: diseaseInput

      }

    });

  };

  const handleKeyPress = (key: string) => {

    if (activeField === 'diseaseInput') {

      setDiseaseInput(prev => prev + key);

    } else if (activeField === 'customSymptom') {

      setCustomSymptom(prev => prev + key);

    }

  };

  const handleBackspace = () => {

    if (activeField === 'diseaseInput') {

      setDiseaseInput(prev => prev.slice(0, -1));

    } else if (activeField === 'customSymptom') {

      setCustomSymptom(prev => prev.slice(0, -1));

    }

  };

  return (

    <div className="container mx-auto px-4 py-8">

      <Header />

      <VoiceGuidance />

      {currentScreen === 'diseaseInput' ? (

        <Card className="w-full">

          <CardHeader>

            <CardTitle className="text-center text-2xl">

              {t('अपने लक्षण बताएँ', 'Tell us your symptoms')}

            </CardTitle>

            <p className="text-center text-muted-foreground">

              {t('सभी लागू लक्षणों का चयन करें या बीमारी का नाम बताएँ', 'Select all symptoms that apply or enter disease name')}

            </p>

          </CardHeader>

          <CardContent>

            <div className="space-y-4">

              <div>

                <h2 className="text-xl font-semibold text-center mb-4">

                  {t('बीमारी का नाम दर्ज करें', 'Enter Disease Name')}

                </h2>

                <div className="flex gap-2">

                  <Input

                    type="text"

                    placeholder={t('जैसे: बुखार, सर्दी, आदि', 'Ex: Fever, Cold, etc.')}

                    value={diseaseInput}

                    onChange={(e) => setDiseaseInput(e.target.value)}

                    onFocus={() => setActiveField('diseaseInput')}

                    className="flex-1"

                  />

                  <Button

                    onClick={handleDiseaseSubmit}

                    disabled={isLoading}

                    className="bg-blue-500 hover:bg-blue-600"

                  >

                    {isLoading ? t('खोज रहे हैं...', 'Searching...') : t('खोजें', 'Find')}

                  </Button>

                </div>

                {error && (

                  <Alert variant="destructive" className="mt-2">

                    <AlertDescription>{error}</AlertDescription>

                  </Alert>

                )}

              </div>

              {activeField === 'diseaseInput' && (

                <TouchKeyboard

                  onKeyPress={handleKeyPress}

                  onBackspace={handleBackspace}

                  keyboardType={keyboardType}

                  onChangeKeyboardType={setKeyboardType}

                />

              )}

              <div className="flex justify-center mt-8">

                <Button

                  onClick={() => navigate('/')}

                  variant="outline"

                  className="mr-2"

                >

                  {t('वापस', 'Back')}

                </Button>

              </div>

            </div>

          </CardContent>

        </Card>

      ) : (

        <Card className="w-full">

          <CardHeader>

            <CardTitle className="text-center text-2xl">

              {t('लक्षणों का चयन करें', 'Select Symptoms')}

            </CardTitle>

            <p className="text-center text-muted-foreground">

              {t(`${diseaseInput} के लिए लक्षणों का चयन करें`, `Select symptoms for ${diseaseInput}`)}

            </p>

          </CardHeader>

          <CardContent>

            <div className="space-y-6">

              {/\* Symptom checkboxes \*/}

              <div className="grid grid-cols-1 md:grid-cols-2 gap-3">

                {symptoms.map((symptom) => (

                  <div className="flex items-center space-x-2" key={symptom.id}>

                    <Checkbox

                      id={symptom.id}

                      checked={selectedSymptoms.includes(symptom.description)}

                      onCheckedChange={() => handleSymptomToggle(symptom.description)}

                    />

                    <Label htmlFor={symptom.id}>{symptom.description}</Label>

                  </div>

                ))}

              </div>

              {/\* Custom symptom input \*/}

              <div className="space-y-2">

                <Label htmlFor="customSymptom">

                  {t('अपने अन्य लक्षण यहाँ लिखें', 'Write your other symptoms here')}

                </Label>

                <Textarea

                  id="customSymptom"

                  value={customSymptom}

                  onChange={(e) => setCustomSymptom(e.target.value)}

                  onFocus={() => setActiveField('customSymptom')}

                  placeholder={t('अन्य लक्षण...', 'Other symptoms...')}

                  className="min-h-[80px]"

                />

              </div>

              {activeField === 'customSymptom' && (

                <TouchKeyboard

                  onKeyPress={handleKeyPress}

                  onBackspace={handleBackspace}

                  keyboardType={keyboardType}

                  onChangeKeyboardType={setKeyboardType}

                />

              )}

              {error && (

                <Alert variant="destructive">

                  <AlertDescription>{error}</AlertDescription>

                </Alert>

              )}

              <div className="flex justify-center space-x-4 mt-6">

                <Button

                  onClick={() => setCurrentScreen('diseaseInput')}

                  variant="outline"

                  className="min-w-[100px]"

                >

                  {t('वापस', 'Back')}

                </Button>

                <Button

                  onClick={handleSubmit}

                  className="bg-green-500 hover:bg-green-600 min-w-[100px]"

                >

                  {t('आगे बढ़ें', 'Next')} →

                </Button>

              </div>

            </div>

          </CardContent>

        </Card>

      )}

    </div>

  );

};

export default SymptomInput;