

Phase 5: Project Demonstration & Documentation

Title: Healthcare Diagnostic and Treatment System

Abstract:

The Healthcare Diagnostic and Treatment System aims to enhance clinical efficiency and patient outcomes through the integration of intelligent diagnosis, personalized treatment plans, and real-time monitoring. Utilizing artificial intelligence, natural language processing, and IoT devices, this system automates symptom analysis, suggests treatment pathways, and collects live patient data for ongoing care. The final phase includes system integration, robust data security, ERP compatibility, and scalability testing. This report presents a comprehensive overview of the demonstration, technical documentation, performance evaluation, and system handover, complete with screenshots, diagrams, and codebase references to illustrate system functionality.

1. Project Demonstration

Overview:

The Healthcare Diagnostic and Treatment System will be presented in a live demo to stakeholders, emphasizing its diagnostic accuracy, treatment recommendation capabilities, and real-time health monitoring.

Demonstration Details:

System Walkthrough: A complete demo of the platform from user entry to treatment suggestion, highlighting chatbot interaction.

AI-Based Diagnosis: Demonstrates how the system identifies probable conditions using symptom inputs and patient history.

Treatment Recommendation Engine: AI generates tailored treatment options based on diagnosis and real-time data.

IoT Integration: Live readings from devices (e.g., heart rate, BP, glucose levels) are shown feeding into the system.

Performance Metrics: Tests system responsiveness, concurrent user load capacity, and data throughput.

Security & Compliance: Demonstrates encryption of medical data and HIPAA-compliant access protocols.

Outcome:

Stakeholders will gain insight into the system’s readiness for real-world deployment, especially its diagnostic precision, treatment logic, and secure handling of sensitive health data.

2. Project Documentation

Overview:

All aspects of the Healthcare Diagnostic and Treatment System are documented, ensuring clarity for developers, users, and system administrators.

Documentation Sections:

System Architecture: Visual representation of AI layers, treatment modules, data flow, and IoT connections.

Code Documentation: Detailed breakdowns of AI algorithms, backend integrations, and interface logic.

User Guide: Step-by-step instructions for accessing the system, inputting symptoms, and understanding treatment suggestions.

Administrator Guide: Guidance on backend management, system updates, and real-time data oversight.

Testing Reports: Comprehensive data on system reliability, error rates, and compliance checks.

Outcome:

Documentation ensures that the system is maintainable, extensible, and understandable for future development and deployment.

3. Feedback and Final Adjustments

Overview:

Following the demonstration, structured feedback will be collected and used to refine performance and usability.

Steps:

Feedback Collection: Responses from faculty, domain experts, and test users will be logged using forms and live feedback.

System Refinement: Key issues like interface lag, diagnostic discrepancies, or data sync lags will be resolved.

Final Testing: Regression testing, scalability validation, and security checks will be performed.

Outcome:

The final build will be stable, accurate, and ready for institutional or commercial application.

4. Final Project Report Submission

Overview:

A cumulative report summarizes the full project lifecycle, innovation points, challenges, and validated outcomes.

Report Sections:

Executive Summary: Brief recap of goals, methodology, and results.

Phase Breakdown: Each development phase is outlined—diagnosis module, treatment logic, device integration, and testing.

Challenges & Solutions: Discusses hurdles like AI misclassification, sensor noise, and integration bugs.

Outcomes: Overview of working features, system robustness, and user satisfaction metrics.

Outcome:

The report provides a foundation for stakeholders, investors, or institutional adopters to understand the project in full.

5. Project Handover and Future Works

Overview:

Outlines potential directions for enhancing and deploying the Healthcare Diagnostic and Treatment System.

Handover Details:

Next Steps: Suggestions include incorporating multilingual AI, expanding disease models, and scaling cloud infrastructure.

Deployment Plan: Recommendations for initial rollout in clinics or telemedicine platforms.

Outcome:

A detailed and structured handover will ensure the continuity of development and smooth system onboarding.

# CODE

from cryptography.fernet import Fernet

# Mock database of symptoms

diagnosis\_db = {

"fever": {"diagnosis": "Viral Infection", "treatment": "Paracetamol, rest, and hydration"},

"cough": {"diagnosis": "Respiratory Infection", "treatment": "Cough syrup and steam inhalation"},

"headache": {"diagnosis": "Migraine or Stress", "treatment": "Pain reliever and rest"},

"sore throat": {"diagnosis": "Throat Infection", "treatment": "Warm saline gargle and lozenges"},

}

# Encryption key and cipher

key = Fernet.generate\_key()

cipher = Fernet(key)

def encrypt\_data(data):

return cipher.encrypt(data.encode())

def decrypt\_data(encrypted):

return cipher.decrypt(encrypted).decode()

def diagnose(symptoms):

recommendations = []

for symptom in symptoms:

symptom = symptom.strip().lower()

if symptom in diagnosis\_db:

entry = diagnosis\_db[symptom]

recommendations.append(f"Symptom: {symptom}\nDiagnosis: {entry['diagnosis']}\nTreatment: {entry['treatment']}")

else:

recommendations.append(f"Symptom: {symptom}\nDiagnosis: Not Found\nRecommendation: Consult a doctor.")

return recommendations

def chatbot():

print("AI Health Assistant: Hello! Please enter your symptoms (comma-separated):")

user\_input = input("You: ")

# Encrypt input

encrypted = encrypt\_data(user\_input)

print("\n[Encrypted input]:", encrypted)

# Decrypt for processing

decrypted = decrypt\_data(encrypted)

symptoms = decrypted.split(',')

print("\nAI Health Assistant: Here are your diagnostic results and suggestions:\n")

for result in diagnose(symptoms):

print(result + "\n")

if \_name\_ == "\_main\_":

chatbot()

## Source code:

app.py

from flask import Flask, render\_template, request, jsonify

from model import diagnose

from ehr\_mock import get\_patient\_history

from treatments import get\_treatment

app = Flask(\_name\_)

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/diagnose', methods=['POST'])

def diagnose\_user():

data = request.json

symptoms = data.get("symptoms")

patient\_id = data.get("patient\_id")

history = get\_patient\_history(patient\_id)

diagnosis = diagnose(symptoms, history)

treatment = get\_treatment(diagnosis)

return jsonify({

"diagnosis": diagnosis,

"treatment": treatment,

"history": history

})

if \_name\_ == '\_main\_':

app.run(debug=True)

---

model.py

def diagnose(symptoms, history):

if "fever" in symptoms and "cough" in symptoms:

return "Flu"

elif "chest pain" in symptoms:

return "Possible Heart Issue"

else:

return "General Checkup Recommended"

---

ehr\_mock.py

def get\_patient\_history(patient\_id):

dummy\_data = {

"101": {"age": 45, "conditions": ["hypertension"], "last\_visit": "2024-11-20"},

"102": {"age": 30, "conditions": [], "last\_visit": "2025-01-15"},

}

return dummy\_data.get(str(patient\_id), {"age": "Unknown", "conditions": [], "last\_visit": "N/A"})

---

treatments.py

def get\_treatment(diagnosis):

treatments = {

"Flu": ["Rest", "Hydration", "Paracetamol"],

"Possible Heart Issue": ["ECG Test", "Cardiologist Referral"],

"General Checkup Recommended": ["Routine Blood Test", "Consult Physician"]

}

return treatments.get(diagnosis, ["Consult Specialist"])

---

templates/index.html

<!DOCTYPE html>

<html>

<head>

<title>Healthcare Diagnostics</title>

</head>

<body>

<h2>Enter Symptoms</h2>

<input id="symptoms" placeholder="e.g., fever, cough" />

<input id="patient\_id" placeholder="Patient ID" />

<button onclick="submitData()">Diagnose</button>

<div id="result"></div>

<script>

function submitData() {

const symptoms = document.getElementById('symptoms').value.split(',');

const patient\_id = document.getElementById('patient\_id').value;

fetch('/diagnose', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ symptoms, patient\_id })

})

.then(res => res.json())

.then(data => {

document.getElementById('result').innerHTML =

`<p><b>Diagnosis:</b> ${data.diagnosis}</p>

<p><b>Treatment:</b> ${data.treatment.join(', ')}</p>

<p><b>Patient History:</b> ${JSON.stringify(data.history)}</p>`;

});

}

</script>

</body>

</html>

# Sample Output (User enters: fever, headache, cough)

AI Health Assistant: Hello! Please enter your symptoms (comma-separated):

You: fever, headache, cough

[Encrypted input]: b'gAAAAABk...'

AI Health Assistant: Here are your diagnostic results and suggestions:

Symptom: fever

Diagnosis: Viral Infection

Treatment: Paracetamol, rest, and hydration

Symptom: headache

Diagnosis: Migraine or Stress

Treatment: Pain reliever and rest

Symptom: cough

Diagnosis: Respiratory Infection

Treatment: Cough syrup and steam inhalation