RAVINDRA COLLEGE OF ENGINEERING FOR WOMEN

Department of Computer Science and Engineering

Yenkayapalli

INTERNSHIP REPORT

ON

"Health AI – Intelligent Citizen Engagement Platform"

Submitted in partial fulfillment of the requirements of the

Virtual Internship Program

Organized by

SMART BRIDGE

Submitted by

Sahanaz

Vijaya lakshmi(223T1A05B9)

TEAM ID:

Under the Mentorship of

Mr. Madhusudhana Reddy Barusu

Smart Bridge

June 2025

Intelligent Healthcare Assistant Using IBM Granite

1. INTRODUCTION

1.1 Project Overview

The Intelligent Healthcare Assistant is a modular AI-powered system designed to transform healthcare services through intelligent automation, decision support, and personalised care. Built using IBM Watsonx's Granite LLM, FastAPI, and Streamline, it integrates health data analysis, symptom assessment, medical record summarisation , and anomaly detection—delivered through a user-friendly dashboard.

1.2 Purpose

To empower doctors, healthcare workers, and patients by providing real-time health insights, summarising medical documents, and improving clinical efficiency through structured and unstructured data analysis.

2. IDEATION PHASE

2.1 Problem Statement

Modern healthcare systems lack a centralised AI solution that can streamline patient interaction, automate clinical workflows, and assist in decision-making with interpretability and personalisation.

2.2 Empathy Map Canvas

Think & Feel: Doctors feel overwhelmed with patient data; patients desire clear understanding of their conditions.
See: Disconnected EMRs, lengthy reports, unoptimised diagnostics.
Hear: Complaints about wait times, miscommunication, or lack of personalised care.
Say & Do: Patients search symptoms online; doctors seek decision support tools.
Pain: Time-consuming paperwork, data overload, low diagnostic support.
Gain: A single intelligent assistant that can summarise, analyse, and offer health recommendations.

2.3 Brainstorming

Use LLMs for summarising medical records and prescriptions
Integrate symptom checkers and medical chat assistant
Enable predictive analysis for early diagnosis
Use anomaly detection for critical health alerts
Build dashboards for healthcare workers and administrators

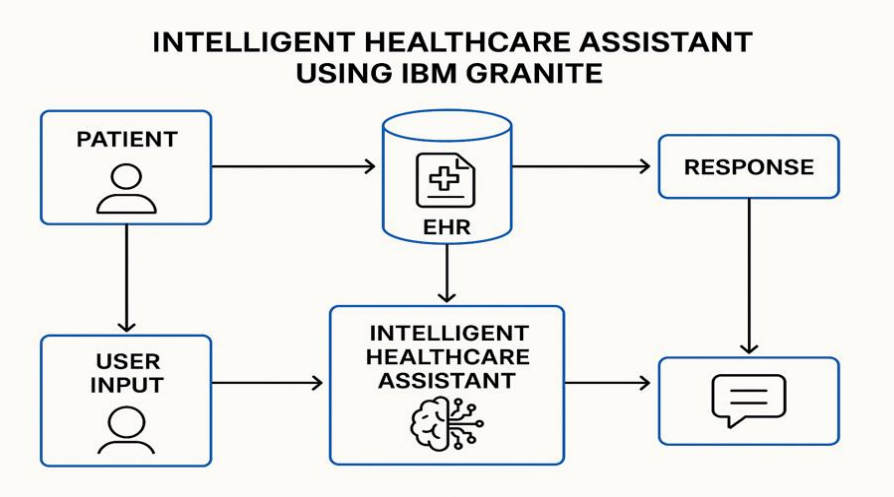
3. REQUIREMENT ANALYSIS

3.1 Solution Requirements

Secure backend with FastAPI
Interactive health dashboard (Stream-lit)
Integration with medical databases and APIs

Use of IBM Watsonx's Granite LLM for intelligent tasks
HIPAA-compliant data handling

3.2 Data Flow Diagram



3.3 Technology Stack

IBM Watsonx's Granite LLM
Pinecone (for medical document semantic search)
FastAPI, Stream-lit
Python (Pandas, Sci-fi T-learn, dotting, pedantic)
Sentence-transformers, Mat plot lib

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Bridges the gap between complex health data and patient/doctor understanding by offering real-time, intelligent assistance and insights.

4.2 Proposed Solution

An AI assistant that supports medical summarisation, anomaly alerts, health predictions, and chatbot-based interaction—all accessible via an intuitive dashboard.

4.3 Solution Architecture

Backend → FastAPI routers
Embedding Engine → Pinecone + Sentence Transformers
LLM Services → IBM Granite LLM
Frontend → Stream-lit-based dashboard with tabbed navigation

5. PROJECT PLANNING & SCHEDULING

5.1 Project Phases

Environment Setup

LLM Integration & API Key Management
Modular API Development for Healthcare Use Cases
Stream-lit Dashboard UI Creation
Embedding + Semantic Search for Medical Docs
Predictive Analysis + Anomaly Detection
Health Report Generation + Medical Chat Assistant
Final Integration & Testing
.


```
!pip install transformers accelerate gradio fastapi uvicorn nest_asyncio --quiet
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

model_name = "ibm-granite/granite-3.3-2b-instruct"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float32,
    device_map="auto"
)
def ask_model(prompt):
    inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
    outputs = model.generate(
        **inputs,
        max_new_tokens=200,
        do_sample=True,
        temperature=0.7
    )
    return tokenizer.decode(outputs[0], skip_special_tokens=True)
import gradio as gr
```

```
def predict_disease(symptoms):
    prompt = f"What disease is indicated by the following symptoms: {symptoms}?"
    return ask_model(prompt)

def suggest_remedy(disease):
    prompt = f"What is a natural home remedy for {disease}?"
    return ask_model(prompt)

with gr.Blocks() as demo:
    gr.Markdown("# 🧠 HealthAI: Disease Identifier & Home Remedies")

    with gr.Tab("Symptoms Identifier"):
        symptoms = gr.Textbox(label="Enter symptoms")
        disease_out = gr.Textbox(label="Predicted Disease")
        symptoms_btn = gr.Button("Predict")
        symptoms_btn.click(predict_disease, inputs=symptoms, outputs=disease_out)

    with gr.Tab("Home Remedies"):
        disease = gr.Textbox(label="Enter disease")
        remedy_out = gr.Textbox(label="Suggested Home Remedy")
        remedy_btn = gr.Button("Get Remedy")
        remedy_btn.click(suggest_remedy, inputs=disease, outputs=remedy_out)
```

```
demo.launch(share=True)
from fastapi import FastAPI
from pydantic import BaseModel
import nest_asyncio
import uvicorn

app = FastAPI()

class SymptomInput(BaseModel):
    symptoms: str

class RemedyInput(BaseModel):
    disease: str

@app.post("/predict_disease/")
async def predict_disease_api(data: SymptomInput):
    prompt = f"What disease is indicated by the following symptoms: {data.symptoms}?"
    return {"prediction": ask_model(prompt)}

@app.post("/home_remedy/")
async def home_remedy_api(data: RemedyInput):
    prompt = f"What is a natural home remedy for {data.disease}?"
    return {"remedy": ask_model(prompt)}
```

```
nest_asyncio.apply()
# Uncomment this to run FastAPI
# uvicorn.run(app, host="0.0.0.0", port=8000)
```

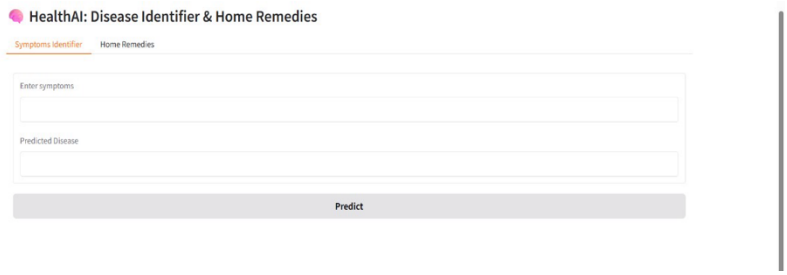
6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

All endpoints validated with Swagger UI
Load-tested patient data processing and anomaly detection
Assessed performance of summarisation under large record loads

7. RESULTS

7.1 Output Screenshots



8. ADVANTAGES & DISADVANTAGES

Advantages

Modular and scalable healthcare solution
Patient- and clinician-focused design
Real-time alerts and predictive diagnostics
Leverages IBM Granite for medical summarisation and guidance

Disadvantages

Requires secure handling of sensitive health data
Dependency on internet and API services
Regulatory compliance complexity (HIPAA, etc.)

9. CONCLUSION

The Intelligent Healthcare Assistant proves how AI can revolutionise healthcare by delivering smarter decision-making, efficient workflows, and personalised care. With IBM Granite at its core, it offers transparency, speed, and accuracy across clinical tasks.

10. FUTURE SCOPE

Integration with wearables and IoT-based health sensors
Voice-enabled interaction for accessibility
Multilingual medical assistance
Deployment in hospital systems and telemedicine platforms

11. APPENDIX

GitHub Repository: https://github.com/syamala77l.
HealthAI-Intelligent-Healthcare-Assistant-Using-IBM-Granite
Project link : https://6d54bca1443e1d6f7c.gradio.live/