Generative AI with IBM Cloud

Project Documentation format

1. Introduction

- **Project Title: HealthAI: Intelligent Healthcare Assistant Using IBM Granite**

- **Team Members:** Reddy Deepthi - code written

  Syed Aswiya - demo video

  Pathan Afrin Khatun- documentation

2. Project Overview

- **Purpose:** The purpose of a Health AI project is to enhance healthcare delivery by leveraging artificial intelligence to improve diagnostics, personalize treatments, optimize operations, and support decision-making for better patient outcomes.

- **Features:** Health AI features include predictive analytics, diagnostic support, personalized treatment plans, medical imaging analysis, remote monitoring, natural language processing, workflow automation, real-time alerts, data integration, risk assessment, and virtual health assistants..

3. Architecture

- **Frontend:** It provides a user-friendly web and mobile interface for seamless citizen interaction, service access and real time communication.

- **Backend:** It leverages of AI-engines, data analytics, and secure APIs to process request, manage citizen data and power intelligent decision making.

- **Database:** It securely stores structured and unstructured citizen data, interaction logs.

4. Setup Instructions

- **Prerequisites:** Visual Studio Code

- **Installation:** 1.lnstall VS code

  2. Install Language extensions(Python)

  3. Run code in VS code

5. Folder Structure

- VS code structure includes an activity bar, side bar, editor area, status bar and terminal panel, all organized to support efficient code development.

6. Running the Application

- Provide commands to start the frontend and backend servers locally i.e file.py, node app.js.

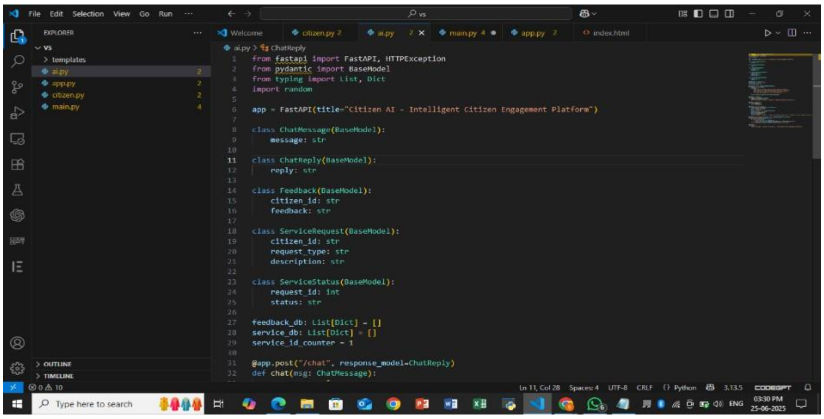
7. API Documentation

- sk-proj-
GOOGLE_API_KEY = "AIzaSyALFLzUNAAH9TxHjCLAXqx4IwERbWuLO5c" # <-- Replace this
genai.configure(api_key=GOOGLE_API_KEY)-
P6npBtaPdQYYO9G4DQ0XNUAruxNh6TNyfKoJdSfykc7zhbMOZV3qRu6RkA

8. Authentication

- Health Ai platform involves verifying users via secure tokens (like JWT) and managing sessions to maintain users state across interactions.

9. User Interface

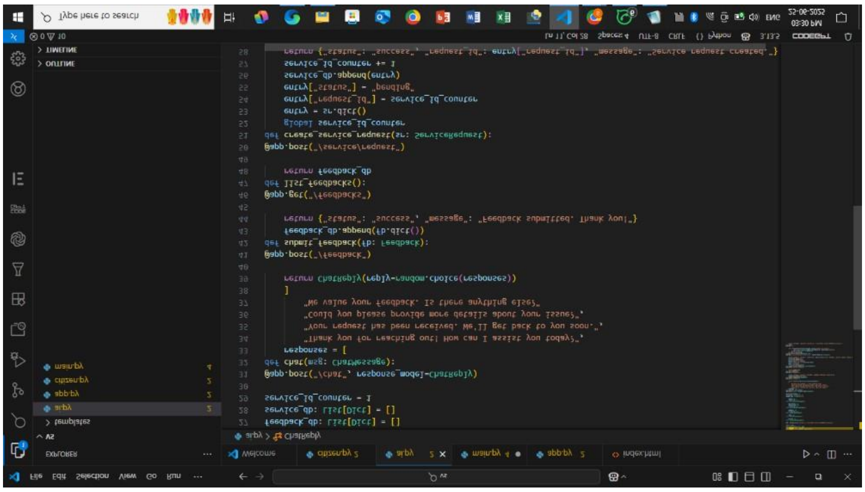


```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List, Dict
import random

app = FastAPI(title="Citizen AI - Intelligent Citizen Engagement Platform")

class ChatMessage(BaseModel):
    message: str

class ChatReply(BaseModel):
    reply: str

class Feedback(BaseModel):
    citizen_id: str
    feedback: str

class ServiceRequest(BaseModel):
    citizen_id: str
    request_type: str
    description: str

class ServiceStatus(BaseModel):
    request_id: int
    status: str

feedback_db: List[Dict] = []
service_db: List[Dict] = []
service_id_counter = 1

@app.post("/chat", response_model=ChatReply)
def chat(msg: ChatMessage):
```

10. Testing

- Testing Explorer UI-Built-in test runner UI

11. Screenshots



```python
feedback_db: List[Dict] = []
service_db: List[Dict] = []
service_id_counter = 1

@app.post("/chat", response_model=ChatReply)
def chat(msg: ChatMessage):
    responses = [
        "Thank you for reaching out! How can I assist you today?",
        "Your request has been received. We'll get back to you soon.",
        "Could you please provide more details about your issue?",
        "We value your feedback. Is there anything else?"
    ]
    return ChatReply(reply=random.choice(responses))

@app.post("/feedback")
def submit_feedback(fb: Feedback):
    feedback_db.append(fb.dict())
    return {"status": "success", "message": "Feedback submitted. Thank you!"}

@app.get("/feedbacks")
def list_feedbacks():
    return feedback_db

@app.post("/service/request")
def create_service_request(sr: ServiceRequest):
    global service_id_counter
    entry = sr.dict()
    entry["request_id"] = service_id_counter
    entry["status"] = "pending"
    service_db.append(entry)
    service_id_counter += 1
    return {"status": "success", "request_id": entry["request_id"], "message": "Service request created."}
```

12. Known Issues

- known issues like high CPU usage, extension conflicts, slow startup or failed Git integration

13. Future Enhancements

- It enhanced by adding intuitive no-code model training, automated data validation.