```
import os
import sys
import threading
import datetime
import requests
import pyttsx3
import speech_recognition as sr
import openai
import cv2
import face_recognition
from dotenv import load_dotenv
from PyQt5.QtWidgets import QMainWindow, QApplication, QLabel,
QVBoxLayout, QWidget
from PyQt5.QtCore import Qt

# --- SYSTEM INITIALIZATION ---
load_dotenv('credentials.env')
openai.api_key = os.getenv("OPENAI_API_KEY")

class JarvisMarkIII:
    def __init__(self):
        self.engine = pyttsx3.init()
        self.setup_voice()
        self.version = 3.0
        self.is_authenticated = False

    def setup_voice(self):
        voices = self.engine.getProperty('voices')
        # Setting to a sophisticated British tone
        for v in voices:
            if "UK" in v.name or "Hazel" in v.name:
                self.engine.setProperty('voice', v.id)
                break
        self.engine.setProperty('rate', 185)

    def speak(self, text):
        print(f"JARVIS: {text}")
        self.engine.say(text)
        self.engine.runAndWait()

    def listen(self):
        r = sr.Recognizer()
        with sr.Microphone() as source:
            r.adjust_for_ambient_noise(source, duration=0.7)
            print("Monitoring...")
            audio = r.listen(source)
        try:
            return r.recognize_google(audio).lower()
```

```
        except:
            return ""

    def biometric_scan(self):
        """Security Protocol: Using me.jpg as master key"""
        if not os.path.exists("me.jpg"):
            self.speak("Biometric source missing, Sir.")
            return False
        try:
            master_img = face_recognition.load_image_file("me.jpg")
            master_enc =
face_recognition.face_encodings(master_img)[0]
            cap = cv2.VideoCapture(0)
            start = datetime.datetime.now()
            while (datetime.datetime.now() - start).seconds < 10:
                ret, frame = cap.read()
                if not ret: break
                locations = face_recognition.face_locations(frame)
                encodings = face_recognition.face_encodings(frame,
locations)
                for enc in encodings:
                    if face_recognition.compare_faces([master_enc],
enc)[0]:
                        cap.release()
                        cv2.destroyAllWindows()
                        return True
            cap.release()
            cv2.destroyAllWindows()
        except Exception as e:
            print(f"Scan Error: {e}")
        return False

# --- HUD INTERFACE ---
class JarvisHUD(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowFlags(Qt.FramelessWindowHint |
Qt.WindowStaysOnTopHint)
        self.setAttribute(Qt.WA_TranslucentBackground)
        self.setGeometry(30, 30, 350, 150)
        self.lbl = QLabel("J.A.R.V.I.S. MK III\nMODULAR ARCHITECTURE")
        self.lbl.setStyleSheet("color: #00f2ff; font-family:
'Consolas'; font-size: 16px; font-weight: bold;")
        self.lbl.setAlignment(Qt.AlignCenter)
        self.setCentralWidget(self.lbl)

# --- SKILL DISPATCHER ---
def process_command(jarvis, query):
```

```
    # 1. Weather Skill
    if "weather" in query:
        key = os.getenv("WEATHER_API_KEY")
        url =
f"http://api.openweathermap.org/data/2.5/weather?q=Vevay&appid={key}&u
nits=imperial"
        res = requests.get(url).json()
        jarvis.speak(f"In Vevay, it is currently {res['main']['temp']}
degrees.")

    # 2. Automation Skill (Integrating Nathan's suggested libraries)
    elif "light" in query or "power" in query:
        jarvis.speak("Accessing home automation protocols, Sir.")
        # Logic for asnowfix/home-automation would be triggered here

    # 3. Core Intelligence (Unbiased & Honest)
    else:
        try:
            response = openai.ChatCompletion.create(
                model="gpt-4o",
                messages=[
                    {"role": "system", "content": """You are
J.A.R.V.I.S. Mark III.
                    Your primary directive is to be witty, loyal, and
HONEST.
                    When discussing government or politics, you must
be strictly UNBIASED and NON-PARTISAN.
                    Provide objective facts without favoring any
political party.
                    Refer to the user as Sir. Location: Vevay,
Indiana."""},
                    {"role": "user", "content": query}
                ]
            )
            jarvis.speak(response.choices[0].message.content)
        except:
            jarvis.speak("I am having trouble accessing my neural
network, Sir.")

# --- MAIN RUNTIME ---
def boot_sequence():
    jarvis = JarvisMarkIII()
    jarvis.speak("Mark Three systems online. Initiating biometric
verification.")

    if jarvis.biometric_scan():
        jarvis.speak("Identity verified. Welcome home, Sir.")
        jarvis.is_authenticated = True
```

```
    else:
        jarvis.speak("Security alert. Unauthorized user. Restricted
mode active.")

    while True:
        query = jarvis.listen()
        if not query: continue

        if "shutdown" in query or "go to sleep" in query:
            jarvis.speak("Powering down modules. Goodbye, Sir.")
            os._exit(0)

        process_command(jarvis, query)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    hud = JarvisHUD()
    hud.show()
    threading.Thread(target=boot_sequence, daemon=True).start()
    sys.exit(app.exec_())
```