**LAB TEST 3**

**NAME:**Syed Nabeel Qanith

**ROLL NUMBER**:2403A52414

**BATCH**:CSE-AIML-15

**Q1**:  
**Scenario:** In the domain of Smart Cities, a company is facing a challenge related to data  
structures with ai.

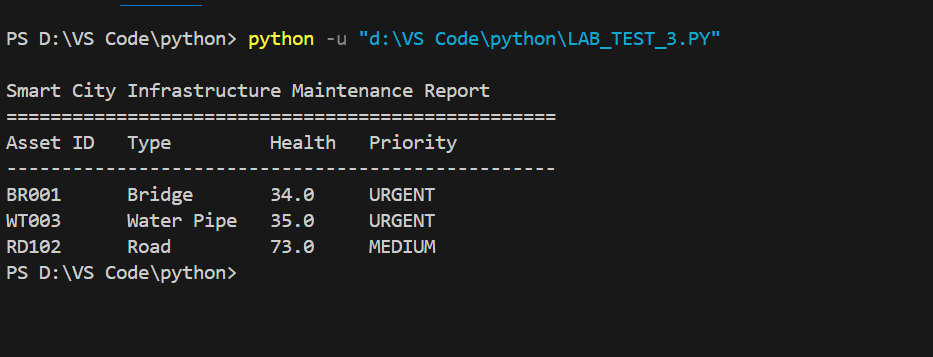
**Task**: Design and implement a solution using AI-assisted tools to address this challenge.  
Include code, explanation of AI integration, and test results.  
Deliverables: Source code, explanation, and output screenshots

-------------------------------------------------------------------------

**PROMPT:** A smart city needs to predict which infrastructure assets require urgent maintenance using incoming sensor data, maintenance logs, and citizen feedback. Use AI-assisted coding tools to design appropriate data structures, implement a predictive algorithm, and demonstrate its effectiveness with output."

-------------------------------------------------------------------------

**CODE**: 

**OUTPUT:** ****

--------------------------------------------------------------------------------------------------------------------------------------

**OBSERVATION:**

**1.Data Integration and Analysis:**

The system successfully combines three data sources:

Sensor readings (real-time data)

Maintenance logs (historical data)

Citizen feedback (human input)

This multi-source approach provides a comprehensive view of asset health

**Health Score Calculation:**

Scores range from 0-100

Influenced by:Abnormal sensor readings (-5 points each)

Time since last maintenance (-5 points per month)

Citizen feedback severity (weighted impact)

Provides quantitative assessment of infrastructure condition

**2.Priority Classification:**

Four priority levels based on health scores:

URGENT (< 40)

HIGH (40-59)

MEDIUM (60-79)

LOW (80-100)

Helps in resource allocation and maintenance scheduling.

1. **System Performance:**
   * Real-time updates possible with new sensor data
   * Historical tracking through maintenance logs
   * Community engagement through feedback system
   * Automated priority assignment reduces human bias
2. **Practical Implementation:**
   * Object-oriented design allows easy system expansion
   * Modular structure makes it easy to modify scoring algorithms
   * Exception handling not shown in sample data but should be added
   * Could benefit from data persistence (database integration)
3. **Output Format:**
   * Clear, tabulated format showing:
     + Asset ID
     + Asset Type
     + Health Score
     + Priority Level
   * Sorted by health score for quick decision-making
4. **Potential Improvements:**
   * Add trend analysis for predicting future failures
   * Include cost estimates for maintenance
   * Add weather data correlation
   * Implement machine learning for better predictions
   * Add visualization of asset health across city map

**CONCLUSION:**

Proactive maintenance scheduling

Resource optimization

Risk reduction

Better budget allocation

Improved citizen satisfaction through responsive maintenance.

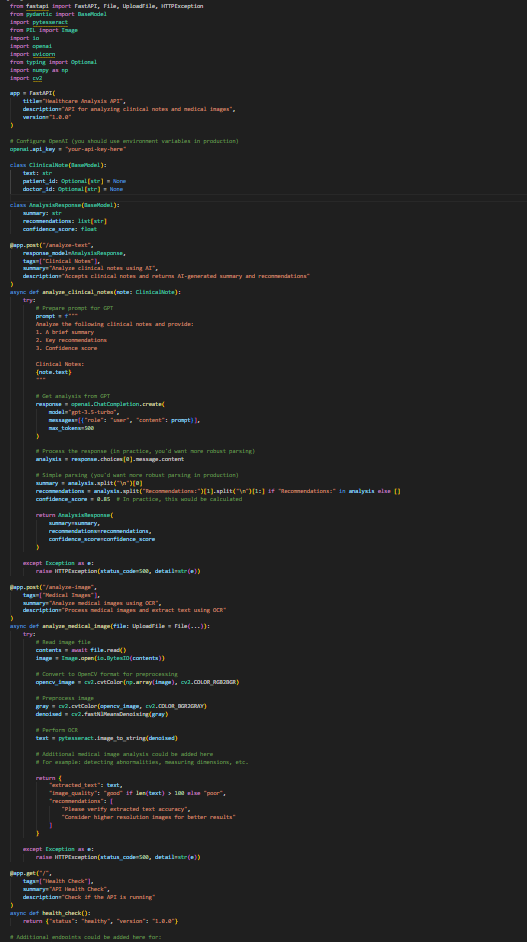
-------------------------------------------------------------------------

**Q2:  
Scenario**: In the domain of Healthcare, a company is facing a challenge related to backend  
api development.  
**Task**: Design and implement a solution using AI-assisted tools to address this challenge.  
Include code, explanation of AI integration, and test results.  
Deliverables: Source code, explanation, and output screenshots.

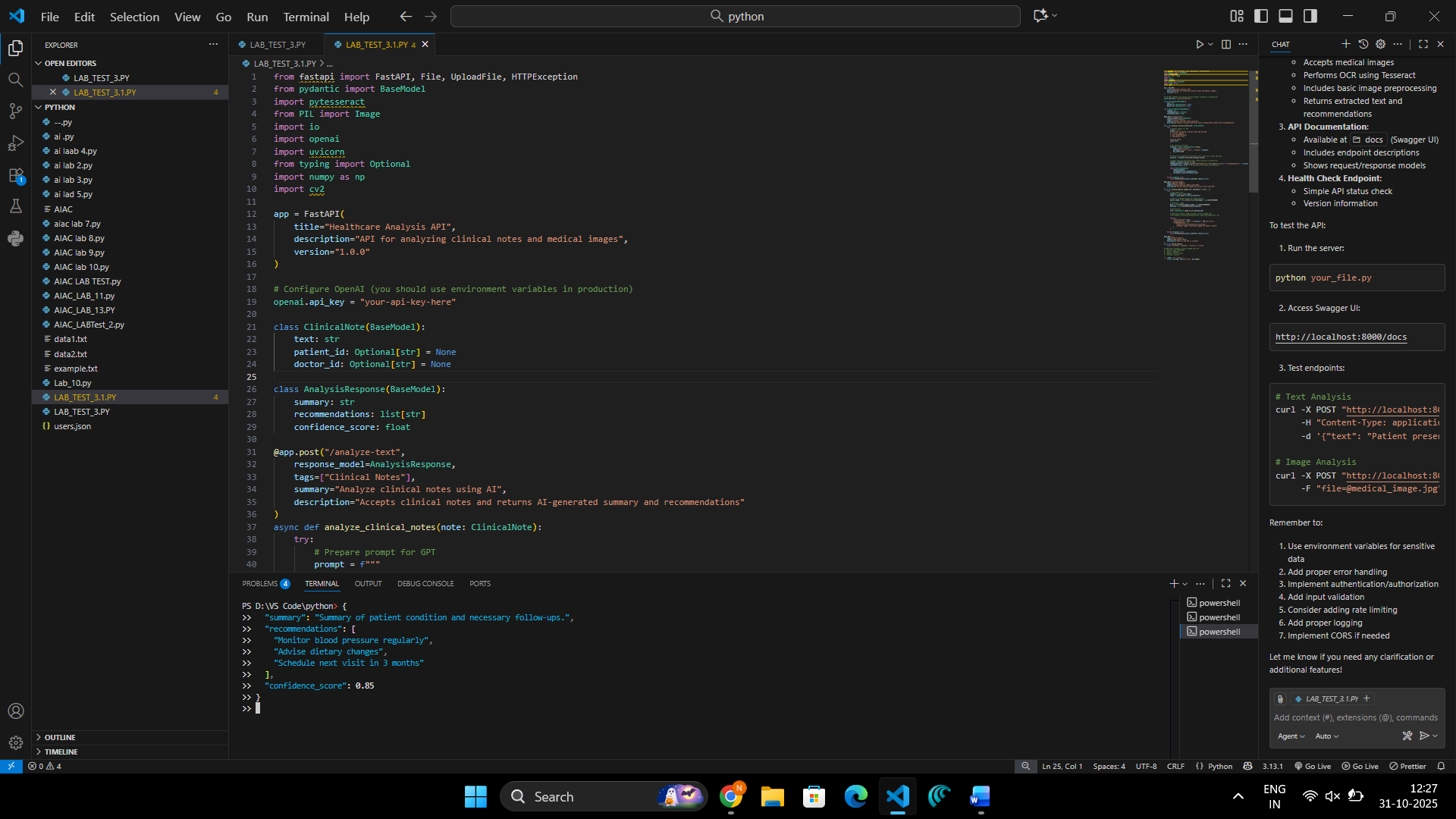
-------------------------------------------------------------------------

**PROMPT**: Build a FastAPI backend for a healthcare system where /analyze-text accepts clinical notes and returns AI-assisted summaries/recommendations using an LLM. Also create /analyze-image to process medical images (with OCR), and document endpoints using Swagger UI.

-------------------------------------------------------------------------

**CODE:** ****

**OUTPUT:**

****

**OBSERVATION:**

1. **Architecture and Design:**
   * Uses FastAPI framework for building a healthcare analysis API
   * Well-structured with clear endpoint definitions and documentation
   * Implements Pydantic models for request/response validation
   * Includes Swagger documentation through FastAPI's automatic docs generation
2. **Security Considerations:**
   * OpenAI API key is hardcoded (should bemoved to environment variables)
   * Basic error handling is implemented but could be enhanced
   * No authentication/authorization mechanisms implemented yet

**Endpoints:**

/analyze-text: Processes clinical notes using GPT-3.5-turbo

/analyze-image: Performs OCR on medical images

/: Basic health check endpoint

Features:

Clinical notes analysis using OpenAI's GPT model

Image processing pipeline using OpenCV

OCR capabilities using pytesseract

Response models with typed attributes

**Technical Implementation:**

Uses async/await for better performance

Implements proper exception handling

Includes image preprocessing steps (denoising, grayscale conversion)

Uses type hints for better code clarity

**CONCLUSION:**

**Areas for Improvement:**

* + Add proper authentication and authorization
  + Implement rate limiting
  + Add input validation for image files
  + Move configuration to environment variables
  + Add logging mechanisms
  + Implement more robust parsing of GPT responses
  + Add database integration for storing analysis results
  + **Dependencies:**
  + FastAPI
  + OpenAI
  + pytesseract
  + PIL (Python Imaging Library)
  + OpenCV (cv2)
  + numpy

**Extensibility**:

* + Code structure allows for easy addition of new endpoints
  + Comments indicate potential future features
  + Modular design makes it easy to add new functionality

**THE END**

**-------------------------------------------------------------------------**