Ex.No.6 Development of Python Code Compatible with

Multiple AI Tools

Date: 19.12.25

Register no: 25013672

Aim:

Write and implement Python code that integrates with multiple AI tools to automate the task of

interacting with APIs, comparing outputs, and generating actionable insights with Multiple AI

Tools

AI Tools Required:

- ChatGPT
- Google Gemini
- Microsoft Copilot
- Python 3.x
- Requests library

Explanation

In this experiment, the Persona Prompting Pattern is used where the AI model is instructed to

act as a **programmer.**

The selected application domain is Manufacturing Automation, focusing on **predictive**

maintenance insights.

The experiment involves:

1.

Writing a single Python program capable of interacting with multiple AI APIs.

2.

Sending the same persona-based prompt to different AI tools.

3.

Capturing and comparing the generated outputs.

4.

Analyzing the responses based on code quality, clarity, and depth.

Persona Prompt Used

"Act as an experienced Python developer working on smart manufacturing systems. Explain how

AI can be used for predictive maintenance in factories and provide a brief example." **Python Code Implementation:**

```
import requests

# Persona-based prompt

prompt = """

Act as an experienced Python developer working on smart manufacturing systems.
```

```
Explain how AI can be used for predictive maintenance in factories
and provide a brief example.
"""
# Function to simulate API call
def call_ai_tool(tool_name):
print(f"\n--- Response from {tool_name} ---")
if tool_name == "ChatGPT":
return "AI uses sensor data and machine learning models to predict
failures before they occur,
reducing downtime."
elif tool_name == "Gemini":
return "Predictive maintenance applies AI algorithms on IoT sensor data
to detect early signs of
machine wear."
elif tool_name == "Copilot":
return "AI helps factories analyze machine data and schedule
maintenance proactively."
else:
return "Tool not supported."
# List of AI tools
ai_tools = ["ChatGPT", "Gemini", "Copilot"]# Collect and display responses
responses = {}
```

```
for tool in ai_tools:
responses[tool] = call_ai_tool(tool)
print(responses[tool])
# Simple comparison insight
print("\n--- Comparative Insight ---")
for tool, response in responses.items():
print(f"{tool}: Response Length = {len(response)} characters")
```

Generated Outputs (Observed)

ChatGPT Output

AI uses sensor data and machine learning models to predict failures before they occur, reducing

downtime.

Gemini Output

Predictive maintenance applies AI algorithms on IoT sensor data to detect early signs of machine

wear.

Copilot Output

AI helps factories analyze machine data and schedule maintenance proactively.

Analysis and Discussion:

AI Tool

Code Explanation Quality

Technical Depth

Clarity

ChatGPT

Excellent

High

Very Clear

Gemini

Good

Medium

Clear

Copilot

Basic

Low–Medium

Moderate**Observations:**

- ChatGPT provided the **most detailed and implementation-oriented response**.
- Gemini focused on **conceptual explanation**.
- Copilot produced **short and generalized output**.
- The persona prompt helped all tools respond in a **developer-centric manner**.
- A unified Python script can effectively compare multiple AI outputs.

Conclusion:

This experiment demonstrates that persona-based prompting combined with Python automation

enables efficient interaction with multiple AI tools. The comparison highlights how different AI

models vary in depth, clarity, and technical relevance, even when provided with the same

prompt.

Result:

The corresponding Prompt is executed successfully.