

Lab 18– API Integration: Connecting to external services with error handling

Lab Objective

The objective of this lab is to provide students with hands-on experience

in integrating external APIs into Python applications using AI-assisted coding tools. Students will:

1. Understand the fundamentals of API requests, responses, and authentication mechanisms.
2. Learn to use AI-assisted coding to generate and optimize scripts for fetching, parsing, and handling API data.
3. Practice error handling strategies to manage common issues such as invalid responses, timeouts, and missing API keys.
4. Develop the ability to design robust and reusable API integration pipelines, balancing automation through AI tools with human judgment and debugging.

Lab Question 1: Weather Forecasting API

A travel company wants to show real-time weather updates for its customers. You are given access to a public weather API that requires an

API key and provides weather data in JSON format.

- Task 1: Use AI-assisted coding to write a script that fetches the current temperature and weather description for a given city. The script should handle errors if the API key is invalid or missing.
- Task 2: Extend the script to save the weather data into a local CSV file, ensuring that duplicate entries are avoided. Implement error handling for file I/O exceptions

```
import csv
from datetime import datetime

# Simulated API response (pretend this came from an online service)
def get_weather(city):
    api_response = {
        "Hyderabad": {"temp": 30.5, "description": "Clear Sky"},
        "Mumbai": {"temp": 32.1, "description": "Humid and Cloudy"},
        "Delhi": {"temp": 28.9, "description": "Sunny"},
    }
    # simulate an API returning data for that city
    if city not in api_response:
        raise ValueError("City not found or invalid API key")
    return api_response[city]

def save_to_csv(city, temp, desc):
    filename = "weather_data.csv"
    try:
        with open(filename, "a+", newline="", encoding="utf-8") as f:
            f.seek(0)
            existing = f.read()
            if f"{city},{temp},{desc}" in existing:
                print("Duplicate entry - skipping save.")
                return
            writer = csv.writer(f)
            writer.writerow([datetime.now().strftime("%Y-%m-%d %H:%M:%S"), city, temp, desc])
        print("Weather data saved successfully!")
    except Exception as e:
        print("Error saving data:", e)
```

```
def main():
    city = input("Enter city name: ").strip().title()
    try:
        data = get_weather(city)
        temp, desc = data["temp"], data["description"]
        print(f"Temperature in {city}: {temp}°C | Weather: {desc}")
        save_to_csv(city, temp, desc)
    except ValueError as e:
        print("Error:", e)
    except Exception as e:
        print("Unexpected error:", e)

if __name__ == "__main__":
    main()
```

```
| Enter city name: hyderabad
| Temperature in Hyderabad: 30.5°C | Weather: Clear Sky
| Weather data saved successfully!
```

The screenshot shows a Jupyter Notebook environment. On the left, a sidebar titled "Files" lists a directory structure with "sample_data" and "weather_data.csv". The main area contains a Python script:

```

[1] ✓ 5s
def main():
    city = input("Enter city name: ").strip().title()
    try:
        data = get_weather(city)
        temp, desc = data["temp"], data["description"]
        print(f"Temperature in {city}: {temp}°C | Weather: {desc}")
        save_to_csv(city, temp, desc)
    except ValueError as e:
        print("Error:", e)
    except Exception as e:
        print("Unexpected error:", e)

if __name__ == "__main__":
    main()

```

Below the code, a terminal window shows the output of running the script with "hyderabad" as input, displaying the temperature and weather description, and confirming the data was saved successfully.

On the right, a screenshot of an Excel spreadsheet shows a single row of data:

A	B	C	D	E	F	G
#####	Hyderabad	30.5	Clear Sky			
1						
2						
3						
4						
5						
7						
8						
9						

Lab Question 2: Currency Exchange Rate API

A financial startup needs a tool to convert amounts between currencies

using an exchange rate API. However, the API occasionally fails due to server downtime.

- Task 1: Write a script (with AI assistance) that takes user input (amount, source currency, target currency) and fetches the latest exchange rate from the API. Include errors in handling invalid currency codes.
- Task 2: Add logic to retry the request up to three times if the API

call fails due to network or server issues and log all failed attempts into a local error log file.

Prompt: Generate a Python script that simulates fetching an exchange rate from an API for currency conversion. Take user input for amount, source, and target currencies. Include retry logic up to 3 times if a simulated network error occurs. Log failed attempts to `error_log.txt`

```
▶ import time
  import random

def get_exchange_rate(source, target):
    rates = {
        "USD": {"INR": 83.12, "EUR": 0.92},
        "INR": {"USD": 0.012, "EUR": 0.011},
        "EUR": {"USD": 1.09, "INR": 90.8},
    }

    if random.random() < 0.3:
        raise ConnectionError("Network/server error")

    if source not in rates or target not in rates[source]:
        raise ValueError("Invalid currency code")

    return rates[source][target]

def main():
    amount = float(input("Enter amount: "))
    source = input("From currency: ").upper()
    target = input("To currency: ").upper()

    for attempt in range(3):
        try:
            rate = get_exchange_rate(source, target)
            print(f"{amount} {source} = {amount * rate:.2f} {target}")
            break
        except ConnectionError as e:
            print(f"Attempt {attempt+1}: {e}")
            time.sleep(1)
        except ValueError as e:
            print("Error:", e)
            break
    else:
        with open("error_log.txt", "a") as f:
            f.write("Failed to get rate after 3 attempts\n")
            print("All attempts failed. Logged to error_log.txt")

if __name__ == "__main__":
    main()
```

```
→ Enter amount: 5000
  From currency: inr
  To currency: usd
  5000.0 INR = 60.00 USD
```

Lab Question 3: News Headlines API

A news aggregator wants to display the latest technology news headlines

using a news API. Sometimes, the API responds slowly or returns incomplete data.

- Task 1: Use AI-assisted coding to fetch the top 5 technology headlines and print them neatly in the console. Implement error handling for timeout errors by setting a maximum request time.

Task 2: Clean and preprocess the headlines by removing special characters and converting text to title case. Handle the scenario where the API response contains empty or null values.

Prompt: Generate a Python program that simulates fetching the top 5 technology news headlines from an API.

Clean the text by removing special characters and converting it to title case.

Add timeout error handling

```
▶ import re
import random
import time

def get_headlines():
    headlines = [
        "AI Breakthrough: New Model Beats Records!",
        "Cyber Security Threats Increasing Worldwide",
        "Tech Giants Announce Cloud Partnerships",
        "Quantum Computing Sets New Milestone",
        "Startups Raise Billions In 2025 Funding Round",
        None
    ]

    if random.random() < 0.2:
        time.sleep(2)
        raise TimeoutError("Request timed out")
    return headlines

def clean_text(text):
    if not text:
        return None
    cleaned = re.sub(r'^[^\w\s]', '', text)
    return cleaned.title().strip()

def main():
    try:
        data = get_headlines()
        print("\nTop 5 Technology Headlines:\n")
        count = 0
```

```

        return None
cleaned = re.sub(r'[^A-Za-z0-9\s]', '', text)
return cleaned.title().strip()

def main():
    try:
        data = get_headlines()
        print("\nTop 5 Technology Headlines:\n")
        count = 0
        for h in data:
            cleaned = clean_text(h)
            if cleaned:
                count += 1
                print(f"{count}. {cleaned}")
            if count == 5:
                break
    except TimeoutError as e:
        print("Error:", e)

if __name__ == "__main__":
    main()

```

Error: Request timed out

Output after waiting for sometime and running:



Top 5 Technology Headlines:

1. Ai Breakthrough New Model Beats Records
2. Cyber Security Threats Increasing Worldwide
3. Tech Giants Announce Cloud Partnerships
4. Quantum Computing Sets New Milestone
5. Startups Raise Billions In 2025 Funding Round