AI ASSISTED CODING

LAB 18– *API Integration: Connecting to external services with error handling.*

Roll no: 2503A51L17

Name: Simra Tahseen

Batch: 25BTCAICSB19

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.

Code Generated:

task1-18.py

Assignment-18 > task1-18.py > ...

```python
"""Fetch current temperature and weather description for a given city.

This script is written for OpenWeatherMap's current weather API (https://openweathermap.org/current).
It expects an API key and a city name. The API key may be provided via the
WEATHER_API_KEY environment variable or the --api-key command-line option.

Example:
    python task1.1-18.py --city "London" --api-key YOUR_KEY_HERE

If the API key is missing or invalid the script will print a helpful error.
If you use a different weather provider, adjust the URL and JSON parsing in
fetch_weather().
"""
from __future__ import annotations
import argparse
import os
import sys
from typing import Tuple, Any

def fetch_weather_openweathermap(city: str, api_key: str, units: str = "metric", timeout: int = 10) -> Tuple[float, str, Any]:
    """Call OpenWeatherMap current weather API and return (temp, description, raw_json).
    Raises ValueError with a helpful message for known error conditions.
    """
    try:
        import requests
    except ImportError as e:
        raise RuntimeError("The 'requests' package is required. Install it with: pip install requests") from e

    url = "https://api.openweathermap.org/data/2.5/weather"
    params = {"q": city, "appid": api_key, "units": units}

    try:
        resp = requests.get(url, params=params, timeout=timeout)
    except requests.exceptions.Timeout:
        raise ValueError("Request timed out contacting the weather API")
    except requests.exceptions.RequestException as e:
        raise ValueError(f"Network error when contacting weather API: {e}")

    # Handle authentication/invalid key
    if resp.status_code == 401:
        # Try to extract message from JSON
        try:
            body = resp.json()
            msg = body.get("message") or body
        except Exception:
            msg = resp.text
        raise ValueError(f"Unauthorized: invalid API key or access denied. API message: {msg}")
    # Other non-success codes
    if resp.status_code != 200:
        try:
            body = resp.json()
            msg = body.get("message") or body
        except Exception:
            msg = resp.text
        raise ValueError(f"Weather API returned HTTP {resp.status_code}: {msg}")

    try:
        data = resp.json()
    except Exception as e:
        raise ValueError(f"Failed to decode JSON response: {e}") from e

    # Parse temperature and weather description (OpenWeatherMap structure)
    try:
        temp = data.get("main", {}).get("temp")
        weather_list = data.get("weather") or []
        description = weather_list[0].get("description") if weather_list else "(no description)"
    except Exception as e:
        raise ValueError(f"Unexpected JSON structure from the weather API: {e}") from e

    if temp is None:
        raise ValueError("Temperature value not found in API response")

    return float(temp), str(description), data

def parse_args() -> argparse.Namespace:
    p = argparse.ArgumentParser(description="Fetch current temperature and weather description for a city")
    p.add_argument("--city", "-c", required=False, help="City name (e.g. London,UK or Tokyo)")
    p.add_argument("--api-key", "-k", required=False, help="Weather API key (overrides WEATHER_API_KEY env var)")
    p.add_argument("--units", "-u", choices=("metric", "imperial", "standard"), default="metric", help="Units to request from API")
    return p.parse_args()

def main() -> int:
    args = parse_args()

    city = args.city or os.environ.get("CITY")
    api_key = args.api_key or os.environ.get("WEATHER_API_KEY")

    if not city:
        print("Error: No city provided. Pass --city or set CITY environment variable.")
        return 2

    if not api_key:
        print("Error: No API key provided. Set WEATHER_API_KEY env var or pass --api-key.")
        return 3

    try:
        temp, description, raw = fetch_weather_openweathermap(city, api_key, units=args.units)
    except RuntimeError as e:
        print(f"Environment error: {e}")
        return 4
    except ValueError as e:
        print(f"API error: {e}")
        return 5

    # Print a concise, human-friendly line
    unit_symbol = "°C" if args.units == "metric" else ("°F" if args.units == "imperial" else "K")
    print(f"Weather for {city}: {temp}{unit_symbol}, {description}")

    # Optionally return raw JSON for downstream processing (print as one-line JSON)
    # Uncomment to see the full API payload:
    # import json; print(json.dumps(raw))

    return 0

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

Output:

```
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> $env:WEATHER_API_KEY = 'd44aecee9991b2fd1f45d39f7a74db24'
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> $env:CITY = 'New York,US'
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> python "Assignment-18\task1-18.py"
Weather for New York,US: 10.83°C, light intensity drizzle
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding>
```

- **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.

Code Generated:

```python
def save_weather_to_csv(city: str, temp: float, description: str, raw: Any, api_unix_dt: int | None = None, *, filename: str = "weather_history.csv") -> bool:
    """Append the weather record to a CSV file if not already present.

    Duplicate avoidance: if the file already contains a row with the same (city, dt)
    pair (where dt is the API-provided unix timestamp if available), the row is
    considered duplicate and will not be written.

    Returns True if a new row was written, False if skipped as duplicate.
    Raises OSError on file I/O errors so callers can handle/report them.
    """
    path = Path(filename)

    # Normalize city for comparison
    city_norm = city.strip()

    # Determine canonical timestamp to store
    if api_unix_dt is not None:
        dt_unix = int(api_unix_dt)
        dt_iso = datetime.fromtimestamp(dt_unix, tz=timezone.utc).isoformat()
    else:
        # fallback to current time
        dt_unix = int(datetime.now(tz=timezone.utc).timestamp())
        dt_iso = datetime.fromtimestamp(dt_unix, tz=timezone.utc).isoformat()

    # Read existing file to detect duplicates
    existing_keys: set[tuple[str, str]] = set()
    if path.exists():
        try:
            with path.open("r", encoding="utf-8", newline="") as fh:
                reader = csv.DictReader(fh)
                for row in reader:
                    k = (row.get("city", "").strip(), row.get("dt", ""))
                    existing_keys.add(k)
        except OSError:
            # Reraise to allow caller to decide how to handle I/O problems
            raise

    key = (city_norm, str(dt_unix))
    if key in existing_keys:
        return False

    # Prepare row and write (create file with header if missing)
    fieldnames = ["city", "dt", "datetime_iso", "temp", "description", "raw_json"]
    row = {
        "city": city_norm,
        "dt": str(dt_unix),
        "datetime_iso": dt_iso
```

Output:

```
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> $env:WEATHER_API_KEY = 'd44aecee9991b2fd1f45d39f7a74db24'
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> $env:CITY = 'New York,US'
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> python "Assignment-18\task1-18.py"
Weather for New York,US: 10.99°C, light intensity drizzle
Saved weather to weather_history.csv
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> $env:WEATHER_API_KEY = 'd44aecee9991b2fd1f45d39f7a74db24'
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> $env:CITY = 'London'
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> python "Assignment-18\task1-18.py"
Weather for London: 7.57°C, scattered clouds
Saved weather to weather_history.csv
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding>
```

task1-18.py | weather_history.csv

Assignment-18 > weather_history.csv

```
1  city,dt,datetime_iso,temp,description,raw_json
2  "New York,US",1761814233,2025-10-30T08:50:33+00:00,10.99,light intensity drizzle,"{""coord"": {""lon"": -74.006, ""lat"": 40.7143}, ""weather"": [{""id"": 300, ""main"": ""Drizzle"", ""descrip
3  London,1761814264,2025-10-30T08:51:04+00:00,7.57,scattered clouds,"{""coord"": {""lon"": -0.1257, ""lat"": 51.5085}, ""weather"": [{""id"": 802, ""main"": ""Clouds"", ""description"": ""scatte
4
```

Observation: The program successfully retrieved real-time weather details, including temperature and conditions, for the specified city using the Weather API. It handled invalid or missing API keys effectively and saved the data into a local CSV file while avoiding duplicates and managing file I/O errors smoothly.

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.

Code Generated:

task2-18.py

```python
import requests
import time
import logging

LOG_FILE = "error_log.txt"

logging.basicConfig(
    filename=LOG_FILE,
    level=logging.ERROR,
    format="%(asctime)s - %(levelname)s - %(message)s"
)

def fetch_exchange_rate(source_currency, target_currency, retries=3, delay=2):
    for attempt in range(1, retries + 1):
        try:
            url = f"https://open.er-api.com/v6/latest/{source_currency.upper()}"
            response = requests.get(url, timeout=5)
            response.raise_for_status()
            data = response.json()

            # Check valid result
            if data["result"] != "success":
                raise ValueError(data.get("error-type", "Invalid API response"))

            rate = data["rates"].get(target_currency.upper())
            if rate is None:
                raise ValueError("Invalid target currency code.")
            return rate

        except (requests.exceptions.RequestException, ValueError) as e:
            logging.error(f"Attempt {attempt} failed: {e}")
            print(f"⚠️ Attempt {attempt} failed: {e}")
            if attempt < retries:
                print(f"Retrying in {delay} seconds...")
                time.sleep(delay)
            else:
                print("❌ All attempts failed. Please check log file for details.")
                return None

def main():
    print("💱 Currency Converter Tool (Public API)")
    print("=====================================")

    try:
        amount = float(input("Enter amount: "))
        source_currency = input("Enter source currency (e.g. USD): ").upper().strip()
        target_currency = input("Enter target currency (e.g. INR): ").upper().strip()

        rate = fetch_exchange_rate(source_currency, target_currency)
        if rate:
            converted = amount * rate
            print(f"\n✅ {amount:.2f} {source_currency} = {converted:.2f} {target_currency}")
        else:
            print("\n⚠️ Conversion failed. Check log file for details.")

    except ValueError:
        print("❌ Invalid input. Please enter numeric amount only.")

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

Output:

```
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> & C:/Us
g/Assignment-18/task2-18.py"
💱 Currency Converter Tool (Public API)
=====================================
Enter amount: 100
Enter source currency (e.g. USD): USD
Enter target currency (e.g. INR): INR

✅ 100.00 USD = 8873.42 INR
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> & C:/Us
g/Assignment-18/task2-18.py"
💱 Currency Converter Tool (Public API)
=====================================
Enter amount: 10
Enter source currency (e.g. USD):
Enter target currency (e.g. INR):
⚠️Attempt 1 failed: 404 Client Error: Not Found for url: https://open.er-api.com/v6/latest/
Retrying in 2 seconds...
⚠️Attempt 2 failed: 404 Client Error: Not Found for url: https://open.er-api.com/v6/latest/
Retrying in 2 seconds...
⚠️Attempt 3 failed: 404 Client Error: Not Found for url: https://open.er-api.com/v6/latest/
❌ All attempts failed. Please check log file for details.

⚠️Conversion failed. Check log file for details.
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding>
```

Observation: The script efficiently converted currencies by fetching real-time exchange rates from the API while validating user inputs for incorrect codes. It also implemented a retry mechanism to handle network or server issues and logged failed attempts, ensuring reliable performance and smooth execution.

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.

Code Generated:

task3-18.py

Assignment-18 > task3-18.py > fetch_headlines

```python
import requests
import re

# --- Configuration ---
API_KEY = "d8e9a7ca21c6445b822fd3bf8c5c4006"  # Replace with your valid NewsAPI key
URL = f"https://newsapi.org/v2/top-headlines?category=technology&language=en&pageSize=5&apiKey={API_KEY}"

def fetch_headlines():
    """Fetch top 5 technology headlines with timeout and error handling."""
    try:
        response = requests.get(URL, timeout=5)  # ⏰ timeout after 5 seconds
        response.raise_for_status()  # raise HTTPError for bad responses
        data = response.json()

        # Ensure articles exist
        if "articles" not in data or not data["articles"]:
            print("⚠️ No articles found in the API response.")
            return []

        # Extract headlines
        headlines = [article.get("title", "") for article in data["articles"]]
        return headlines

    except requests.exceptions.Timeout:
        print("❌ Request timed out. Please try again later.")
        return []
    except requests.exceptions.RequestException as e:
        print(f"❌ Error fetching data: {e}")
        return []

def clean_headlines(headlines):
    """Clean and preprocess headlines."""
    cleaned = []
    for title in headlines:
        if not title or title.strip() == "":
            continue  # skip empty/null titles
        # Remove special characters, keep letters/numbers/spaces
        title = re.sub(r'[^A-Za-z0-9 ]+', '', title)
        # Convert to Title Case
        title = title.title()
        cleaned.append(title)
    return cleaned

```

```python
# --- Main Execution ---
raw_headlines = fetch_headlines()
processed_headlines = clean_headlines(raw_headlines)

# --- Display neatly ---
if processed_headlines:
    print("\n📰 Top 5 Technology Headlines:\n")
    for i, headline in enumerate(processed_headlines, 1):
        print(f"{i}. {headline}")
else:
    print("⚠️ No valid headlines to display.")

```

Output:

```
ustom Office Templates/Desktop/AIAssistedCoding/Assignment-18/task3-18.py"

📰 Top 5 Technology Headlines:

1. 18 People Whose Lives Got Completely And Totally Blown Up By The Consequences Of Their Unfortunate Decisions Last Week  Buzzfeedcom
2. Ikea Just Took Over Your Smart Home  The Verge
3. 12 Apps I Urge Nontechies To Install On Their Android Phones  Android Authority
4. Hyrule Warriors Age Of Imprisonment Voice Actors Officially Revealed  Nintendo Life
5. Last Day  Best Buy Week 2 Black Friday Doorbusters Galaxy S25 Ultra 400 Off Samsung Tv 1100 Off More  9To5Toys
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding>
```

Observation: The program successfully fetched and displayed the latest technology headlines from the News API with proper timeout and error handling. It also cleaned and formatted the headlines by removing special characters, converting text to title case, and ignoring null entries for a neat and readable output.