

AI ASSISTED CODING

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

Roll no: 2503A51L41

Name: Gundeti Hasini

Batch: 24BTCAICSB20

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 > task1-18py > ...
1  """Fetch current temperature and weather description for a given city.
2
3 This script is written for OpenWeatherMap's current weather API (https://openweathermap.org/current).
4 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.
5
6 Example:
7   python task1-18.py --city "London" --api-key YOUR_KEY_HERE
8
9 If the API key is missing or invalid the script will print a helpful error.
10 If you use a different weather provider, adjust the URL and JSON parsing in
fetch_weather().
11 """
12
13 from __future__ import annotations
14 import argparse
15 import os
16 import sys
17 from typing import Tuple, Any
18
19 def fetch_weather_openweathermap(city: str, api_key: str, units: str = "metric", timeout: int = 10) -> Tuple[float, str, Any]:
20     """Call OpenWeatherMap current weather API and return (temp, description, raw_json).
21     Raises ValueError with a helpful message for known error conditions.
22     """
23     try:
24         import requests
25     except ImportError as e:
26         raise RuntimeError(f"The 'requests' package is required. Install it with: pip install requests") from e
27
28     url = "https://api.openweathermap.org/data/2.5/weather"
29     params = {"q": city, "appid": api_key, "units": units}
30
31     try:
32         resp = requests.get(url, params=params, timeout=timeout)
33     except requests.exceptions.Timeout:
34         raise ValueError("Request timed out contacting the weather API")
35     except requests.exceptions.RequestException as e:
36         raise ValueError(f"Network error when contacting weather API: {e}")
37
38     # Handle authentication/invalid key
39     if resp.status_code == 401:
40         # Try to extract message from JSON
41         try:
42             body = resp.json()
43             msg = body.get("message") or body
44         except Exception:
45             msg = resp.text
46         raise ValueError(f"Unauthorized: invalid API key or access denied. API message: {msg}")
47
48
49     # Other non-success codes
50     if resp.status_code != 200:
51         try:
52             body = resp.json()
53             msg = body.get("message") or body
54         except Exception:
55             msg = resp.text
56         raise ValueError(f"Weather API returned HTTP {resp.status_code}: {msg}")
57
58     try:
59         data = resp.json()
60     except Exception as e:
61         raise ValueError(f"Failed to decode JSON response: {e}") from e
62
63     # Parse temperature and weather description (OpenWeatherMap structure)
64     try:
65         temp = data.get("main", {}).get("temp")
66         weather_list = data.get("weather") or []
67         description = weather_list[0].get("description") if weather_list else "(no description)"
68     except Exception as e:
69         raise ValueError(f"Unexpected JSON structure from the weather API: {e}") from e
70
71     if temp is None:
72         raise ValueError("Temperature value not found in API response")
73
74     return float(temp), str(description), data
75
76 def parse_args() -> argparse.Namespace:
77     p = argparse.ArgumentParser(description="Fetch current temperature and weather description for a city")
78     p.add_argument("--city", "-c", required=False, help="City name (e.g. London/UK or Tokyo)")
79     p.add_argument("--api-key", "-k", required=False, help="Weather API key (overrides WEATHER_API_KEY env var)")
80     p.add_argument("--units", "-u", choices=("metric", "imperial", "standard"), default="metric", help="Units to request from API")
81     return p.parse_args()
82
83 def main() -> int:
84     args = parse_args()
85
86     city = args.city or os.environ.get("CITY")
87     api_key = args.api_key or os.environ.get("WEATHER_API_KEY")
88
89     if not city:
90         print("Error: No city provided. Pass --city or set CITY environment variable.")
91         return 2
92
93     if not api_key:
94         print("Error: No API key provided. Set WEATHER_API_KEY env var or pass --api-key.")
95         return 3
96
97     try:
98         temp, description, raw = fetch_weather_openweathermap(city, api_key, units=args.units)
99     except RuntimeError as e:
100         print(f"Environment error: {e}")
101         return 4
102     except ValueError as e:
103         print(f"API error: {e}")
104         return 5
105
106     # Print a concise, human-friendly line
107     unit_symbol = "°C" if args.units == "metric" else ("°F" if args.units == "imperial" else "K")
108     print(f"weather for {city}: ({temp}){unit_symbol}, {description}")
109
110     # Optionally return raw JSON for downstream processing (print as one-line JSON)
111     # Uncomment to see the full API payload:
112     # import json; print(json.dumps(raw))
113
114     return 0
115
116 if __name__ == "__main__":
117     raise SystemExit(main())
118

```

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:

```
81 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:
82     """Append the weather record to a CSV file if not already present.
83
84     Duplicate avoidance: if the file already contains a row with the same (city, dt)
85     pair (where dt is the API-provided unix timestamp if available), the row is
86     considered duplicate and will not be written.
87
88     Returns True if a new row was written, False if skipped as duplicate.
89     Raises OSError on file I/O errors so callers can handle/report them.
90     """
91
92     path = Path(filename)
93
94     # Normalize city for comparison
95     city_norm = city.strip()
96
97     # Determine canonical timestamp to store
98     if api_unix_dt is not None:
99         dt_unix = int(api_unix_dt)
100        dt_iso = datetime.fromtimestamp(dt_unix, tz=timezone.utc).isoformat()
101    else:
102        # Fallback to current time
103        dt_unix = int(datetime.now(tz=timezone.utc).timestamp())
104        dt_iso = datetime.fromtimestamp(dt_unix, tz=timezone.utc).isoformat()
105
106    # Read existing file to detect duplicates
107    existing_keys: set[tuple[str, str]] = set()
108    if path.exists():
109        try:
110            with path.open("", encoding="utf-8", newline="") as fh:
111                reader = csv.DictReader(fh)
112                for row in reader:
113                    k = (row.get("city", "").strip(), row.get("dt", ""))
114                    existing_keys.add(k)
115        except OSError:
116            # Reraise to allow caller to decide how to handle I/O problems
117            raise
118
119    key = (city_norm, str(dt_unix))
120    if key in existing_keys:
121        return False
122
123    # Prepare row and write (create file with header if missing)
124    fieldnames = ["city", "dt", "datetime_iso", "temp", "description", "raw_json"]
125    row = [
126        "city": city_norm,
127        "dt": str(dt_unix),
128        "datetime_iso": dt_iso
129    ]
130
```

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 X
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 X
Assignment-18 > task2-18.py ...
1 import requests
2 import time
3 import logging
4
5 LOG_FILE = "error_log.txt"
6
7 logging.basicConfig(
8     filename=LOG_FILE,
9     level=logging.ERROR,
10    format="%(asctime)s - %(levelname)s - %(message)s"
11 )
12
13 def fetch_exchange_rate(source_currency, target_currency, retries=3, delay=2):
14     for attempt in range(1, retries + 1):
15         try:
16             url = f"https://open.er-api.com/v6/latest/{source_currency.upper()}"
17             response = requests.get(url, timeout=5)
18             response.raise_for_status()
19             data = response.json()
20
21             # Check valid result
22             if data["result"] != "success":
23                 raise ValueError(data.get("error-type", "Invalid API response"))
24
25             rate = data["rates"].get(target_currency.upper())
26             if rate is None:
27                 raise ValueError("Invalid target currency code.")
28             return rate
29
30         except (requests.exceptions.RequestException, ValueError) as e:
31             logging.error(f"Attempt {attempt} failed: ({e})")
32             print(f"\n⚠ Attempt {attempt} failed: ({e})")
33             if attempt < retries:
34                 print(f"\nRetrying in {delay} seconds...")
35                 time.sleep(delay)
36             else:
37                 print("\n❌ All attempts failed. Please check log file for details.")
38                 return None
39
40 def main():
41     print("Currency Converter Tool (Public API)")
42     print("====")
43
44     try:
45         amount = float(input("Enter amount: "))
46         source_currency = input("Enter source currency (e.g. USD): ").upper().strip()
47         target_currency = input("Enter target currency (e.g. INR): ").upper().strip()
48
49         rate = fetch_exchange_rate(source_currency, target_currency)
50         if rate:
51             converted = amount * rate
52             print(f"\n\n{amount:.2f} {source_currency} = {converted:.2f} {target_currency}")
53         else:
54             print("\n⚠ Conversion failed. Check log file for details.")
55
56     except ValueError:
57         print("\n❌ Invalid input. Please enter numeric amount only.")
58
59 if __name__ == "__main__":
60     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.

```
task3-18.py X
Assignment-18 > task3-18.py > fetch_headlines

1 import requests
2 import re
3
4 # --- Configuration ---
5 API_KEY = "d8e9a7ca21c6445b022fd8f8c5c4006" # Replace with your valid NewsAPI key
6 URL = f"https://newsapi.org/v2/top-headlines?category=technology&language=en&pageSize=5&apiKey={API_KEY}"
7
8 def fetch_headlines():
9     """Fetch top 5 technology headlines with timeout and error handling."""
10    try:
11        response = requests.get(URL, timeout=5) # ⚠ timeout after 5 seconds
12        response.raise_for_status() # raise HTTPError for bad responses
13        data = response.json()
14
15        # Ensure articles exist
16        if "articles" not in data or not data["articles"]:
17            print("⚠ No articles found in the API response.")
18            return []
19
20        # Extract headlines
21        headlines = [article.get("title", "") for article in data["articles"]]
22        return headlines
23
24    except requests.exceptions.Timeout:
25        print("⚠ Request timed out. Please try again later.")
26        return []
27    except requests.exceptions.RequestException as e:
28        print(f"⚠ Error fetching data: {e}")
29        return []
30
31 def clean_headlines(headlines):
32     """Clean and preprocess headlines."""
33     cleaned = []
34     for title in headlines:
35         if not title or title.strip() == "":
36             continue # skip empty/null titles
37         # Remove special characters, keep letters/numbers/spaces
38         title = re.sub("[^A-Za-z0-9 ]+", '', title)
39         # Convert to Title Case
40         title = title.title()
41         cleaned.append(title)
42     return cleaned
43
```

Code Generated:

```
44 # --- Main Execution ---
45 raw_headlines = fetch_headlines()
46 processed_headlines = clean_headlines(raw_headlines)
47
48 # --- Display neatly ---
49 if processed_headlines:
50     print("\n■ Top 5 Technology Headlines:\n")
51     for i, headline in enumerate(processed_headlines, 1):
52         print(f"{i}. {headline}")
53     else:
54         print("⚠ No valid headlines to display.")
55
```

Output:

```
ustom Office Templates/Desktop/AIAssistedCoding/Assignment-18/task3-18.py"
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\Desktop\AIAssistedCoding> []

[+] 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.