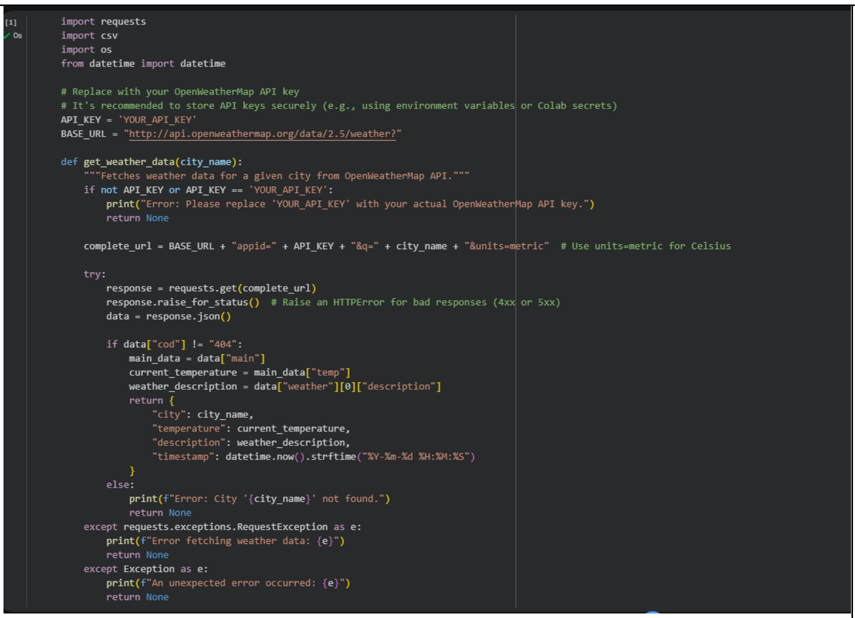
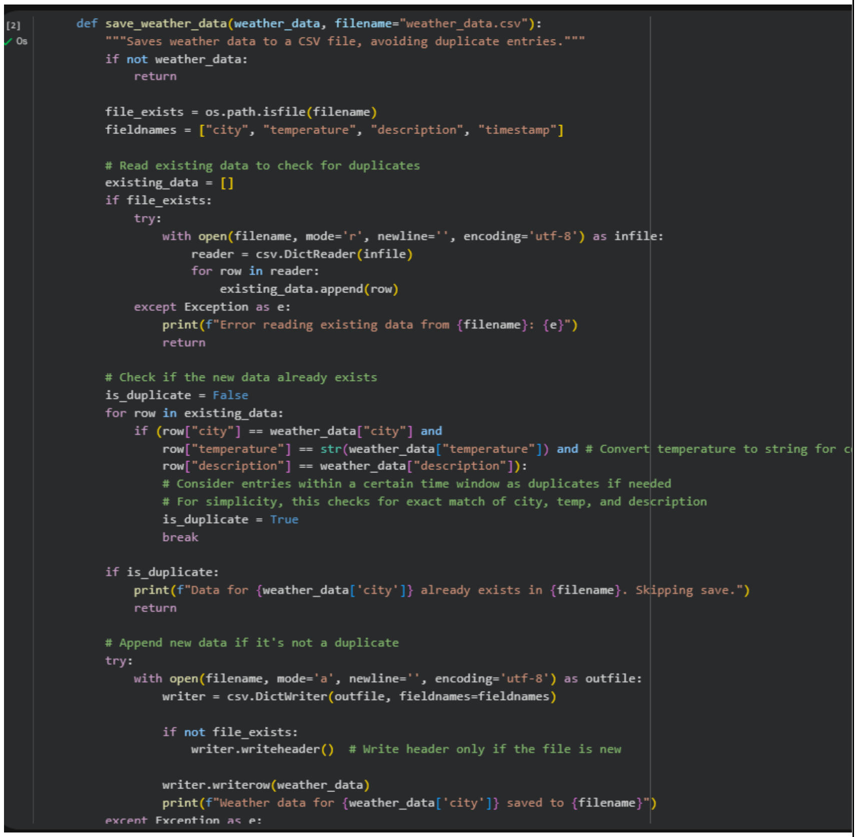
SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE				DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**		**Academic Year:**2025-2026	
Course Coordinator Name		Venkataramana Veeramsetty			
Instructor(s) Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)			
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding		
Year/Sem	II/I	**Regulation**	R24		
Date and Day of Assignment	Week5 - Monday	**Time(s)**			
Duration	2 Hours	**Applicable to Batches**			
Assignment Number: 18.5(Present assignment number)/**24**(Total number of assignments)					

Q.No.	**Question**	***Expected Time to complete***

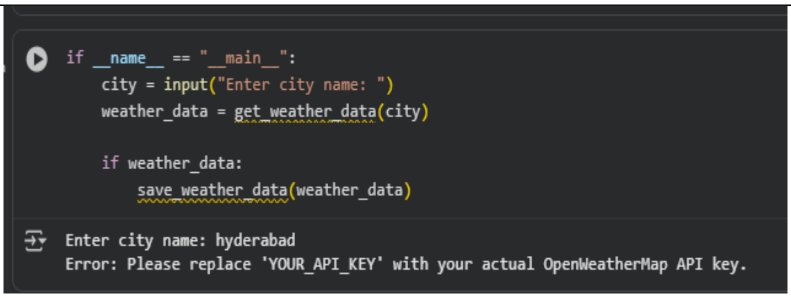
	Lab 18– API Integration: Connecting to external services with error handling **Lab Objective** The objective of this lab is to provide students with hands-on experience in integrating external APIs into Python applications using AI-assisted coding tools. Students will: 1. Understand the fundamentals 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.	
1	**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. **ANSWER:** **Prompt:** **"Write a Python script that uses a public weather API (like OpenWeatherMap) to fetch and display the current temperature and weather description for a given city. The script should: Handle errors for invalid or missing API keys. Save the weather data (city, temperature, description, date/time) into a local CSV file. Avoid duplicate entries when saving data. Include proper error handling for network and file I/O exceptions."**	Week 9 - Monday



```
[1]
✓ 0s
import requests
import csv
import os
from datetime import datetime

# Replace with your OpenWeatherMap API key
# It's recommended to store API keys securely (e.g., using environment variables or Colab secrets)
API_KEY = 'YOUR_API_KEY'
BASE_URL = "http://api.openweathermap.org/data/2.5/weather?"

def get_weather_data(city_name):
    """Fetches weather data for a given city from OpenWeatherMap API."""
    if not API_KEY or API_KEY == 'YOUR_API_KEY':
        print("Error: Please replace 'YOUR_API_KEY' with your actual OpenWeatherMap API key.")
        return None

    complete_url = BASE_URL + "appid=" + API_KEY + "&q=" + city_name + "&units=metric"  # Use units=metric for Celsius

    try:
        response = requests.get(complete_url)
        response.raise_for_status()  # Raise an HTTPError for bad responses (4xx or 5xx)
        data = response.json()

        if data["cod"] != "404":
            main_data = data["main"]
            current_temperature = main_data["temp"]
            weather_description = data["weather"][0]["description"]
            return {
                "city": city_name,
                "temperature": current_temperature,
                "description": weather_description,
                "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
            }
        else:
            print(f"Error: City '{city_name}' not found.")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Error fetching weather data: {e}")
        return None
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return None
```



```
[2]
✓ 0s
def save_weather_data(weather_data, filename="weather_data.csv"):
    """Saves weather data to a CSV file, avoiding duplicate entries."""
    if not weather_data:
        return

    file_exists = os.path.isfile(filename)
    fieldnames = ["city", "temperature", "description", "timestamp"]

    # Read existing data to check for duplicates
    existing_data = []
    if file_exists:
        try:
            with open(filename, mode='r', newline='', encoding='utf-8') as infile:
                reader = csv.DictReader(infile)
                for row in reader:
                    existing_data.append(row)
        except Exception as e:
            print(f"Error reading existing data from {filename}: {e}")
            return

    # Check if the new data already exists
    is_duplicate = False
    for row in existing_data:
        if (row["city"] == weather_data["city"] and
            row["temperature"] == str(weather_data["temperature"]) and # Convert temperature to string for c
            row["description"] == weather_data["description"]):
            # Consider entries within a certain time window as duplicates if needed
            # For simplicity, this checks for exact match of city, temp, and description
            is_duplicate = True
            break

    if is_duplicate:
        print(f"Data for {weather_data['city']} already exists in {filename}. Skipping save.")
        return

    # Append new data if it's not a duplicate
    try:
        with open(filename, mode='a', newline='', encoding='utf-8') as outfile:
            writer = csv.DictWriter(outfile, fieldnames=fieldnames)

            if not file_exists:
                writer.writeheader()  # Write header only if the file is new

            writer.writerow(weather_data)
            print(f"Weather data for {weather_data['city']} saved to {filename}")
    except Exception as e:
```



```
if __name__ == "__main__":
    city = input("Enter city name: ")
    weather_data = get_weather_data(city)

    if weather_data:
        save_weather_data(weather_data)
```

```
Enter city name: hyderabad
Error: Please replace 'YOUR_API_KEY' with your actual OpenWeatherMap API key.
```

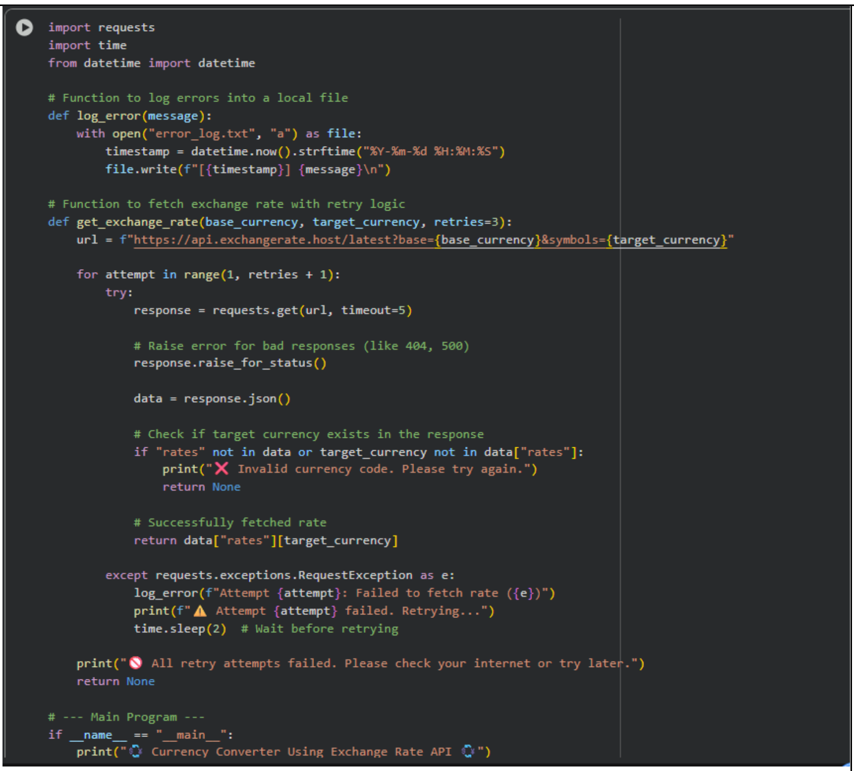
Lab Question 2: Currency Exchange Rate API

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.

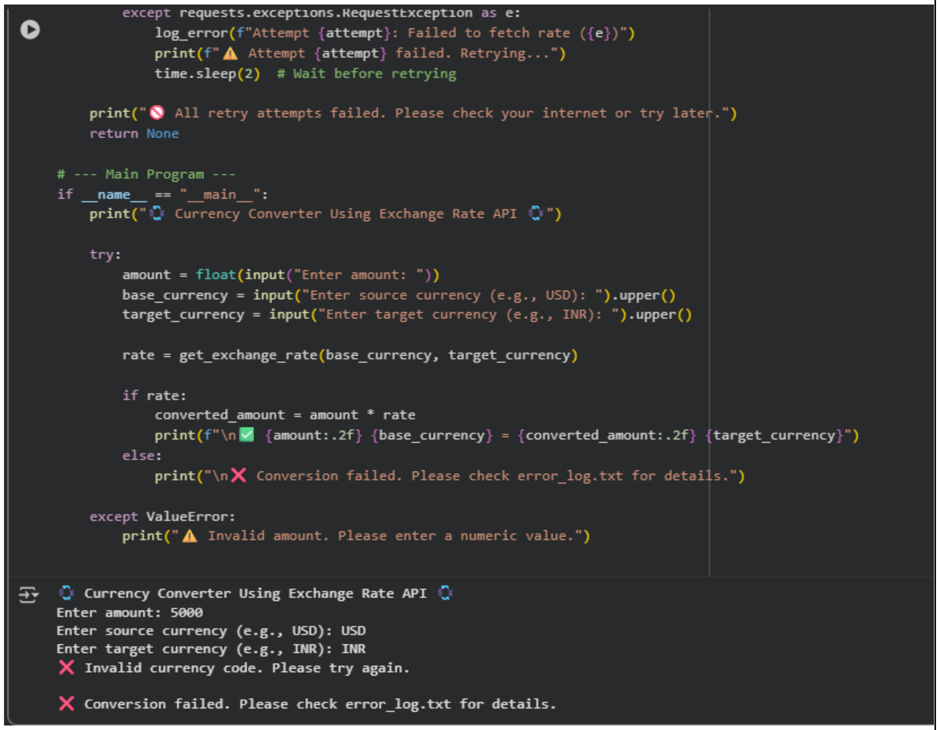
- **Task 1:** Write a script (with AI assistance) that takes user input (amount, source currency, target currency) and fetches the latest exchange rate from the API. Include errors in handling invalid currency codes.
- **Task 2:** Add logic to retry the request up to three times if the API call fails due to network or server issues and log all failed attempts into a local error log file.

  PROMPT: Write a Python script that converts an amount between two currencies using a public Exchange Rate API. The program should: Take user input (amount, source currency, target currency). Fetch the latest exchange rate from the API. Handle invalid currency codes with proper error messages. Retry the API request up to 3 times if it fails due to network or server errors. Log all failed attempts (with timestamps and error details) to a local file named error_log.txt. implement this cleanly with comments explaining each step.

  SCREENSHOTS:



```
import requests
import time
from datetime import datetime

# Function to log errors into a local file
def log_error(message):
    with open("error_log.txt", "a") as file:
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        file.write(f"[{timestamp}] {message}\n")

# Function to fetch exchange rate with retry logic
def get_exchange_rate(base_currency, target_currency, retries=3):
    url = f"https://api.exchangerate.host/latest?base={base_currency}&symbols={target_currency}"

    for attempt in range(1, retries + 1):
        try:
            response = requests.get(url, timeout=5)

            # Raise error for bad responses (like 404, 500)
            response.raise_for_status()

            data = response.json()

            # Check if target currency exists in the response
            if "rates" not in data or target_currency not in data["rates"]:
                print("❌ Invalid currency code. Please try again.")
                return None

            # Successfully fetched rate
            return data["rates"][target_currency]

        except requests.exceptions.RequestException as e:
            log_error(f"Attempt {attempt}: Failed to fetch rate ({e})")
            print(f"⚠ Attempt {attempt} failed. Retrying...")
            time.sleep(2)  # Wait before retrying

    print("🚫 All retry attempts failed. Please check your internet or try later.")
    return None

# --- Main Program ---
if __name__ == "__main__":
    print("🔄 Currency Converter Using Exchange Rate API 🔄")
```



```
        except requests.exceptions.RequestException as e:
            log_error(f"Attempt {attempt}: Failed to fetch rate ({e})")
            print(f"⚠ Attempt {attempt} failed. Retrying...")
            time.sleep(2)  # Wait before retrying

    print("🚫 All retry attempts failed. Please check your internet or try later.")
    return None

# --- Main Program ---
if __name__ == "__main__":
    print("🔄 Currency Converter Using Exchange Rate API 🔄")

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

        rate = get_exchange_rate(base_currency, target_currency)

        if rate:
            converted_amount = amount * rate
            print(f"\n✅ {amount:.2f} {base_currency} = {converted_amount:.2f} {target_currency}")
        else:
            print("\n❌ Conversion failed. Please check error_log.txt for details.")

    except ValueError:
        print("⚠ Invalid amount. Please enter a numeric value.")
```

```
🔄 Currency Converter Using Exchange Rate API 🔄
Enter amount: 5000
Enter source currency (e.g., USD): USD
Enter target currency (e.g., INR): INR
❌ Invalid currency code. Please try again.

❌ Conversion failed. Please check error_log.txt for details.
```

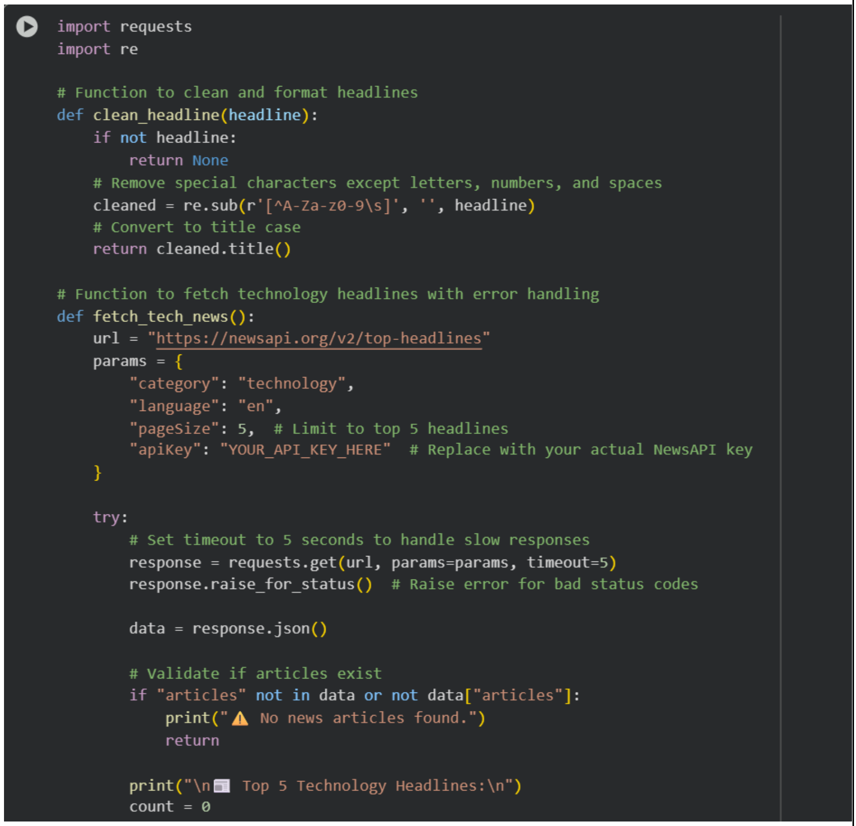
Lab Question 3: News Headlines API

A news aggregator wants to display the latest technology news headlines using a news API. Sometimes, the API responds slowly or returns incomplete data.

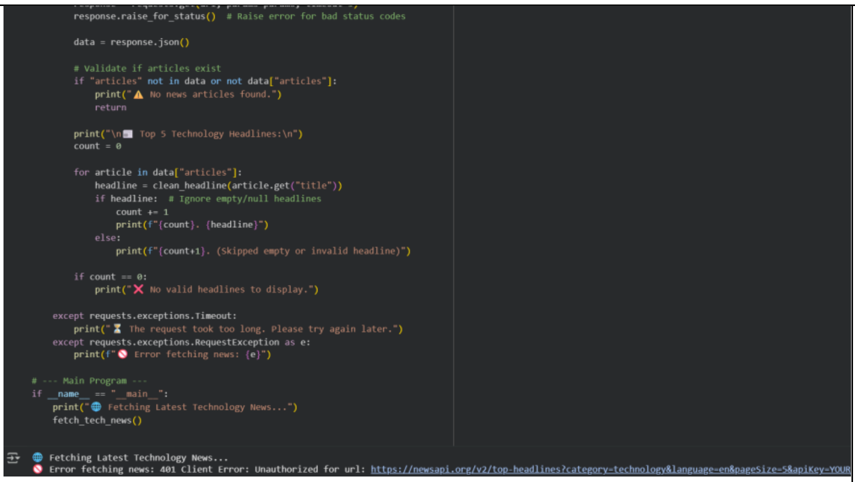
- **Task 1:** Use AI-assisted coding to fetch the top 5 technology headlines and print them neatly in the console. Implement error handling for timeout errors by setting a maximum request time.
- **Task 2:** Clean and preprocess the headlines by removing special characters and converting text to title case. Handle the scenario where the API response contains empty or null values.

  PROMPT: **Write a Python script that fetches the latest top 5 technology news headlines using a public News API. The program should:Fetch data from the API with a timeout limit to handle slow responses.Handle errors such as timeouts, network issues, or incomplete responses.Clean and preprocess the headlines by removing special characters and converting them to title case. Ignore or handle cases where the API returns empty or null headlines.Display the cleaned top 5 headlines neatly in the console.Use AI-assisted coding and include comments explaining each step.**

  Screenshots:



```python
import requests
import re

# Function to clean and format headlines
def clean_headline(headline):
    if not headline:
        return None
    # Remove special characters except letters, numbers, and spaces
    cleaned = re.sub(r'[^A-Za-z0-9\s]', '', headline)
    # Convert to title case
    return cleaned.title()

# Function to fetch technology headlines with error handling
def fetch_tech_news():
    url = "https://newsapi.org/v2/top-headlines"
    params = {
        "category": "technology",
        "language": "en",
        "pageSize": 5,  # Limit to top 5 headlines
        "apiKey": "YOUR_API_KEY_HERE"  # Replace with your actual NewsAPI key
    }

    try:
        # Set timeout to 5 seconds to handle slow responses
        response = requests.get(url, params=params, timeout=5)
        response.raise_for_status()  # Raise error for bad status codes

        data = response.json()

        # Validate if articles exist
        if "articles" not in data or not data["articles"]:
            print("⚠ No news articles found.")
            return

        print("\n📰 Top 5 Technology Headlines:\n")
        count = 0
```

```
        response.raise_for_status()  # Raise error for bad status codes

        data = response.json()

        # Validate if articles exist
        if "articles" not in data or not data["articles"]:
            print("⚠ No news articles found.")
            return

        print("\n📰 Top 5 Technology Headlines:\n")
        count = 0

        for article in data["articles"]:
            headline = clean_headline(article.get("title"))
            if headline:  # Ignore empty/null headlines
                count += 1
                print(f"{count}. {headline}")
            else:
                print(f"{count+1}. (Skipped empty or invalid headline)")

        if count == 0:
            print("❌ No valid headlines to display.")

    except requests.exceptions.Timeout:
        print("⌛ The request took too long. Please try again later.")
    except requests.exceptions.RequestException as e:
        print(f"🚫 Error fetching news: {e}")

# --- Main Program ---
if __name__ == "__main__":
    print("🌐 Fetching Latest Technology News...")
    fetch_tech_news()
```

```
🌐 Fetching Latest Technology News...
🚫 Error fetching news: 401 Client Error: Unauthorized for url: https://newsapi.org/v2/top-headlines?category=technology&language=en&pageSize=5&apiKey=YOUR
```