UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Student Name: Arghyanil Chowdhury	**UID:** 24MCC20021
Branch: MCA (CCD)	**Section/Group:** 24MCD1-A
Semester: 1st	**Date of Performance:** 30/10/2024
Subject Name: Linux Administration Lab	**Subject Code:** 24CAP-607

Q. Title of Project: Weather Data Tracking and Historical Report System

1. Aim of the Practical:

The aim of this practical is to develop a weather data tracking and retrieval system that fetches real-time weather information for specified cities using the OpenWeatherMap API. The project enables storage of weather data in JSON format, allowing users to access both current weather conditions and historical records. Additionally, the system provides functionality to search and display past weather reports by city, date, or date range, facilitating easy access to comprehensive weather information.

2. Hardware and Software Requirements:

- Hardware Requirements:
  - A computer system with a minimum of 2 GB RAM (recommended: 4 GB or more).
  - 500 MB of free disk space for storing data files and required software.
  - Stable internet connection for API data fetching.

- Software Requirements:
  - Operating System: Red Hat Enterprise Linux (RHEL) or any Linux-based OS that supports Bash scripting.
  - Software Packages:
    - Bash: For scripting and running the program.
    - curl: To handle HTTP requests and fetch data from the OpenWeatherMap API.
    - jq: For parsing and manipulating JSON data.
  - OpenWeatherMap API Key: A valid API key to access weather data.
  - Text Editor: Any text editor for script development (e.g., VS Code, Nano, Vim)

3. Task to be Done:

- Setup the environment by installing Red Hat Enterprise Linux (RHEL) and ensuring required software packages (curl, jq) are installed and configured.

- Obtain a valid API key from OpenWeatherMap by signing up for an account.

- Develop a Bash script to fetch current weather data for a specified city, extract relevant information (temperature, humidity, wind speed, and weather condition), display the information in a user-friendly manner, and store the fetched data in a JSON file for historical reference.

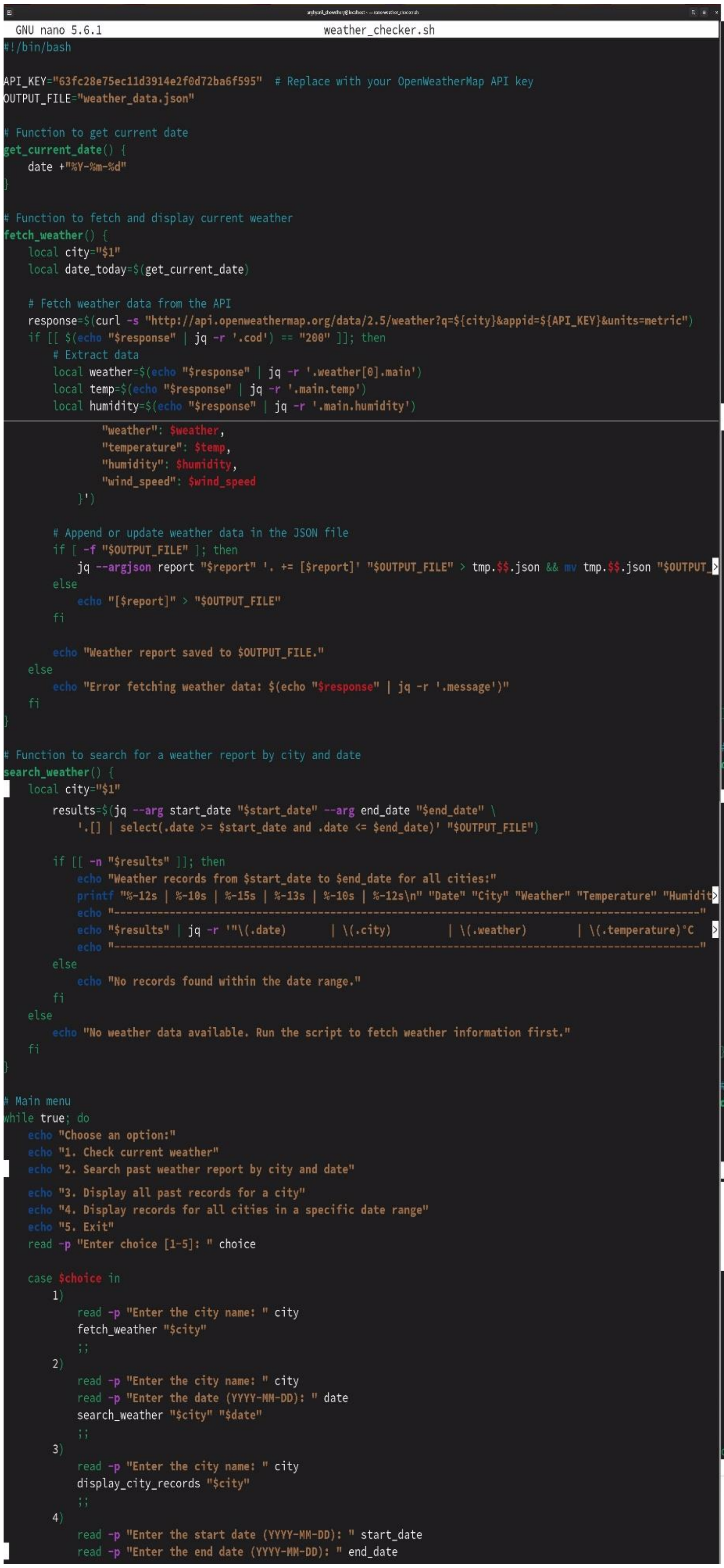
- Implement functions to search for past weather reports by city and date, display all weather records for a specific city, and retrieve weather records across multiple cities within a defined date range.

- Design a command-line menu system for user interaction, allowing easy navigation of options to fetch current weather, search past reports, or display historical data.

- Test the script for various cities and date ranges to ensure accurate data retrieval and reporting, and validate the correctness of stored data and the functionality of search features.

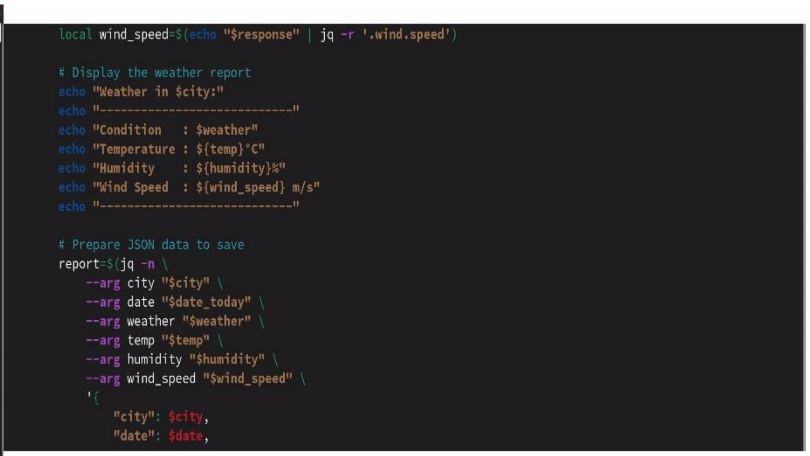
- Prepare documentation detailing the script’s functionality, usage instructions, and any configuration requirements for users.

4. Code for Experiment/Practical:

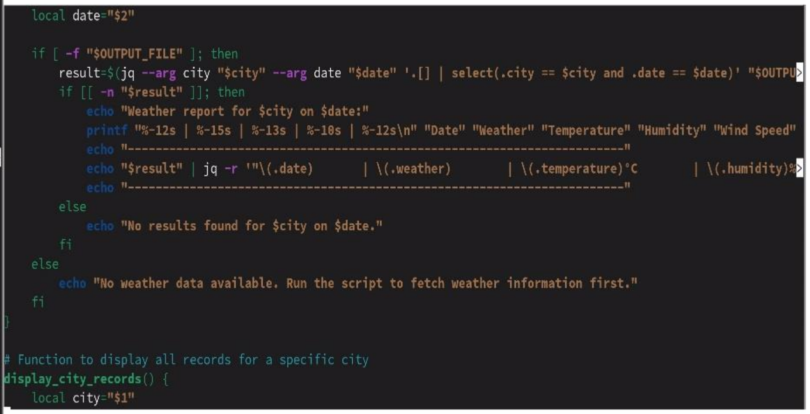


```
GNU nano 5.6.1                                    weather_checker.sh
#!/bin/bash

API_KEY="63fc28e75ec11d3914e2f0d72ba6f595"  # Replace with your OpenWeatherMap API key
OUTPUT_FILE="weather_data.json"

# Function to get current date
get_current_date() {
    date +"%Y-%m-%d"
}

# Function to fetch and display current weather
fetch_weather() {
    local city="$1"
    local date_today=$(get_current_date)

    # Fetch weather data from the API
    response=$(curl -s "http://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API_KEY}&units=metric")
    if [[ $(echo "$response" | jq -r '.cod') == "200" ]]; then
        # Extract data
        local weather=$(echo "$response" | jq -r '.weather[0].main')
        local temp=$(echo "$response" | jq -r '.main.temp')
        local humidity=$(echo "$response" | jq -r '.main.humidity')
```

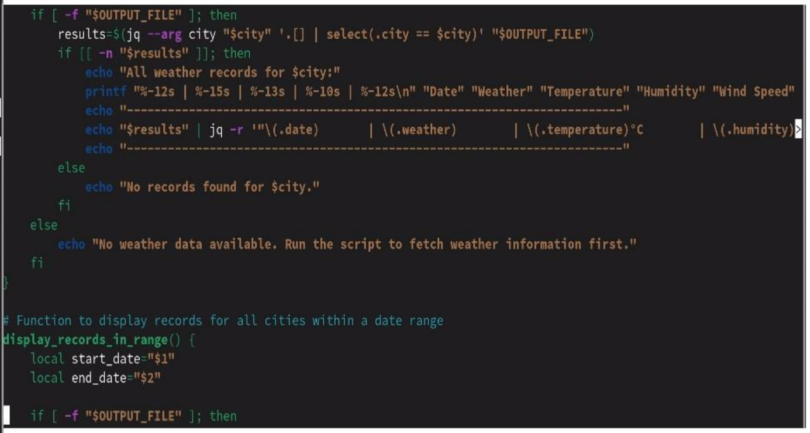
```
                "weather": $weather,
                "temperature": $temp,
                "humidity": $humidity,
                "wind_speed": $wind_speed
            }')

        # Append or update weather data in the JSON file
        if [ -f "$OUTPUT_FILE" ]; then
            jq --argjson report "$report" '. += [$report]' "$OUTPUT_FILE" > tmp.$$.json && mv tmp.$$.json "$OUTPUT_>
        else
            echo "[$report]" > "$OUTPUT_FILE"
        fi

        echo "Weather report saved to $OUTPUT_FILE."
    else
        echo "Error fetching weather data: $(echo "$response" | jq -r '.message')"
    fi
}

# Function to search for a weather report by city and date
search_weather() {
    local city="$1"
        results=$(jq --arg start_date "$start_date" --arg end_date "$end_date" \
            '.[] | select(.date >= $start_date and .date <= $end_date)' "$OUTPUT_FILE")

        if [[ -n "$results" ]]; then
            echo "Weather records from $start_date to $end_date for all cities:"
            printf "%-12s | %-10s | %-15s | %-13s | %-10s | %-12s\n" "Date" "City" "Weather" "Temperature" "Humidit>
            echo "-----------------------------------------------------------------------------------------------"
            echo "$results" | jq -r '"\(.date)       | \(.city)         | \(.weather)        | \(.temperature)°C  >
            echo "-----------------------------------------------------------------------------------------------"
        else
            echo "No records found within the date range."
        fi
    else
        echo "No weather data available. Run the script to fetch weather information first."
    fi
}

# Main menu
while true; do
    echo "Choose an option:"
    echo "1. Check current weather"
    echo "2. Search past weather report by city and date"
    echo "3. Display all past records for a city"
    echo "4. Display records for all cities in a specific date range"
    echo "5. Exit"
    read -p "Enter choice [1-5]: " choice

    case $choice in
        1)
            read -p "Enter the city name: " city
            fetch_weather "$city"
            ;;
        2)
            read -p "Enter the city name: " city
            read -p "Enter the date (YYYY-MM-DD): " date
            search_weather "$city" "$date"
            ;;
        3)
            read -p "Enter the city name: " city
            display_city_records "$city"
            ;;
        4)
            read -p "Enter the start date (YYYY-MM-DD): " start_date
            read -p "Enter the end date (YYYY-MM-DD): " end_date
```



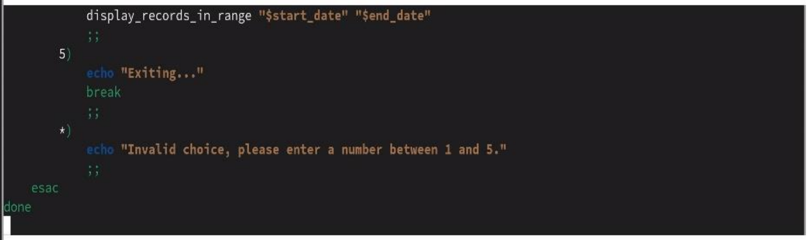
```
        local wind_speed=$(echo "$response" | jq -r '.wind.speed')

        # Display the weather report
        echo "Weather in $city:"
        echo "----------------------------"
        echo "Condition   : $weather"
        echo "Temperature : ${temp}°C"
        echo "Humidity    : ${humidity}%"
        echo "Wind Speed  : ${wind_speed} m/s"
        echo "----------------------------"

        # Prepare JSON data to save
        report=$(jq -n \
            --arg city "$city" \
            --arg date "$date_today" \
            --arg weather "$weather" \
            --arg temp "$temp" \
            --arg humidity "$humidity" \
            --arg wind_speed "$wind_speed" \
            '{
                "city": $city,
                "date": $date,
```



```
    local date="$2"

    if [ -f "$OUTPUT_FILE" ]; then
        result=$(jq --arg city "$city" --arg date "$date" '.[] | select(.city == $city and .date == $date)' "$OUTPU>
        if [[ -n "$result" ]]; then
            echo "Weather report for $city on $date:"
            printf "%-12s | %-15s | %-13s | %-10s | %-12s\n" "Date" "Weather" "Temperature" "Humidity" "Wind Speed"
            echo "-----------------------------------------------------------------------"
            echo "$result" | jq -r '"\(.date)       | \(.weather)        | \(.temperature)°C        | \(.humidity)%>
            echo "-----------------------------------------------------------------------"
        else
            echo "No results found for $city on $date."
        fi
    else
        echo "No weather data available. Run the script to fetch weather information first."
    fi
}

# Function to display all records for a specific city
display_city_records() {
    local city="$1"
```

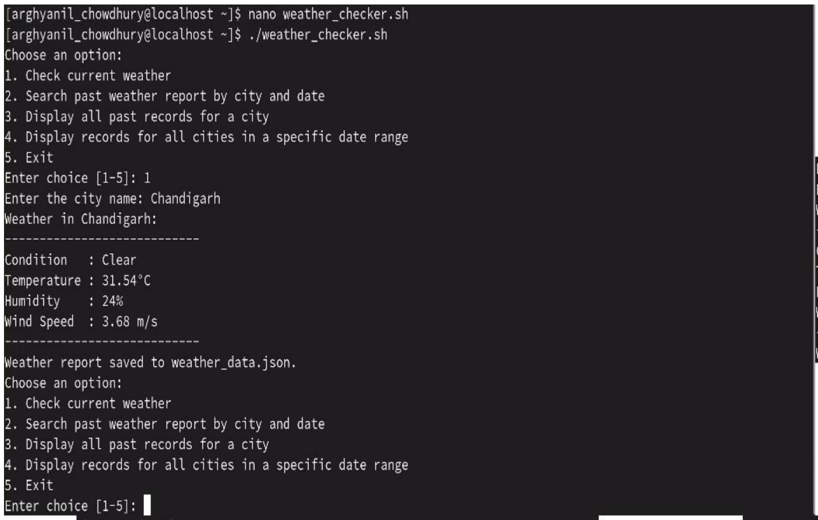


```
    if [ -f "$OUTPUT_FILE" ]; then
        results=$(jq --arg city "$city" '.[] | select(.city == $city)' "$OUTPUT_FILE")
        if [[ -n "$results" ]]; then
            echo "All weather records for $city:"
            printf "%-12s | %-15s | %-13s | %-10s | %-12s\n" "Date" "Weather" "Temperature" "Humidity" "Wind Speed"
            echo "-----------------------------------------------------------------------"
            echo "$results" | jq -r '"\(.date)       | \(.weather)        | \(.temperature)°C        | \(.humidity)>
            echo "-----------------------------------------------------------------------"
        else
            echo "No records found for $city."
        fi
    else
        echo "No weather data available. Run the script to fetch weather information first."
    fi
}

# Function to display records for all cities within a date range
display_records_in_range() {
    local start_date="$1"
    local end_date="$2"

    if [ -f "$OUTPUT_FILE" ]; then
```

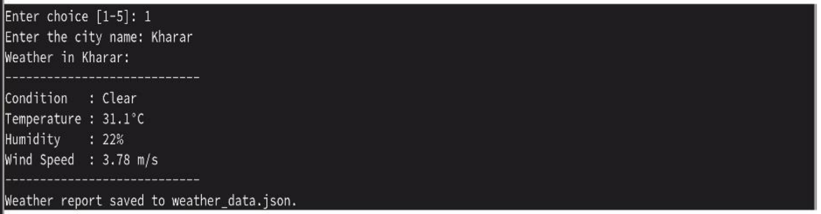


```
            display_records_in_range "$start_date" "$end_date"
            ;;
        5)
            echo "Exiting..."
            break
            ;;
        *)
            echo "Invalid choice, please enter a number between 1 and 5."
            ;;
    esac
done
```

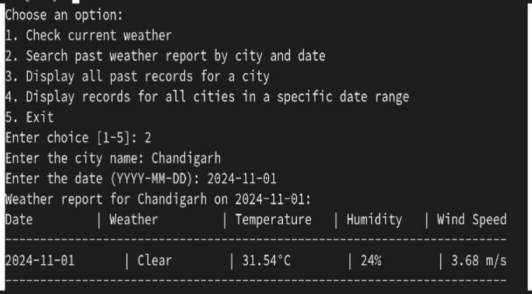
5. Result/Output/Writing Summary:



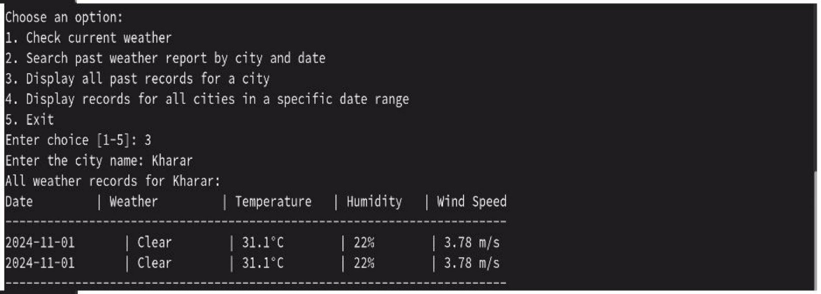
```
[arghyanil_chowdhury@localhost ~]$ nano weather_checker.sh
[arghyanil_chowdhury@localhost ~]$ ./weather_checker.sh
Choose an option:
1. Check current weather
2. Search past weather report by city and date
3. Display all past records for a city
4. Display records for all cities in a specific date range
5. Exit
Enter choice [1-5]: 1
Enter the city name: Chandigarh
Weather in Chandigarh:
----------------------------
Condition   : Clear
Temperature : 31.54°C
Humidity    : 24%
Wind Speed  : 3.68 m/s
----------------------------
Weather report saved to weather_data.json.
Choose an option:
1. Check current weather
2. Search past weather report by city and date
3. Display all past records for a city
4. Display records for all cities in a specific date range
5. Exit
Enter choice [1-5]:
```



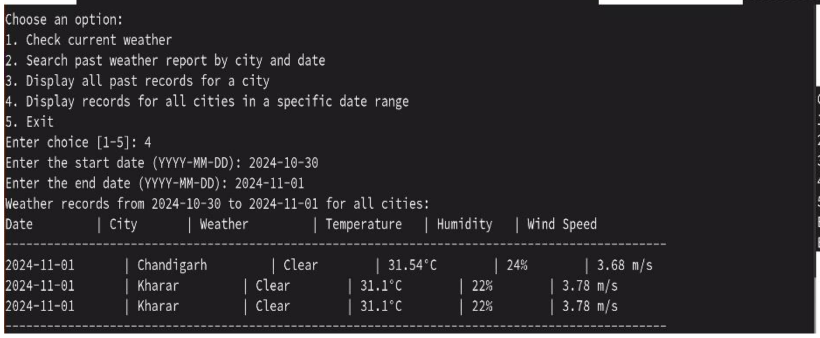
```
Enter choice [1-5]: 1
Enter the city name: Kharar
Weather in Kharar:
----------------------------
Condition   : Clear
Temperature : 31.1°C
Humidity    : 22%
Wind Speed  : 3.78 m/s
----------------------------
Weather report saved to weather_data.json.
```



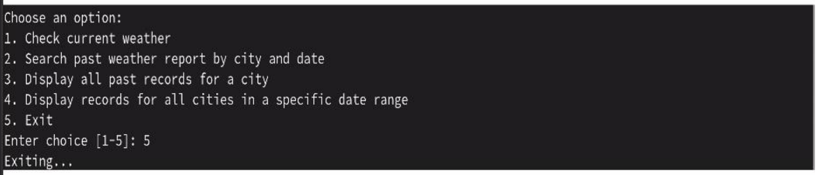
```
Choose an option:
1. Check current weather
2. Search past weather report by city and date
3. Display all past records for a city
4. Display records for all cities in a specific date range
5. Exit
Enter choice [1-5]: 2
Enter the city name: Chandigarh
Enter the date (YYYY-MM-DD): 2024-11-01
Weather report for Chandigarh on 2024-11-01:
Date         | Weather         | Temperature   | Humidity    | Wind Speed
--------------------------------------------------------------------------
2024-11-01       | Clear        | 31.54°C        | 24%        | 3.68 m/s
--------------------------------------------------------------------------
```



```
Choose an option:
1. Check current weather
2. Search past weather report by city and date
3. Display all past records for a city
4. Display records for all cities in a specific date range
5. Exit
Enter choice [1-5]: 3
Enter the city name: Kharar
All weather records for Kharar:
Date         | Weather         | Temperature    | Humidity    | Wind Speed
---------------------------------------------------------------------------
2024-11-01       | Clear        | 31.1°C        | 22%        | 3.78 m/s
2024-11-01       | Clear        | 31.1°C        | 22%        | 3.78 m/s
---------------------------------------------------------------------------
```



```
Choose an option:
1. Check current weather
2. Search past weather report by city and date
3. Display all past records for a city
4. Display records for all cities in a specific date range
5. Exit
Enter choice [1-5]: 4
Enter the start date (YYYY-MM-DD): 2024-10-30
Enter the end date (YYYY-MM-DD): 2024-11-01
Weather records from 2024-10-30 to 2024-11-01 for all cities:
Date         | City       | Weather         | Temperature    | Humidity    | Wind Speed
------------------------------------------------------------------------------------------------
2024-11-01       | Chandigarh         | Clear        | 31.54°C        | 24%        | 3.68 m/s
2024-11-01       | Kharar         | Clear        | 31.1°C        | 22%        | 3.78 m/s
2024-11-01       | Kharar         | Clear        | 31.1°C        | 22%        | 3.78 m/s
------------------------------------------------------------------------------------------------
```



```
Choose an option:
1. Check current weather
2. Search past weather report by city and date
3. Display all past records for a city
4. Display records for all cities in a specific date range
5. Exit
Enter choice [1-5]: 5
Exiting...
```

The weather-checking application successfully retrieves and displays current weather information for specified cities. Upon executing the program, users can choose from several options, including checking the current weather, searching for past weather reports by city and date, displaying all past records for a specific city, and reviewing records across multiple cities within a defined date range. The application correctly fetched and presented the current weather conditions for cities such as Chandigarh and Kharar, including details like temperature, humidity, wind speed, and weather condition. For instance, in Chandigarh, the temperature was reported at 31.54°C with clear weather. Additionally, the program successfully retrieved past weather data for specific dates, showing accurate historical records with all relevant details, such as the weather report for November 1, 2024, which returned the correct data for both Chandigarh and Kharar. The fetched weather data is stored in a JSON file (weather_data.json), allowing for easy access and management of weather records over time, enabling users to track and analyze weather patterns effectively. The command-line interface is intuitive, guiding users through the various options with clear prompts and formatted output, which enhances the overall user experience. In conclusion, the project demonstrates effective automation in fetching and displaying weather data, meeting the specified requirements and functionalities.

Learning outcomes (What I have learnt):

- Developed a deeper understanding of shell scripting, specifically Bash, enhancing my ability to automate tasks and create functional command-line applications.

- Gained hands-on experience in working with APIs, learning how to fetch real-time weather data from the OpenWeatherMap API and process JSON responses using tools like jq.

- Improved skills in data management by implementing a system to store and retrieve weather information in a structured JSON format, emphasizing the importance of organized data for accessibility and analysis.

- Enhanced my ability to handle errors in scripts, allowing for graceful management of potential issues when fetching data from external sources.

- Strengthened problem-solving skills through debugging and refining the application based on user feedback, leading to a more robust final product.

- Focused on creating an intuitive user interface, ensuring the program is user-friendly and accessible to a wider audience.

- Acquired practical knowledge applicable to future software development projects, particularly in areas of automation, data retrieval, and user interaction design.