CPT304 Assignment2 Development Process Report

Overview

This report details the development process of a Holiday and Weather application, which helps users find public holidays within 16 days, check the weather forecast for a specific holiday, and search for hotels in the selected city. The application utilizes Python and various APIs to gather holiday and weather data and employs the tkinter library for the graphical user interface.

Outline and Analysis of Requirements

Before starting the development process, I analyzed the requirements and outlined the necessary steps:

1. A user interface for entering a country name and city.
2. A method to search for country codes using the entered country name.
3. Gather public holidays for the specified country based on the country codes.
4. Filter holidays within 16 days.
5. Retrieve weather data for the selected city and date.
6. Display weather information.
7. Search for hotels in the city.

Components Selection and Validation Process

The following components and libraries were selected:

1. `tkinter` : A Python library for creating graphical user interfaces.
2. `pycountry` : A Python library to interact with the ISO country codes.
3. `requests` : A Python library for making HTTP requests.
4. `geopy` : A Python library for dealing with geocoding operations.
5. `matplotlib` : A Python library for creating visualizations.
6. `webbrowser` for searching suitable hotels.
7. Holiday data retrieval: The Nager.Date API (https://date.nager.at/) was used to fetch public holiday data for the specified country code.
8. Weather data retrieval: The Open-Meteo API (https://www.open-meteo.com/) was used to obtain weather data for a specific city and date.

These components were validated by examining their compatibility with the project requirements, their ease of use, and their maintainability. Furthermore, we ensured that they were well-documented and had a strong user community to support troubleshooting during the development process.

Components Composition

I will shown the components composition by the three requirements.

1. **Public Holiday Search.** To retrieve public holidays, a free online API is utilized: `https://date.nager.at/api/v3/publicholidays/2023/{CountryCode}` . The `requests` library is employed to access this HTTP resource. However, this API requires a `CountryCode` based on `ISO 3166-1 alpha-2` to search for public holidays, which many users may not be familiar with. Consequently, a converter is needed to transform the country name into a country code. The `pycountry` library fulfills this requirement, enabling users to input a country name and subsequently select and click the corresponding country code to search for public holidays.

   The Holiday API returns a JSON response, for example:

```
[
    {
        "date": "2023-01-01",
        "localName": "Neujahr",
        "name": "New Year's Day",
        ... // other properties.
    },
    ... // other Public Holidays.
]
```

From this response, the application extracts and displays the `date`, `name`, and `localName`.

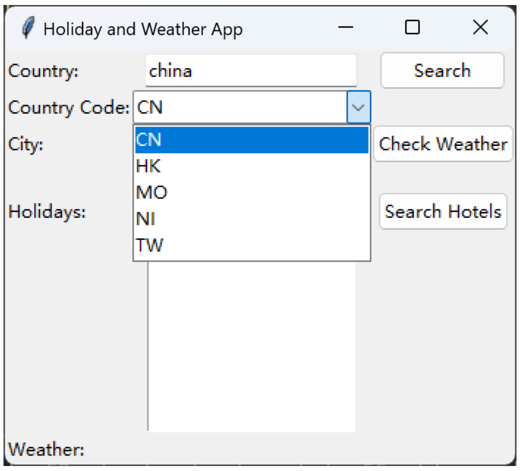


Figure 1: User input the country name and choose the country code.

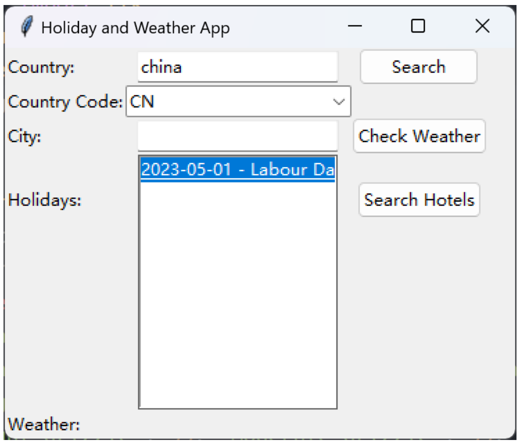
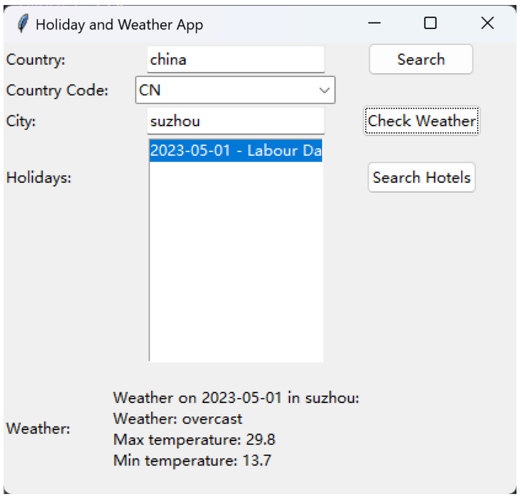


Figure 2: User choose the country code and get the in 16 days holidays.

2. **Search Weather.** Once the holiday date has been obtained, the weather can be searched for the selected date. The Open-Meteo API (https://www.open-meteo.com/) is used to gather weather data for a specific city and date. However, this API does not accept a city name as input; instead, it requires `longitude` and `latitude`. Therefore, the `geopy` library is employed as a converter to transform the city name into `longitude` and `latitude`. The `requests` library is then used to access the URL: `https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude={longitude}&daily=weathercode,temperature_2m_max,temperature_2m_min&forecast_days=1&start_date={date}&end_date={end_date}&timezone=Asia%2FSingapore`. In this response, the application retrieves the `temperature_2m_min`, `temperature_2m_max`, and `weathercode` in JSON format. Since the weather code is an integer, a converter is needed to transform the number into a human-readable string. The start date corresponds to the user-selected public holiday, while the end date is one day after the start date.



Holiday and Weather App

Country: china

Country Code: CN

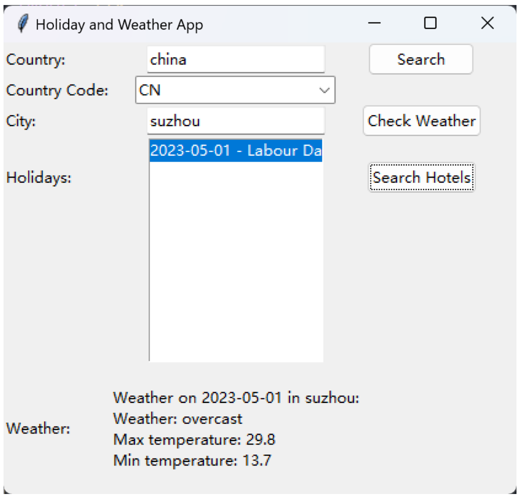
City: suzhou

Holidays: 2023-05-01 - Labour Da

Weather:
```
Weather on 2023-05-01 in suzhou:
Weather: overcast
Max temperature: 29.8
Min temperature: 13.7
```

Figure 3: User choose the holiday and get the weather information.

3. **Search the Hotels.** There are no readily available and free APIs to obtain hotel information. For instance, Airbnb and OYO do not offer APIs to users, while Booking.com provides paid APIs. Consequently, the `webbrowser` library is used to open a web browser and search for hotel information on Google. Google's comprehensive hotel search and booking visualization on its web pages provide users with an effective and convenient way to find accommodations.



Holiday and Weather App

Country: china

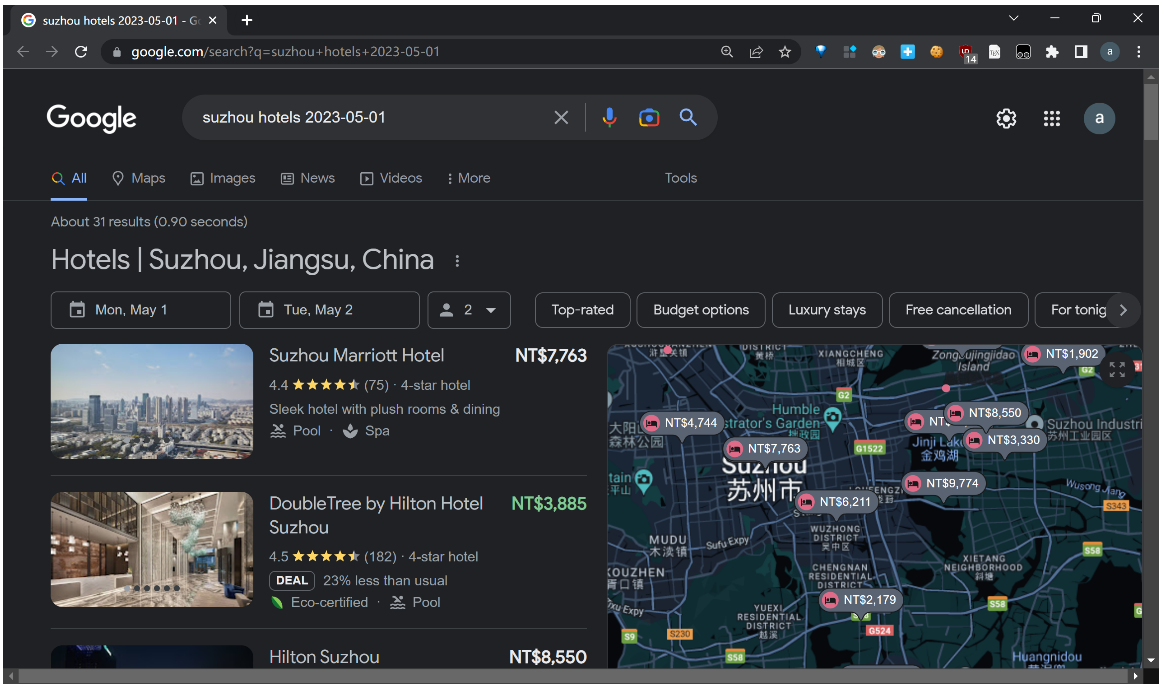
Country Code: CN

City: suzhou

Holidays: 2023-05-01 - Labour Da

Weather:
```
Weather on 2023-05-01 in suzhou:
Weather: overcast
Max temperature: 29.8
Min temperature: 13.7
```

Figure 4: User click the "Search Hotels" button.


suzhou hotels 2023-05-01

About 31 results (0.90 seconds)

Hotels | Suzhou, Jiangsu, China

Mon, May 1 · Tue, May 2 · 2

Top-rated · Budget options · Luxury stays · Free cancellation · For tonig

Suzhou Marriott Hotel NT$7,763
4.4 ★★★★★ (75) · 4-star hotel
Sleek hotel with plush rooms & dining
Pool · Spa

DoubleTree by Hilton Hotel Suzhou NT$3,885
4.5 ★★★★★ (182) · 4-star hotel
DEAL 23% less than usual
Eco-certified · Pool

Hilton Suzhou NT$8,550

Figure 5: check the hotel information on the holiday in the area.

Conclusion

The development process of the application was successful, resulting in a functional application that meets the outlined requirements. By utilizing the selected components and libraries, the application enables users to find public holidays, check weather forecasts, and search for hotels in the chosen city.

Appendix

```python
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import tkinter as tk
from tkinter import ttk
import pycountry
import requests
from geopy.geocoders import Nominatim


import webbrowser
# Function to search
# for country codes
weather_code = {
    0: 'clear sky',
    1: 'mainly clear', 2: 'partly cloudy', 3: 'overcast',
    45: 'fog', 48: 'depositing rime fog',
    51: 'light drizzle', 53: 'moderate drizzle', 55: 'dense drizzle',
    56: 'light freezing drizzle', 57: 'dense freezing drizzle',
    61: 'slight rain', 63: 'moderate rain', 65: 'heavy rain',
    66: 'light freezing rain', 67: 'dense freezing rain',
    71: 'slight snow fall', 73: 'moderate snow fall', 75: "heavy snow fall",
    77: 'snow grains',
    80: 'slight rain showers', 81: 'moderate rain showers', 82: 'violent rain showers',
    85: 'slight snow shower', 86: 'heavy snow showers',
    95: 'thunderstorm',
    96: 'thunderstorm with slight hail', 99: 'thunderstorm with heavy hail',
}

def open_hotel_search():
    if holiday_list.curselection():
        selected_date = holiday_list.get(holiday_list.curselection())
        date = selected_date.split(" - ")[0]
        city = city_entry.get()
        search_query = f"{city} hotels {date}"
        url = f"https://www.google.com/search?q={search_query.replace(' ', '+')}"
        webbrowser.open(url)
    else:
        weather_text.set("Please select a date from the list.")

def search_country_codes(query):
    countries = [
        country.alpha_2 for country in pycountry.countries.search_fuzzy(query)]
    return countries

# Function to get public holidays


def get_public_holidays(country_code):
    url = f'https://date.nager.at/api/v3/publicholidays/2023/{country_code}'
    response = requests.get(url)
    holidays = response.json()
    return holidays

# Function to handle the country search button click


def on_country_search_button_click():
    query = country_entry.get()
    country_codes = search_country_codes(query)
    country_code_combobox['values'] = country_codes
    country_code_combobox.current(0)

# Function to handle country code selection
```

```python
def on_country_code_selected(event):
    country_code = country_code_combobox.get()
    holidays = get_public_holidays(country_code)
    filtered_holidays = filter_holidays_within_16_days(holidays)
    holiday_list.delete(0, tk.END)
    for holiday in filtered_holidays:
        holiday_list.insert(
            tk.END, f"{holiday['date']} - {holiday['name']} ({holiday['localName']})")

# Function to get longitude and latitude


def filter_holidays_within_16_days(holidays):
    today = datetime.today()
    next_16_days = today + timedelta(days=16)
    filtered_holidays = [
        holiday for holiday in holidays
        if today <= datetime.strptime(holiday['date'], '%Y-%m-%d') <= next_16_days
    ]
    return filtered_holidays


def get_longitude_latitude(city_name):
    if city_name == "suzhou":
        return 120.60, 31.30
    geolocator = Nominatim(user_agent="holiday_weather_app")
    location = geolocator.geocode(city_name)
    return location.longitude, location.latitude

# Function to check the weather
def get_weather(latitude, longitude, date, end_date):
    url = f"https://api.open-meteo.com/v1/forecast?latitude={latitude}&longitude=
{longitude}&daily=weathercode,temperature_2m_max,temperature_2m_min&forecast_days=1&start_date=
{date}&end_date={end_date}&timezone=Asia%2FSingapore"
    response = requests.get(url)
    weather_data = response.json()
    return weather_data


# Function to handle the check weather button click
def plot_weather_data(weather_data):
    time = [datetime.strptime(t, "%Y-%m-%dT%H:%M")
            for t in weather_data['hourly']['time']]
    temperature = weather_data['hourly']['temperature_2m']

    fig, ax = plt.subplots()
    ax.plot(time, temperature, marker='o', linestyle='-')

    ax.set(xlabel='Time', ylabel='Temperature (°C)',
           title='Temperature Forecast')
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d %H:%M'))
    plt.xticks(rotation=45)
    plt.grid()
    plt.tight_layout()
    plt.show()


def on_check_weather_button_click():
    if holiday_list.curselection():
        selected_date = holiday_list.get(holiday_list.curselection())
        date = selected_date.split(" - ")[0]
        city = city_entry.get()
        longitude, latitude = get_longitude_latitude(city)
        weather_data = get_weather(latitude, longitude, date, date)
        # plot_weather_data(weather_data)
        s = weather_data
        weather_text.set(f"""
Weather on {s['daily']['time'][0]} in {city}:
Weather: {weather_code[s['daily']['weathercode'][0]]}
Max temperature: {s['daily']['temperature_2m_max'][0]}
Min temperature: {s['daily']['temperature_2m_min'][0]}
""")
    else:
        weather_text.set("Please select a date from the list.")

# Create the main window
root = tk.Tk()
root.title("Holiday and Weather App")

# Create input elements
country_label = ttk.Label(root, text="Country:")
country_label.grid(column=0, row=0, sticky=tk.W)
country_entry = ttk.Entry(root)
country_entry.grid(column=1, row=0)
country_search_button = ttk.Button(
    root, text="Search", command=on_country_search_button_click)
country_search_button.grid(column=2, row=0)

country_code_label = ttk.Label(root, text="Country Code:")
country_code_label.grid(column=0, row=1, sticky=tk.W)
country_code_combobox = ttk.Combobox(root)
country_code_combobox.grid(column=1, row=1)
country_code_combobox.bind("<<ComboboxSelected>>", on_country_code_selected)

# Add city input elements
city_label = ttk.Label(root, text="City:")
city_label.grid(column=0, row=2, sticky=tk.W)
city_entry = ttk.Entry(root)
city_entry.grid(column=1, row=2)

# Add holiday list
holiday_label = ttk.Label(root, text="Holidays:")
holiday_label.grid(column=0, row=3, sticky=tk.W)
holiday_list = tk.Listbox(root)
holiday_list.grid(column=1, row=3, rowspan=4)

# Add check weather button
check_weather_button = ttk.Button(
    root, text="Check Weather", command=on_check_weather_button_click)
check_weather_button.grid(column=2, row=2)

# Add weather display
weather_label = ttk.Label(root, text="Weather:")
weather_label.grid(column=0, row=7, sticky=tk.W)
weather_text = tk.StringVar()
weather_display = ttk.Label(root, textvariable=weather_text)
weather_display.grid(column=1, row=7, sticky=tk.W)
hotel_search_button = ttk.Button(root, text="Search Hotels", command=open_hotel_search)
hotel_search_button.grid(column=2, row=3)

# Start the GUI event loop
root.mainloop()
```