SE 226 Spring 23-24 Project

BEST HOTELS FOR YOU

ÖZLEM NUR NAZLIOĞLU

20210601043

Project Title: Best Hotels for You

Project Description: The Hotel Recommendation System is a graphical user interface (GUI) application designed to assist users in finding top hotels in European cities based on their preferences. The system allows users to select a city, specify check-in and check-out dates, and choose their preferred currency for pricing. It then fetches data from an online source, processes it, and displays the top 5 recommended hotels matching the user's criteria.

REQUIREMENTS OF THE PROJECT

GUI Development Requirements

I. Use Tkinter or any other suitable GUI library to create the graphical interface for the program.

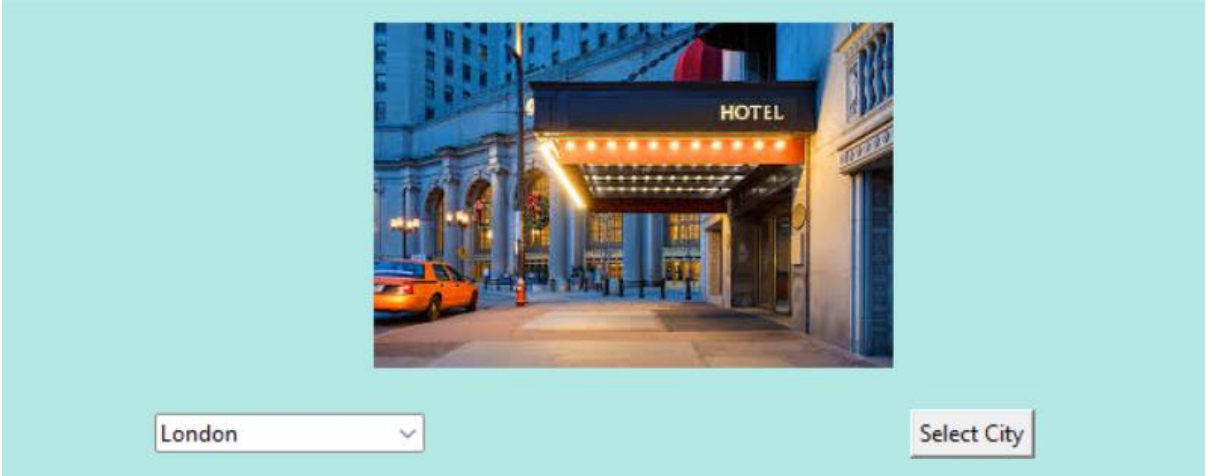
```
self.select_button = Button(self.master,
                            text="Select City",
                            command=self.select_city)
self.select_button.grid(row=1, column=1, padx=10, pady=10)

# Create a DateEntry widget for check-in date selection
self.check_in_date_entry = DateEntry(self.master, width=12, background='darkblue', foreground='white',
                                     borderwidth=2, date_pattern='yyyy-mm-dd', mindate=datetime.today())
self.check_in_date_entry.grid(row=2, column=0, padx=10, pady=10)

# Button to confirm check-in date selection
self.check_in_button = Button(self.master, text="Select Check-in Date", command=self.select_check_in_date)
self.check_in_button.grid(row=2, column=1, padx=10, pady=10)

# Create a DateEntry widget for check-out date selection
self.check_out_date_entry = DateEntry(self.master, width=12, background='darkblue', foreground='white',
                                      borderwidth=2, date_pattern='yyyy-mm-dd', mindate=datetime.today())
self.check_out_date_entry.grid(row=3, column=0, padx=10, pady=10)

# Button to confirm check-out date selection
self.check_out_button = Button(self.master, text="Select Check-out Date", command=self.select_check_out_date)
self.check_out_button.grid(row=3, column=1, padx=10, pady=10)

# Create a label for currency selection
self.title_label2 = Label(self.master,
                          text="Please select euro or tl",
                          font=("Arial", 13, "bold"),
                          bg="#b3e8e4")
self.title_label2.grid(row=4, column=0, columnspan=2, pady=10)
```

The code utilizes the tkinter library to create a GUI interface. It imports tkinter modules and uses them to construct various GUI components such as labels, buttons, frames, radio buttons, and text widgets.

The HotelListingGUI class serves as the main interface for the program. It creates a window with a specific title, size, and background color. Within this window, it constructs various GUI elements such as labels, buttons, combo boxes, date pickers, and text widgets to facilitate user interaction.

II. Design and implement a dropdown list to allow users to select a city (at least 10 Europe cites).

```
european_cities = [
    "London",
    "Paris",
    "Berlin",
    "Madrid",
    "Rome",
    "Amsterdam",
    "Vienna",
    "Prague",
    "Athens",
    "Stockholm"
]

# Create a Combobox for city selection

self.city_combobox = Combobox(self.master,
                              values=european_cities,
                              width=20,
                              state="readonly")
self.city_combobox.current(0)  # Set default selection
self.city_combobox.grid(row=1, column=0, pady=10)

# Button to confirm city selection
self.select_button = Button(self.master,
                            text="Select City",
                            command=self.select_city)
self.select_button.grid(row=1, column=1, padx=10, pady=10)
```



European_cities is a list of European cities defined in the code. Each city name is a string element in the list . Self.city_combobox creates an instance of the Combobox widget, which is used for selecting items from a dropdown list. It is initialized with the list of European cities as its values. State sets the state of the Combobox to "readonly", which means the user can't type in their own value but can only select from the predefined options. Self.city_combobox.current(0) sets the default selection of the Combobox to the first city in the list .

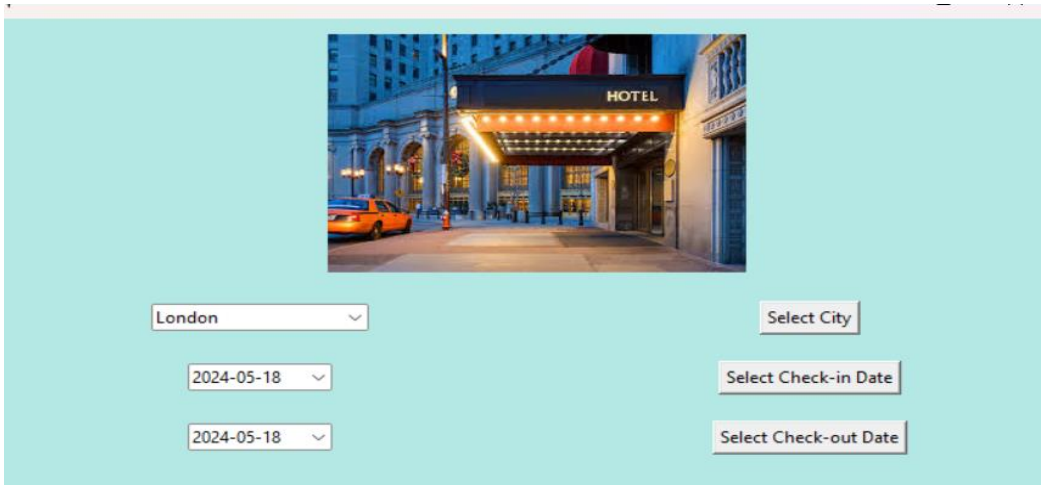
III. Design and implement a GUI item to allow users to enter check-in and check-out dates.

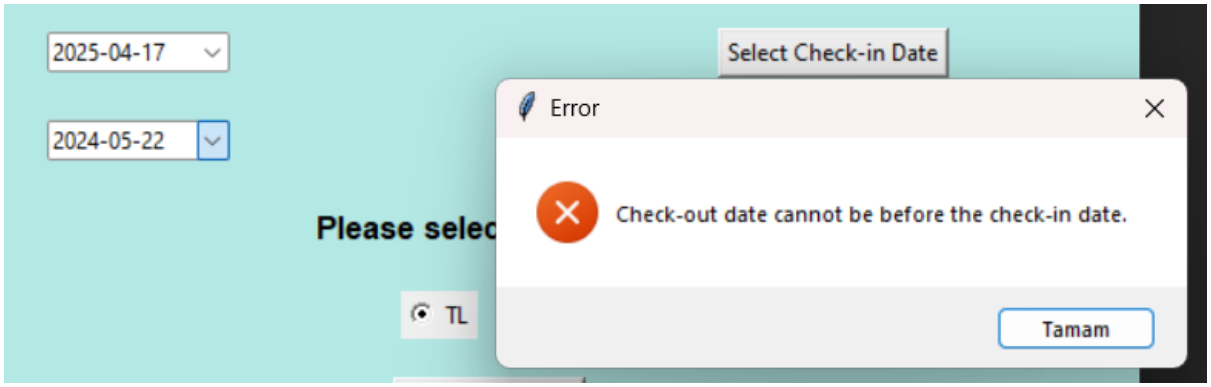
```
self.check_in_date_entry = DateEntry(self.master, width=12, background='darkblue', foreground='white',
                                     borderwidth=2, date_pattern='yyyy-mm-dd', mindate=datetime.today())
self.check_in_date_entry.grid(row=2, column=0, padx=10, pady=10)

# Button to confirm check-in date selection
self.check_in_button = Button(self.master, text="Select Check-in Date", command=self.select_check_in_date)
self.check_in_button.grid(row=2, column=1, padx=10, pady=10)

# Create a DateEntry widget for check-out date selection
self.check_out_date_entry = DateEntry(self.master, width=12, background='darkblue', foreground='white',
                                      borderwidth=2, date_pattern='yyyy-mm-dd', mindate=datetime.today())
self.check_out_date_entry.grid(row=3, column=0, padx=10, pady=10)

# Button to confirm check-out date selection
self.check_out_button = Button(self.master, text="Select Check-out Date", command=self.select_check_out_date)
self.check_out_button.grid(row=3, column=1, padx=10, pady=10)
```

Code creates DateEntry widgets for selecting check-in and check-out dates, along with buttons to confirm the date selections. Each DateEntry widget is initialized with a width of 12 characters, a dark blue background, white foreground, and a border width of 2 pixels. The date pattern is set to 'yyyy-mm-dd', and the mindate is set to today's date to prevent selecting past dates. These widgets are placed in the GUI window using the grid layout manager, specifying their positions in rows 2 and 3 and columns 0, and 1, respectively. Corresponding buttons labeled "Select Check-in Date" and "Select Check-out Date" are created, associated with callback functions for handling the date selections. Overall, this setup provides an intuitive interface for users to pick their check-in and check-out dates conveniently.

```
1 usage
def select_check_in_date(self):
    selected_date = self.check_in_date_entry.get_date()
    print("Check-in date:", selected_date)

1 usage
def select_check_out_date(self):
    selected_date = self.check_out_date_entry.get_date()
    print("Check-out date:", selected_date)

1 usage
def select_check_out_date(self):
    try:
        selected_check_in_date = self.check_in_date_entry.get_date()
        selected_check_out_date = self.check_out_date_entry.get_date()

        # Ensure check-out date is not before check
        if selected_check_out_date < selected_check_in_date:
            raise ValueError("Check-out date cannot be before the check-in date.")

        print("Check-out date:", selected_check_out_date)
```

Select_check_in_date: This method retrieves the selected check-in date from the DateEntry widget self.check_in_date_entry using the get_date() method and prints it to the console.Select_check_out_date: Similarly, this method retrieves the selected check-out date from the DateEntry widget self.check_out_date_entry and prints it to the console. Select_check_out_date : This method is a modification of the previous select_check_out_date method. It first attempts to retrieve both check-in and check-out dates. Then, it checks if the selected check-out date is before the selected check-in date. If so, it raises a ValueError with an appropriate error message. Otherwise, it prints the selected check-out date to the console. If a ValueError is raised, it displays an error message box with the error message.

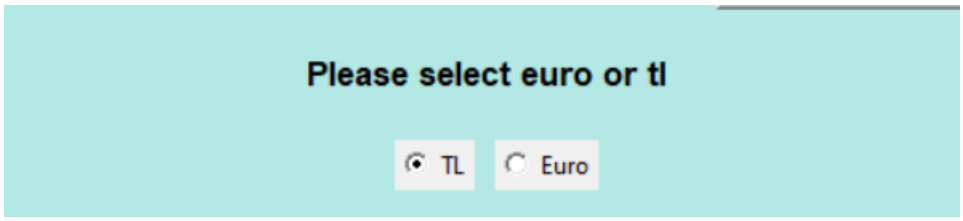




IV. Create a radio-button to show the hotel prices in Euro or TL (Use conversion 1Euro =30TL)

```
self.title_label2 = Label(self.master,
                          text="Please select euro or tl",
                          font=("Arial", 13, "bold"),
                          bg="#b3e8e4")
self.title_label2.grid(row=4, column=0, columnspan=2, pady=10)

# Create a frame for buttons
self.button_frame = Frame(self.master,
                          bg="#b3e8e4")
self.button_frame.grid(row=5, column=0, columnspan=2, pady=10)

# Currency variable
self.currency_var = StringVar(value="")

# Create TL button
self.button1 = Radiobutton(self.button_frame, text="TL", variable=self.currency_var, value="TL")
self.button1.grid(row=0, column=0, padx=10)

# Create Euro button
self.button2 = Radiobutton(self.button_frame, text="Euro", variable=self.currency_var, value="EURO")
self.button2.grid(row=0, column=1)
```

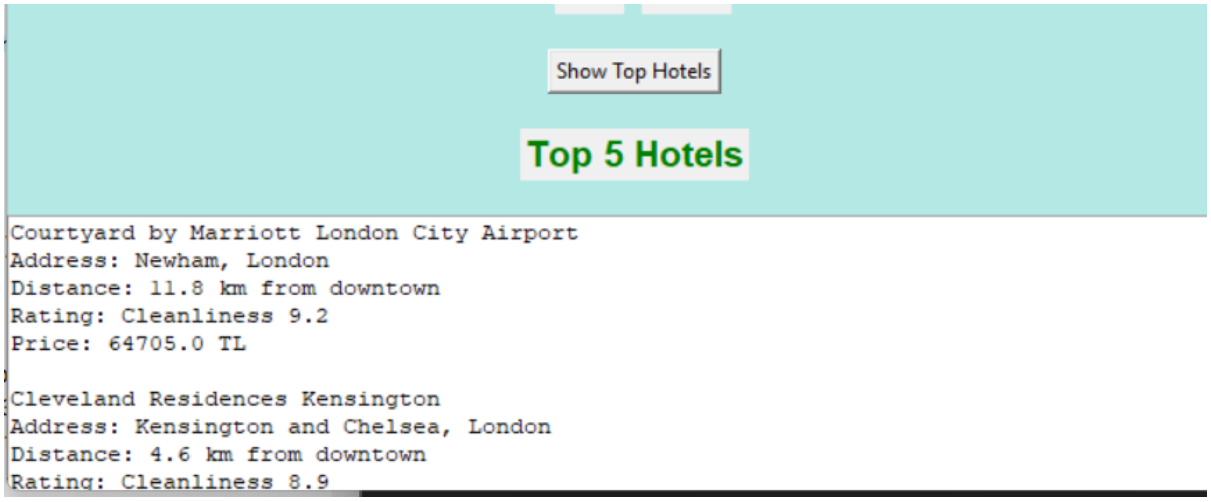
This segment of the code establishes a section within the graphical user interface to enable users to choose between Turkish Lira and Euro currencies. Initially, a descriptive label, "Please select euro or tl", is added to provide guidance to the users. Subsequently, a frame is created to contain the radio buttons facilitating currency selection. The frame is configured with a background color matching the GUI's theme. Two radio buttons are added within this frame: one labeled "TL" and the other "Euro". These buttons are linked to a StringVar variable, self.currency_var, allowing the GUI to track the user's selection. Positioned side by side within the frame, they offer a visually clear and intuitive means for users to indicate their currency preference, enhancing the overall usability of the interface.



V. Create an area in the GUI to display top 5 hotels (all attributes will be presented).

```
self.top_hotels_label = Label(self.master, text="Top 5 Hotels", font=("Helvetica", 16, "bold"),
                              foreground="green")
self.top_hotels_label.grid(row=7, column=0, columnspan=2, pady=10)

# Text widget to display top 5 hotels
self.top_hotels_text = Text(self.master, height=15, width=90)
self.top_hotels_text.grid(row=8, column=0, columnspan=2, pady=10)
```

This portion of the code sets up the interface for displaying the top 5 recommended hotels based on the user's selections. It consists of a button labeled "Show Top Hotels" (self.display_button), which, when clicked, triggers the display_top_hotels method to fetch and display the recommended hotels. Below the button, a label (self.top_hotels_label) with the text "Top 5 Hotels" is placed to indicate the purpose of the subsequent display. Finally, a Text widget (self.top_hotels_text) is added to the GUI to present the details of the top 5 hotels, such as their names, addresses, distances, ratings, and prices.



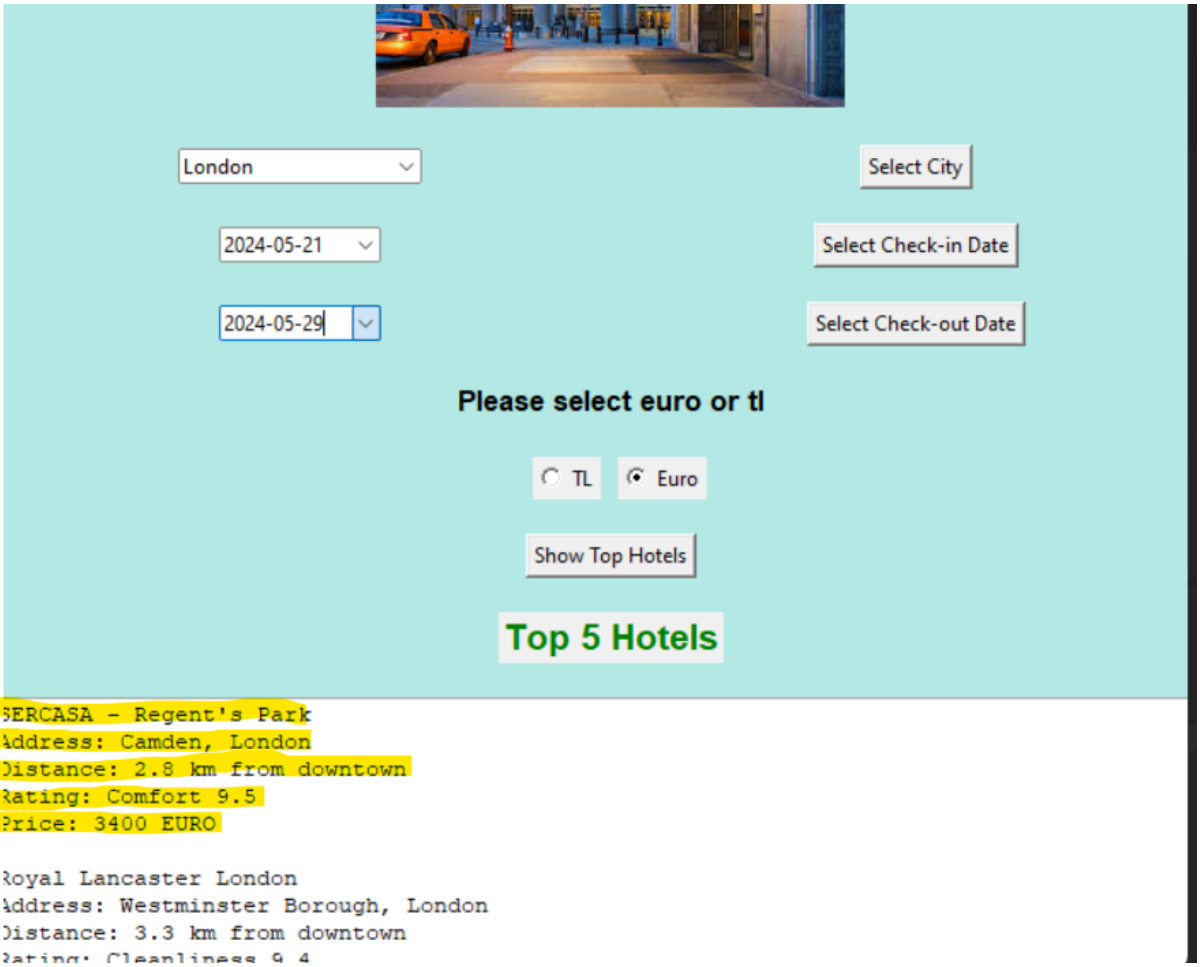
Web Scraping Requirements

Utilize the “requests” library to send HTTP requests and fetch the hotel information from Booking.com. Do not change the currency while you scrape the price. Retrieve the price in Euro or TL from web page then convert it whenever it is required. (Use conversion 1Euro =30TL whenever is required) ,Use the “beautifulsoup” library to parse the HTML content and extract the necessary data for the given user query and If any attribute/field does not hold a valid value assign “NOT GIVEN” to it.

```python
def fetch_hotels(self, city, check_in, check_out, currency):
    top_hotels_data = []

    try:
        url = self.get_url(city, check_in, check_out)
        headers = {
            'User-Agent': 'Mozilla/5.0 (X11; CrOS x86_64 8172.45.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.64 Safari/537.36',
            'Accept-Language': 'en-US, en;q=0.5'
        }
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.text, features: 'html.parser')
        hotels = soup.find_all( name: 'div', attrs: {'data-testid': 'property-card'})

        for hotel in hotels[:10]:  # Fetch data for the first 10 hotels
            # Extract hotel information
            name_element = hotel.find('div', {'data-testid': 'title'})
            name = name_element.text.strip() if name_element else "NOT GIVEN"
            address_element = hotel.find('span', {'data-testid': 'address'})
            address = address_element.text.strip() if address_element else "NOT GIVEN"
            distance_element = hotel.find('span', {'data-testid': 'distance'})
            distance = distance_element.text.strip() if distance_element else "NOT GIVEN"
            rating_element = hotel.find('span', {'class': 'a3332d346a'})
            rating_str = rating_element.text.strip() if rating_element else "Rating not available"

            price_element = hotel.find('span', {'data-testid': 'price-and-discounted-price'})
            price_str = price_element.text.strip() if price_element else "NOT GIVEN"
```

```python
            top_hotels_data.append({
                'title': name,
                'address': address,
                'distance': distance,
                'rating': rating_str,
                'price': price_tl
            })

    except Exception as e:
        print("Error occurred:", str(e))

    # Convert list of dictionaries to DataFrame
    top_hotels_df = pd.DataFrame(top_hotels_data)
```

Fetch_hotels method: This method takes in parameters like city, check-in and check-out dates, and currency. It then fetches data from a hotel Booking.com for the specified city and time range. It extracts information about the top hotels, including their name, address, distance, rating, and price. It also converts the price from Turkish Lira to EURO if the specified currency is EURO. It finally returns this hotel data in a Pandas DataFrame and code includes a try-except block to catch any exceptions that may occur during the process of fetching hotel data. If an exception occurs, it prints an error message.




```
1 usage
def get_url(self, city, check_in, check_out):
    # Assume 'cities' is a dictionary mapping city names to their IDs
    city_id = cities.get(city, "")
    base_url = 'https://www.booking.com/searchresults.html?ss={city}&ssne={city}&ssne_untouched={city}&efdco
    return base_url.format(city=city, city_id=city_id, check_in=check_in, check_out=check_out)
```

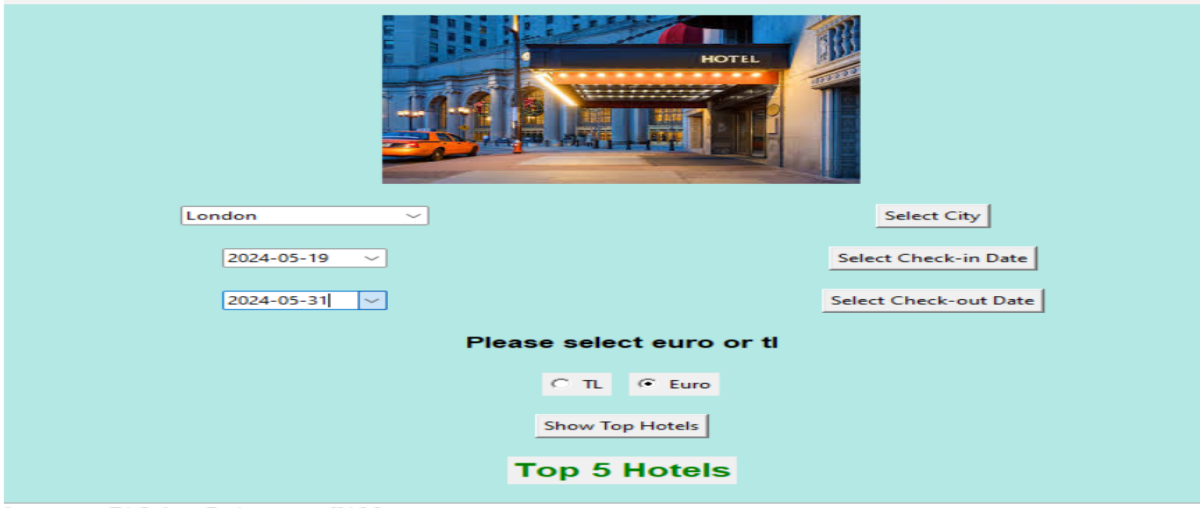
```
cities = {
    "London": "-2601889",
    "Paris": "-1456928",
    "Berlin": "-1746443",
    "Madrid": "-390625",
    "Rome": "-126693",
    "Amsterdam": "-2140479",
    "Vienna": "-1995499",
    "Prague": "-1476801",
    "Athens": "-814876",
    "Stockholm": "-2524279"
}
```

Get_url method: This method generates the URL for fetching hotel data from Booking.com. It takes the city name, check-in, and check-out dates as input parameters. It uses a base URL template and substitutes placeholders with actual values. The city ID is obtained from a predefined dictionary called cities, which maps city names to their corresponding IDs on Booking.com. This dictionary maps city names to their respective IDs on Booking.com. These IDs are used in the URL generation process by the get_url method.

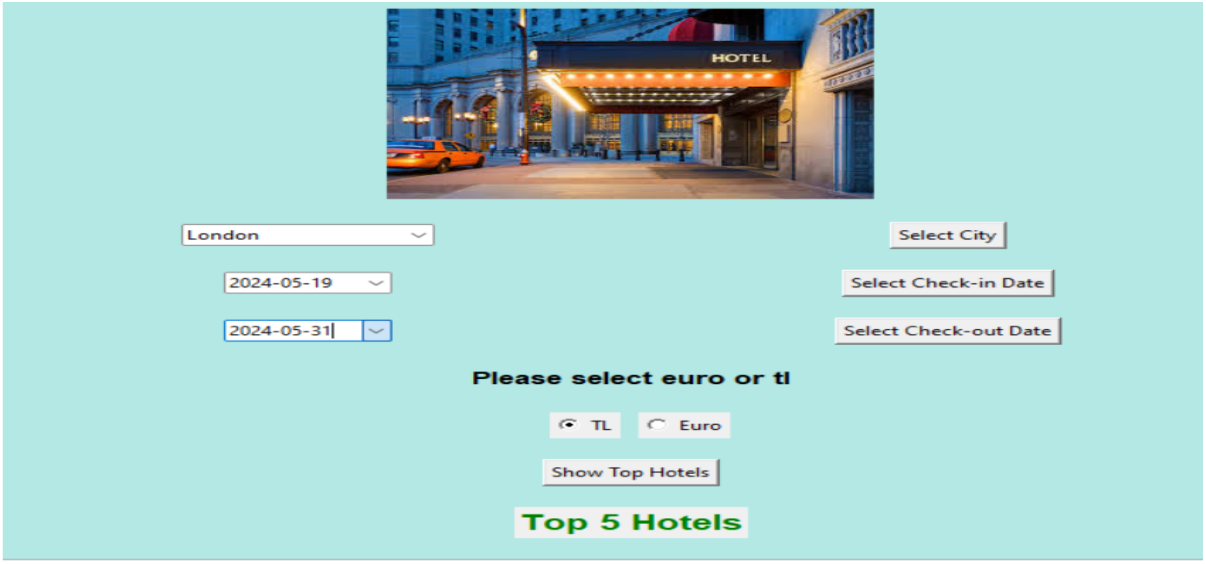
```
price_element = hotel.find('span', {'data-testid': 'price-and-discounted-price'})
price_str = price_element.text.strip() if price_element else "NOT GIVEN"

# Extract numerical value from price string and remove commas
price_value = price_str.split()[1].replace(',', '')
price_tl = float(price_value)

# Convert TL to EURO if currency is EURO
if currency == 'EURO':
    price_tl = math.floor(price_tl / 30)
```

Price_element = hotel.find('span', {'data-testid': 'price-and-discounted-price'}): This line searches for a element with the attribute data-testid set to 'price-and-discounted-price'. If found, it assigns the found element to price_element, otherwise, it assigns "NOT GIVEN". price_str = price_element.text.strip() if price_element else "NOT GIVEN": This line extracts the text content from the price_element found in the previous step. It then strips any leading or trailing whitespaces. If price_element exists ,it assigns the stripped text to price_str. Otherwise, it assigns "NOT GIVEN".

Price_value = price_str.split()[1].replace(',', '') It splits the price_str by whitespace into a list of strings and selects the second element , assuming that the price is represented. It then removes any commas from this string, as commas are often used as thousand separators in numeric representations. if currency == 'EURO': price_tl = price_tl / 30: This conditional statement checks if the specified currency is 'EURO'. If it is, it converts the price value from Turkish Lira to Euro by dividing it by 30.




```
Mason & Fifth, Primrose Hill
Address: Camden, London
Distance: 3.8 km from center
Rating: Comfort 9.6
Price: 3760 EURO
```




```
Mason & Fifth, Primrose Hill
Address: Camden, London
Distance: 3.8 km from downtown
Rating: Comfort 9.6
Price: 112814.0 TL
```

Hotel Information Storage/Display Requirements

I. If your Student ID's last digit is an odd number, the retrieved hotels will be sorted based on hotel ra1ng (considering only the number) (in descending order) and the sorted hotel list (top 5) will be presented in GUI.

```
if not top_hotels_df.empty:

    top_hotels_df = top_hotels_df.sort_values(by='rating', ascending=False,
                                              key=lambda x: pd.to_numeric(x.str.extract(r'(\d+\.\d+|\d+)')[0],
                                                                          errors='coerce')).head(5)
```

Code verifies whether the DataFrame top_hotels_df contains data, and if so, it sorts the DataFrame based on the 'rating' column in descending order, ensuring that hotels with

higher ratings appear first. The sorting is accomplished by converting the 'rating' column values into numeric format using a lambda function that employs regular expressions to extract numerical values. The extracted values are then converted to numeric format with any non-numeric values coerced to NaN. Finally, it selects the first 5 rows from the sorted DataFrame, representing the top 5 hotels with the highest ratings, effectively filtering the DataFrame to include only the most highly-rated hotels for further analysis or display.

Top 5 Hotels

```
Three-Bedroom Bohemian Haven
Address: Kensington and Chelsea, London
Distance: 5.5 km from downtown
Rating: Comfort 9.5
Price: 6696 EURO

Cove Landmark Pinnacle
Address: Tower Hamlets, London
Distance: 7.2 km from downtown
Rating: Cleanliness 9.2
```

```
Distance: 7.2 km from downtown
Rating: Cleanliness 9.2
Price: 1994 EURO

Ember Locke Kensington
Address: Kensington and Chelsea, London
Distance: 4.8 km from downtown
Rating: Comfort 9.0
Price: 1272 EURO
```

Top 5 Hotels

```
Rating: Comfort 9.0
Price: 1272 EURO

Wonderful Double Room In House
Address: London
Distance: 13.3 km from downtown
Rating: Comfort 8.6
Price: 469 EURO

Marigold Private double room
```

II. Store the hotel information (all attributes and all hotels) in a text or csv file.

```python
def display_top_hotels(self):
    city = self.city_combobox.get()
    check_in = self.check_in_date_entry.get()
    check_out = self.check_out_date_entry.get()
    currency = self.currency_var.get()
    top_hotels_df = self.fetch_hotels(city, check_in, check_out, currency)

    print("Type of top_hotels:", type(top_hotels_df))

    self.top_hotels_text.delete( index1: 1.0, END)
    if not top_hotels_df.empty:
        for index, row in top_hotels_df.iterrows():
            self.top_hotels_text.insert(END,
                                        chars: f"{row['title']}\nAddress: {row['address']}\nDistance: {row['distance']}\nRating: {row['rating']}\nPrice: {row['price']} {c
    else:
        self.top_hotels_text.insert(END,  chars: "No hotels found for the selected criteria.")

    self.store_hotel_data(top_hotels_df)  # Store the hotel data in a CSV file
```

```python
def store_hotel_data(self, hotels):
    try:
        hotels.to_csv('myhotels.csv', header=True, index=False)
        messagebox.showinfo( title: "Success",  message: "Hotel data successfully stored in CSV file.")
    except Exception as e:
        messagebox.showerror( title: "Error",  message: f"An error occurred while storing hotel data: {str(e)}")
        print("Error occurred while storing hotel data:", str(e))
```

Display_top_hotels: It retrieves the selected city, check-in and check-out dates, and currency from GUI elements like combo boxes and entry widgets. It calls the fetch_hotels method with the retrieved parameters to fetch hotel data for the specified city, dates, and currency. The resulting DataFrame is stored in top_hotels_df. It clears any existing text in a text widget , which is used for displaying hotel information. If the DataFrame top_hotels_df is not empty, it iterates over each row in the DataFrame using iterrows() and inserts hotel information (title, address, distance, rating, and price) into the text widget. The price is formatted with the selected currency.If no hotels are found for the selected criteria, it inserts a message indicating so into the text widget. It then calls a method store_hotel_data to store the hotel data in a CSV file.

Store_hotel_data: It takes a DataFrame hotels as input, containing hotel data to be stored. Inside a try-except block, it attempts to export the hotel data to a CSV file named 'myhotels.csv' using the to_csv method of Pandas DataFrame. If the CSV export is successful, it displays an information message box with the title "Success" and the content "Hotel data successfully stored in CSV file." If an exception occurs during the CSV export, it catches the exception, displays an error message box with details about the error, and prints the error message to the console.

```
title,address,distance,rating,price
WELTWIEN Luxury Art Apartments,"12. Meidling, Vienna",4.3 km from center,Cleanliness 9.7,723
Pension Haus Sanz,"23. Liesing, Vienna",8.2 km from center,Cleanliness 9.5,543
Grand Quarters City Residence,"03. Landstraße, Vienna",1.3 km from center,Cleanliness 9.4,757
Ruby Lissi Hotel Vienna,"01. Innere Stadt, Vienna",400 m from center,Location 9.4,599
Flemings Selection Hotel Wien-City,"08. Josefstadt, Vienna",1.5 km from center,Cleanliness 9.1,498
```

Text Data

Vienna

Select City

2024-05-19

Select Check-in Date

2024-05-22

Select Check-out Date

Please select euro or tl

TL Euro

Show Top Hotels

Top 5 Hotels

```
WELTWIEN Luxury Art Apartments
Address: 12. Meidling, Vienna
Distance: 4.3 km from center
Rating: Cleanliness 9.7
Price: 723 EURO

Pension Haus Sanz
Address: 23. Liesing, Vienna
Distance: 8.2 km from center
Rating: Cleanliness 9.5
```