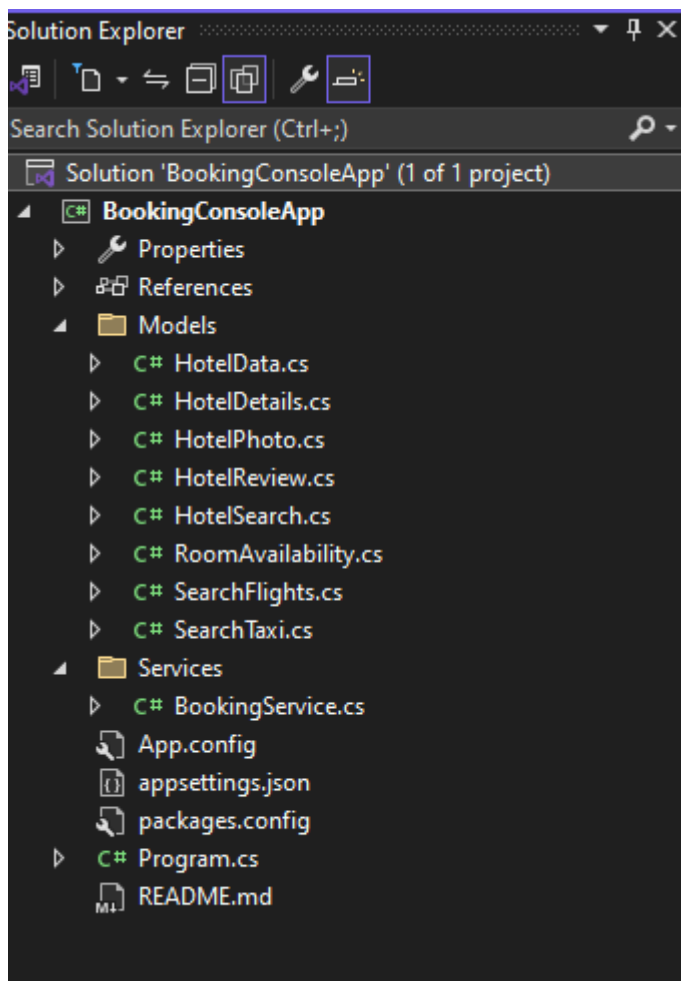


Demo of the AceBooking – Travel and Accommodation Booking System

Hello everyone, my name is Shyam Dattani. Today I'm going to show you a demo of our application, AceBooking, which we created as part of the Programming with APIs and Frameworks lesson. Beginning with this session, I will offer an overview of the code that supports the application, highlighting some of the main components that are required for operating. Following that, we'll go over how everything works from the viewpoint of the user.

Now, as I walk through the code, I'll point out classes and methods that are responsible for crucial functionality such as connecting to the Booking.com API, processing user input, and managing data replies. This should give you a clearer feel of the technology infrastructure we've created.

Then, in the application demo, I explain how these elements will fall into place, allowing us to conduct things like search for hotels, assess room availability, and so on. It offers you an overview of the route we take from code to execution, as well as how much thinking and reasoning goes into our work. So, let's get directly to the code.

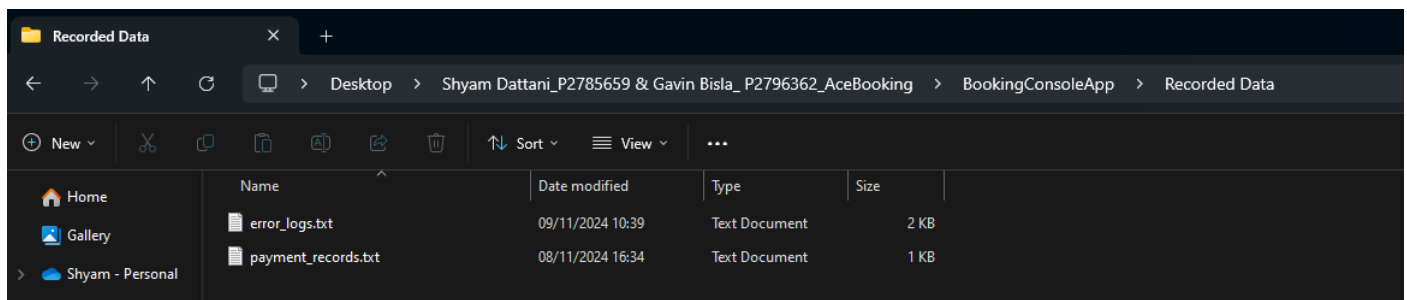


First, let's take a step back and look at the big picture: our codebase's structure, as displayed in this view from Visual Studio's Solution Explorer. This will present an orderly view of all the components that make up the program AceBooking.

We have a number of classes in the Models folder that are responsible for processing data obtained from multiple Booking.com APIs. For example, `HotelData.cs` and `HotelDetails.cs` give generic hotel information, whereas `RoomAvailability.cs` keeps room availability information. These classes are vital for structuring the data that we will send to the user.

The Services folder contains `BookingService.cs`, which is responsible for interacting with the Booking.com API. This file comprises our requests and answers for hotel searches, room details, and much more; it works as the backbone of our API connection.

Aside from this, there are a couple additional files: `appsettings.json`, which contains configuration settings such as API keys, and `Program.cs`, which functions as the application's entry point and handles the essential functionality and user interaction.



Finally, the Recorded Data folder contains a file labelled `payment_record.txt` and `error_logs.txt` file. The `payment_records.txt` file is a text file, it temporarily retains information such as the booking date, hotel ID, and payment details. The `error_logs.txt` file is an error handling text file which contains the timestamp and the issue. In this demo version, it's only a crude record simulation.

This project structure helps us to keep the code orderly and makes it simpler to work on any given task. Now I'll go over in detail some of the files that make up this project and how they operate together to deliver crucial functionality like hotel search and reservations.

Service's Folder – `BookingService.cs`

Now, we're moving on to the **`BookingService.cs`** file, which contains the core logic for interacting with the Booking.com API. This file is crucial for performing hotel booking operations.

First, let's look at the **import statements**:

```
using BookingConsoleApp.Models; // Importing models from the BookingConsoleApp.Models namespace
using System.Collections.Generic; // For using List<T>
using System.Net.Http; // For making HTTP requests
using System.Net.Http.Json; // For JSON serialisation and deserialisation
using System.Threading.Tasks; // For asynchronous programming
using System.Text.Json; // For working with JSON data
using Newtonsoft.Json; // For additional JSON functionality, if needed
using System; // For general functionalities
```

In this section, we import various namespaces necessary for the functionality of our application, including models, HTTP client capabilities, asynchronous programming, and JSON handling.

Next, we define the **`BookingService` class**:

```
namespace BookingConsoleApp.Services
{
    // BookingService class to handle hotel booking operations
    public class BookingService
    {
        private readonly HttpClient _httpClient; // HttpClient instance for making API calls

        // Constructor to initialise HttpClient with API key
        public BookingService(string apiKey)
        {
            _httpClient = new HttpClient(); // Create a new HttpClient instance
            // Add required headers for authentication
            _httpClient.DefaultRequestHeaders.Add("x-rapidapi-key", apiKey);
            _httpClient.DefaultRequestHeaders.Add("x-rapidapi-host", "booking-com15.p.rapidapi.com");
        }
    }
}
```

Here, we encapsulate the functionality within the `BookingService` class, where we define an instance of `HttpClient` to manage our API calls. The constructor sets up this client with the necessary authentication headers.

Now, we will examine the **SearchHotelDetailsAsync** method:

```
public async Task<List<SearchHotel>> SearchHotelDetailsAsync(int destId, string searchType, string arrivalDate, string departureDate)
{
    // Construct the GET request to search for hotel details with the specified parameters
    var response = await _httpClient.GetAsync($"https://booking-com15.p.rapidapi.com/api/v1/hotels/searchHotels?dest_id={destId}&search_type={searchType}&arrival_date={arrivalDate}&departure_date={departureDate}&page_number=1&units=metric&temperature_unit=c&language=de-en-us&currency_code=EUR");

    // Ensure the response indicates success; otherwise, throw an exception
    response.EnsureSuccessStatusCode();

    // Read the response content as a string
    var jsonResponse = await response.Content.ReadAsStringAsync();

    // Parse the JSON response to create a JsonDocument for processing
    var jsonDocument = JsonDocument.Parse(jsonResponse);

    // Initialize a list to hold hotel details
    var hotelDetailList = new List<SearchHotel>();

    // Check if the 'data' property exists in the JSON response
    if (jsonDocument.RootElement.TryGetProperty("data", out JsonElement dataElement))
    {
        // Iterate through each hotel in the 'data' array
        foreach (var hotelJson in dataElement.EnumerateArray())
        {
            // Create a new SearchHotel object and populate its properties from the JSON data
            var hotelDetails = new SearchHotel
            {
                HotelId = hotelJson.GetProperty("hotel_id").GetInt32(), // Extract hotel ID
                Name = CleanupString(hotelJson.GetProperty("name").TryGetProperty("name", out JsonElement nameElement) ? nameElement.GetString() : string.Empty), // Extract and clean up hotel name
                AccessibilityLabel = CleanupString(hotelJson.GetProperty("accessibility_label", out JsonElement accessibilityElement) ? accessibilityElement.GetString() : string.Empty), // Extract and clean up accessibility label
                Price = hotelJson.GetProperty("property").TryGetProperty("price_readmore", out JsonElement priceReadmoreElement) & priceReadmoreElement.TryGetProperty("gross_price", out JsonElement grossPriceElement) ? grossPriceElement.GetProperty("value").GetDouble() : 0, // Extract gross price
                Currency = hotelJson.GetProperty("property").TryGetProperty("price_currency", out JsonElement priceCurrencyElement) & priceCurrencyElement.TryGetProperty("gross_price", out JsonElement grossPriceElement) ? grossPriceElement.GetProperty("currency").GetString() : string.Empty, // Extract currency code
                Rating = hotelJson.GetProperty("property").TryGetProperty("review_score", out JsonElement reviewScoreElement) ? reviewScoreElement.GetDouble() : 0, // Extract review score
                ImageUrl = hotelJson.GetProperty("property").TryGetProperty("photo_url", out JsonElement photoUrlElement) & photoUrlElement.GetString().Length > 0 ? photoUrlElement.GetString() : string.Empty, // Extract image URL if available
                Status = CleanupString(hotelJson.GetProperty("status", out JsonElement statusElement) ? statusElement.GetString() : string.Empty), // Extract and clean up hotel status
            };

            // Add the hotel details to the list
            hotelDetailList.Add(hotelDetails);
        }
    }

    // Return the list of hotel details after processing
    return hotelDetailList;
}
```

In this method, we perform a hotel search based on various parameters. After sending a request to the Booking.com API, we check for a successful response and parse the JSON data to extract details about each hotel, which are then added to a list.

Next, we have the utility method **CleanUpString**:

```
// Utility method to clean up unwanted characters from strings
3 references
private string CleanUpString(string input)
{
    return input.Replace("?", "").Trim(); // Remove '?' and trim whitespace
}
```

This method is a helper that cleans up strings by removing unwanted characters, ensuring that the data we handle is in a proper format.

Now, let's take a look at the **GetHotelPhotosAsync** method:

```
public async Task<List<HotelPhoto>> GetHotelPhotosAsync(int hotelId)
{
    // Construct the URL for fetching hotel photos based on the provided hotel ID
    var url = $"https://booking-com15.p.rapidapi.com/api/v1/hotels/getHotelPhotos?hotel_id={hotelId}&languagecode=en-us";

    // Send a GET request to the specified URL
    var response = await _httpClient.GetAsync(url);

    // Ensure the response indicates success; otherwise, an exception will be thrown
    response.EnsureSuccessStatusCode();

    // Read the content of the response as a string
    var jsonResponse = await response.Content.ReadAsStringAsync();

    // Parse the JSON response to create a JsonDocument for further processing
    var jsonDocument = JsonDocument.Parse(jsonResponse);

    // Initialise a list to hold the hotel photos
    var photosList = new List<HotelPhoto>();

    // Iterate through each photo in the JSON data
    foreach (var photoJson in jsonDocument.RootElement.GetProperty("data").EnumerateArray())
    {
        // Create a new HotelPhoto object and populate its properties from the JSON data
        var photo = new HotelPhoto
        {
            Id = photoJson.GetProperty("id").GetInt32(), // Extract the photo ID
            Url = photoJson.GetProperty("url").GetString() // Extract the photo URL
        };

        // Add the photo to the list of photos
        photosList.Add(photo);
    }

    // Return the list of photos after processing
    return photosList;
}
```

This method retrieves photos for a specific hotel by sending a request to the API. The response is processed in a similar manner as before, extracting photo data and returning it as a list.

Finally, let's review the **SearchFlightsAsync** method:

```
// SearchFlightsAsync
public async Task<List<FlightOffer>> SearchFlightsAsync(string fromId, string toId, DateTime departDate, DateTime returnDate)
{
    // Construct the URL for searching flights with the provided parameters
    var url = $"https://booking-com15.p.rapidapi.com/api/v1/flights/searchFlights?fromId={fromId}&toId={toId}&departDate={departDate:yyyy-MM-dd}&returnDate={returnDate:yyyy-MM-dd}&pageNo=1&adults=1&children=0,17&sort=BEST&cabinClass=ECONOMY&currency_code=AED";

    // Send a GET request to the specified URL to search for flights
    var response = await _httpClient.GetAsync(url);

    // Ensure the response indicates success; otherwise, throw an exception
    response.EnsureSuccessStatusCode();

    // Read the response content as a string
    var jsonResponse = await response.Content.ReadAsStringAsync();

    // Parse the JSON response to create a JsonDocument for processing
    var jsonDocument = JsonDocument.Parse(jsonResponse);

    // Initialise a list to hold flight offers
    var flightOffersList = new List<FlightOffer>();

    // Check if the 'data' property exists in the JSON response
    if (jsonDocument.RootElement.TryGetProperty("data", out var dataElement))
    {
        // Check if the 'aggregation' property exists within 'data'
        if (dataElement.TryGetProperty("aggregation", out var aggregationElement))
        {
            // Iterate through each airline in the 'airlines' array
            foreach (var airline in aggregationElement.GetProperty("airlines").EnumerateArray())
            {
                // Create a new FlightOffer object and populate its properties from the JSON data
                var offer = new FlightOffer
                {
                    Key = airline.GetProperty("iataCode").GetString(), // Extract IATA code of the airline
                    Price = airline.GetProperty("minPrice").GetProperty("units").GetInt32(), // Extract minimum price
                    Currency = airline.GetProperty("minPrice").GetProperty("currencyCode").GetString(), // Extract currency code
                    DurationMin = aggregationElement.GetProperty("durationMin").GetInt32(), // Extract minimum duration
                    DurationMax = aggregationElement.GetProperty("durationMax").GetInt32() // Extract maximum duration
                };

                // Add the flight offer to the list
                flightOffersList.Add(offer);
            }
        }
    }

    // Return the list of flight offers after processing
    return flightOffersList;
}
```

This technique allows the user to search for flights with precise departure and return dates. After establishing the request URL, we inspect the answer as mentioned above, extracting flight or flight information and returning them in a list.

The BookingService.cs file is detailed above, including the methods necessary to successfully search for hotels and flights, acquire hotel images, and process JSON data. Notice how each of the ways is created to connect effectively with the API for receiving and showing information to our users for their travel requirements.

Program.cs

Now, we're moving on to the Program.cs file, which is the core of our Booking Console Application. This file manages the application's configuration, user interactions, and serves as the entry point.

First, let's look at the **namespaces and using directives**:

```
using Microsoft.Extensions.Configuration; // For configuration management
using System;
using System.IO; // For file handling
using System.Threading.Tasks; // For asynchronous programming
using BookingConsoleApp.Services; // For booking service operations
using System.Collections.Generic; // For collection management
using BookingConsoleApp.Models; // For model definitions
using System.Globalization; // For culture-specific operations
using System.Linq; // For LINQ methods like All
using System.Text.RegularExpressions; // For Regex
```

In this section, we import various namespaces essential for configuration management, file handling, asynchronous programming, and our application's models and services.

Next, let's examine the **Main method**, which is where the application starts:

```
namespace BookingConsoleApp
{
    0 references
    class Program
    {
        0 references
        static async Task Main(string[] args)
        {
            // Build configuration from appsettings.json file
            var configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory()) // Set base path to the current directory
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true) // Load configuration settings
                .Build();

            // Retrieve API key from configuration
            string apiKey = configuration["ApiSettings:ApiKey"];

            // Initialise booking service with the retrieved API key
            var bookingService = new BookingService(apiKey);
        }
    }
}
```

Here, we build the application configuration from the appsettings.json file. The API key for our Booking service is retrieved from the configuration, allowing for easy management of sensitive information.

Now, let's look at the **user interaction** part of the application:

```
// Display a welcome message to the user
Console.WriteLine("Welcome to AceBooking - Travel and Accommodation Booking System");

bool continueBooking = true; // Variable to control the menu loop

while (continueBooking) // Start a loop that continues until the user opts to exit
{
    // Display the main menu options to the user
    Console.WriteLine("\nPlease select an option:");
    Console.WriteLine("1. Search Hotel Destination");
    Console.WriteLine("2. Search Hotel Details");
    Console.WriteLine("3. Get Hotel Details");
    Console.WriteLine("4. Get Room Availability");
    Console.WriteLine("5. Get Hotel Reviews");
    Console.WriteLine("6. Get Hotel Photos");
    Console.WriteLine("7. Search Flights");
    Console.WriteLine("8. Search Taxi");
    Console.WriteLine("9. Exit");

    // Prompt the user to enter their choice from the menu
    Console.Write("Enter your choice (1-9): ");
    string choice = Console.ReadLine(); // Read user input
}
```

The following stage adds a loop that presents a menu of alternatives for the user to choose from. That makes the program more dynamic, allowing users to explore via a variety of features.

In other words, Program.cs will be necessary for dealing with the complete flow. It conducts important installs, initialises the services, and gives a user-friendly interface for interacting with the booking features.

Models Folder – HotelSearch.cs

Now, we're moving on to the HotelSearch.cs file, which defines the model for hotel search results in our Booking Console Application, belonging to the BookingConsoleApp.Models namespace.

```
// Namespace declaration for the BookingConsoleApp.Models
namespace BookingConsoleApp.Models
{
    // Definition of the SearchHotels class, representing a hotel in search results
    4 references
    public class SearchHotels
    {
        // Property to store the unique identifier for the hotel
        2 references
        public int HotelId { get; set; }

        // Property to store the name of the hotel
        2 references
        public string Name { get; set; }

        // Property to store accessibility information for the hotel
        2 references
        public string AccessibilityLabel { get; set; }

        // Property to store the price of the hotel
        2 references
        public double Price { get; set; }

        // Property to store the currency in which the price is stated
        2 references
        public string Currency { get; set; }

        // Property to store the rating of the hotel, usually on a scale (e.g., 1-10)
        2 references
        public double Rating { get; set; }

        // Property to store a description of the room offered at the hotel
        0 references
        public string RoomDescription { get; set; }

        // Property to store the URL for the hotel's image
        2 references
        public string ImageUrl { get; set; }

        // Property to store the current status of the hotel (e.g., available, booked)
        1 reference
        public string Status { get; set; }
    }
}
```

The Class SearchHotels aims to collect all relevant information that may be retrieved for a given hotel when a user does a search. It contains the following properties: The HotelId is a unique identifier for each hotel. The hotel's name is: AccessibilityLabel outlines the hotel's accessible qualities. amount is the amount one must pay to stay at the hotel. Currency is the currency in which the price is stated. Rating is the rating of the hotel, often on a scale of 1 to 10; RoomDescription is the specs of the room given by the hotel. The ImageUrl is the URL of the hotel's image, while the Status specifies whether or not the hotel is now accessible. Taken together, these pieces present a full perspective of vital information that customers will have access to in order to make wise selections while searching for an apartment.

HotelDetails.cs

Next, we're moving onto the hoteldetails.cs file.

```
// Namespace declaration for the BookingConsoleApp.Models
namespace BookingConsoleApp.Models
{
    // Definition of the HotelDetails class, representing detailed information about a hotel
    4 references
    public class HotelDetails
    {
        // Property to store the unique identifier for the hotel
        0 references
        public int Ufi { get; set; }

        // Property to store the hotel ID
        0 references
        public int HotelId { get; set; }

        // Property to store the name of the hotel
        2 references
        public string HotelName { get; set; }

        // Property to store the URL for more details about the hotel
        2 references
        public string Url { get; set; }

        // Property to store the address of the hotel
        2 references
        public string Address { get; set; }

        // Property to store the city where the hotel is located
        2 references
        public string City { get; set; }

        // Property to store the country code where the hotel is located
        2 references
        public string CountryCode { get; set; }

        // Property to store the number of available rooms in the hotel
        2 references
        public int AvailableRooms { get; set; }

        // Property to store the average room size of the hotel
        2 references
        public string AverageRoomSize { get; set; }
    }

    // Definition of the HotelDetailsResponse class, representing the response structure for hotel details
    0 references
    public class HotelDetailsResponse
    {
        // Property to indicate the status of the API response
        0 references
        public bool Status { get; set; }

        // Property to store any message associated with the response
        0 references
        public string Message { get; set; }

        // Property to store the timestamp of the API response
        0 references
        public long Timestamp { get; set; }

        // Property to store the hotel details data
        0 references
        public HotelDetails Data { get; set; }
    }
}
```

The HotelDetails.cs file has two classes that serve as models and maintain the individual information for a single hotel. First and foremost, there is a class named HotelDetails that describes several properties that include crucial data for this hotel, such as Ufi, the unique identification, Hotel ID, and the hotel's name. It also has properties that display the hotel's URL address data, such as location, city, and country code, available room count, and average room size. Such a framework enables simple access to and modification of all hotel-specific information. The second class, HotelDetailsResponse, will represent the response structure received from an API query for hotel data. It has attributes that display the response's status, messages (if any), a timestamp, and a Data property that holds an instance of the HotelDetails class. This is how the API response successfully delivers extensive information about a hotel. This file is vital to the application's overall operation, including the display and management of hotel information.

RoomAvailability.cs

Finally, let's move to the RoomAvailability.cs file before we start the demo of our application.

```
// Importing necessary namespace for using generic collections
using System.Collections.Generic;

// Namespace declaration for the BookingConsoleApp.Models
namespace BookingConsoleApp.Models
{
    // Definition of the RoomAvailability class, representing room availability details
    3 references
    public class RoomAvailability
    {
        // Property to store the currency used for pricing
        3 references
        public string Currency { get; set; }

        // Property to store the length of stay in days
        1 reference
        public int LengthsOfStay { get; set; }

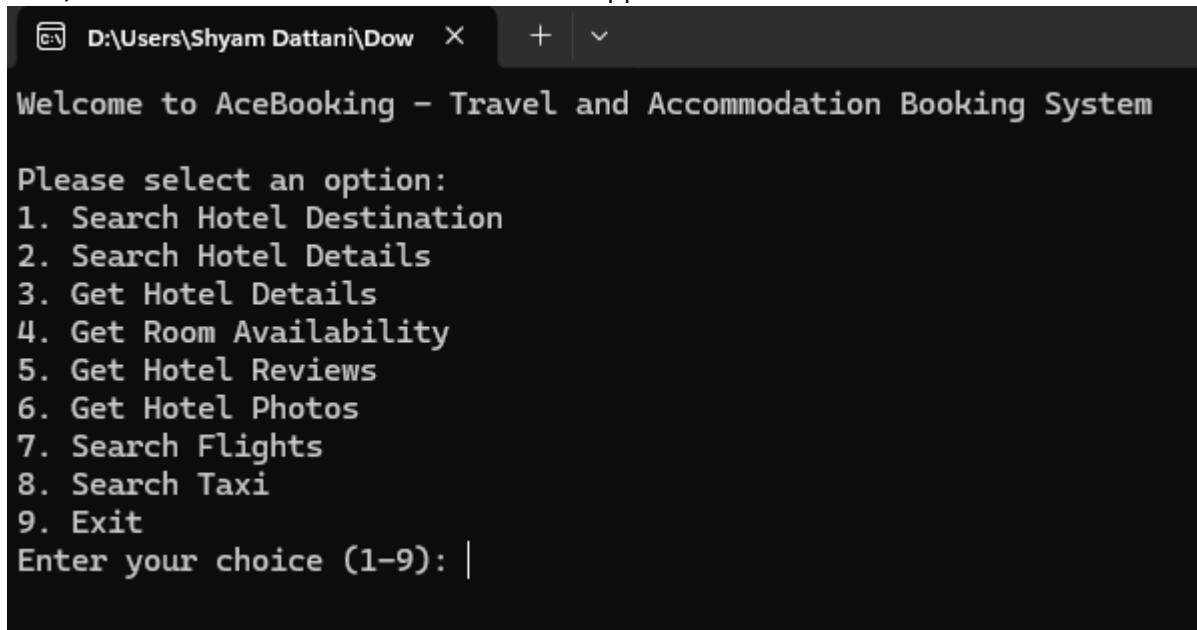
        // Dictionary to store available dates and their corresponding prices
        // Key: Date as a string, Value: Price as a decimal
        3 references
        public Dictionary<string, decimal> AvDates { get; set; } = new Dictionary<string, decimal>();
    }
}
```

The RoomAvailability.cs class offers a model that contains the facts about the availability of rooms in your application. First and first, the namespace must be imported in order to employ generic collections, which is necessary when utilising a type like Dictionary in this class. The RoomAvailability class in the BookingConsoleApp.Models namespace is aimed to capture the most relevant portions of room availability information.

It has a property named Currency, which is meant to maintain the currency in which the price is given. This affords purchasers enough background to grasp the financial ramifications of the lodging expense. It also has a property named LengthsOfStay, which records the history of lengths of stay in days and providing context for how the price is established. It makes substantial use of an AvDates Dictionary, which converts the multiple dates-string representations to their associated prices-decimal representation. This gives for more price flexibility for different dates while also providing room availability information to the user in a stunningly clear and orderly fashion.

Demonstration of the Application

Now, let's move onto the demonstration of the application itself.

A screenshot of a terminal window with a dark background and light green text. The window title bar shows the file path 'D:\Users\Shyam Dattani\ Dow' and standard window controls. The application output reads: 'Welcome to AceBooking - Travel and Accommodation Booking System'. Below this, it says 'Please select an option:' followed by a numbered list of nine options: 1. Search Hotel Destination, 2. Search Hotel Details, 3. Get Hotel Details, 4. Get Room Availability, 5. Get Hotel Reviews, 6. Get Hotel Photos, 7. Search Flights, 8. Search Taxi, and 9. Exit. At the bottom, it prompts 'Enter your choice (1-9):' followed by a vertical bar cursor.

```
D:\Users\Shyam Dattani\ Dow X + v
Welcome to AceBooking - Travel and Accommodation Booking System

Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): |
```

The AceBooking program greets users, stating, "Welcome to AceBooking-Travel and Accommodation Booking System." Then this is the primary menu that appears with choices for people to decide from.

1. Search Hotel Destination - Users may track hotels by location.
2. Search Hotel information: This would need showing information on matching hotels obtained via the user's search.
3. Hotel Details Retrieve detailed information on a certain hotel.
4. Room Availability: Available rooms and prices are returned based on the dates specified.
5. Review Hotel - Collects user reviews to help buyers make educated choices.
6. Hotel Photo Retrieval - Displays hotel photographs for increased visual awareness.
7. Search Flights- Flight choices to supplement your hotel plans.
8. Search Taxi: It identifies sources of transportation for consumers throughout their stay.
9. Exit - Allows the user to exit the app when done with their job.

The "Enter your choice (1-9):" popup seeks the user's decision and brings them to the relevant functionality. Okay, let's look at option 1.

Selected Option 1: Search Hotel Destination

```
D:\Users\Shyam Dattani\Desktop X + v
Welcome to AceBooking - Travel and Accommodation Booking System

Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 1
Enter a destination: Manchester
An error occurred: Response status code does not indicate success: 429 (Too Many Requests).
Press Enter to return to the main menu...
```

As you can see in the CMD, an error message shows on the CMD interface and is also noted in the error_logs.txt file.

Error_logs.txt

```
Timestamp: 09/11/2024 10:50:09; Error: Response status code does not indicate success: 429 (Too Many Requests).; User Inputs: Destination: Manchester
```

The error_logs file as you can see records user inputs, date, and error information. The Error_logs.txt file had all of this logged.

```
Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 1
Enter a destination: manchester
Hotels in manchester:
Destination ID: -2602512
Name: Manchester
Region: Greater Manchester
Country: United Kingdom
Latitude: 53.4812, Longitude: -2.23615
Image URL: https://cf.bstatic.com/xdata/images/city/150x150/976977.jpg?k=8d13c94917fa00569d115c9123c7b5789ad41f7383b6fad32a1bda8e215e8936&o=
-----
Destination ID: 20079942
Name: Manchester
Region: New Hampshire
Country: United States
Latitude: 42.9956, Longitude: -71.4553
Image URL: https://cf.bstatic.com/xdata/images/city/150x150/980138.jpg?k=13118c0c36fd028243e14cd7fbd70f7e9b0364ef8dfc82e82791d3022aac8bc7&o=
-----
Destination ID: 1077
Name: Manchester City Centre
Region: Greater Manchester
Country: United Kingdom
Latitude: 53.47925, Longitude: -2.241755
Image URL: https://cf.bstatic.com/xdata/images/district/150x150/56351.jpg?k=81acc465a1c7b3a58b068901ae18dd157da72b472a9753515b9801dfa1c1f832&o=
-----
Destination ID: 20019012
Name: Manchester
Region: Connecticut
Country: United States
Latitude: 41.7758, Longitude: -72.5219
Image URL: https://cf.bstatic.com/xdata/images/city/150x150/898223.jpg?k=a3493c27ae3c023fe988ee4f8e31c8327954d842326920a732208325d32219b4&o=
-----
Destination ID: 47
Name: Manchester Airport
Region: Greater Manchester
Country: United Kingdom
Latitude: 53.362, Longitude: -2.27344
Image URL: https://cf.bstatic.com/static/img/plane-100.jpg
-----
Press Enter to return to the main menu...
|
```

After picking option 1, the user is requested for a destination. When you key in "Manchester," the application gives a list of hotels near the specified area. The output presents some of the hotels with their details, as seen below:

- Destination ID: Unique identity for each destination.
- Name: This is the name of the location or hotel.

- Area: The wider area within which the hotel is located.
- Country: The destination country.
- Longitude/Latitude: Geographic coordinates for the site.
- Image URL: A link to a photo that shows the hotel or location.

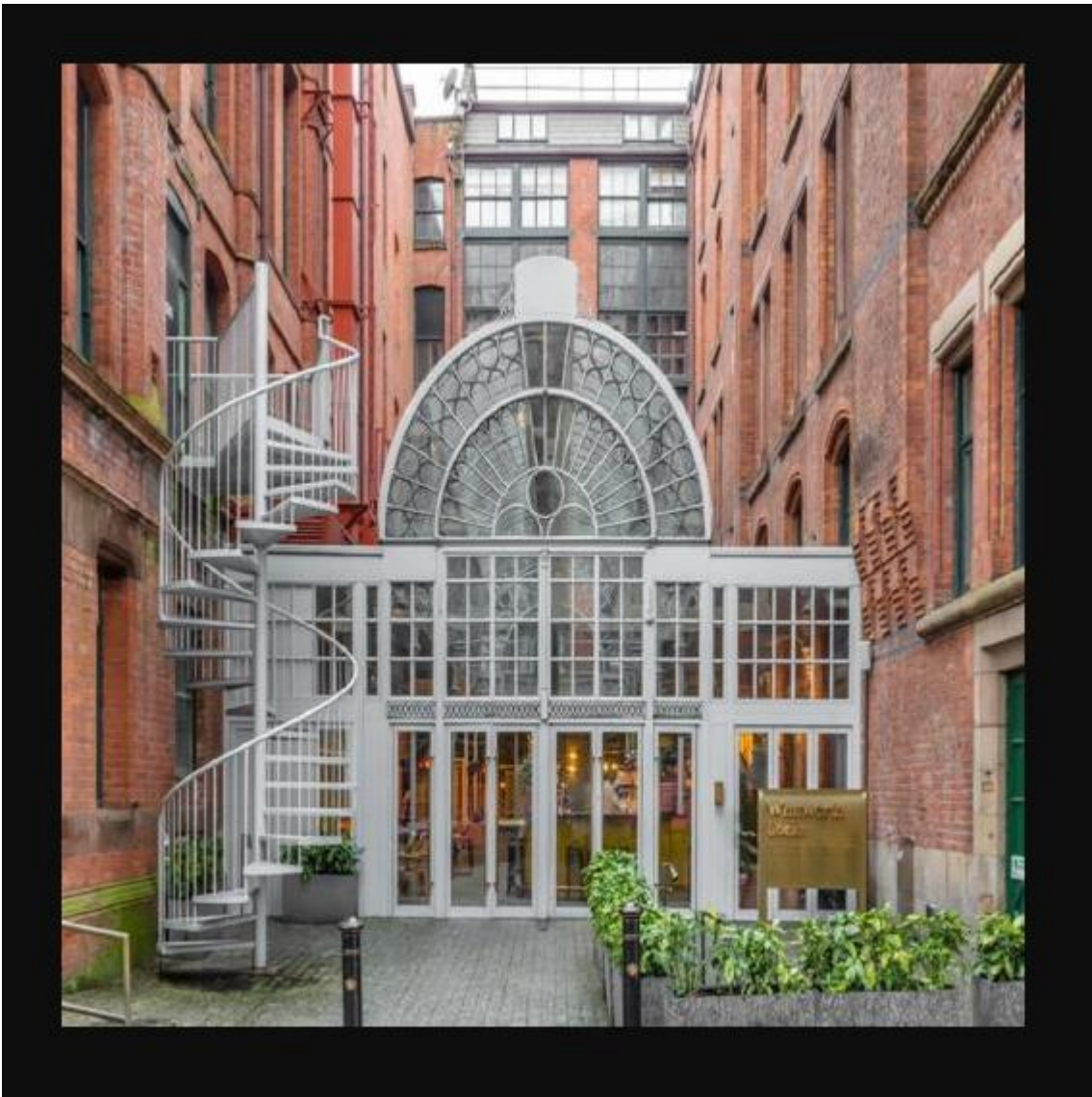
The results show a handful such locations, for example, with their details: "Manchester", Greater Manchester, United Kingdom, and "Manchester", New Hampshire, United States, with accompanying images. After examining the results, the user may click Enter to return to the main menu.

Selected Option 2: Search Hotel Details

Enter your choice (1-9): 2
Enter destination ID: -2682512
Enter search type (e.g., City): city
Enter arrival date (DD-MM-YYYY): 05-11-2024
Enter departure date (DD-MM-YYYY): 26-11-2024
Hotels found:
Hotel ID: 2077759
Name: CitySuites Aparthotel
Accessibility Label: CitySuites Aparthotel.
5 out of 5 stars.
8.6 Excellent 7332 reviews.
?Salford? ?850 m from downtown?.
Entire apartment - 43 m² : 1 bed 1 bedroom 1 living room 1 bathroom.
Original price 26126 AED. Current price 21632 AED..
+120 AED taxes and charges.
Price: 21631.9748267315 AED
Rating: 8.6
Image URL: <https://cf.bstatic.com/xdata/images/hotel/square500/576384976.jpg?k=0daa4a349c60d56eedc3a4f4319abdaf6f1c3d86611c0a264442313a3820f9a2&o=>

Hotel ID: 7678562
Name: Motel One Manchester-St. Peter's Square
Accessibility Label: Motel One Manchester-St. Peter's Square.
8 out of 5 stars.
8.9 Excellent 14971 reviews.
?In downtown?.
Hotel room : 1 bed.
12965 AED.
+100 AED taxes and charges.
No prepayment needed.
Price: 12965.3057361539 AED
Rating: 8.9
Image URL: <https://cf.bstatic.com/xdata/images/hotel/square500/482034988.jpg?k=cd1384d7413a3a10425ff221d6657135c331b3bcce2af172e11f3b80b3582422&o=>

Hotel ID: 1346818
Name: Motel One Manchester-Piccadilly
Accessibility Label: Motel One Manchester-Piccadilly.
8 out of 5 stars.
8.6 Excellent 18678 reviews.
?In downtown?.
Hotel room : 1 bed.
12823 AED.
+100 AED taxes and charges.
No prepayment needed.
Price: 12822.5160254033 AED
Rating: 8.6
Image URL: <https://cf.bstatic.com/xdata/images/hotel/square500/570703239.jpg?k=c42364279333d2f770a24cf2da811196ed2be3317b3579dc65a64413660881f1&o=>



When option 2 is chosen, the Program enables the user to provide the destination ID, search type, arrival date, and departure date. For example, when a user enters destination ID "-2602512" for Manchester, picks "city" as the search type, and adds the arrival and departure dates, the computer produces a list of hotels that fulfil the relevant parameters. If they wish to, they can also click on the image URL, which will send them to the browser for them to view hotel.

The results for each hotel are as follows:

- Hotel ID - A unique identifier for each hotel.
- Name: The name of the hotel will be revealed.
- Accessibility Label: This is a description of the hotel in general.
- Rating: This is the rating score out of 5 stars, followed by the number of customer reviews, for example, "8.6 Excellent" plus the number.
- Location: This characteristic would display if a hotel is near downtown or other locations.
- Room Specifics: Accommodation type, size, number of bedrooms, and bathrooms.
- Price details include the beginning price, current price, taxes & levies, and final price in AED.
- Img Url: A hotel's picture URL.

For example, the result offers information for the "CitySuites Aparthotel" with an 8.6 rating and a current price of 21,632 AED, as well as other alternatives such as "Motel One Manchester-St. Peter's Square" and "Motel One Manchester-Piccadilly." Each hotel item is split by a line to create clarity and allow the user to more simply examine their options. If the user wishes to click on the Image URL, then they are able to do so, by clicking on the URL which will take them to the browser and it will show the image of the hotel, like shown above. Similarly, as before, the user has the option of returning to the main menu.

Selected Option 3: Get Hotel Details

```
Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 3
Enter hotel ID: 2077759
Enter arrival date (DD-MM-YYYY): 05-11-2024
Enter departure date (DD-MM-YYYY): 29-11-2024
Hotel Name: CitySuites Aparthotel
Address: 16 Chapel Street
City: Manchester
Country: gb
Available Rooms: 3
Average Room Size: 9.77 m²
Hotel Link: https://www.booking.com/hotel/gb/citysuites.html
Press Enter to return to the main menu...
```

After choosing option 3, the client is required to give hotel ID, arrival date, and departure date. For example, by entering "2077759" - "CitySuites Aparthotel" - and the dates of arrival "05-11-2024" and "29-11-2024," the computer retrieves comprehensive information about the specified hotel.

The result shows the following information:

- Hotel Name: Just the name of the hotel, such as "CitySuites Aparthotel".
- Street Address: The hotel's street address is "16 Chapel Street".
- City: You may indicate the city where the hotel is situated e.g., Manchester.
- Country: This is the country's code; for example, gb denotes Great Britain.
- Rooms Available: The number of rooms that are now available for booking. Example: "3".
- Specify the average room size in metric units, such as "9.77 m²".
- Hotel Link: This may be a URL link to the hotel's website with extra information. Exemplary URL: "https://www.booking.com/hotel/gb/citysuites.html" .

Simply tapping the Enter key would allow the user to access the main menu

Selected Option 4: Get Room Availability

```
Welcome to AceBooking - Travel and Accommodation Booking System

Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 4
Enter hotel ID: 48347
Enter minimum date (DD-MM-YYYY): 11-11-2024
Enter maximum date (DD-MM-YYYY): 20-11-2024
Currency: USD
Date: 2024-11-11, Price: 128.105999425358 USD
Date: 2024-11-12, Price: 197.981999111918 USD
Date: 2024-11-13, Price: 142.339999361509 USD
Date: 2024-11-14, Price: 109.989999506621 USD
Date: 2024-11-15, Price: 160.455999280247 USD
Date: 2024-11-16, Price: 241.977998914566 USD
Date: 2024-11-17, Price: 95.75599957047 USD
Date: 2024-11-18, Price: 109.989999506621 USD
Date: 2024-11-19, Price: 142.339999361509 USD
Date: 2024-11-20, Price: 169.513999239616 USD
Enter card number: 6745689218367012
Enter expiry date (MM/YY): 02/25
Enter CVV: 369
Payment processed and recorded.
Thank you for Booking with Ace Booking
Press Enter to return to the main menu...
```

Payments.txt file:

```
Hotel ID: 48347, Start Date: 2024-11-11, End Date: 2024-11-20, Card Number: 6745689218367012, Expiry Date: 02/25, CVV: 369
```

If option 4 is chosen, the computer will need the user to provide the Hotel ID as well as the minimum and maximum dates of their stay. For example, if a user enters Hotel ID 48347 with a minimum of 11-11-2024 and a maximum of 20-11-2024, the computer will return and show hotel pricing in USD for each date within the needed time range. Once a valid card number, expiration date, and CVV information are supplied, the program confirms that the payment was registered and processed appropriately. As you can see above, the booking information is logged to payments.txt for future reference and verification of the transaction. Also, above is the error_logs.txt which contains the error logs. Finally, it thanks the client for using Ace Booking and allows them to return to the main menu if they wish.

Selected Option 5: Get Hotel Reviews

```
Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 5
Enter hotel ID: 2877759
Hotel Reviews:
Title: Exceptional
Pros: Location and convenience
Cons:
Date: 2024-11-02 23:50:01
Rating: 4/5
Purpose: leisure
-----
Title: excellent!
Pros: excellent location and great staff
Cons: it was so hot! we turned the heating down but was quite uncomfortable in the night
Date: 2024-11-02 22:07:54
Rating: 3/5
Purpose: leisure
-----
Title: This is one of the best hotels I have stayed. Excellent!
Pros: The location, the room very clean, the size of the room, the staff very friendly almost everything is perfect
Cons: Takes like close to an hour before you are being served during breakfast but the quality of the food compliment the waiting
Date: 2024-11-01 23:13:24
Rating: 3/5
Purpose: business
-----
Title: Excellent
Pros: Really close to lots of amenities and the hotel was remarkable.

Cleanest (felt brand new) - both times we used the pool it was empty and immaculate. Warm and 1.2 m deep - ideal for my 9yr old learning to swim. Breakfast was brilliant - good choice and absolutely amazing service.
Cons: 100 damage deposit taken at check in- however this was given back within an hour of check out.
Date: 2024-11-01 22:22:33
Rating: 3/5
Purpose: leisure
-----
Title: Exceptional
Pros: Great facilities and lovely staff
Cons:
Date: 2024-11-01 22:05:05
Rating: 4/5
Purpose: leisure
-----
```

The fifth option asks the user to submit a hotel ID, following which the system shows a list of hotel reviews. Each review gives a title, the benefits and negatives of the stay, the date, the rating, and the purpose of the visit. One is labelled "Exceptional" and boasts how everything is amazing, including the location and convenience; another laments his nighttime misery. These evaluations offer thorough input on the visitors' experiences and are highly important for any future buyers. If the user wish to return to the main menu, they may.

Selected Option 6: Get Hotel Photos

```
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 6
Enter hotel ID: 2077759
Hotel Photos:
ID: 576384976, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576384976.jpg?k=0daa4a349c60d56eedc3a4f4319abdaf6f1c3d86611c0a264442313a3820f9a2&o=
ID: 131175953, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/131175953.jpg?k=bc8f039835c6e85a05d177f8ae46ac52a20fe1877ea3fac2488f7d1d306cc1c2&o=
ID: 263927245, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263927245.jpg?k=42f86de8dbe1ba5f8aaafbbff8aab17a853f79223c9579822340236aefdec6ec9&o=
ID: 576387334, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576387334.jpg?k=c295d9bc198a490e87a3facb82f14112a3b99cd458aa568752c0da3a6e0cf59a&o=
ID: 263927243, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263927243.jpg?k=ed1ad8e73af16ac3721bca1709261013a4e404645a376add5cf52d39ef1290d7&o=
ID: 576386883, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576386883.jpg?k=b26a5e47079637d1bca674260798158e10512ffe8f5bab3483f45dc34ee3b154&o=
ID: 576386884, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576386884.jpg?k=04edbe44fccc0ec1d691660f8002e437d4c41211a83ba3abb95618c57cf624fd&o=
ID: 263927381, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263927381.jpg?k=991a1fd8db195a997f2e0256c97d64b5c86d8cbba846bef28d555e464667c59&o=
ID: 263928462, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263928462.jpg?k=2fa8d568d7785b910b33ecbd724097454cf257282ef36a925631a37a5cd327c9&o=
ID: 576390827, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576390827.jpg?k=c5b95e6388a8f747d9db4f84c8080f9d6a2d4a82c2ba4915ca5df9c45be960cf&o=
ID: 576384217, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576384217.jpg?k=adff9ea53c55fa08eea7c0edd6f50d492029bc53ead0c9ca3d4826eb130232fc&o=
ID: 576384477, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576384477.jpg?k=dc9f7b48edac77202f4a3922c11fc75eaf642b298d2e37da58021f1cf5a15a85&o=
ID: 576385946, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576385946.jpg?k=63f7d6b42f0c06885df908c5764f5c313e3fad0f7242a74d3315bf12799ab9d&o=
ID: 576386349, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576386349.jpg?k=7f62b7de911533673b2e8d373def50dd670b7a6b84f472ad6f0a09d078284dae&o=
ID: 205823901, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/205823901.jpg?k=b88185449471e06b01628b29b943ec83ddff19f4cf9f26a2bdf04b157a1b9b2ae&o=
ID: 263928064, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263928064.jpg?k=1c4d212673c9bfe2a1076db634d6b1ecf6dcf3bf12cfeea62ea7fa00b257ee9a&o=
ID: 263927895, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263927895.jpg?k=85d1b94773ed8e43e311e75297a1577a6e8f4629793a574e1781a61baeb9504a&o=
ID: 263927043, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263927043.jpg?k=c9bf107e4710eb697a6524db2106139d6274b8e26866c9c107546724c075d215&o=
ID: 263926965, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926965.jpg?k=a83b8afe5dd82fabe49139a2f69b127764b1a0135ba18999ab98d20607451cd&o=
ID: 263926971, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926971.jpg?k=dcf5e4844999b9ea74d8bf2b777fe614f1ddfd2668d36cae807591290c1677834&o=
ID: 263926618, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926618.jpg?k=135201be02adb9d49ed6a09b4af915510e8ebe71b2e8b898554780550b0a555b&o=
ID: 263926623, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926623.jpg?k=d31cfff746d8f869927480d07707c4599b978b14705b85c719dd06f196ad6a382&o=
ID: 263926625, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926625.jpg?k=17d6abe387f688e42e7d1acc23efa16d8d67297c7933f480ee266b22f421ba87&o=
ID: 263926631, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926631.jpg?k=5f188470750699b9eba06d2c202086c249f2c39b25a39b165c6aecd1ad0c1b0d08&o=
ID: 263926642, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926642.jpg?k=6220c751d444c4538317451b40d560965109b139995107bba2da46c46c1dbdf&o=
ID: 263926645, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926645.jpg?k=1b554cd5641f3d52ac8a03da7359f6dbdf3ba9dbd5d6c3386097f2bf8d14f057&o=
ID: 263926650, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926650.jpg?k=3eac0281cea91a6bc79379cc38417ab6b65811c050139ed11268035d6939f0ec&o=
ID: 263926662, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926662.jpg?k=01d9e95c0d3bd21fb06e152b3327fcbce9c358d1e935ed1467dbab4a4846d33c&o=
ID: 205821772, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/205821772.jpg?k=26c6850ef7c68162a343f17abc78775a79a7491370c9ae0e225c121d028d3cd&o=
ID: 205825779, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/205825779.jpg?k=9af11993c2f4dc60ae9445d72504991ba703f5fcfe17a0028c0bb66719dbe1df&o=
ID: 576389103, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/576389103.jpg?k=8337c0f66c62b75803e38951e4a23b845dee2a9cfa704730626ec0947d3d59ae&o=
ID: 186409219, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/186409219.jpg?k=431f2ba1fb80ec8317f6bed6e91d2f4a28a07d4bd4d1ce92df4f00c58b710538&o=
ID: 263926636, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/263926636.jpg?k=17068ffab65b021e692b75deec78b7ade0ec8b8da504f30acf5d7a6505a19bd95&o=
ID: 205821788, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/205821788.jpg?k=2428703694a59047eb07c426ea9b9f198da1ab4443de524cb9d3392e10ca178a&o=
ID: 228116180, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/228116180.jpg?k=f8697fe2496418a105875f9941a8a9aea02c61c7cfb2eaf82d8e965235d3e7822&o=
ID: 205825768, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/205825768.jpg?k=3cb87b1d9a558bd4f72cfe2daee487670ecfcadd6f54b0dc03923585bbb0461a&o=
ID: 205824739, URL: https://cf.bstatic.com/xdata/images/hotel/square1024/205824739.jpg?k=98ff85598b0fde661ae783080d79d6228fdd0e2a9089bbe9716268716c3b3932&o=
```

This section illustrates the AceBooking app's "Get Hotel Photos" functionality. The program obtains a hotel id from the end user and requests images of the hotel using this ID. After providing the hotel ID, the program offers a selection of hotel photos with unique IDs and URLs. This would improve the user experience by delivering a visual depiction of each hotel while also supplying prospective customers with a better idea of the stay to be booked. This tool will improve the search process by allowing users to visually examine possibilities. This option allows the user to return to the main menu if that is what they prefer.

Option 7: Search Flights

```
Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 7
Enter departure location ID: BOM.AIRPORT
Enter arrival location ID: DEL.AIRPORT
Enter departure date (DD-MM-YYYY): 11-11-2024
Enter return date (DD-MM-YYYY): 25-11-2024
Searching for flight offers, please wait...
Flight Offers:
Key: SG
Price: 1431 AED
Duration (Min - Max): 3 - 27 hours
-----
Key: UK
Price: 1113 AED
Duration (Min - Max): 3 - 27 hours
-----
Key: AI
Price: 1121 AED
Duration (Min - Max): 3 - 27 hours
-----
Key: IX
Price: 1145 AED
Duration (Min - Max): 3 - 27 hours
-----
Key: 6E
Price: 1113 AED
Duration (Min - Max): 3 - 27 hours
-----
Press Enter to return to the main menu...
|
```

In this chapter, I will look at the logic that supports flight searches inside the Booking Console application. The user then picks option 7 from the main menu, which asks them to enter the departure location ID, "BOM.AIRPORT," followed by the arrival location ID, "DEL.AIRPORT." Then he announces the departure date as "11-11-2024" and the return date as "25-11-2024." After choosing Search, the program examines the request and displays possible flight possibilities. In summary, the results offer essential flight identifiers: SG, 1431 AED, 3-27 hours; UK, 1113 AED, 3-27 hours; AI, 1121 AED, 3-24 hours; IX, 1145 AED, 3-27 hours; and 6E, 1113 AED, 3-27 hours. This allows the user to explore all available flights before pressing Enter to return to the main menu, exhibiting the power of flight search.

Option 8: Search Taxi

```
Please select an option:
1. Search Hotel Destination
2. Search Hotel Details
3. Get Hotel Details
4. Get Room Availability
5. Get Hotel Reviews
6. Get Hotel Photos
7. Search Flights
8. Search Taxi
9. Exit
Enter your choice (1-9): 8
Enter pick-up location ID: ChIJRym9mVDI5zsRrqh0xGAazB4
Enter drop-off location ID: ChIJ____b8DR5zsRVz_XpIUEKcA
Enter pick-up date (DD-MM-YYYY): 05-11-2024
Enter pick-up time (HH:MM in 24-hour format): 13:00
Searching for taxi offers, please wait...
Taxi Offers:
Result ID: 41453768-1827-4839-a539-7d826690ffe1
Vehicle Type: STANDARD
Description: Perfect for solo travellers, couples and small families.
Supplier Name: Transferz
Passenger Capacity: 3
Price: 18.12 EUR
Duration: 54 minutes
Image URL: https://cdn.rideways.com/images/cars/standard.jpg
Meet & Greet: True
-----
Result ID: d9d70d78-4736-4fa7-9fc3-6f72ebf28423
Vehicle Type: LARGE
Description: Great choice for families.
Supplier Name: Transferz
Passenger Capacity: 4
Price: 21.78 EUR
Duration: 54 minutes
Image URL: https://cdn.rideways.com/images/cars/people-carrier.jpg
Meet & Greet: True
-----
Result ID: 95066700-345a-4f1a-a2e0-6a3b52a67644
Vehicle Type: STANDARD
Description: 100% Electric
Supplier Name: Travelodesk India
Passenger Capacity: 3
Price: 25.26 EUR
Duration: 54 minutes
Image URL: https://cdn.rideways.com/images/cars/standard.jpg
Meet & Greet: True
-----
```

Here, I'll explain how the taxi search works inside the Booking Console Application. It then prompts the user to submit information after picking option 8 from the menu. The pick-up location ID is "ChIJRym9mVDI5zsRrqh0xGAazB4," but the drop-off location ID is "ChIJ____b8DR5zsRVz_XpIUEKcA." The pickup date is "05-11-2024," with a pickup time of "13:00." In this case, the user begins the search, and the program processes his request by delivering a list of taxi choices. Various results comprise the result ID, vehicle type, description, supplier name, passenger capacity, price in EUR, and anticipated duration. Example offers are: The first option is a STANDARD Transferz vehicle perfect for lone travellers, couples, and small families. Capacity for three individuals, price of 18.12 EUR, and length of 54 minutes. Another possibility is the Transferz LARGE vehicle, which can transport four individuals for 21.78 EUR. Then there's Travelodesk India's 100% electric STANDARD car, which seats three people and costs 25.26 EUR. Each package shows whether a meet and greet service is included. The user may either continue with their earlier option of taxi availability or return to the home menu.

Option 9: Exit

```
Please select an option:  
1. Search Hotel Destination  
2. Search Hotel Details  
3. Get Hotel Details  
4. Get Room Availability  
5. Get Hotel Reviews  
6. Get Hotel Photos  
7. Search Flights  
8. Search Taxi  
9. Exit  
Enter your choice (1-9): 9  
Press any key to exit...
```

In the last part, I'll explain the Booking Console Application's exit functionality. In this regard, after the user chooses option 9 from the main menu, the program will challenge them to confirm their decision to end the application. It will enable the user to end their session by providing the message "Press any key to exit." In this approach, any pushed key will finish the running program, allowing the user to exit the present application instantly and easily. This now enables users to stop the program quickly after they have done their obligations, increasing the overall experience and usefulness of the Booking Console tool.

Thank you for taking the time to review my demonstration; I hope it was informative and presented you with some guidance on how to use the AceBooking - Travel and Accommodation Booking System.