**Wellness Retreat Booking System**

**Aim of the Project:**

The aim of this project is to design a functional system for wellness retreat bookings that simplifies the process for users, enabling them to seamlessly select available time slots, provide guest details, and secure bookings at their preferred locations. It intends to provide a well-structured and user-friendly interface where customers can easily input relevant booking details and receive feedback on their bookings instantly. The core objective is to automate the booking process, minimize errors, and offer basic functionality like saving and retrieving booking details, ensuring that a user’s booking experience is smooth and hassle-free.

Another key aim is to ensure the system integrates proper error handling mechanisms, which account for invalid inputs and unforeseen disruptions, ultimately providing a safe and reliable booking platform. This project also aims to incorporate storage of bookings via file handling so that users can refer back to their bookings. Through these objectives, the project is expected to streamline wellness retreat bookings and reduce the administrative burden of handling manual bookings.

**Business Problem or Problem Statement:**

In today’s busy world, managing wellness retreat bookings can be a complicated and time-consuming process for both organizers and participants. Manual booking methods often result in double-bookings, missed appointments, or incomplete data, which causes friction in the client experience. Clients expect to have a seamless digital experience when booking wellness services, which includes clear time slot availability, easy payment methods, and reliable confirmation processes. On the other hand, businesses require an efficient system that minimizes errors and ensures customer satisfaction without overwhelming staff with manual scheduling tasks.

The problem at hand is to develop a streamlined, automated booking system that can handle multiple customer bookings, ensure no overlaps, store information securely, and provide clear feedback to users. Without such a system, wellness retreat businesses risk mismanaging client data, overscheduling, or losing revenue due to inefficient booking methods. Thus, a digital booking system that efficiently addresses these pain points is critical to improving overall business operations and customer satisfaction.

**Project Description**

This project presents a digital system for handling bookings at wellness retreats. It allows users to input details like the number of guests, select from predefined time slots, enter payment information, and choose a location. The system then stores the booking details securely in text files, ensuring data persistence, and alerts users if a booking with the same user ID already exists to avoid duplications. The interface is simple yet functional, making it accessible to users who may not be tech-savvy.

The system is designed to have two main components: user interaction and file handling. During interaction, users are guided through the booking process with clear prompts, ensuring they can easily complete the booking steps. Once the booking is completed, the system saves the information in a file for future reference. By structuring the bookings into text files, it also ensures that each booking is independent and traceable by user ID. Additionally, the program incorporates basic input validation, ensuring users cannot enter negative guest numbers or invalid time slots.

**Functionalities**

The system supports a range of functionalities that allow users to complete their bookings smoothly. These include:

1. **User Interaction and Input**: Users are prompted to input their details, including the number of guests, time slot selection, payment details, and location. The system ensures that the input provided is valid and follows the expected format. Invalid inputs, such as selecting unavailable time slots or entering non-numeric guest counts, are caught, and users are alerted accordingly.
2. **File Handling**: Bookings are saved in a text file that is uniquely named based on the user ID. The system also checks if a booking already exists for that user ID to avoid duplication. This ensures a well-organized booking record for future reference.
3. **Time Slot Management**: The system presents a list of available time slots for users to select from, ensuring no conflicts in scheduling and a clear overview of booking options.
4. **Error Handling and Validation**: The program includes robust validation processes to prevent incorrect inputs. For instance, it handles exceptions such as invalid time slot selection, negative numbers for guest count, and missing inputs.
5. **User-Friendly Interface**: The step-by-step booking process is user-friendly, with prompts guiding users through each phase, making it easy for anyone to book a session.

**Input Versatility with Error Handling and Exception Handling:**

One of the project’s key focuses is to ensure that input versatility is paired with effective error handling and exception management. The system is designed to accept a wide range of inputs, while also validating that the inputs conform to expected formats. For instance, the system checks if the guest number input is a positive integer and ensures that users only select valid time slots from the list provided.

If users input invalid data, the system immediately alerts them and provides them with an opportunity to correct the input. For example, if a user selects a time slot that doesn’t exist or enters non-numeric values for the guest count, the system does not proceed with the booking process. Instead, it guides the user to make the correct selection or input.

Error handling is also extended to file operations. For instance, the system checks whether a booking file already exists for a given user ID before proceeding with saving new data. This helps avoid overwriting existing bookings and ensures data integrity. Any file handling errors, such as permission issues or non-existent directories, are caught, and meaningful messages are displayed to users to address the issue.

**Code Overview**

* **Class Definition (WellnessRetreat)**: This class encapsulates all functionality related to a wellness retreat booking. It holds methods for creating, displaying, and saving bookings to files.
  + **Attributes**:
    - available\_time\_slots: A list of predefined time slots available for booking.
    - \_\_init\_\_(): Initializes the booking with the user’s ID, number of guests, selected time slot, payment details, and location.
  + **Methods**:
    - display\_details(): Displays the booking information.
    - save\_to\_file(): Saves booking details to a text file in the bookings/ directory. It checks if a booking already exists with the same user\_id to prevent duplicate bookings.
    - create\_from\_input(): A static method that handles user input, validates it, and returns the input details for the booking.
* **Main Program Loop**: This loop presents the user with options to create a new booking or exit the system. If the user opts to create a booking, it gathers input using create\_from\_input() and processes the booking with the WellnessRetreat class methods.

**Code Implementation:**

import os

# Class for handling wellness retreat bookings

class WellnessRetreat:

available\_time\_slots = [

'09:00 AM',

'10:00 AM',

'11:00 AM',

'01:00 PM',

'02:00 PM',

]

def \_\_init\_\_(self, user\_id, guests, time\_slot, payment\_details, location):

self.user\_id = user\_id

self.guests = guests

self.time\_slot = time\_slot

self.payment\_details = payment\_details

self.location = location

def display\_details(self):

print("\nBooking Details:")

print(f"User ID: {self.user\_id}")

print(f"Guests: {self.guests}")

print(f"Time Slot: {self.time\_slot}")

print(f"Payment Details: {self.payment\_details}")

print(f"Location: {self.location}")

def save\_to\_file(self):

try:

# Ensure directory exists

if not os.path.exists("bookings"):

os.makedirs("bookings")

# Check if the booking file already exists

if os.path.exists(f"bookings/booking\_{self.user\_id}.txt"):

print("Booking with this User ID already exists. Please choose a different User ID.")

return

# Save booking details to a file

with open(f"bookings/booking\_{self.user\_id}.txt", "w") as file:

file.write(f"User ID: {self.user\_id}\n")

file.write(f"Guests: {self.guests}\n")

file.write(f"Time Slot: {self.time\_slot}\n")

file.write(f"Payment Details: {self.payment\_details}\n")

file.write(f"Location: {self.location}\n")

print("Booking saved successfully.")

except Exception as e:

print(f"An error occurred while saving the booking: {e}")

@staticmethod

def create\_from\_input():

try:

user\_id = input("Enter User ID: ")

guests = input("Enter number of guests: ")

if not guests.isdigit() or int(guests) <= 0:

print("Please enter a valid number of guests.")

return None

print("Available time slots:")

for i, slot in enumerate(WellnessRetreat.available\_time\_slots, start=1):

print(f"{i}. {slot}")

time\_slot\_index = int(input("Select a time slot by number: ")) - 1

if time\_slot\_index < 0 or time\_slot\_index >= len(WellnessRetreat.available\_time\_slots):

print("Invalid selection. Please try again.")

return None

time\_slot = WellnessRetreat.available\_time\_slots[time\_slot\_index]

payment\_details = input("Enter payment details (Card/Paypal etc.): ")

location = input("Enter preferred location: ")

return user\_id, guests, time\_slot, payment\_details, location

except Exception as e:

print(f"An error occurred during booking: {e}")

return None

# Main program loop

def main():

while True:

try:

print("\nWelcome to the Wellness Retreat Booking System")

print("1. Create a new booking")

print("2. Exit")

choice = int(input("Enter your choice: "))

if choice == 1:

details = WellnessRetreat.create\_from\_input()

if details is not None:

user\_id, guests, time\_slot, payment\_details, location = details

booking = WellnessRetreat(user\_id, guests, time\_slot, payment\_details, location)

booking.display\_details()

booking.save\_to\_file()

elif choice == 2:

print("Exiting the system. Have a great day!")

break

else:

print("Invalid choice, please try again.")

except ValueError:

print("Invalid input, please enter a number.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Key Features of Code:**

* **Time Slot Selection**: Users are presented with a list of available time slots and can select their preferred time by number.
* **Input Validation**: Ensures that inputs such as the number of guests and time slot selection are valid before proceeding.
* **File Handling**: Ensures bookings are saved in a specific folder, checking for duplicates based on user\_id.
* **Error Handling**: Catches exceptions during user input and file operations, ensuring that errors are handled gracefully without crashing the program.

This system is a basic implementation of a real-world booking system, with room for extensions such as adding a GUI, integrating a database, or expanding functionality (e.g., editing or canceling bookings).

**Results and Outcomes:**

The system successfully implements a booking solution for wellness retreats that automates the entire process from user input to file storage. The project results demonstrate the effectiveness of object-oriented design in solving a practical problem, with the added benefit of input validation and error handling to enhance user experience. Each booking is saved as a text file, ensuring that users and the retreat organizer can access booking details in a structured manner.

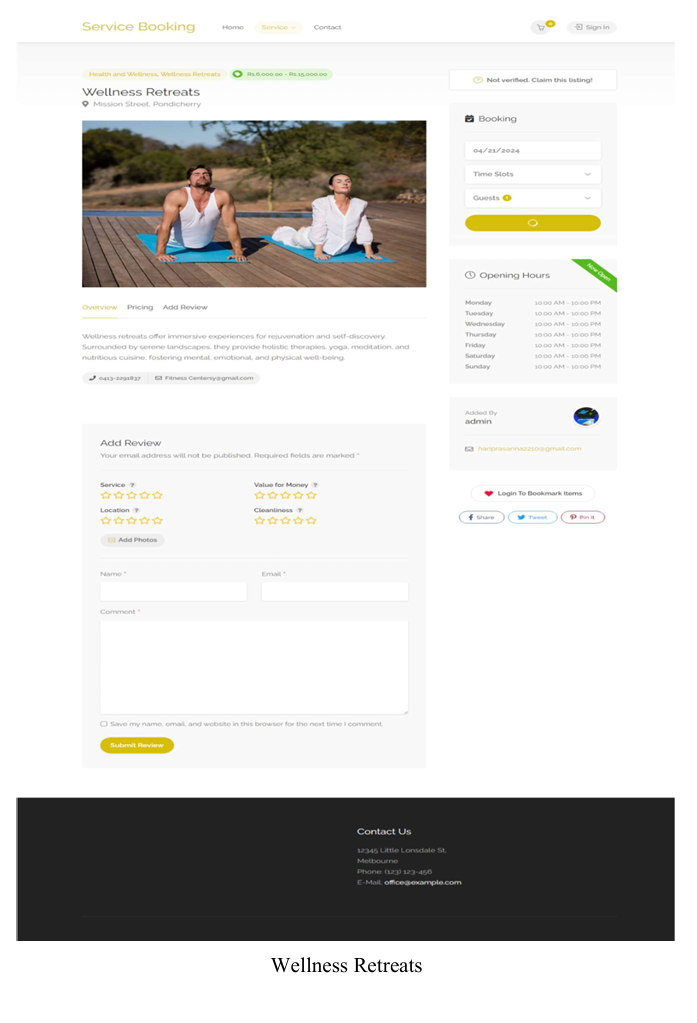
The outcome also highlights the importance of error handling. By validating user inputs, the system avoids common pitfalls like scheduling conflicts, invalid data entries, and file-handling errors. Users are informed of issues in real-time, allowing them to correct mistakes immediately. This provides a robust system that is user-friendly and minimizes potential for errors.

**Conclusion:**

In conclusion, this project addresses the challenges associated with manual booking systems by providing a simple, automated solution for wellness retreat bookings. By leveraging Python’s file handling and input validation capabilities, the system delivers a smooth and reliable user experience. The integration of input validation, error handling, and exception management ensures that users have minimal friction when entering their booking details.

Ultimately, the project’s success lies in its ability to reduce administrative load on wellness retreat businesses while providing a positive experience for clients. The flexibility of the system also allows for future enhancements, such as adding online payment integrations or expanding the available time slots dynamically. This booking system serves as a solid foundation for improving operational efficiency and customer satisfaction in wellness retreat operations.

**Outlook :**

****