KING ABDULAZIZ UNIVERSITY - FACULTY OF ENGINEERING - DEPARTMENT OF INDUSTRIAL ENGINEERING

Computer Applications in Industrial Engineering 1 (IE-322)

Final Report U Hotel

M	Name	ID
1	Abdullah Khalid Bakhsh	2237067
2	Ibraheem Abdulaaly Alshaikh	2242744
3	Abdulrahman Hani Althinayyan	2241794
4	Zeyad Saleh Althinayyan	2339104
5	Mohammed Almarghalani	2236407

Instructor: Dr Mohammed Atif Shahzad

Section: FA

Submitted on 4/5/2025





Table of Contents

Introduction

The U Hotel Room Reservation App is a Python-based desktop application that provides a simple and functional solution for room booking in a hotel-like setting. Designed using the Tkinter library, this lightweight application simulates real-world hotel booking operations including guest registration, room selection, date-based stay planning, and dynamic pricing with discounts.

The app allows users to input their name, choose room types (Standard, Deluxe, Suite), select check-in and check-out dates, and optionally apply a discount code. This solution is ideal for learning how to create GUI-based reservation systems using Python without relying on complex databases or web frameworks. It provides a user-friendly interface for hotel guests and is a valuable prototype for small-scale hotels or training scenarios.

Setting up the Python Application

1) Install Python

   First, make sure Python is installed on your computer. It's best to use version 3.8 or later. You can download it from python.org.

2) Install the Calendar Tool

   The app uses a tool called tkcalendar to let you pick dates easily. To install it, open your terminal or command prompt and type:

   pip install tkcalendar

3) Open the App in an Editor

   You can open the Python file using any editor you like—Visual Studio Code, PyCharm, or even Python's built-in editor (IDLE).

4) Run the App

   In the terminal, go to the folder where your file is saved and run:

   python filename.py

   Replace filename.py with the actual name of your file.)

5) How It Works

   The app uses Tkinter to build the interface and tkcalendar for choosing dates. You don't need any extra setup like a database or a server. That means it's easy to run on any computer that has Python installed.

Function Definitions

```
# Room prices per night
room_prices = {
    "Standard": 100,
    "Deluxe": 150,
    "Suite": 200
}

def calculate_total_price(nights, room_type, discount_percent=0):
    price_per_night = room_prices.get(room_type, 0)
    total_before_discount = nights * price_per_night
    discount_amount = total_before_discount * (discount_percent / 100)
    total_after_discount = total_before_discount - discount_amount
    return total_before_discount, discount_amount, total_after_discount
```

- The system's main logic is handled by two key functions. One of them, calculate_total_price(nights, room_type, discount_percent), is responsible for computing the total cost of a guest's stay. It uses a predefined dictionary called room_prices that lists the nightly rates for three room types: Standard, Deluxe, and Suite. This function takes in the number of nights, the type of room selected, and an optional discount percentage. It then calculates the total amount before any discount, figures out how much the discount saves, and determines the final price after the discount is applied. Finally, it returns all three values for further use in the reservation process.

```
def reserve_room():
    guest_name = entry_name.get().strip()
    room_type = room_var.get()
    check_in_date = check_in.get_date()
    check_out_date = check_out.get_date()
    entered_code = entry_discount.get().strip()

    if not guest_name or room_type == "Select a room":
        messagebox.showwarning("Warning", "Please enter your name and select a room type.")
        return

    if check_out_date <= check_in_date:
        messagebox.showerror("Error", "Check-out date must be after Check-in date.")
        return
```

- The reserve_room() function manages the full reservation process by collecting user inputs such as the guest's name, chosen room type, check-in and check-out dates, and an optional discount code. It first validates the inputs to ensure all required fields are filled and that the check-out date comes after the check-in date. If the validation fails, appropriate warning or error messages are shown, and the process is stopped. If everything is valid, it calls the pricing function to calculate the total cost, generates a new discount code for future use, displays a confirmation message with all booking details, and finally logs the reservation into a local file for record-keeping.

```
    # Calculate number of nights
    nights = (check_out_date - check_in_date).days

    # Check entered discount code
    valid_discount = 20 if entered_code.startswith("UHOTEL-") else 0

    # Generate new code for future
    new_discount_code = f"UHOTEL-{random.randint(1000, 9999)}"

    # Calculate prices
    total_before, discount_amount, final_price = calculate_total_price(nights, room_type, valid_discount)

    # Build confirmation message
    message = f"""
Reservation Successful!
Guest Name: {guest_name}
Room Type: {room_type}
Check-in: {check_in_date.strftime('%Y-%m-%d')}
Check-out: {check_out_date.strftime('%Y-%m-%d')}
Nights: {nights}
Total Before Discount: ${total_before:.2f}
Discount Applied: {valid_discount}% (${discount_amount:.2f})
Final Price: ${final_price:.2f}

🎁 Your Discount Code for Next Time: {new_discount_code}
"""
```

- The function begins by calculating the number of nights between the check-in and check-out dates. It then checks if a discount code has been entered and whether it starts with “UHOTEL-”; if so, a 20% discount is applied. After that, a new discount code is generated to be used for future bookings. Using the pricing function, it calculates both the total cost before the discount and the final amount after applying the discount. Finally, it composes a confirmation message that includes all the booking details along with the newly generated discount code.

```
# Create main window
window = tk.Tk()
window.title("U Hotel - Room Reservation")
window.geometry("450x600")
window.resizable(False, False)

# Title
tk.Label(window, text="U Hotel Reservation", font=("Segoe UI", 18, "bold")).pack(pady=10)

# Guest Name
tk.Label(window, text="Guest Name:", font=("Segoe UI", 12)).pack()
entry_name = tk.Entry(window, font=("Segoe UI", 12), width=30)
entry_name.pack(pady=5)

# Room Type
tk.Label(window, text="Room Type:", font=("Segoe UI", 12)).pack()
room_var = tk.StringVar(value="Select a room")
room_options = ["Standard", "Deluxe", "Suite"]
room_menu = tk.OptionMenu(window, room_var, *room_options)
room_menu.config(font=("Segoe UI", 12))
room_menu.pack(pady=5)
```

- The application starts by creating the main window titled “U Hotel - Room Reservation,” setting its dimensions to 450×600 pixels and making it non-resizable to maintain a consistent layout. A prominent title label is then added at the top of the window, displaying “U Hotel Reservation” in bold to serve as the header. For guest information input, a label reading “Guest Name” is shown alongside a text entry field where users can type in their name. Following that, the app introduces a section for room type selection, featuring a label “Room Type” and a dropdown menu offering three room options: Standard, Deluxe, and Suite.

```
# Check-in Date
tk.Label(window, text="Check-in Date:", font=("Segoe UI", 12)).pack()
check_in = DateEntry(window, font=("Segoe UI", 12), width=20, background='darkblue', foreground='white', borderwidth=2)
check_in.pack(pady=5)

# Check-out Date
tk.Label(window, text="Check-out Date:", font=("Segoe UI", 12)).pack()
check_out = DateEntry(window, font=("Segoe UI", 12), width=20, background='darkblue', foreground='white', borderwidth=2)
check_out.pack(pady=5)

# Discount Code Input
tk.Label(window, text="Enter Discount Code (Optional):", font=("Segoe UI", 12)).pack(pady=5)
entry_discount = tk.Entry(window, font=("Segoe UI", 12), width=30)
entry_discount.pack(pady=5)

# Reserve Button
tk.Button(window, text="Reserve", font=("Segoe UI", 12, "bold"), command=reserve_room, width=20).pack(pady=20)

# Footer
tk.Label(window, text="Thank you for choosing U Hotel!", font=("Segoe UI", 10), fg="gray").pack(side="bottom", pady=10)

# Run the app
window.mainloop()

```

- The interface continues with date selection fields, starting with a check-in date section that includes a label and a calendar widget (DateEntry) allowing users to choose their arrival date. This is followed by a similar setup for the check-out date, where users can select their departure date using another calendar input. Next, there's an optional field for entering a discount code, consisting of a label and a text entry box. The core action is triggered by a prominently displayed "Reserve" button, which executes the reserve_room function when clicked. At the bottom of the window, a friendly footer message reads "Thank you for choosing U Hotel!" to enhance the user experience. Finally, the application launches and remains active through the window.mainloop() call, which keeps the GUI responsive and running until the user closes it.

GUI Components

The GUI is built using Tkinter widgets arranged in a simple vertical layout. The main window includes:

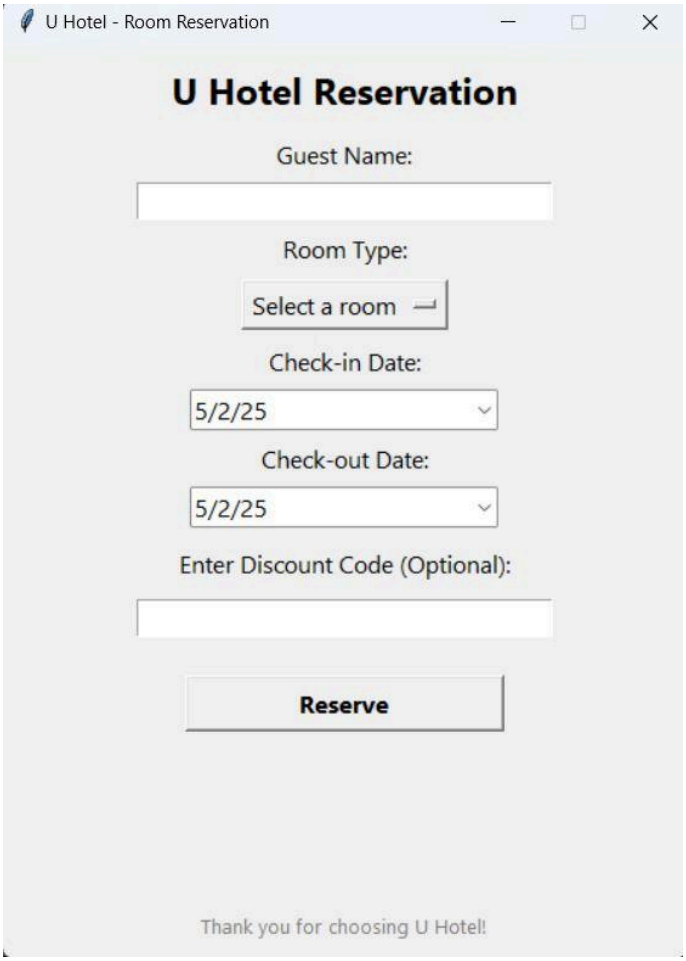

***Figure 1**: U Hotel Reservation Interface*

- **Title Label**: Shows “U Hotel Reservation” at the top.
- **Guest Name Field**: A text input for entering the guest’s name.
- **Room Type Dropdown**: Allows selection between Standard, Deluxe, or Suite.
- **Date Pickers**: Two `tkcalendar.DateEntry` widgets for selecting check-in and check-out dates.
- **Discount Code Entry**: An optional field for entering a discount code.
- **Reserve Button**: Initiates the reservation process when clicked.
- **Footer Label**: Displays a thank-you message at the bottom.

All elements use the "Segoe UI" font and are center-aligned to enhance readability and user experience.

Error Type

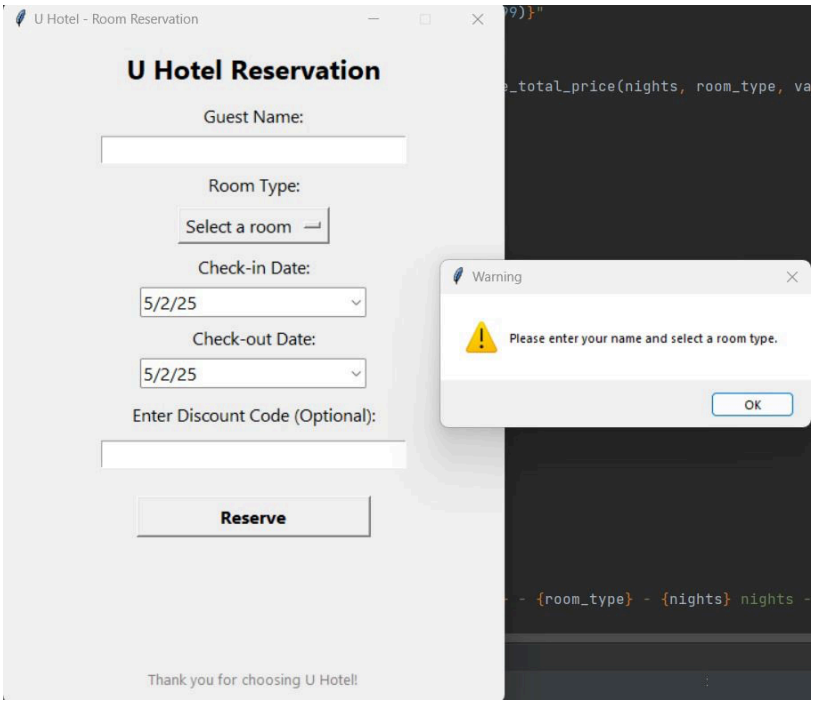

***Figure 2**: Input Validation Prompt Displayed on Missing Fields*

- In figure 2, the user attempted to submit a room reservation without filling in the guest name or selecting a room type. As a result, the system responded with a pop-up warning message that says, “Please enter your name and select a room type.” This is a built-in validation feature in the application that helps prevent incomplete bookings. It ensures that essential information—like the guest’s name and chosen room category—is entered before the reservation can be processed. This step improves the overall user experience by guiding users to complete all required fields properly.

Discount Logic & Validation

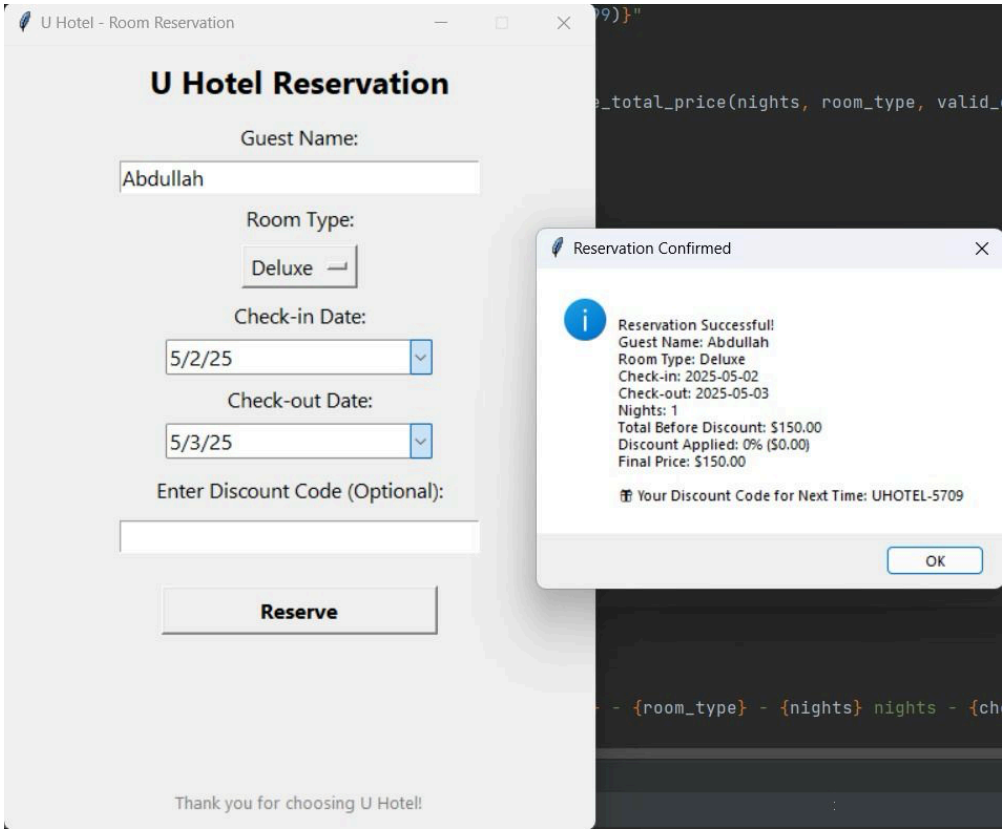

***Figure 3**: Successful Booking Confirmation Without Discount Code*

The application features a basic discount system to mimic promotional codes:

- A 20% discount is applied if the entered code starts with "UHOTEL-".
- After each booking, a new randomly generated code (e.g., UHOTEL-1234) is issued to encourage future use.

Input validation checks include:

- Verifying that the guest's name and room type are entered.
- Confirming the check-out date comes after the check-in date.
- Blocking empty or invalid inputs.

Warnings and confirmations are displayed using `messagebox.showwarning()` and `messagebox.showinfo()`.

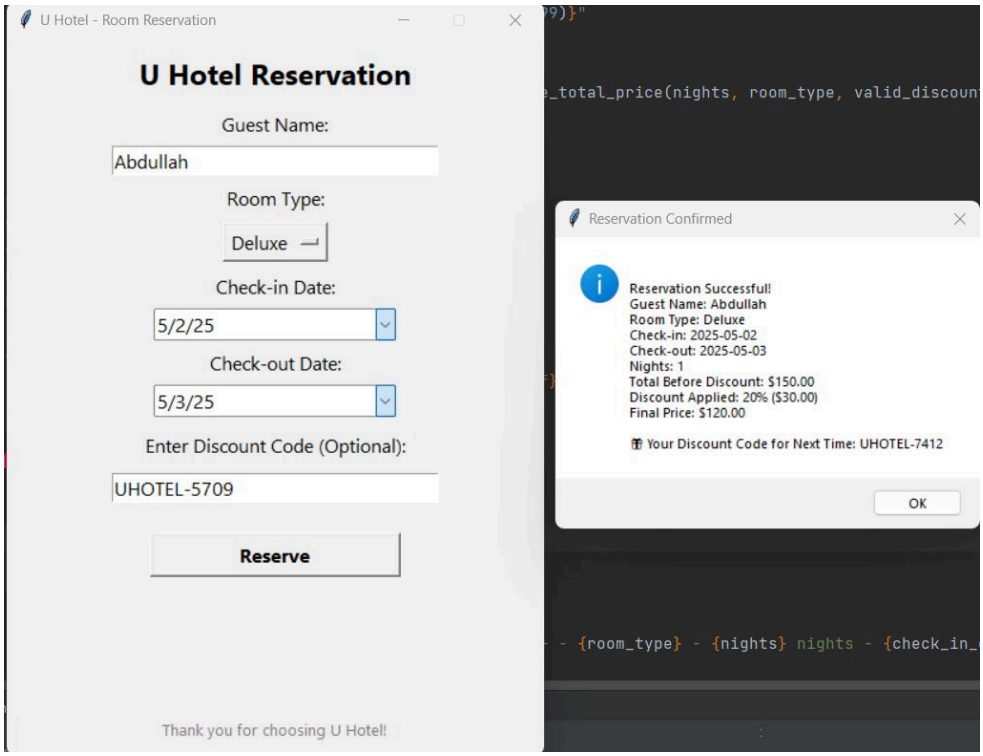

***Figure 4**: Successful Booking with Valid Discount Code*

Clear and Exit Buttons

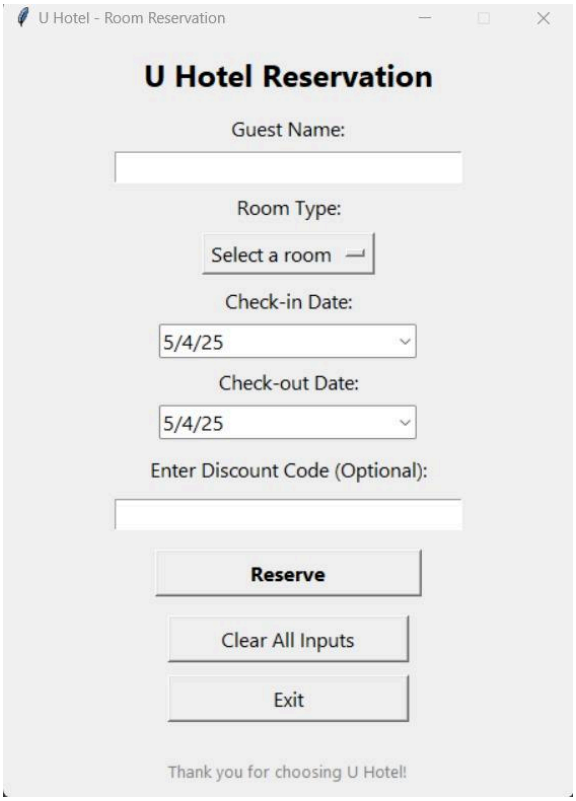

***Figure 5**: Clear and Exit Buttons in the U Hotel Reservation Interface*

- To make the app more user-friendly, we added two buttons: **“Clear All Inputs”** and **“Exit”**. The Clear button resets all fields—name, room type, dates, and discount code—using the clear_fields() function, added withtk.Button(window, text="Clear All Inputs", font=("Segoe UI", 12), command=clear_fields, width=20).pack(pady=5).
- The Exit button lets users close the app quickly with
- tk.Button(window, text="Exit", font=("Segoe UI", 12), command=window.quit, width=20).pack(pady=5)
- Both buttons improve usability by making the form easier to manage and exit.

Saving Reservation Data

Each reservation is saved locally in a text file named reservations.txt. This file logs:

- Date and time of booking
- Guest name
- Room type
- Number of nights
- Stay duration (check-in to check-out)
- Final price
- Discount used and new code generated

  This ensures a simple form of persistence without using databases, suitable for small or educational applications.

Running the Application

The program operates using Tkinter's standard `mainloop()`. Upon launch, it displays a window titled "U Hotel - Room Reservation" featuring all input fields and the reservation button.

All interface components are fully interactive, and the application functions independently, without the need for an internet connection or external services.

Conclusion

The U Hotel Reservation App showcases how Python's Tkinter library can be leveraged to create a simple yet functional room booking system. It integrates features like pricing calculations, discount code processing, and data storage within an intuitive graphical interface. This project provides a solid starting point for developing more advanced hospitality and reservation management solutions.