**Hotel Menu Mangement**

CASE STUDY

***Submitted by***

**Priyansh Kumar UID- 24MCI10262**

***in partial fulfillment for the award of the degree of***

**Master of Computer Applications**

**IN**

**Artificial Intelligence and Machine Learning**



**Chandigarh University**

November 2024

**ACKNOWLEDGEMENT**

I would like to express my deepest gratitude to all those who have supported and guided me throughout the development of this project.

First and foremost, I extend my sincere appreciation to DR.GAGANDEEP KAUR whose expertise, encouragement, and invaluable insights guided me through the complexities of this work. Their feedback and attention to detail were instrumental in refining the ideas and methodologies that shaped this project.

I am grateful to CHANDIGARH UNIVERSITY for providing the resources and support that made this work possible. The access to facilities, tools, and databases was essential in gathering data and conducting necessary analyses.

Special thanks go to AWINASH KUMAR for their collaboration and support throughout the project. Their enthusiasm and innovative ideas were invaluable, especially during the brainstorming and problem-solving phases.

I would also like to acknowledge the contribution of MS.SHRUTI SHARMA sharing their knowledge and experience, which greatly expanded my understanding of HOTEL MENU MANAGEMENT.

Lastly, I am thankful to my family and friends, who provided unwavering support and encouragement, enabling me to stay motivated throughout this journey.

Thank you all for your contributions to this project.

——————————— ——————————

### Analysis and Critical Thinking

**1. Introduction**

* **Purpose**: This report examines a Python-based program simulating a basic ordering system for a restaurant menu. The program allows customers to order items from a set menu, calculates the total cost, and provides feedback on item availability.
* **Background**: This type of program can be useful for small cafes or restaurants looking to implement a digital ordering system with limited functionality, such as adding single or multiple items to an order and displaying the total.
* **Scope**: This report will analyze the code's structure, functionality, and user experience, followed by recommendations for improvement.

**2. Analysis of the Code**

* **Overview of Key Features**: The program allows users to select items from a predefined menu and calculates the total cost. The program also checks if ordered items are available, and it prompts users if they wish to add additional items.
* **Data and Evidence**: The menu is stored in a dictionary with items as keys and their prices as values. The code uses conditional statements to verify if items are available and to update the total order amount.
* **Underlying Factors**: The simplicity of the code structure makes it beginner-friendly. However, it is limited to handling only two items in a single order, which restricts functionality for users ordering multiple items.

**3. Critical Evaluation**

* **Perspective Analysis**:
  + **User Experience**: The program is easy to use but could be more flexible. Currently, the customer can only order up to two items, which limits usability for larger orders.
  + **Code Efficiency**: The code uses a dictionary for menu storage, which is efficient for quick lookups. However, the structure could be improved to handle more complex order scenarios.
  + **Error Handling**: The program lacks robust error handling. For example, entering an incorrect or misspelled item name results in the item not being recognized, which may frustrate users.
* **Arguments For and Against**:
  + **Strengths**: Simple and easy to read. The code effectively checks if items exist in the menu and updates the total order amount accordingly.
  + **Weaknesses**: Limited to a two-item order, lacking flexibility for more complex ordering scenarios. Additionally, the prompt asks twice for new items rather than using a loop to continuously accept multiple items.
* **Limitations and Assumptions**:
  + The program assumes users will enter item names exactly as specified in the menu, which can lead to errors with minor spelling differences.
  + The program is limited in functionality, as it does not offer options to remove items, view a summary of the order, or handle quantities.

**4. Application of Critical Thinking Skills**

* **Logical Reasoning**: The logic for checking item availability and updating the total amount is straightforward, making it suitable for a simple ordering system. However, a loop could replace the hardcoded item\_2 input to allow for an arbitrary number of items.
* **Problem-Solving Approach**: To improve functionality, adding a while loop for continuous order prompts could simplify adding multiple items. Additionally, implementing input validation would improve user experience by accepting case-insensitive inputs or suggesting correct item names for minor typos.
* **Bias and Subjectivity**: There is an implicit assumption that users will correctly input item names and will only want to order two items. Broadening this perspective could help design a more flexible and user-friendly system.

**5. Conclusion**

* **Summary of Findings**: The program successfully allows users to order up to two items from a set menu, calculates the total, and provides feedback if an item is unavailable. However, the code could be made more flexible by allowing for multiple items and including error handling for user input.
* **Implications**: Improvements to the program could make it more suitable for practical applications, especially if it were intended for a real café setting. Adding features like continuous ordering, handling multiple items, and improving input validation would enhance usability.

Code Implementation

The Hotel Menu management System is built using Python. Below is a breakdown of the core features.

#Define the menu of resturant

menu = {

    'pizza':40,

    'pasta':50,

    'burger':60,

    'salad':50,

    'coffee':80,

    'hot coffee':100,

}

#Greet

print("Welcome to PYTHON cafe")

print("pizza: Rs40\n pasta: Rs50\nBurger: RS60\nsalad: Rs50\ncoffee: Rs80\nHot coffee: Rs100 ")

order\_total = 0

#80 + 100 = 180

item\_1 = input("Enter the name of items you want to order = ")

if item\_1 in menu:

    order\_total += menu [item\_1] #0 + 50

    print(f"youritem {item\_1} has been added to your order")

else:

    print(f"Ordered item {item\_1} is not avaialable yet!")

another\_order = input ("DO you want to add another items? (yes/no) ")

if another\_order == "yes":

    item\_2 = input("Enter the name of second item = ")

    if item\_2 in menu:

        order\_total += menu[item\_2]

        print(f"Item  {item\_2} has been  added to order")

    else:

        print(f"orderd item { item\_2} is not avaialabel!")

print(f"your total amount of items to pay is {order\_total}")

# Output:

Welcome to PYTHON cafe

pizza: Rs40

pasta: Rs50

Burger: RS60

salad: Rs50

coffee: Rs80

Hot coffee: Rs100

Enter the name of items you want to order = pizza

youritem pizza has been added to your order

DO you want to add another items? (yes/no) yes

Enter the name of second item = coffee

Item coffee has been added to order

your total amount of items to pay is 120

References

 **Python Official Documentation**

* Python Software Foundation. (n.d.). *The Python Standard Library*. Retrieved from <https://docs.python.org/3/library/>
* This source provides detailed explanations of Python’s built-in data types, control structures, and libraries, which are foundational for building simple menu-based applications.

 **Automate the Boring Stuff with Python**

* Sweigart, A. (2015). *Automate the Boring Stuff with Python: Practical Programming for Total Beginners*. No Starch Press.
* This book is an excellent resource for learning Python through practical projects, including beginner-friendly examples for handling user inputs, creating menus, and performing basic calculations.

 **Real Python – Python Dictionary Guide**

* Real Python. (n.d.). *Python Dictionaries: A Comprehensive Guide*. Retrieved from https://realpython.com/python-dicts/
* This guide explains how dictionaries work in Python, including tips on efficient lookups, which is useful when implementing a menu system where items are matched with prices.

 **Think Python: How to Think Like a Computer Scientist**

* Downey, A. B. (2015). *Think Python: How to Think Like a Computer Scientist*. O'Reilly Media.
* This book provides a comprehensive overview of Python programming and encourages critical thinking and problem-solving, which are valuable for designing flexible and user-friendly applications.

 **Python.org Tutorials**

* Python Software Foundation. (n.d.). *Python Tutorials*. Retrieved from <https://docs.python.org/3/tutorial/>