International University

Project-Report

Course code : CSE 316

Course tittle : Artificial Intelligence Lab

Submitted By

Name: Mehedi Hasan

ID: 213-15-4307

Name: Raktim Mondol

ID: 213-15-4287

Name: Yeamin Khan

ID: 213-15-4300

Section: 60_C

Dept. of CSE

Daffodil International University.

Project Proposal

Project Tittle: Restaurant Management System.

Introduction: Online restaurant management system is the process of ordering food from a website. The product can be either ready-to-eat food. The aim of developing online restaurant management project is to replace the traditional way of taking orders with computerized system.Another important reason for developing this project is to prepare order summary reports quickly and in correct format at any point of time when required. A restaurant management system is a valuable tool that streamlines various tasks and processes within a restaurant, such as managing menu items, processing orders, handling payments, and generating reports. By developing a custom system with Python and Tkinter, we can create a tailored solution that meets the specific needs of a restaurant.

Language: Python,Tkinter.

IDE: Pycharm.

Objective:

Restaurant management software aims to simplify some of the most common management duties in the restaurant industry.

For example,

The main objective of this system is to manage the details of item category,food,delivery address,order,and shopping cart. It manages all the information about item category,customer,cart item,item category.

To provide fast, efficient and reliable system by the way of managing the records of all their transaction.

Features and Benefits:

- Fill in the order the customer wants
- Complete bill management
- Managing menu items
- Calculate the total data
- Reset the data
- Feedback form

Benefits of Building a Restaurant Management System

1. **Customization:** By developing your own system, you can tailor it to fit the specific requirements of your restaurant, ensuring that it aligns perfectly with your workflow and business processes.
2. **Cost-effectiveness:** While commercial restaurant management systems can be expensive, building your own

with Python and Tkinter can significantly reduce costs, as open-source tools and libraries are freely available.

3. **Scalability:** As your restaurant grows, you can easily scale and expand your custom system to accommodate new features and functionality.
4. **Learning opportunity:** Developing a restaurant management system is an excellent learning experience, allowing you to enhance your Python programming skills and gain a deeper understanding of software development.

Full Source Code

```
import tkinter as tk
from tkinter import messagebox

class RestaurantManagementSystem:
    def __init__(self, root):
        self.root = root
        self.root.title("Restaurant Management System")

        self.customer_name = tk.StringVar()
        self.customer_contact = tk.StringVar()

        self.items = {
            "Burger": 100,
            "Pizza": 200,
            "Pasta": 150,
            "Sandwich": 80,
            "Salad": 90
        }

        self.orders = {}

        self.gst_percentage = 18

        self.create_gui()
```

```
    def create_gui(self):
        details_frame = tk.LabelFrame(self.root, text="Customer
Details")
        details_frame.pack(fill="x", padx=10, pady=10)

        name_label = tk.Label(details_frame, text="Name:")
        name_label.grid(row=0, column=0, padx=5, pady=5, sticky="e")
        name_entry = tk.Entry(details_frame,
textvariable=self.customer_name)
        name_entry.grid(row=0, column=1, padx=5, pady=5, sticky="w")

        contact_label = tk.Label(details_frame, text="Contact:")
        contact_label.grid(row=1, column=0, padx=5, pady=5,
sticky="e")
        contact_entry = tk.Entry(details_frame,
textvariable=self.customer_contact)
        contact_entry.grid(row=1, column=1, padx=5, pady=5,
sticky="w")
        contact_entry.configure(validate="key")

contact_entry.configure(validatecommand=(contact_entry.register(self.
validate_contact), "%P"))

        menu_frame = tk.LabelFrame(self.root, text="Menu")
        menu_frame.pack(fill="both", expand=True, padx=10, pady=10)

        item_header = tk.Label(menu_frame, text="Items")
        item_header.grid(row=0, column=0, padx=5, pady=5, sticky="w")
        quantity_header = tk.Label(menu_frame, text="Quantity")
        quantity_header.grid(row=0, column=1, padx=5, pady=5,
sticky="w")

        row = 1
        for item, price in self.items.items():
            item_var = tk.IntVar()
            item_label = tk.Label(menu_frame, text=f"{item} -
{self.convert_to_inr(price)}")
            item_label.grid(row=row, column=0, padx=5, pady=5,
sticky="w")

            quantity_entry = tk.Entry(menu_frame, width=5)
```

```
            quantity_entry.grid(row=row, column=1, padx=5, pady=5,
sticky="w")

            self.orders[item] = {"var": item_var, "quantity":
quantity_entry}

            row += 1

        buttons_frame = tk.Frame(self.root)
        buttons_frame.pack(fill="x", padx=10, pady=10)

        print_bill_button = tk.Button(buttons_frame, text="Print
Bill", command=self.show_bill_popup)
        print_bill_button.pack(side="left", padx=5)

        past_record_button = tk.Button(buttons_frame, text="Past
Records", command=self.past_records)
        past_record_button.pack(side="left", padx=5)

        clear_selection_button = tk.Button(buttons_frame, text="Clear
Selection", command=self.clear_selection)
        clear_selection_button.pack(side="left", padx=5)

        self.sample_bill_text = tk.Text(self.root, height=10)
        self.sample_bill_text.pack(fill="x", padx=10, pady=10)

        # Update sample bill when quantity or item is selected
        for item, info in self.orders.items():
            info["quantity"].bind("<FocusOut>", lambda event,
item=item: self.update_sample_bill(item))
            info["quantity"].bind("<Return>", lambda event,
item=item: self.update_sample_bill(item))
            info["quantity"].bind("<KeyRelease>", lambda event,
item=item: self.update_sample_bill(item))
            info["var"].trace("w", lambda *args, item=item:
self.update_sample_bill(item))

    def show_bill_popup(self):
        # Check if customer name is provided
        if not self.customer_name.get().strip():
            messagebox.showwarning("Warning", "Please enter customer
name.")
            return
```

```
        selected_items = []
        total_price = 0

        for item, info in self.orders.items():
            quantity = info["quantity"].get()
            if quantity:
                selected_items.append((item, int(quantity)))
                total_price += self.items[item] * int(quantity)

        if not selected_items:
            messagebox.showwarning("Warning", "Please select at least
one item.")
            return

        gst_amount = (total_price * self.gst_percentage) / 100

        bill = f"Customer Name: {self.customer_name.get()}\n"
        bill += f"Customer Contact:
{self.customer_contact.get()}\n\n"
        bill += "Selected Items:\n"
        for item, quantity in selected_items:
            bill += f"{item} x {quantity} -
{self.convert_to_inr(self.items[item] * quantity)}\n"
        bill += f"\nTotal Price:
{self.convert_to_inr(total_price)}\n"
        bill += f"GST ({self.gst_percentage}%):
{self.convert_to_inr(gst_amount)}\n"
        bill += f"Grand Total: {self.convert_to_inr(total_price +
gst_amount)}"

        messagebox.showinfo("Bill", bill)

    def past_records(self):
        messagebox.showinfo("Past Records", "This feature is not
implemented yet.")

    def clear_selection(self):
        for item, info in self.orders.items():
            info["var"].set(0)
            info["quantity"].delete(0, tk.END)

    def update_sample_bill(self, item):
```

```
        selected_items = []
        total_price = 0

        for item, info in self.orders.items():
            quantity = info["quantity"].get()
            if quantity:
                selected_items.append((item, int(quantity)))
                total_price += self.items[item] * int(quantity)

        gst_amount = (total_price * self.gst_percentage) / 100

        bill = f"Customer Name: {self.customer_name.get()}\n"
        bill += f"Customer Contact:
{self.customer_contact.get()}\n\n"
        bill += "Selected Items:\n"
        for item, quantity in selected_items:
            bill += f"{item} x {quantity} -
{self.convert_to_inr(self.items[item] * quantity)}\n"
        bill += f"\nTotal Price:
{self.convert_to_inr(total_price)}\n"
        bill += f"GST ({self.gst_percentage}%):
{self.convert_to_inr(gst_amount)}\n"
        bill += f"Grand Total: {self.convert_to_inr(total_price +
gst_amount)}"

        self.sample_bill_text.delete("1.0", tk.END)  # Clear previous
contents
        self.sample_bill_text.insert(tk.END, bill)

    def validate_contact(self, value):
        return value.isdigit() or value == ""

    @staticmethod
    def convert_to_inr(amount):
        return "₹" + str(amount)

root = tk.Tk()
restaurant_system = RestaurantManagementSystem(root)
root.mainloop()
```

Conclusion: The restaurant management system project is crucial for the success and efficiency of a modern restaurant. By implementing this

system, the restaurant can significantly enhance it’s operations,improve customers satisfaction,and ultimately increase profitability.The project aims to deliver a user-friendly,secure,and scalable solution that will meet the restaurant’s needs and adapt to it’s future requirements ensuring it’s long-term success.