**PROJECT TITLE**

**CAFE MANAGEMENT SYSTEM**

**PROJECT REPORT**

*Submitted in fulfilment of the requirement for the*

*Semester project*



IQRA UNIVERSITY

By

**PROJECT MEMBERS**

**<< Suneel Kumar\_67559>>**

**<<Anil Kumar \_67140>>**

**<<Muhammad Arslan\_65337>>**

**DEPARTMENT OF COMPUTER SCIENCE**

**CERTIFICATE**



IQRA UNIVERSITY

This is to certify that mini-project report entitled **“CAFE MANAGEMENT SYSTEM”** is confide record of work carried out by **<<PROJECT MEMBERS>>** during the year 2024 in fulfilment of the Requirement of SEMESTER PROJECT of the Degree of Bachelor of Computer Science

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**<<TEACHER NAME>> Instructor Signature**

**ACKNOWLEDGEMENT**

We bring out the report of our project work endeavoring gratitude to the Almighty Allah. We have received valuable guidance and help from many people among whom some require special mention.

We take this opportunity to express gratitude to our mam **<<** **UMAT UL SHAIYA >>** for giving us the opportunity to express our ideas.

Our sincere thanks are also to the lab assistants for the help they rendered to our project. Also our sincere thanks to the **M**anagement of **I**QRA **U**NIVERSITY for providing us with all the facilities for completing the project.

Last but not the least we would like to thank our dear parents & our colleagues for their unending support and valuable suggestions for implementing this project.

**Project: Cafe Management System in Python**

**Introduction**

This report details the development of a cafe management system using Python. The system streamlines various cafe operations, enhancing efficiency and providing valuable insights.

Key functionalities include:

* **Order Processing:** Take orders from customers, calculate costs accurately, and generate receipts with itemized details.
* **Admin Mangement**: We ask for admin ID and Password. After successful verification of admin we give options for changing menu items, and to see sales details.
* **Customer Management:** We first show the menu, then ask for order and also give them options to remove and add items till the time he confirms the order, then print their bill. And all these sales details are saved in a text file.
* **Reporting and Analytics:** Generate reports on sales, popular items, customer trends, and inventory levels, guiding informed decision-making by using bar graph in which Green color shows most bought item Yellow never bought items and Red other.

**2. Objectives**

* Automate order processing and management to reduce errors and improve efficiency.
* Track inventory in real-time.
* Enhance customer service through quicker order processing and personalized interactions.
* Generate reports for sales, inventory, and financial analysis to aid in decision-making By Graph.

**3.System Design**

1. **Data Structures**

* **Menu:** A CSV file to can store menu items, including ID, Names, prices,and pricing
* **ADMIN\_INFO:** A CSV File to store admin information incluing name and password
* **CART:** A CSV FILE represent orders, storing details of customer and ordered items (including quantities), and total costs.
* **Order\_History:**A CSV file to store order history for decision making

inventory items ( drinks), including names, costs, stock levels, and reorder points.

|  |  |
| --- | --- |
| NAME | DESCRIPTION |
| Admin\_info.csv | In this text file we stored the information of Admin and their password. |
| Cafe\_menu.csv | In this file we stored the menu details. |
| Cart.csv | In this file we write the order details of a customer. |
| Order\_history.csv | In this file we save the sales details. |

**2 Credentials for Login as a Admin**

|  |  |
| --- | --- |
| **Admin Name** | **Password** |
| anil | anil123 |
| suneel | sun456 |
| arslan | ars789 |
| amna | amna234 |

1. **Source Code python scripts**
2. **project.py file:**

|  |
| --- |
| # import projectfun  from projectfun import \*  while True :        # Print The Greeting's        greeting()      print()        # Getting the Type of User        user\_type = input("Please Specify The User Type! [Type : a for admin or c for customer] : ",)      user\_type = user\_type.lower()        # Admin/Master's Page        if user\_type == "a":            # Entering Into The Page For The Varification!            login = log\_in()            # If Login successfully With Correct Login Credentials!            if login == True:                # Admin System                while True :                    # Adding of New Menu Items!                    add\_menu = input("Do You Want to Add a Item in The Menu! [Type : y for YES or n for NO] : ",)                  add\_menu = add\_menu.lower()                    if add\_menu == "y":                      while add\_menu == "y":                          add\_success = add\_menu\_item()                            if add\_success == True :                              print("Added Successfully!!\n")                              add\_menu = input("Do You Want to Add More Item's in The Menu! [Type : y for YES or n for NO] : ",)                              add\_menu = add\_menu.lower()                                if add\_menu == "y":                                  continue                                elif add\_menu == "n":                                  break                                else :                                  print("Please Write a Valid Input From Next Time!\n")                            else :                              pass                    elif add\_menu == "n":                      pass                    else :                      print("Please Write a Valid Input From Next Time!\n")                      # Display The Sales Details Done Till Now!                    see\_history = input("Do You Want to See The Order History! [Type : y for YES or n for NO] : ",)                  see\_history = see\_history.lower()                    if see\_history == "y":                      order\_history()                    elif see\_history == "n":                      pass                    else :                      print("Please Write a Valid Input From Next Time!\n")                    # Asking Admin Whether He/She Want to Logout From The Admin/Master Page!                    ask\_exit = input("Do You Want To Logout and Go To The Home Page! [Type : y for YES or n for NO] : ",)                  ask\_exit = ask\_exit.lower()                    if ask\_exit == "y":                      break                    elif ask\_exit == "n":                      continue                    else :                      print("Please Write a Valid Input From Next Time!\n")                      continue                # Print After Loging Out From The Admin/Master Page!                print()              print("Successfully LOGOUT From The Admin/Master Page!\n")              print("Going To The Main/Home Page!")              print("Please Wait!\n")        # Customer Page For Order        elif user\_type == "c":          customer\_name = input("Please Enter The Customer Name Here! : ",)            # Printing The Menu            print()          print("Please Have a Look in The Today's Menu!\n")          print("       \*\* Menu \*\*       ")          menu\_display()            # Customer Page            while True :                # Take Order From The Customer!                ask\_order = input("Would You Like to Place Your Order! [Type : y for YES or n for NO] : ",)              ask\_order = ask\_order.lower()                if ask\_order == "y":                  while ask\_order == "y":                      print()                      print("\*\*\* TAKING ORDER \*\*\*\n")                      order\_item\_id = input("Please Enter The Item Id Here! : ",)                        id\_check = check\_item\_details(order\_item\_id)                        if id\_check == (True , False) :                            # Printing The Item Information                            menu\_info = menu\_data()                          item\_info = menu\_info[order\_item\_id]                          print()                          print("." \* 64)                          print(":  Item Name  :  " , item\_info[0] , " "\*(15 - len(item\_info[0])) , "::  Item Price  :  "                                   , str(item\_info[1]) , " "\*(6 - len(str(item\_info[1]))) , ":"  )                          print("." \* 64,"\n")                            try :                              item\_qty = int(input("Please Enter The Quantity Here! {Hint : Please Enter The Quantity in Integer!} : ",))                            except :                              print("Please Enter a Valid Input!\n")                              continue                            else :                              # Adding the Item To The Cart!                                add\_cart\_success = take\_order(order\_item\_id , item\_qty)                                if add\_cart\_success == True :                                  print("Successfully Added To The Cart!\n")                                  ask\_order = input("Would You Like to Order Something Else! [Type : y for YES or n for NO] : ",)                                  ask\_order = ask\_order.lower()                                  continue                                else :                                  continue                        else :                          print("Please Enter a Valid ITEM ID!\n")                          continue                elif ask\_order == "n":                  print("Please Take Your Time, See The Menu, And Then Order!!\n")                  continue                else :                  print("Please Write a Valid Input!\n")                  continue                # Asking The Customer, If He/She Want's To Remove Some Item From The Cart!                ask\_remove = input("Do You Want To Remove a Item From The Cart! [Type : y for YES or n for NO] : ",)              ask\_remove = ask\_remove.lower()                if ask\_remove == "y":                  while ask\_remove == "y":                        # Printing The Current Item's Present In The Cart!                        cart\_item = retrieve\_data("cart.csv")                      print("Current Item In The Cart!")                      for item in cart\_item :                          print(item)                        remove\_item\_id = input("Please Enter The Item Id Here! : ",)                      remove\_order\_success = item\_remove\_order(remove\_item\_id)                        if remove\_order\_success == True :                          print("Successfully Removed From The Cart!\n")                          ask\_remove = input("Do You Want To Remove more Item From The Cart! [Type : y for YES or n for NO] : ",)                          ask\_remove = ask\_remove.lower()                            if ask\_remove == "y":                              continue                            elif ask\_remove == "n":                              break                            else :                              print("Please Enter A Valid Input!\n")                              continue                      else :                          print("Please Write a Valid Input From Next Time!\n")                          break                elif ask\_remove == "n" :                  pass                else :                  print("Please Write a Valid Input From Next Time!\n")                  pass                # Asking The Customer, To Confirm The Order!                confirm\_order = input("Please Confirm Your Order For The Billing [Type : y for YES or n for NO] : ",)              confirm\_order = confirm\_order.lower()                if confirm\_order == "y":                  final\_order\_list = retrieve\_data("cart.csv")                  for item\_list in final\_order\_list :                      item\_list[2] = float(item\_list[2])                      item\_list[3] = int(item\_list[3])                    # Transfer The Item Info. From The Cart To The Order History                    store\_order\_history = cart\_to\_order(customer\_name)                    # Generate and Print The Bill!                    bill\_success = generate\_bill(final\_order\_list)                    # Empty The Cart!                    cart\_empty\_success = empty\_cart()                    if store\_order\_history == True and bill\_success == True and cart\_empty\_success == True :                      print()                      print("Thanks For Choosing Us! Hope You Will Come Back!\n")                      break                    else :                      print("An ERROR Occured!")                      print("Sorry, For The Inconvenience!")                      print("Please wait\n")                      continue                else :                  cart\_empty\_success = empty\_cart()                  print("Your Order is Cancelled!")                  print("Sorry, For The Inconvenience!\n")                  break        # In Case If The User Type Is Wrong!        else :          print()          print("Please Enter A Valid Input!!\n")          continue |

1. **projectfun.py file**

|  |
| --- |
| # \*\*Funtion to print Greeting!\*\*  # In[7]:  def greeting():      """      This Function is use to print the greeting at the start of the program.      Arguments:          No Argument Taken      Returns:          print greeting message.      """      print(r"                                                                                            \_")      print(r" \_\_                \_\_                           \_\_\_\_\_\_\_        \_\_\_\_\_\_\_                     | |")      print(r" \ \              / /                          |\_\_   \_\_|      | \_\_\_\_\_\_|                    | |")      print(r"  \ \     \_\_     / /                              | |         | |                          | |")      print(r"   \ \   /\_\_\   / /  \_\_       \_  \_   \_   \_   \_    | |   \_     | |          \_     \_\_   \_\_   |\_|")      print(r"    \ \\_//  \\\_/ /  |\_   |   |  | | | \\_/ | |\_    | |  | |    | |\_\_\_\_\_    /\_\   |\_   |\_     \_ ")      print(r"     \\_\_/    \\_\_/   |\_\_  |\_  |\_ |\_| |     | |\_    |\_|  |\_|    |\_\_\_\_\_\_\_|  /   \  |    |\_\_   |\_|")      #LOGIN FUNCTION  import csv  from getpass import getpass  # \*\*Function to Log-in into the Admin system \*\*  def log\_in(attempts=3):      """      This is a login Function for the Admin, It takes the Admin Name and Admin Password and      Check the input from the Admin\_info.csv File(Data Base).      The password is taken using getpass method.        Argumentss:          Attempts : Number of attempts provided to login.                     Default value is 3.        Inputs:          Take the Admin Name and Password as the input.      Returns:          returns True, if it is is executed succesfully.          In case of any error, it returns False.          But after first try function returns None value (Thats why to check use login\_value).      Raises:          UnknownError: Raises an exception when a error is found.      """        user\_list = user\_data()        for attempt in range(attempts, 0, -1):          print("Please Enter the following details!\n")          user\_name = input("User Name: ")          # password = getpass("Password: ")          password=input("userpassword")          print()            if user\_name in user\_list and password == user\_list[user\_name]:              print("~~~ Log-in successfully!! ~~~\n")              print("Welcome", user\_name, "!")              return True          else:              print(f"This is not a valid Username/Password. You have {attempt-1} attempts left.\n")              if attempt == 1:                  print("This was your last attempt! Please try later!\n")                  return False    # \*\*Get The User Data in the form of dictionary\*\*  def user\_data():      """      This Function is use to retrieve the data from the file Admin\_info.csv,      into the function using retrieve\_data function.      Then convert it into a dictonary with Key as admin name and Value as the password.      Arguments:          No Argument Taken      Returns:          returns a dictonary with Key as admin name and Value as the password.      Raises:          UnknownError: Raises an exception when a error is found.          Admin\_info.csv file not found!      """      user\_dictionary = {}      try:          with open("C:\\Users\\LENOVO\\Desktop\\python froject\\Admin\_info.csv", mode='r') as file:              reader = csv.reader(file)              next(reader)  # Skip the header row if there is one              for row in reader:                  user\_name = row[1]                  password = row[2]                  user\_dictionary[user\_name] = password      except FileNotFoundError:          print("Admin\_info.csv file not found!")        return user\_dictionary  # ////////////////////  import csv  def add\_menu\_item():      """      This Function is use to Add New Menu Items into the File cafe\_menu.txt,      By Using the data\_entry function.      To verify that the Item\_Id provided does not exists in the menu to avoid Duplication of the Item\_Id,      check\_item\_details function is used (as Item\_Id is a unique Key/Primary Key).      After updating the quantity it stores the order back into the cafe\_menu.csv File.      Arguments:          No Argument Taken.        Inputs:          Take the Unique Item Id and Item Name as an input from the user.      Returns:         returns True, if it is is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      old\_menu = retrieve\_data("cafe\_menu.csv")      print("       \*\*\* The Current Menu Details \*\*\*")      menu\_display()      print("Please Enter the Following Details!\n")      item\_id = input("Unique Item ID: ")      item\_name = input("Item Name: ")      try:          item\_price = float(input("Item Price: "))      except ValueError:          print("\nPlease Enter a valid Item Price!\n")          return False        validity = check\_item\_details(item\_id, item\_name)      if validity == (False, False) and item\_name != "":          item\_detail = [item\_id, item\_name, f"{item\_price:.2f}"]          old\_menu.append(item\_detail)          data\_entry("cafe\_menu.csv", old\_menu)          return True      else:          print("\nPlease Enter valid Inputs!")          print("ERROR: Duplication of Input was Found!\n")          return False      # \*\*Funtion to get data from the file into a list!!\*\*  def retrieve\_data(path):      """      This Function is use to get the data from the File,      which contains the data in comma seprated form and      convert it into a nested list and return that list.      Arguments:          path : The path of the .csv file to read in string format.      Returns:          Returns a nested list, in which data in each line is converted into a list.          In the case of an error, it returns None.      Raises:          FileNotFoundError: Raises an exception when the file is not found in the specified path.      """      try:          with open(path, mode='r') as file:              reader = csv.reader(file)              return [row for row in reader if row]      except FileNotFoundError:          print("Please make sure that the path you have provided is correct!")          return []      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return []  # \*\*Funtion to Display the menu!!\*\*  def menu\_display():      """      This Function is use to retrieve the data from the file cafe\_menu.csv,      into the function using retrieve\_data function.      Then, Print the current Menu Page.      Arguments:          No Argument Taken      Returns:          returns the Menu in a Tabular form.      Raises:          Cafe\_menu.csv not found!.          UnknownError: Raises an exception when a error is found.      """      try:          print("\n" + "." \* 46)          print(": Item I.D :        Item        :   Price    :")          menu\_list = retrieve\_data("C:\\Users\\LENOVO\\Desktop\\python froject\\cafe\_menu.csv")          for item in menu\_list:              print("." \* 46)              print(f": {item[0]:<9} : {item[1]:<15} : {item[2]:>8} :")          print("." \* 46, "\n")      except Exception as e:          print("cafe\_menu.csv not found!")          print("ERROR:", e, "\n")    # \*\*Get The Menu Data in the form of dictionary\*\*  def menu\_data():      """      This Function is use to retrieve the data from the file cafe\_menu.txt,      into the function using retrieve\_data function.      Then, convert it into a dictonary with Key as Item\_Id and Value as a tuple of name and price.      Arguments:          No Argument Taken      Returns:          returns a dictonary with Key as Item\_Id and Value as a tuple of name and price.      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          menu\_dictionary = {}          menu\_list = retrieve\_data("C:\\Users\\LENOVO\\Desktop\\python froject\\cafe\_menu.csv")          for item in menu\_list:              item\_id = item[0]              item\_name = item[1]              item\_price = float(item[2])              menu\_dictionary[item\_id] = (item\_name, item\_price)          return menu\_dictionary      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return {}    # check\_item\_details(1818)  def check\_item\_details(item\_id, item\_name=None):      """      This Function is use to check whether the Item\_Id and Item\_Name      provided matches with that in the DataBase(cafe\_menu.csv) File.      And Returns a tuple with 2 boolean values.      For Example : if item\_id is correct and item\_name is wrong      output      : (True,False)      Arguments:          item\_id   : The Item ID is written in string format.          item\_name : The Item Name is written in string format and it is not necessary to include(optional argument).                      Default Value is None.      Returns:          Returns a tuple with 2 boolean values.          For Example : if item\_id is correct and item\_name is wrong          output      : (True,False)      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          menu = menu\_data()          for value in menu:              if value == item\_id:                  if menu[value][0] == item\_name:                      return True, True                  else:                      return True, False              elif value != item\_id:                  if menu[value][0] == item\_name:                      return False, True              else:                  pass          return False, False      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return False, False  # \*\*Funtion to enter data in the file from a list!!\*\*  def data\_entry(path, data):      """      This Function is use to enter the data from the Nested List,      Into the file in a comma seprated form.      Each line in the file contains the data of a list(element).      Arguments:          path  : The path of the .txt file is to write in string format.          entry : The Nested List(Which means a List inside a List).      Returns:          returns True, if executed succesfully.          In the case of an error it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          with open(path, mode='w', newline='', encoding='utf-8') as file:              writer = csv.writer(file)              writer.writerows(data)      except Exception as e:          print("An Error Occurred while writing to file!")          print("ERROR:", e, "\n")  # ////////////////////////////////////////////////  import csv  # \*\*Function To Store Order in Cart.csv ; Cart.csv : A Temporary txt file to have the order details\*\*  def cart(order\_list):      """      This Function is use to enter(append) the data from the Nested List,      into the file cart.csv in a comma seperated form.      Each line in the file contains the data of a list(element).      Arguments:          order\_list : The Nested List.(A List Inside A List)      Returns:          returns nothing(None).      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          with open("C:\\Users\\LENOVO\\Desktop\\python froject\\cart.csv", mode="a", newline='', encoding='utf-8') as file\_cart:              writer = csv.writer(file\_cart)              writer.writerow(order\_list)      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")  def take\_order(item\_id, quantity=1):      """      This Function is use to Take Order with the Item\_Id and quantity as the inputs.      And uses check\_item\_details function to check with that in the DataBase(cafe\_menu.csv) File.      And uses cart function to add the item into the cart.csv File.      Arguments:          item\_id   : The Item ID is written in string format.          Quentity  : The Item Quantity is written in integer format.( Default value is 1)      Returns:          returns True, if it is is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          check = check\_item\_details(item\_id)          if check == (True, False):              order\_list = [item\_id]              menu\_dict = menu\_data()              order\_list.append(menu\_dict[item\_id][0])              order\_list.append(str(menu\_dict[item\_id][1]))              order\_list.append(str(quantity))              cart(order\_list)              return True          else:              return False      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e)          print("Please Enter a valid Input!\n")          return False  # \*\*Function to Generate a Bill\*\*  def generate\_bill(order\_list):      """      This Function is use to generate the bill from the data in the order\_list.      Then generate the bill and also the total amount to be payed.      Arguments:          order\_list : The Nested List.(A List Inside A List)        Returns:          returns True, if it is is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      print("\n\n                                            \*\*\*  BILL  \*\*\*                                             ")      print("." \* 106)      print(":    S.No    :    ITEM ID    :       ITEM NAME       :     PRICE     :    QUANTITY   :    TOTAL AMOUNT   :")      s\_no = 1      total\_amount = 0.0      try:          for item in order\_list:              total\_price = float(item[2]) \* float(item[3])              print("." \* 106)              print(f":    {s\_no:<7} :    {item[0]:<9} :    {item[1]:<16} :    {item[2]:<8} :      {item[3]:<8} :     {total\_price:<11} :")              s\_no += 1              total\_amount += total\_price          print("." \* 106)          print("\n\n" + "." \* 55)          print(f":    TOTAL AMOUNT TO PAY IN RUPEES    :    {total\_amount:<8} :")          print("." \* 55)          return True      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return False    # \*\*Function to Transfer the Order Information into the order\_history.txt File\*\*  def cart\_to\_order(customer\_name):      """      This Function is use to enter the data from the cart.csv File into order\_history.csv File,      Data is stored in the order\_history.csv file in a comma seprated form with      additional customer name added at the last of each line.      Arguments:          customer\_name : The Name of the Customer is written in the string format.      Returns:          returns True, if it is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception, when a error is found.      """      try:          final\_order = retrieve\_data("C:\\Users\\LENOVO\\Desktop\\python froject\\cart.csv")          with open("order\_history.csv", mode="a", newline='', encoding='utf-8') as file\_open:              writer = csv.writer(file\_open)              for list\_data in final\_order:                  list\_data.append(customer\_name)                  writer.writerow(list\_data)          return True      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return False  # \*\*Function to Empty the Current Item in the Cart.csv File\*\*  def empty\_cart():      """      This Function is used to empty the data from the cart.csv File.      Arguments:          No Argument Taken.      Returns:          returns True, if it is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          with open("C:\\Users\\LENOVO\\Desktop\\python froject\\cart.csv", mode="w", encoding='utf-8') as file\_open:              file\_open.write("")          return True      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return False  def item\_remove\_order(item\_id):      """      This Function is used to remove an item from the ordered information present in the cart.csv File.      To verify that the item\_id provided exists in the menu, check\_item\_details funcion is used.      After checking, the item's information is removed from the cart.csv File.      Then, the Function update the cart.csv File.      Arguments:          item\_id : The Item ID is written in the string format.      Returns:          returns True, if it is is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      try:          check = check\_item\_details(item\_id)          if check == (True, False):              order\_list = retrieve\_data("C:\\Users\\LENOVO\\Desktop\\python froject\\cart.csv")              new\_order = [value for value in order\_list if value[0] != item\_id]              data\_entry("cart.csv", new\_order)              return True          else:              print("Entered Item ID was wrong. Please Enter A Valid ID!\n")              return False      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return False  # \*\*Function to Get the Order History and to Display it from order\_history.txt File!\*\*  def order\_history():      """      This Function is use to retrieve the data from the File order\_history.txt,      into the Function using retrieve\_data Function.      Then, print the Customer Order History with the total sale done till now.      Arguments:          No Argument Taken      Returns:          returns the Customer Order History in a Tabular form.          returns True, if is executed succesfully.          In case of any error, it returns False.      Raises:          UnknownError: Raises an exception when a error is found.      """      print("\n                                      \*\*\*  ORDER HISTORY  \*\*\*                                             ")      print("." \* 110)      print(":    S.No    :     CUSTOMER NAME     :   ITEM ID    :       ITEM NAME        :     PRICE     :    QUANTITY   :")      s\_no = 1      total\_sale = 0.0      order\_list = retrieve\_data("order\_history.csv")      try:          for item in order\_list:              total\_price = float(item[2]) \* float(item[3])              print("." \* 110)              print(f":    {s\_no:<7} :    {item[-1]:<20} :   {item[0]:<8} :     {item[1]:<18} :     {item[2]:<7} :     {item[3]:<7} :")              s\_no += 1              total\_sale += total\_price          print("." \* 110)          print("\n\n" + "." \* 57)          print(f":    TOTAL SALE TILL NOW IN RUPEES    :    {total\_sale:<10} :")          print("." \* 57)          return True      except Exception as e:          print("An Error Occurred!")          print("ERROR:", e, "\n")          return False |

1. **matplot.py file**

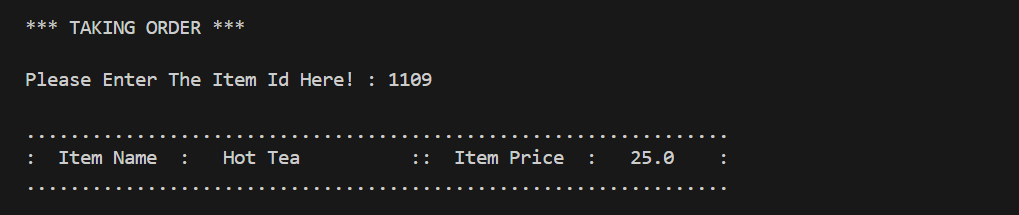
|  |
| --- |
| import csv  import matplotlib.pyplot as plt  from collections import Counter  # Function to retrieve data from a file  def retrieve\_data(file\_path):      try:          with open(file\_path, 'r') as file:              data = [line.strip().split(',') for line in file.readlines()]          return data      except FileNotFoundError:          print(f"File '{file\_path}' not found!")          return []      except Exception as e:          print(f"An error occurred while reading '{file\_path}': {e}")          return []  # Function to load menu items from the CSV  def load\_menu(file\_path):      menu\_data = retrieve\_data(file\_path)      menu\_dict = {item[0]: item[1] for item in menu\_data}  # item\_id: item\_name      return menu\_dict  # Function to count occurrences of each item in the order list  def count\_orders(order\_list):      item\_counts = Counter()      for order in order\_list:          item\_name = order[1]  # Assuming item name is the second element in each order entry          quantity = int(order[3])  # Assuming quantity is the fourth element in each order entry          item\_counts[item\_name] += quantity      return item\_counts  # Function to plot a bar graph of popular orders  def plot\_bar\_graph(item\_counts, menu\_dict):      item\_names = []      counts = []      colors = []      for item\_name in menu\_dict.values():          count = item\_counts.get(item\_name, 0)          item\_names.append(item\_name)          counts.append(max(count, 0.1))  # Ensure all items have at least a small bar          if count > 5:              colors.append('green')          elif count == 0:              colors.append('yellow')          else:              colors.append('red')      plt.figure(figsize=(10, 6))      plt.bar(item\_names, counts, color=colors)      plt.xlabel('Item Name')      plt.ylabel('Number of Orders')      plt.title('Popular Orders')      plt.xticks(rotation=45, ha='right')      plt.tight\_layout()      plt.show()  # Main function to run the script  def main():      menu\_file = "C:\\Users\\LENOVO\\Desktop\\python froject\\cafe\_menu.csv"  # Update with your actual file path      order\_history\_file = "C:\\Users\\LENOVO\\Desktop\\python froject\\order\_history.csv"  # Update with your actual file path      menu\_dict = load\_menu(menu\_file)      orders = retrieve\_data(order\_history\_file)      if orders:          order\_counts = count\_orders(orders)          plot\_bar\_graph(order\_counts, menu\_dict)      else:          print("No orders found.")  # Execute the main function  if \_\_name\_\_ == "\_\_main\_\_":      main() |

**5. Core Functionality**

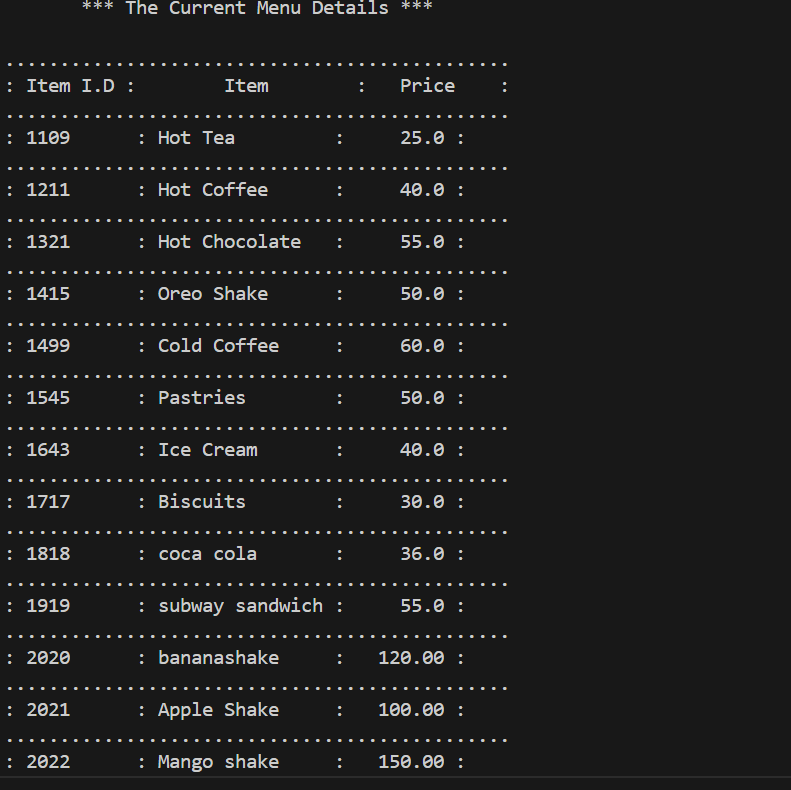
* **greeting():** To print greeting in start of project.



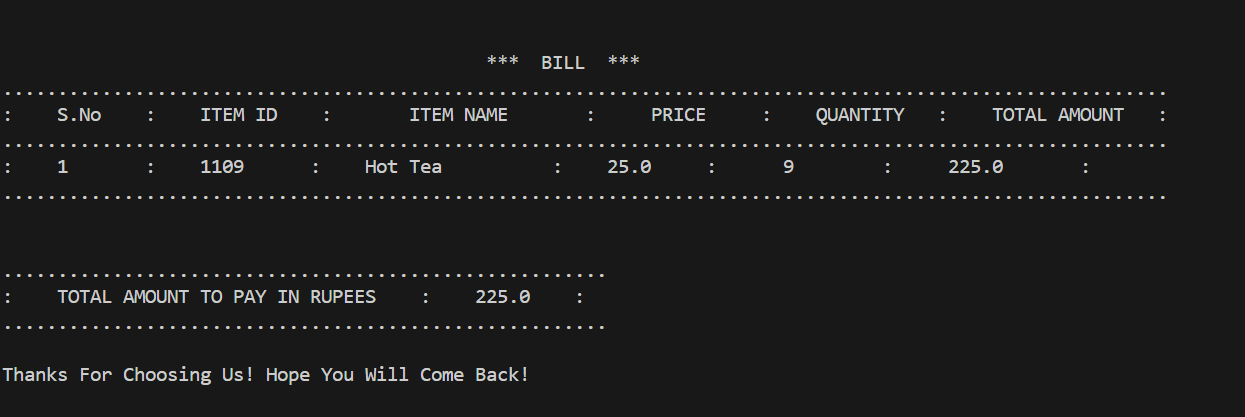
* **log\_in(attempts=3):** To validate admin name and password only in three attempts.
* **take\_order(**item\_id,quantity=1**):** used to take an order and store it in cart



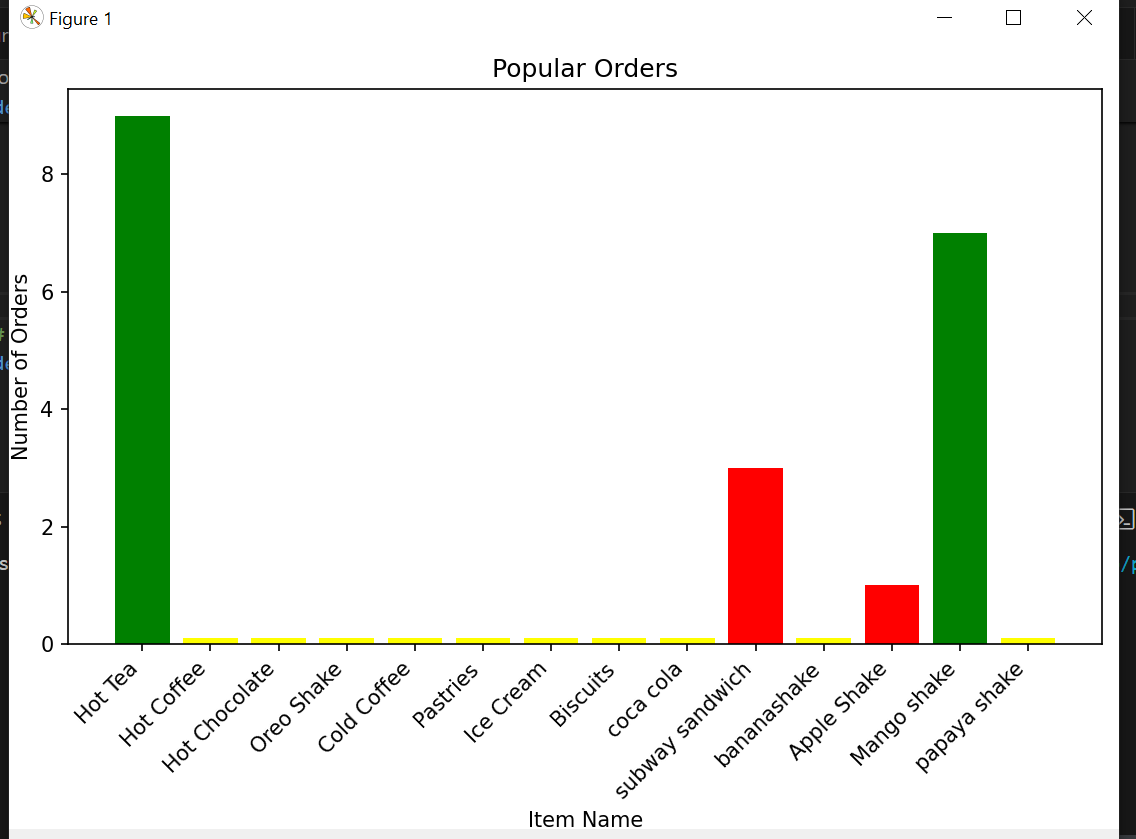
* **add\_menu\_item():** add new menu item and write in csv filewith the help of data\_entry()function.
* **menu\_display():** Display the menu items for selection by customer



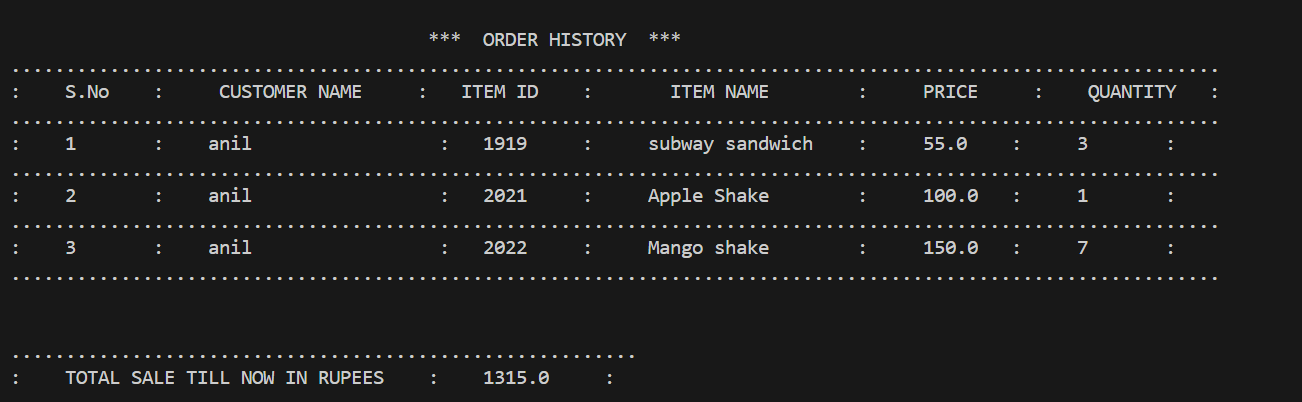
* **Cart(**order\_list**):**to add orderd item in cart
* **Cart\_to\_order(**customer\_name**)**: to add order form cart to order\_history.csv file
* **empty\_cart()**:to empty cart after its contect is written in order\_history.csv file
* **Item\_remove\_order(**item\_id**)**:if user want to cancell a particular ordered item
* **generate\_bill(order\_list):** Generates a detailed receipt with itemized information and total cost print to console.



* **Plot\_bar\_graph** (item\_counts,menu\_dict): to display bar graph



* **Order\_history():** Generate the order history for financial analizing.



**5. Modules and Libraries**

* **Project Script files**

**>> Project.py(main file)** *This is our main file for the program and in this we import the function file.*

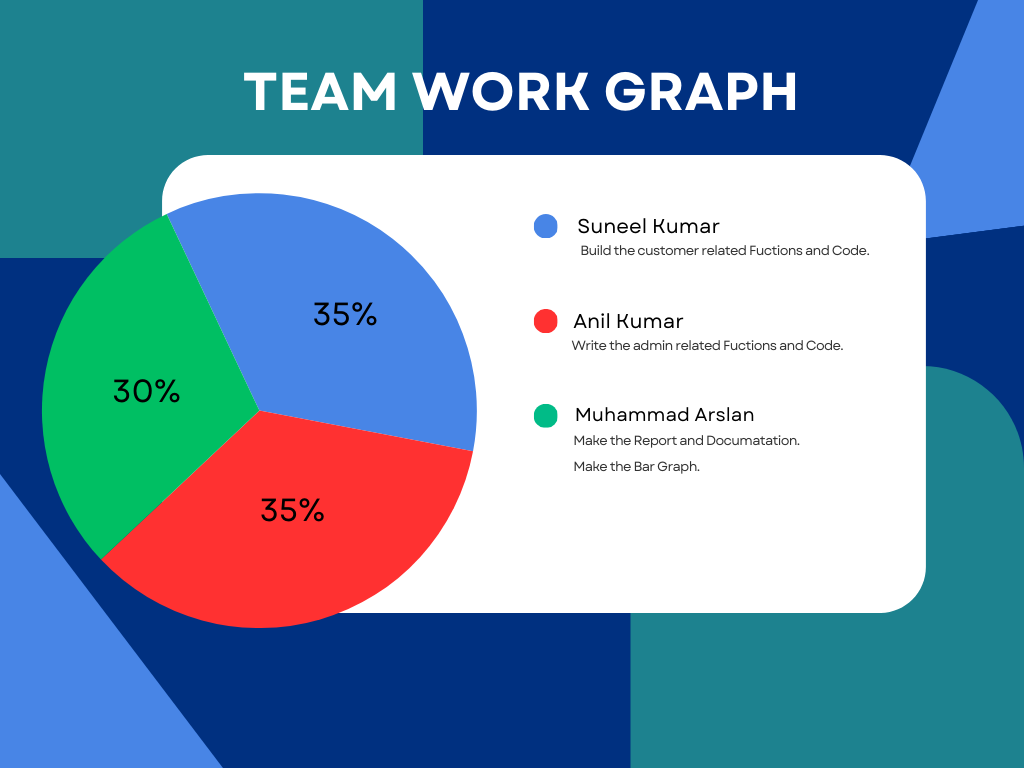
**>> Projectfun.py** *In this file we stored all the functions which we have used in our main file*

***>>Matplotep.py:*** *to print bar Graph*

* **Bar Graph :** matplotlib for advanced data analysis and visualization in reports.
* **CSV**: to create, read and write from csv file
* **Getpass:** to hide admin password during typing
* **Collections:** use counter function to count total quantity of manu\_items

1. **Contribution of group Members:**

**Showing team Work contribution by pie\_chart:**



**1\_Suneel Kumar (35%) :** Build the customer related Fuctions and Code.

**2\_Anil Kumar(35%):**Write the admin related Fuctions and Code.

**3\_Muhammad Arslan (30%) :** Make the Report and Documatation and Bar Graph.