Assignment

Formal Methods in Software Engineering(FMS)

(SE-313)

Group Id: A-24

FOOD ORDERING SYSTEM

Group Members

Aqsa Zaib	SE-21013
Mahnoor Iqbal	SE-21027

FOOD ORDERING SYSTEM:

1) Scope:

The primary goal of the food ordering system is to provide users with a user-friendly platform for browsing a diverse menu, selecting items from various categories, specifying quantities, obtaining the total cost, and finally, placing an order. The system aims to streamline the ordering process, enhance user experience, and efficiently manage orders for both users and the restaurant.

Functionalities:

Menu Display:

- Users can access a comprehensive menu featuring categories such as burgers, pizzas, drinks, and rolls.
- Each category will have a variety of items with detailed information (name, description, price).

Category Selection:

- Users can select a specific category to narrow down their menu choices (e.g., burgers, pizzas).

Item Selection:

- Users can view the details of individual items within a selected category.
- Users can add items to their order with the ability to specify the quantity.

Price Display:

- The system displays the price of each selected item.
- It calculates and displays the total cost of the order in real-time.

Order Placement:

- Users can review their selected items, quantities, and the total cost before placing an order.
- Upon confirmation, the order is placed, and the user receives an order confirmation.

2) 4+1 View:

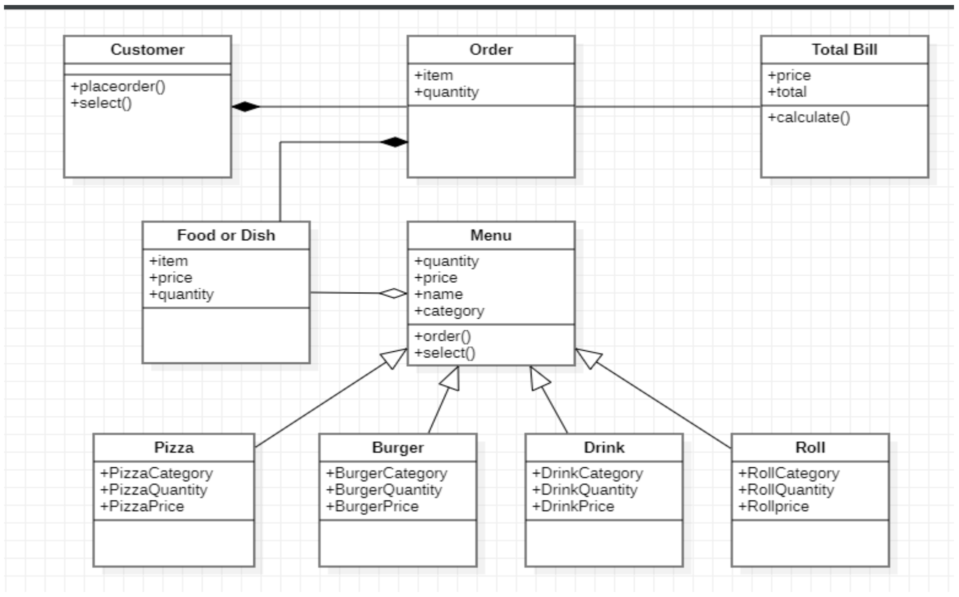
The 4+1 architectural view is an approach that combines multiple perspectives to describe a complex software system. It consists of four distinct views, each addressing different concerns, plus an additional use case view.

2.1. Logical View:

- Diagram Type: Class Diagram

- Description: Focuses on the key abstractions and entities in the system, representing the relationships and interactions between them. It provides a high-level view of the system's structure.

Class diagram showcasing the FoodOrdering class and its relationships with other classes representing menu items, orders, and tables.



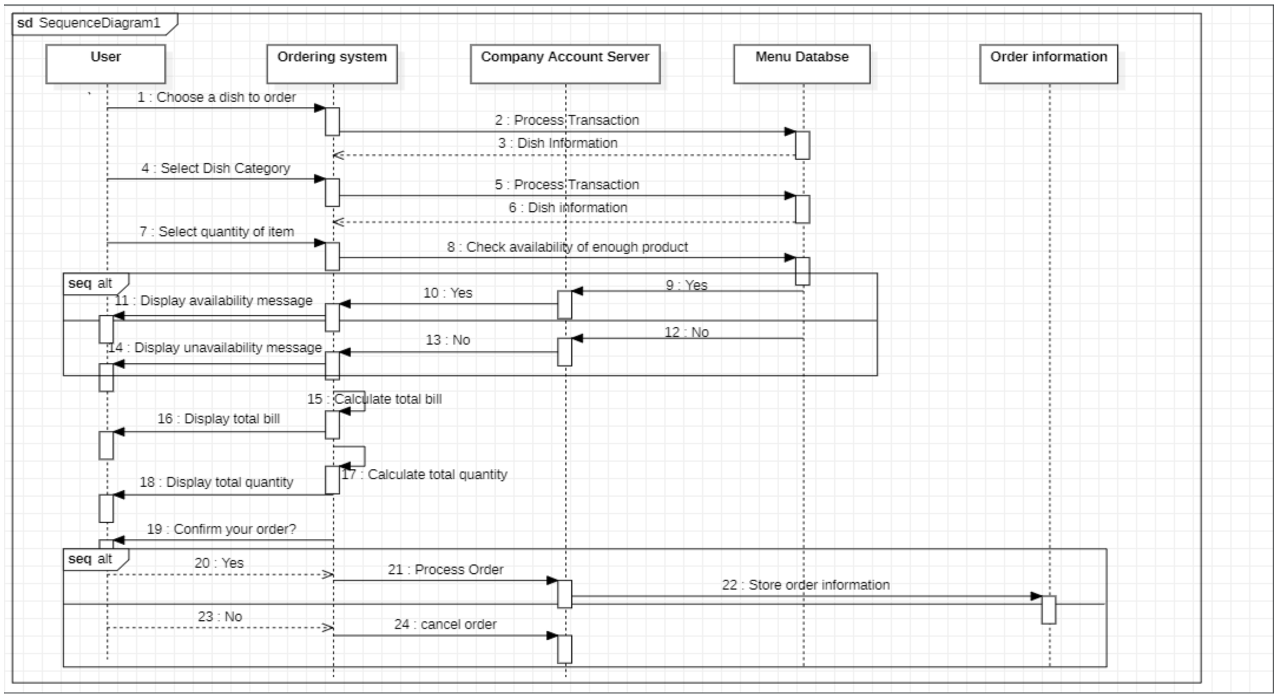
2.2. Process View:

- Diagram Type: Sequence Diagrams or Activity Diagrams

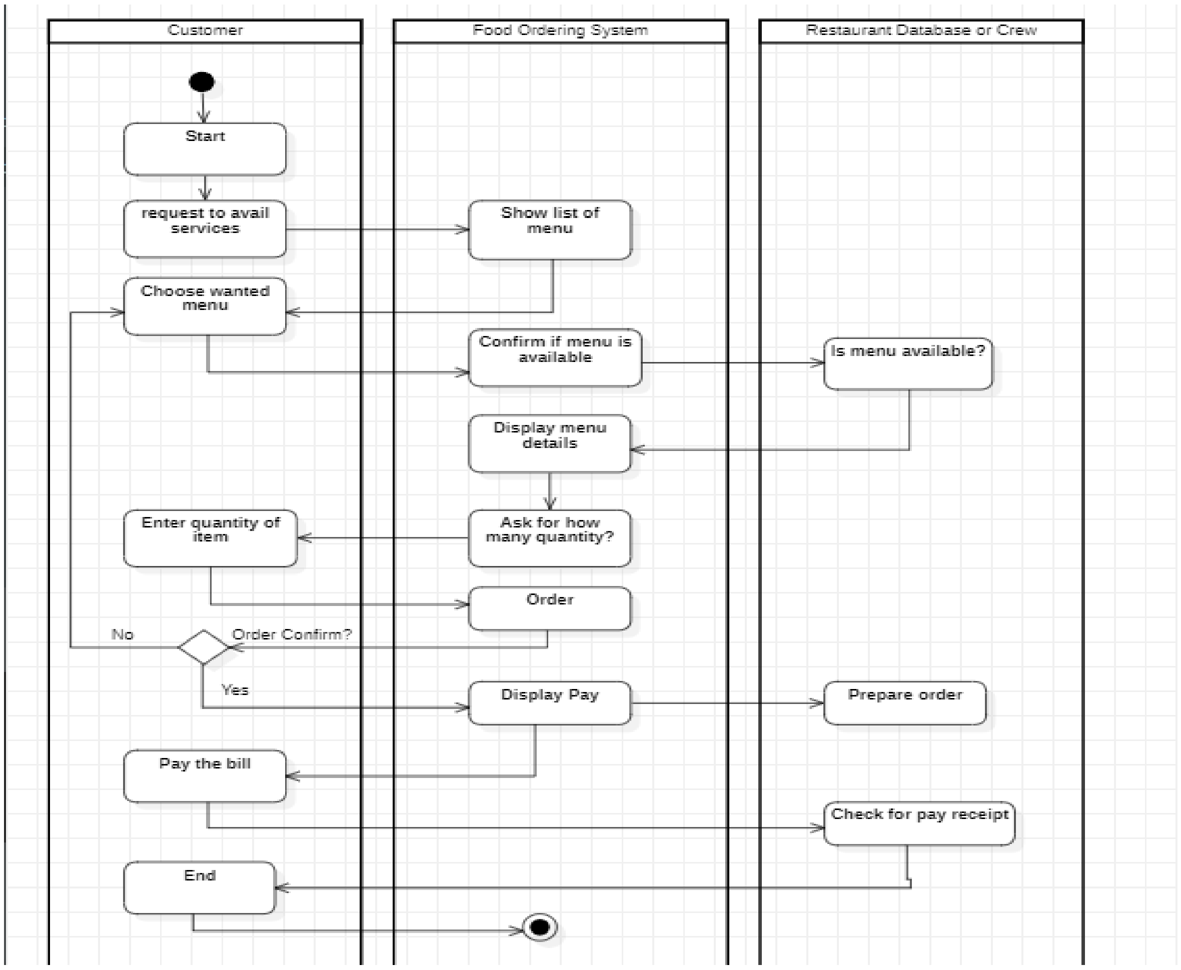
- Description: Illustrates the dynamic aspects of the system, emphasizing the flow of control between different processes or components. It shows how various processes collaborate to achieve specific functionalities.

Sequence diagrams or activity diagrams depicting the flow of control during processes like order processing, table reservation, and bill calculation.

Sequence Diagram:



Activity Diagram:

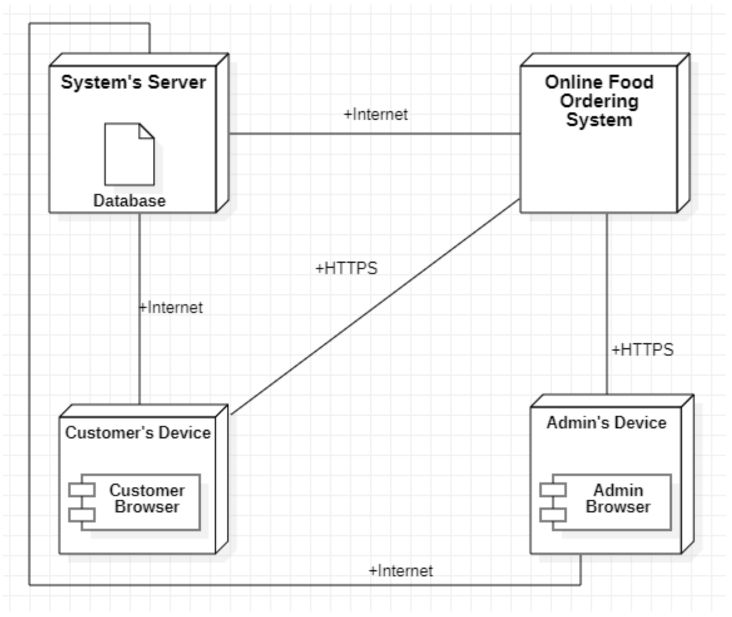


2.3. Physical View:

- Diagram Type: Deployment Diagram

- *Description:* Describes the physical architecture of the system, detailing the distribution of components across hardware nodes. It shows how software components are deployed on hardware resources.

Deployment diagram illustrating the physical distribution of the system components, possibly showing the client device, server, and database.

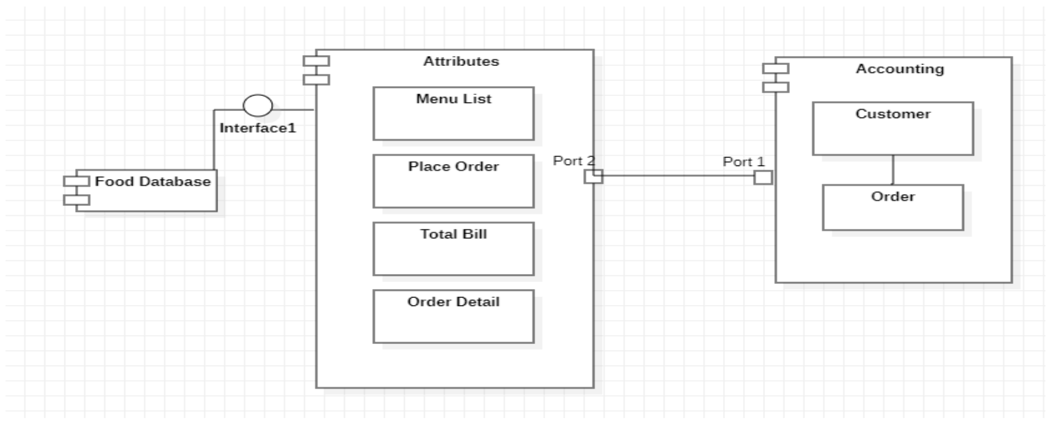


2.4. Development View (Implementation View):

- Diagram Type: Component Diagram or Package Diagram

- Description: Provides insights into the organization and structuring of the system's source code or executable components. It reveals how the software is organized and how components interact at the code level.

Component or package diagram revealing the organization of code components, including the FoodOrdering class and related methods.

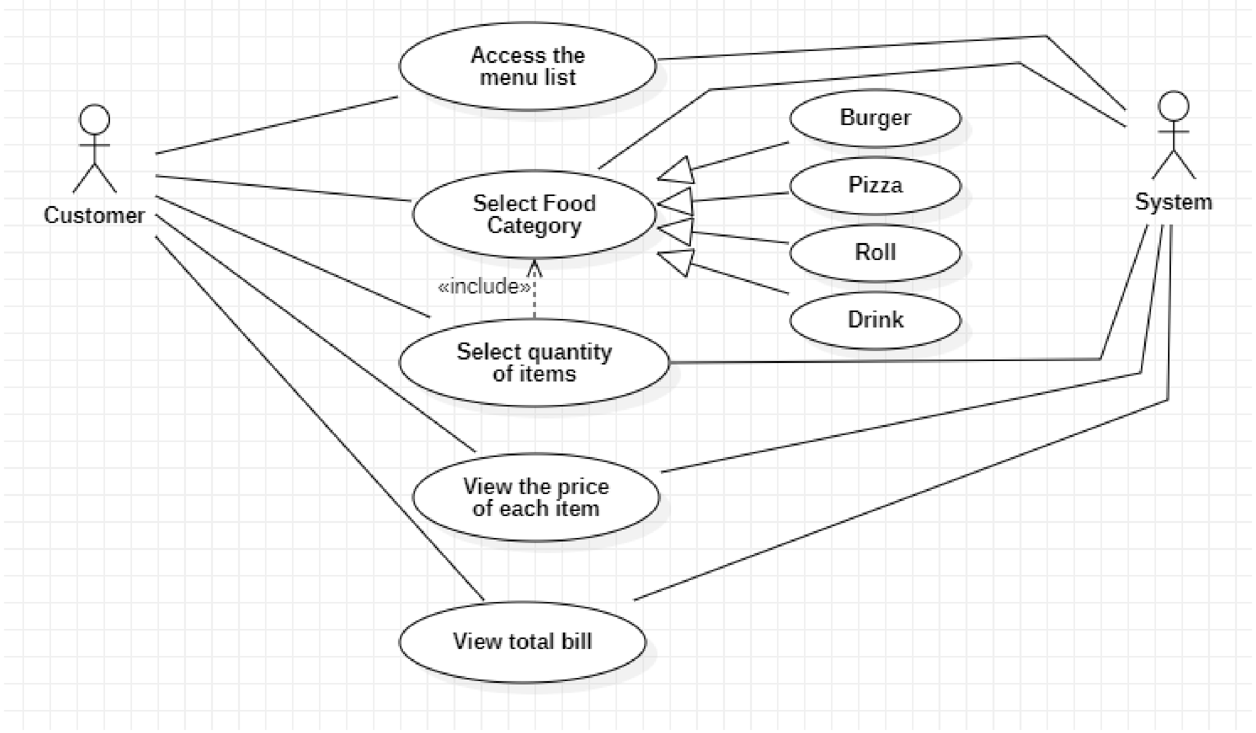


2.5. Use Case View(+1):

- Diagram Type: Use Case Diagram

- Description: Presents various use cases or scenarios that describe how users interact with the system. It helps identify system functionalities from an end-user perspective.

Use case diagram highlighting interactions like ordering different food items, reserving tables, and calculating the total bill from a user's perspective.



3)VDM

```
class Menu

types
  price = real;

instance variables
  burgerPrice: Price := 0;
  pizzaPrice: Price := 0;
  drinkPrice: Price := 0;
  rollPrice: Price := 0;
  qTable: nat := 0;
  lTable: nat := 0;
  bQ: nat := 0;
  rQ: nat := 0;
  dQ: nat := 0;
  pQ: nat := 0;
  totalPrice: Price := 0;

operations
  public Menu: () ==> Menu
  Menu() ==
    burgerPrice := 0;

    pizzaPrice := 0;

    drinkPrice := 0;

    rollPrice := 0;

    qTable := 0;

    lTable := 0;

    bQ := 0;

    rQ := 0;

    dQ := 0;

    pQ := 0;

    totalPrice := 0;


  public orderBurger: real ==> ()

  orderBurger(price) ==

    burgerPrice := burgerPrice + price;

    bQ := bQ + 1;


  public orderPizza: real ==> ()

  orderPizza(price) ==

    pizzaPrice := pizzaPrice + price;

    pQ := pQ + 1;


  public orderDrink: real ==> ()

  orderDrink(price) ==

    drinkPrice := drinkPrice + price;

    dQ := dQ + 1;


  public orderRoll: real ==> ()

  orderRoll(price) ==

    rollPrice := rollPrice + price;

    rQ := rQ + 1;


  public reserveTable: nat ==> ()

  reserveTable(quantity) ==

    IO`println("Enter number of tables you want
to reserve :  q_table");


  public calculateTotalPrice: () ==> ()

  calculateTotalPrice() ==

    totalPrice := burgerPrice + rollPrice +
pizzaPrice + drinkPrice;


  public calcTotalBill: () ==> ()

  calcTotalBill() ==

    IO`println("YOUR BILL:");


IO`println("==========================
================================");

    IO`println("You have Booked " ^ qTable ^ "
tables.");

    IO`println("Product\t\tQuantity\t\tTotal");


    if burgerPrice > 0 then

      IO`println("Burgers\t\t" ^ bQ ^ "\t\t\t" ^
burgerPrice);

    if rollPrice > 0 then

      IO`println("Rolls\t\t" ^ rQ ^ "\t\t\t" ^
rollPrice);
```

```
    if pizzaPrice > 0 then
      IO`println("Pizzas\t\t" ^ pQ ^ "\t\t\t" ^
pizzaPrice);
    if drinkPrice > 0 then
      IO`println("Drinks\t\t" ^ dQ ^ "\t\t\t" ^
drinkPrice);



IO`println("==========================
================================");
    IO`println("Total bill is: " ^ totalPrice);

end Menu

class TestMenu
operations
  public static run: () ==> ()
  run() ==
    let menu = new Menu()
    in
      menu.orderBurger(150);
      menu.orderPizza(800);
      menu.orderDrink(60);
      menu.orderRoll(100);
      menu.reserveTable(3);
      menu.calcTotalBill();
    end TestMenu
TestMenu`run();
```

4) VDM TO C++:

```
  #include <iostream>
#include <string>
#include <Windows.h>

using namespace std;
int menu_choice;
long total_price = 0;
int burger_price = 0;
int pizza_price = 0;
int drink_price = 0;
int roll_price = 0;
int q_table = 0;
int l_table = 0;
int b_q;
int r_q;
int d_q;
int p_q;

/*
I have used burger_price globally because it is initialed
with zero when while loop is executed ,so to overcome
this i globally used
*/
class menu {

public:
    int roll_choice;

    void menu_logo1() {
        HANDLE logo =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(logo, 11);
        cout <<
"\t\t\t||==================================
=====================||\n";

    }
    string cout_for_menu = "\t\t\t||\t\t ***** 5 STAR
FOOD STATION \t\t   ||\t\t";
    void menu_logo2() {

        cout <<
"\t\t\t||==================================
=====================||\n\n";
        cout << "\t\t\t\t\t
**!!!**   WELCOME   **!!!**\n";
```

```
        cout << "\t\t\t\t***  WILL BE HAPPY TO SERVE
OUR CUSTOMERS  *** \n\n";
    }
    void menu_items() {
        HANDLE menu =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(menu, 14);
        cout <<
"\t\t\t||===================================
======================||\n";
        cout << "\t\t\t||\t\t      ***** 5 STAR MENU
\t\t   ||\t\t\n";
        cout <<
"\t\t\t||===================================
======================||\n\n";

        cout << " What would you like to order Sir/Madam
? " << endl;
        cout << " Please select items from MENU CARD "
<< endl;
        cout << "1: Burgers " << endl;
        cout << "2: Rolls " << endl;
        cout << "3: Pizzas " << endl;
        cout << "4: Drinks " << endl;
        cout << "5: Book Tables " << endl;
        cout << "6: Total Bill " << endl;
        cout << "7: See Total Sell " << endl;
        cout << "0: Exit " << endl;
        cin >> menu_choice;
    }
    friend ostream& operator<< (ostream&, const menu);
};

ostream& operator << (ostream& obj, const menu obj2)
{
    obj << obj2.cout_for_menu << endl;
    return obj;
}
class roll :public menu {
private:
    int p = 0;
public:
    int roll_quantity = 0;
    long ind_price = 0;
    long total_r_price = 0;
    void showdata() {
        HANDLE roll =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(roll, 10);
        cout << "Please order a Roll you want !!!" << endl;
        cout << "1: Mayo Roll      Rs: 150 " << endl;
        cout << "2: Sasta Roll     Rs: 100 " << endl;
        cout << "3: Beef Roll      Rs: 180 " << endl;
        cout << "4: Chicken Roll   Rs: 150 " << endl;
        cout << "5: Garlic Roll    Rs: 130 " << endl;
        cout << "enter 0 for exit " << endl;
        cout << "Enter roll you want!!!!!!!!!! " << endl;
        cin >> roll_choice;
    }

    void setdata() {
        HANDLE roll =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(roll, 10);
        while (roll_choice != 0) {
            switch (roll_choice) {
            case 1:
                cout << "Enter Mayo Roll quantity you
want!!! " << endl;
                cin >> roll_quantity;
                cout << "You have ordered " << roll_quantity
<< " Mayo Rolls " << endl;
                break;
            case 2:
                cout << "Enter Sasta Roll quantity you
want!!!" << endl;
                cin >> roll_quantity;
                cout << "You have ordered " << roll_quantity
<< " Sasta Rolls " << endl;
                break;
            case 3:
                cout << "Enter Beef Roll quantity you
want!!!" << endl;
                cin >> roll_quantity;
                cout << "You have ordered " << roll_quantity
<< " Beef Rolls " << endl;
                break;
            case 4:
                cout << "Enter Chicken Roll quantity you
want!!!" << endl;
                cin >> roll_quantity;
                cout << "You have ordered " << roll_quantity
<< " Chicken Rolls " << endl;
                break;
            case 5:
                cout << "Enter Garlic Roll quantity you
want!!!" << endl;
                cin >> roll_quantity;
                cout << "You have ordered " << roll_quantity
<< " Garlic Roll " << endl;
                break;
            default:
                cout << "invalid selection" << endl;
            }
            switch (roll_choice) {
            case 1:
                p = 150;
                roll_price = roll_price + (p * roll_quantity);
                r_q += roll_quantity;
                p_q += roll_quantity;
                break;
            case 2:
                p = 100;
                roll_price = roll_price + (p * roll_quantity);
                r_q += roll_quantity;
                 p_q += roll_quantity;
                break;
            case 3:
                p = 180;
                roll_price = roll_price + (p * roll_quantity);
                r_q += roll_quantity;
                 p_q += roll_quantity;
                break;
            case 4:
                p = 150;
                roll_price = roll_price + (p * roll_quantity);
                r_q += roll_quantity;
                 p_q += roll_quantity;
                break;
            case 5:
                p = 130;
                roll_price = roll_price + (p * roll_quantity);
                r_q += roll_quantity;
                 p_q += roll_quantity;
                break;
            default:
                cout << "invalid selection" << endl;
            }
            cout << endl << "Please again order a Roll you
want !!!" << endl;
            cout << "1: Mayo Roll      Rs: 150 " << endl;
            cout << "2: Sasta Roll     Rs: 100 " << endl;
            cout << "3: Beef Roll      Rs: 180 " << endl;
            cout << "4: Chicken Roll   Rs: 150 " << endl;
            cout << "5: Garlic Roll    Rs: 130 " << endl;
            cout << "enter 0 for exit " << endl;
            cout << "Enter roll you want!!!!!!!!!! " << endl;
            cin >> roll_choice;
        }
        cout << "The Total price of rolls is " << roll_price
<< endl;
    }
};
class burger :public menu {
private:
    int p = 0;
    int burger_quantity = 0;
public:
    int burger_choice;
    long total_r_price = 0;
    void showdata() {
        HANDLE burger =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(burger, 13);
        cout << "Please order a Burger you want !!!" <<
endl;
        cout << "1: Sasta Burger       Rs: 150 " << endl;
        cout << "2: Chicken Burger   Rs: 200 " << endl;
        cout << "3: Zinger Burger    Rs: 300 " << endl;
        cout << "4: Beef Burger      Rs: 250 " << endl;
        cout << "5: Mighty Burger    Rs: 500 " << endl;
        cout << "Enter burger you want!!!!!!!!!! " << endl;
        cin >> burger_choice;
    }
    void setdata() {
        HANDLE burger =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(burger, 13);
        while (burger_choice != 0) {
            switch (burger_choice) {
            case 1:
                cout << "Enter Sasta Burger quantity you
want!!!" << endl;
                cin >> burger_quantity;
                cout << "You have ordered " <<
burger_quantity << " Sasta Burgers " << endl;
                break;
            case 2:
                cout << "Enter Chicken Burger  quantity you
want!!!" << endl;
                cin >> burger_quantity;
                cout << "You have ordered " <<
burger_quantity << " Chicken Burgers " << endl;
                break;
            case 3:
                cout << "Enter Zinger Burger quantity you
want!!!" << endl;
                cin >> burger_quantity;
                cout << "You have ordered " <<
burger_quantity << " Zinger Burgers " << endl;
                break;
            case 4:
                cout << "Enter Beef Burger quantity you
want!!!" << endl;
                cout << "You have ordered " <<
burger_quantity << " Beef Burgers " << endl;
                break;
            case 5:
                cout << "Enter Mighty Burger quantity you
want!!!" << endl;
                cin >> burger_quantity;
                cout << "You have ordered " <<
burger_quantity << " Mighty Burgers " << endl;
                break;
            default:
```

```
            cout << "invalid selection \n" << endl;
            break;
         }
         switch (burger_choice) {
         case 1:
            p = 150;
            burger_price = burger_price + (p *
burger_quantity);
            b_q += burger_quantity;
            break;
         case 2:
            p = 200;
            burger_price = burger_price + (p *
burger_quantity);
            b_q += burger_quantity;
            break;
         case 3:
            p = 300;
            burger_price = burger_price + (p *
burger_quantity);
            b_q += burger_quantity;
            break;
         case 4:
            p = 250;
            burger_price = burger_price + (p *
burger_quantity);
            b_q += burger_quantity;
            break;
         case 5:
            p = 500;
            burger_price = burger_price + (p *
burger_quantity);
            b_q += burger_quantity;
            break;
         default:
            cout << "invalid selection \n" << endl;
            break;
         }
         cout << endl << "Please again order a Burger
you want !!!" << endl;
         cout << "1: Sasta Burger     Rs: 150 " << endl;
         cout << "2: Chicken Burger   Rs: 200 " << endl;
         cout << "3: Zinger Burger    Rs: 300 " << endl;
         cout << "4: Beef Burger      Rs: 250 " << endl;
         cout << "5: Mighty Burger    Rs: 500 " << endl;
         cout << "enter 0 for exit " << endl;
         cout << "Enter burger you want!!!!!!!!!! " <<
endl;

         cin >> burger_choice;
      }
      cout << " The Total price of burgers is " <<
burger_price << endl;
   }
};
/*                Pizza class              */
class pizza :public menu {
private:
   int p = 0;
   int pizza_quantity = 0;
public:
   int pizza_choice;
   long total_r_price = 0;
   void showdata() {
      HANDLE pizza =
GetStdHandle(STD_OUTPUT_HANDLE);
      SetConsoleTextAttribute(pizza, 11);
      cout << "Please order a Pizza you want !!!" <<
endl;
      cout << "1: Arabic Ranch Pizza      Rs: 1000 " <<
endl;
      cout << "2: Dancing Fajila Pizza    Rs: 800 " <<
endl;
      cout << "3: Vegetable Pizza         Rs: 600 " <<
endl;
      cout << "4: Chicken Pizza           Rs: 850 " <<
endl;
      cout << "5: Tikka Masala Pizza      Rs: 1200 " <<
endl;
      cout << "Enter Pizza you want!!!!!!!!!! " << endl;
      cin >> pizza_choice;
   }
   void setdata() {
      HANDLE pizza =
GetStdHandle(STD_OUTPUT_HANDLE);
      SetConsoleTextAttribute(pizza, 11);
      while (pizza_choice != 0) {
         switch (pizza_choice) {
         case 1:
            cout << "Enter Arabic Ranch Pizza quantity
you want!!!" << endl;
            cin >> pizza_quantity;
            cout << "You have ordered " <<
pizza_quantity << " Arabic Ranch Pizzas " << endl;
            break;
         case 2:
            cout << "Enter Dancing Fajila Pizza  quantity
you want!!!" << endl;
            cin >> pizza_quantity;
            cout << "You have ordered " <<
pizza_quantity << " Dancing Fajila Pizzas " << endl;
            break;
         case 3:
            cout << "Enter Vegetable Pizza quantity you
want!!!" << endl;
            cin >> pizza_quantity;
            cout << "You have ordered " <<
pizza_quantity << " Vegetable Pizzas " << endl;
            break;
        case 4:
            cout << "Enter Chicken Pizza quantity you
want!!!" << endl;
            cin >> pizza_quantity;
            cout << "You have ordered " <<
pizza_quantity << " Chicken Pizzas " << endl;
            break;
        case 5:
            cout << "Enter Tikka Masala Pizza quantity
you want!!!" << endl;
            cin >> pizza_quantity;
            cout << "You have ordered " <<
pizza_quantity << " Tikka Masala Pizzas " << endl;
            break;
        default:
            cout << "invalid selection \n" << endl;
            break;
        }
        switch (pizza_choice) {
        case 1:
            p = 1000;
            pizza_price = pizza_price + (p *
pizza_quantity);
             p_q += pizza_quantity;
            break;
        case 2:
            p = 800;
            pizza_price = pizza_price + (p *
pizza_quantity);
             p_q += pizza_quantity;
            break;
        case 3:
            p = 600;
            pizza_price = pizza_price + (p *
pizza_quantity);
             p_q += pizza_quantity;
            break;
        case 4:
            p = 850;
            pizza_price = pizza_price + (p *
pizza_quantity);
             p_q += pizza_quantity;
            break;
        case 5:
            p = 1200;
            pizza_price = pizza_price + (p *
pizza_quantity);
             p_q += pizza_quantity;
            break;
        default:
            cout << "invalid selection \n" << endl;
            break;
        }
        cout << endl << "Please again a order a Pizza
you want !!!" << endl;
        cout << "1: Arabic Ranch Pizza      Rs: 1000 "
<< endl;
        cout << "2: Dancing Fajila Pizza    Rs: 800 " <<
endl;
        cout << "3: Vegetable Pizza         Rs: 600 " <<
endl;
        cout << "4: Chicken Pizza           Rs: 850 " <<
endl;
        cout << "5: Tikka Masala Pizza      Rs: 1200 "
<< endl;
        cout << "enter 0 for exit " << endl;
        cout << "Enter Pizza you want!!!!!!!!!! " <<
endl;

        cin >> pizza_choice;
      }
      cout << " The Total price of Pizzas is " <<
pizza_price << endl;

    }
};
/*                  Drink class              */
class drink :public menu {
private:
    int p = 0;
    int drink_quantity = 0;
public:
    int drink_choice;
    long total_r_price = 0;
    void showdata() {
        HANDLE drink =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(drink, 10);
        cout << "Please order a Drink you want !!!" <<
endl;
        cout << "1: Water Bottle    Rs: 70 " << endl;
        cout << "2: Fanta           Rs: 60 " << endl;
        cout << "3: Pepsi           Rs: 60 " << endl;
        cout << "4: Sting           Rs: 80 " << endl;
        cout << "5: Dew             Rs: 70 " << endl;
        cout << "Enter Drink you want!!!!!!!!!! " << endl;
        cin >> drink_choice;
    }
    void setdata() {
        HANDLE drink =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(drink, 10);
        while (drink_choice != 0) {
            switch (drink_choice) {
            case 1:
```

```
            cout << "Enter Water Bottle quantity you
want!!!" << endl;
            cin >> drink_quantity;
            cout << "You have ordered " <<
drink_quantity << " Water Bottles " << endl;
            break;
        case 2:
            cout << "Enter Fanta  quantity you want!!!"
<< endl;
            cin >> drink_quantity;
            cout << "You have ordered " <<
drink_quantity << " Fantas " << endl;
            break;
        case 3:
            cout << "Enter Pepsi quantity you want!!!" <<
endl;
            cin >> drink_quantity;
            cout << "You have ordered " <<
drink_quantity << " Pepsis " << endl;
            break;
        case 4:
            cout << "Enter Sting quantity you want!!!" <<
endl;
            cin >> drink_quantity;
            cout << "You have ordered " <<
drink_quantity << " Stings " << endl;
            break;
        case 5:
            cout << "Enter Dew quantity you want!!!" <<
endl;
            cin >> drink_quantity;
            cout << "You have ordered " <<
drink_quantity << " Dews " << endl;
            break;
        default:
            cout << "invalid selection \n" << endl;
            break;
        }
        switch (drink_choice) {
        case 1:
            p = 70;
            drink_price = drink_price + (p *
drink_quantity);
            d_q += drink_quantity;
            break;
        case 2:
            p = 60;
            drink_price = drink_price + (p *
drink_quantity);
            d_q += drink_quantity;
            break;
        case 3:
            p = 60;
            drink_price = drink_price + (p *
drink_quantity);
            d_q += drink_quantity;
            break;
        case 4:
            p = 80;
            drink_price = drink_price + (p *
drink_quantity);
            d_q += drink_quantity;
            break;
        case 5:
            p = 70;
            drink_price = drink_price + (p *
drink_quantity);
            d_q += drink_quantity;
            break;
        default:
            cout << "invalid selection \n" << endl;
            break;
        }
        cout << endl << "Please again order a Drink you
want !!!" << endl;
        cout << "1: Water Bottle     Rs: 70 " << endl;
        cout << "2: Fanta           Rs: 60 " << endl;
        cout << "3: Pepsi           Rs: 60 " << endl;
        cout << "4: Sting           Rs: 80 " << endl;
        cout << "5: Dew             Rs: 70 " << endl;
        cout << "enter 0 for exit " << endl;
        cout << "Enter Drink you want!!!!!!!!!! " <<
endl;

        cin >> drink_choice;
      }
      cout << " The Total price of Drinks is " <<
drink_price << endl;

    }
};
class total_bill :public menu, public roll, public burger {
public:
    void cal_total_bill() {
        HANDLE bill =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(bill, 13);

        // Header of the bill
        cout << "YOUR BILL:\n";
        cout <<
"=======================================
===================\n";
        cout << "You have Booked " << q_table << "
tables." << endl;
        cout << "Product\t\tQuantity\t\tTotal\n";
        // Display burger details
        if (burger_price > 0) {
            cout << "Burgers\t\t" << b_q << "\t\t\t" <<
burger_price << "\n";
        }

        // Display roll details
        if (roll_price > 0) {
            cout << "Rolls\t\t" << r_q << "\t\t\t" <<
roll_price << "\n";
        }

        // Display pizza details
        if (pizza_price > 0) {
            cout << "Pizzas\t\t" << p_q << "\t\t\t" <<
pizza_price << "\n";
        }

        // Display drink details
        if (drink_price > 0) {
            cout << "Drinks\t\t" << d_q << "\t\t\t" <<
drink_price << "\n";
        }

        // Footer of the bill
        cout <<
"========================================
====================\n";

        // Calculate and display the total bill
        total_price = burger_price + roll_price +
pizza_price + drink_price;
        cout << "Total bill is: " << total_price << "\n";
    }
};

class table :public menu {
public:
    int table_choice = 0;
    void total_table() {
        HANDLE table =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(table, 11);
        cout << " *** Available number of tables in our
hotel : 10\n\n";
    }
    void reserve_table() {
        HANDLE table =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(table, 11);
        cout << " Enter number of tables you want to
reserve :  ";
        cin >> q_table;
    }
    void booked_table() {
        HANDLE table =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(table, 11);
        l_table = l_table + q_table;
        cout << "You have Booked " << l_table << endl;

    }
};
class total_sell :public menu, public roll, public burger {
public:
    void show_sell() {
        HANDLE sell =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(sell, 14);
        cout << " The total sell of items is given below !!! "
<< endl;
        cout << " Total sell of burgers is " << burger_price
<< endl;
        cout << " Total sell of Pizzas is " << pizza_price
<< endl;
        cout << " Total sell of Drinks is " << drink_price
<< endl;
        cout << " Total sell of rolls is " << roll_price <<
endl;
        HANDLE overallsell =
GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(overallsell, 13);
        cout << " *** Overall Sell is : " << total_price <<
"  *** " << endl;
    }
};
int main() {
    menu obj2;
    obj2.menu_logo1();
    cout << obj2;
    obj2.menu_logo2();
    table t1;
    t1.total_table();
    obj2.menu_items();
    burger b1;
    roll r1;
    pizza p1;
    drink d1;
    total_bill bill;
    total_sell s1;
    if (menu_choice == 1) {
        b1.showdata();
        b1.setdata();
    }
    else if (menu_choice == 2) {
        r1.showdata();
        r1.setdata();
    }
    else if (menu_choice == 3) {
        p1.showdata();
        p1.setdata();
    }
    else if (menu_choice == 4) {
        d1.showdata();
        d1.setdata();
    }
    else if (menu_choice == 5) {
        t1.reserve_table();

        if (q_table > 10 ) {
            HANDLE sorry =
GetStdHandle(STD_OUTPUT_HANDLE);
            SetConsoleTextAttribute(sorry, 7);
            cout << "\t\t!!!!!  SORRY  !!!! " << endl;
            cout << "Number of table you want to book are
not available" << endl;
            cout << "The availabe tables are :10"<< endl;
        }
    }
    else if (menu_choice == 6) {
        bill.cal_total_bill();
    }
    else if (menu_choice == 7) {
        s1.show_sell();
    }
    else if (menu_choice == 0) {
        exit(0);
    }


    while (r1.roll_choice == 0 || b1.burger_choice == 0 ||
t1.table_choice == 0 || p1.pizza_choice == 0 ||
d1.drink_choice == 0) {

        if (r1.roll_choice == 0 || b1.burger_choice == 0 ||
t1.table_choice == 0 || p1.pizza_choice == 0 ||
d1.drink_choice == 0) {
            cout << "\n\n" << endl;
            obj2.menu_items();
            total_price = burger_price + roll_price +
pizza_price + drink_price;
            if (menu_choice == 1) {
                b1.showdata();
                b1.setdata();
            }
            else if (menu_choice == 2) {
                r1.showdata();
                r1.setdata();
            }
            else if (menu_choice == 3) {
                p1.showdata();
                p1.setdata();
            }
            else if (menu_choice == 4) {
                d1.showdata();
                d1.setdata();
            }
            else if (menu_choice == 5) {
                    t1.reserve_table();

        if (q_table > 10 ) {
            HANDLE sorry =
GetStdHandle(STD_OUTPUT_HANDLE);
            SetConsoleTextAttribute(sorry, 7);
            cout << "\t\t!!!!!  SORRY  !!!! " << endl;
            cout << "Number of table you want to book are
not available" << endl;
            cout << "The availabe tables are :10"<< endl;
        }

            }
            else if (menu_choice == 6) {
                bill.cal_total_bill();
            }
            else if (menu_choice == 7) {
                s1.show_sell();
            }
            else if (menu_choice == 0) {
                exit(0);
            }
        }
    }
    return 0;
}
```

5) TESTING CLASS:

```
class TestSwitchCases {

public:
    static void testBurgerOrder() {
        try {
            burger b1;

            // Simulate user input for burger order
        b1.burger_choice = 1;  // Choose Sasta Burger
        b1.setdata();

        // Add assertions or print statements to check the
expected output
        assert(b1.burger_price > 0);  // Ensure that
burger_price is greater than 0
        assert(b1.b_q > 0);  // Ensure that burger quantity
is greater than 0
        // You can add more assertions based on the
expected behavior of your program

        std::cout << "Test for Burger Order passed
successfully!\n";

      } catch (const std::exception& e) {
        std::cerr << "Exception in testBurgerOrder: " <<
e.what() << "\n";
      }
    }

    static void testRollOrder() {
      try {
        roll r1;

        // Simulate user input for roll order
        r1.roll_choice = 1;  // Choose Mayo Roll
        r1.setdata();

        // Add assertions or print statements to check the
expected output
        assert(r1.roll_price > 0);  // Ensure that roll_price
is greater than 0
        assert(r1.r_q> 0);  // Ensure that roll quantity is
greater than 0
        // You can add more assertions based on the
expected behavior of your program

        std::cout << "Test for Roll Order passed
successfully!\n";

      } catch (const std::exception& e) {
        std::cerr << "Exception in testRollOrder: " <<
e.what() << "\n";
      }
    }



    static void testPizzaOrder() {
    try {
        pizza p1;
// Create objects for other menu items
        burger b1;

        // Simulate user input for pizza order
        p1.pizza_choice = 1;  // Choose Arabic Ranch
Pizza
        p1.setdata();

        // Add assertions or print statements to check the
expected output
        assert(p1.pizza_price > 0);  // Ensure that
pizza_price is greater than 0
        assert(p1.p_q > 0);  // Ensure that pizza quantity
is greater than 0
        // You can add more assertions based on the
expected behavior of your program

        std::cout << "Test for Pizza Order passed
successfully!\n";

      } catch (const std::exception& e) {
        std::cerr << "Exception in testPizzaOrder: " <<
e.what() << "\n";
      }
    }

    static void testDrinkOrder() {
      try {
        drink d1;

        // Simulate user input for drink order
        d1.drink_choice = 1;  // Choose Water Bottle
        d1.setdata();

        // Add assertions or print statements to check the
expected output
        assert(d1.drink_price > 0);  // Ensure that
drink_price is greater than 0
        assert(d1.d_q > 0);  // Ensure that drink quantity
is greater than 0
        // You can add more assertions based on the
expected behavior of your program

        std::cout << "Test for Drink Order passed
successfully!\n";

      } catch (const std::exception& e) {
        std::cerr << "Exception in testDrinkOrder: " <<
e.what() << "\n";
      }
    }

    static void testTotalBill() {
      try {
        roll r1;
        pizza p1;
```

```
            drink d1;

            // Simulate user orders for different menu items
b1.burger_choice = 1;  // Choose Sasta Burger
            b1.setdata();
            p1.setdata();

            d1.drink_choice = 2;  // Choose Fanta
            d1.setdata();

            // Create total_bill object and calculate the total
bill
            total_bill bill;
            bill.cal_total_bill();

            // Add assertions or print statements to check the
expected output
            assert(bill.total_price > 0);  // Ensure that
total_price is greater than 0
            // You can add more assertions based on the
expected behavior of your program

            std::cout << "Test for Total Bill Calculation
passed successfully!\n";

        } catch (const std::exception& e) {
            r1.roll_choice = 2;  // Choose Sasta Roll
            r1.setdata();

            p1.pizza_choice = 3;  // Choose Vegetable Pizza
            std::cerr << "Exception in testTotalBill: " <<
e.what() << "\n";
        }
    }


};

int main() {
    // Call your test functions here
    cout<<"TEST FOR BURGER CLASS:"<<endl;
    TestSwitchCases::testBurgerOrder();
    cout<<"TEST FOR ROLL CLASS:"<<endl;
    TestSwitchCases::testRollOrder();
    cout<<"TEST FOR PIZZA CLASS:"<<endl;
    TestSwitchCases::testPizzaOrder();
    cout<<"TEST FOR DRINK CLASS:"<<endl;
    TestSwitchCases::testDrinkOrder();
    cout<<"TEST FOR TOTAL BILL CLASS:"<<endl;
    TestSwitchCases::testTotalBill();
    return 0;
}
```

6) TEST CASES:

MODULE 1: MENU CHOICE

TEST CASE ID	**TEST CASE NAME**	**DESCRIPTION**	**EXPECTED OUTPUT**	**ACTUAL OUTPUT**	**STATUS**
TC001	Order Burgers	Simulate the scenario where the user chooses to order burgers.	The program should display the burger menu, allow the user to enter the quantity for each type of burger.	Displayed the list of burger varieties.	PASS

TC002	Order Rolls	Simulate the scenario where the user chooses to order rolls	The program should display the roll menu, allow the user to enter the quantity for each type of roll.	Displayed the list of Roll varieties.	PASS
TC003	Order Pizzas	Simulate the scenario where the user chooses to order pizzas.	The program should display the pizza menu, allow the user to enter the quantity for each type of pizza.	Displayed the list of Pizza varieties.	PASS
TC004	Order Drinks	Simulate the scenario where the user chooses to order drinks	The program should display the drink menu, allow the user to enter the quantity for each type of drink.	Displayed the list of Drink varieties.	PASS
TC005	Book Tables	Simulate the scenario where the user chooses to book tables.	The program should prompt the user to enter the number of tables to reserve.	Asked the user to enter number of tables to reserve.	PASS
TC006	Calculate Total Bill	Simulate the scenario where the user chooses to calculate the total bill.	The program should display a detailed bill including the quantity and total price for each ordered item, and the overall total bill.	Displayed total bill comprising of quantity, total price and tables reserved.	PASS

MODULE 2: ORDER BURGER

TEST CASE ID	TEST CASE NAME	DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TB001	Order Sasta Burger	Simulate the scenario where the user orders Sasta Burger.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TB002	Order Chicken Burger	Simulate the scenario where the user orders Chicken Burger.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TB003	Order Zinger Burger	Simulate the scenario where the user orders Zinger Burger.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TB004	Order Beef Burger	Simulate the scenario where the user orders Beef Burger.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TB005	Order Mighty Burger	Simulate the scenario where the user orders Mighty Burger.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS

MODULE 3: ORDER ROLLS

TEST CASE ID	TEST CASE NAME	DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TR001	Order Mayo Roll	Simulate the scenario where the user orders Mayo Roll	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TR002	Order Sasta Roll	Simulate the scenario where the user orders Sasta Roll	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TR003	Order Beef Roll	Simulate the scenario where the user orders Beef Roll	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TR004	Order Chicken Roll	Simulate the scenario where the user orders Chicken Roll.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TR005	Order Garlic Roll	Simulate the scenario where the user orders Garlic Roll	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS

MODULE 4: ORDER PIZZA

TEST CASE ID	TEST CASE NAME	DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
TP001	Order Arabic Ranch Pizza	Simulate the scenario where the user orders Arabic Ranch Pizza	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS

TP002	Order Dancing Fajila Pizza	Simulate the scenario where the user orders Dancing Fajila Pizza	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TP003	Order Vegetable Pizza	Simulate the scenario where the user orders Vegetable Pizza	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TP004	Order Chicken Pizza	Simulate the scenario where the user orders Chicken Pizza.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TP005	OrderTikka Masala Pizza	Simulate the scenario where the user orders Tikka Masala Pizza.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS

MODULE 5: ORDER DRINK

TEST CASE ID	**TEST CASE NAME**	**DESCRIPTION**	**EXPECTED OUTPUT**	**ACTUAL OUTPUT**	**STATUS**
TD001	Order Water Bottle	Simulate the scenario where the user orders Water Bottle.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TD002	Order Fanta	Simulate the scenario where the user orders Fanta.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS

TD003	Order Pepsi	Simulate the scenario where the user orders Pepsi.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TD004	Order Sting	Simulate the scenario where the user orders Sting.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS
TD005	Order Dew	Simulate the scenario where the user orders Dew.	The program should display the quantity prompt, confirm the order.	Displayed the quantity prompt and confirmed the order.	PASS