**Application For Cafeteria Card Management**

-------------------------------------------------------------------------------------------------------------------------------------

Need to prepare a console application for Cafeteria management.

The application should allow to do below process.

# Main Menu:

1. **User Registration**
2. **User Login**
3. **Exit**

## User Registration:

On selection of the “User Registration” option from the main menu,

1. Need to get the below details from user for the user registration.
   1. Username
   2. Father Name
   3. Mobile Number
   4. MailID
   5. Gender
   6. WorkStationNumber (WS101)
   7. Balance
2. On completion of the user registration, we need to display the user Id (auto generation ID number).

## User Login

On selection of the “User Login” option from the main menu,

1. Ask for the “UserID” of the user. Check the “UserID” in the user list. If it is not present show” Invalid UserID”.
2. If UserID exists, then show the below menu.
   * 1. **Show My Profile**
     2. **Food Order**
     3. **Modify Order**
     4. **Cancel Order**
     5. **Order History**
     6. **Wallet Recharge**
     7. **Show WalletBalance**
     8. **Exit**
   1. **Show My Profile**

Show the current logged in user’s personal details in console.

* 1. **Food Order**

Show the below items where the user wishes to order and get the option.

**Default Food Details**

|  |  |  |  |
| --- | --- | --- | --- |
| **FoodID** | **FoodName** | **Price** | **AvailableQuantity** |
| FID101 | Coffee | 20 | 100 |
| FID102 | Tea | 15 | 100 |
| FID103 | Biscuit | 10 | 100 |
| FID104 | Juice | 50 | 100 |
| FID105 | Puff | 40 | 100 |
| FID106 | Milk | 10 | 100 |
| FID107 | Popcorn | 20 | 20 |

1. Create a temporary local **carItemtList**.
2. Create an Order object with current UserID, Order date as current DateTime, total price as 0, Order status as “Initiated”.
3. Ask the user to pick a product using **FoodID**, **quantity** required and calculate price of food.
4. If the food and quantity available, reduce the quantity from the food object’s “**AvailableQuantity**” then create **CartItems** object using the available data.
5. Add that object it to local **CartItemsList** to add it to cart wish list.
6. Ask the user whether he want to pick another product.
7. If **“Yes”** then show the updated **Food Details** and repeat from step “**3”**.
8. Repeat the process until the user enters **“No”.**
9. If He says No then,
10. Ask the user whether he confirm the wish list to purchase. If he says “No” then traverse the local **CartItemList** and get the Items one by one and reverse the quantity to the **FoodItem’s** objects in the **foodList**.
11. If he says “Yes” then, Calculate the total price of the food items selected using the local **CartItemList** information and check the balance from the user details whether it has sufficient balance to purchase.
12. If he has enough balance, then deduct the respective amount from the user’s balance.
13. Append the local **CartItemList** toglobal **CartItemList.**
14. Modify Order object created at step 1’s total price as total **OrderPrice** and **OrderStatus** as “**Ordered”**.
15. Then add the Order object to the Order list.
16. Show **“Order placed successfully, and OrderID is OID1001”.**
17. If he doesn’t have enough balance show “In sufficient balance to purchase.” Ask him “Are you willing to recharge.”
18. If he says “**No**” then show **“Exiting without Order due to insufficient balance”.** Then need to return the **localCartList** items to **foodItemsList**.
19. If he says “**Yes**”. Then ask him to Recharge with the total price of Order. If he recharged with that amount means continue from step 12 to continue purchase.
    1. **Modify Order**
20. Show the Order details of current logged in user to pick an Order detail by using **OrderID** and whose status is **“Ordered”.** Check whether the order details is present. If yes then,
21. Show list of Cart Item details and ask the user to pick an Item id.
22. Ensure the **ItemID** is available and ask the user to enter the new quantity of the food. Calculate the Item price and update in the **OrderPrice**.
23. Calculate the total price of Items and update in Order details entry.
24. Show Order modified successfully.
    1. **Cancel Order**
    2. Show the Order details of the current user who’s Order status is “**Ordered**”.
    3. Ask the user to pick an OrderID to cancel.
    4. Check the OrderID is valid. If not, then show “Invalid OrderID”.
    5. If valid, then Return the Order total amount to current user.
    6. Return product orderQuantity to Foodtem list’s FoodQuantity.
    7. Change the OrderStatus as Cancelled.
    8. Show “**Order cancelled successfully”** and product returned to cart.
    9. **Order History**

Show the Order details which placed by the current user.

* 1. **Recharge**

Ask the user whether he want to recharge. If yes then, Get the amount to be recharged and add the recharged amount to the balance of the user.

**f. Show WalletBalance**

Display the user’s wallet balance in console.

**g. Exit**

Exit and go to the main menu.

## Exit

Close the application.

## PersonalDetails Class

**Properties**:

* Name
* FatherName
* Gender {Enum - Select, Male, Female, Transgender}
* Mobile
* MailID

## IBalance Interface

**Properties**: WalletBalance

**Method**: WalletRecharge, DeductAmount

## UserDetails Class:

UserDetails class should inherit **IBalance & PersonalDetails**.

UserDetails class should have below properties and methods.

**Properties:**

* UserID (Auto – SF1001)
* WorkStationNumber
* Field: \_balance
* Read only property: WalletBalance.

**Methods:**

* WalletRecharge, DeductAmount

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **UserID** | **UserName** | **FatherName** | **MobileNumber** | **MailID** | **Gender** | **WorkStationNumber** | **Balance** |
| SF1001 | Ravichandran | Ettapparajan | 8857777575 | [ravi@gmail.com](mailto:ravi@gmail.com) | Male | WS101 | 400 |
| SF1002 | Baskaran | Sethurajan | 9577747744 | baskaran@gmail.com | Male | WS105 | 500 |

**Default User Details:**

## FoodDetails Class

Food Details Class should have below properties and methods.

**Properties:**

* + FoodID (Auto - FID101)
  + FoodName
  + FoodPrice
  + AvailableQuantity

## CartItem Class

Properties:

* ItemID (Auto - ITID101)
* OrderID
* FoodID
* OrderPrice
* OrderQuantity

## OrderDetails Class

Order Class should have below properties.

**Properties:**

* OrderID (Auto – OID1001)
* UserID
* OrderDate
* TotalPrice
* OrderStatus – (Default, Initiated, Ordered, Cancelled)

Default Order Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OrderID** | **UserID** | **OrderDate** | **TotalPrice** | **OrderStatus** |
| OID1001 | SF1001 | 15/06/2022 | 70 | Ordered |
| OID1002 | SF1002 | 15/06/2022 | 100 | Ordered |

Details Of Cart Item Class:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ItemID** | **OrderID** | **FoodID** | **OrderPrice** | **OrderQuantity** |
| ITID101 | OID1001 | FID101 | 20 | 1 |
| ITID102 | OID1001 | FID103 | 10 | 1 |
| ITID103 | OID1001 | FID105 | 40 | 1 |
| ITID104 | OID1002 | FID103 | 10 | 1 |
| ITID105 | OID1002 | FID104 | 50 | 1 |
| ITID106 | OID1002 | FID105 | 40 | 1 |