DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-116
Course Title: Object Oriented Programming
Complex Engineering Problem
FE Batch 2023, Spring Semester 2024
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1	**Areeba Khan**	**CS23110**
S2	**Zara Akram**	**CS23104**
S3		

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the class diagram meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) **[4 marks]**						
1	2	3	4			
The class diagram does not meet the desired specifications and is producing incorrect outputs.	The class diagram partially meets the desired specifications and is producing incorrect or partially correct outputs.	The class diagram meets the desired specifications but is producing incorrect or partially correct outputs.	The class diagram meets all the desired specifications and is producing correct outputs.			
Criterion 2: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) **[6 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 3: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 4: How friendly is the application interface? (CPA-1, CPA-3) **[2 marks]**						
1	2	3	4			
The application interface is difficult to understand and use.	The application interface is easy to understand and but not that comfortable to use.	The application interface is very easy to understand and use.	The application interface is very interesting/ innovative and easy to understand and use.			
Criterion 5: How does the student performed individually and as a team member? (CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			
Criterion 6: Does the report adhere to the given format and requirements? **[2 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

Teacher’s Signature

Frosty Flavors

ONLINE SHOPPING CART

Course: Object Oriented Programming (CS-116)

Instructor: Dr. Maria Waqas (Theory), Ms. Tehreem Khan (Practical)

Semester: Spring

Section: C

Prepared by:
Areeba Khan (CS-23110)
Zara Akram (CS-23104)

Contents

PROBLEM DESCRIPTION:

The objective was to create an online shopping cart which is a virtual shopping trolley, where shoppers can put all of their want-to-buy products in, review to make adjustments in quantity, product attributes, etc., and remove it before or during the checkout if they change their mind. The application also should allow the users to view history of their past purchases.

DISTINGHUSHING FEATURES OF PROJECT:

This shopping cart application, named "FROSTY FLAVOURS," is designed to provide users with a convenient way to manage their shopping experience. It includes features for user registration, login, browsing products, adding/removing products from the cart, checking out, and viewing purchase history. The application utilizes a graphical user interface (GUI) created with Tkinter.

User Management: Users can register with a unique username and password, storing their personal details such as first name, last name, and address.

Product Management: Products are loaded from a text file (products.txt), where each product includes attributes like ID, name, price, description, and quantity. Products can be viewed in a list format, and detailed information is displayed when selected.

Cart Operations: Users can add products to their cart, remove them, view the current contents of their cart, and check out when ready. The application manages stock availability and updates product quantities accordingly.

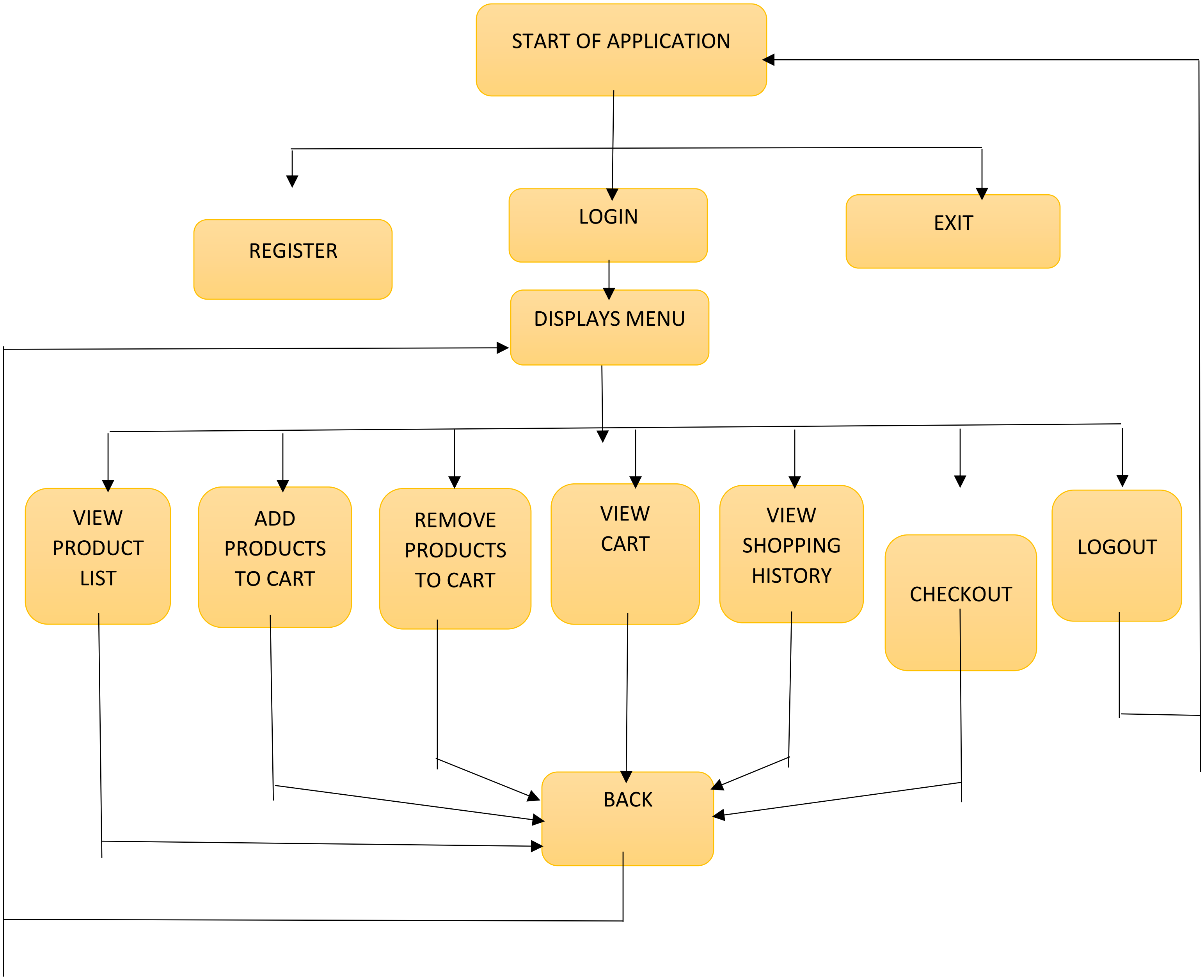
Purchase History: Each user has a purchase history that records details of their orders, including the date, items purchased, quantities, and total cost. History is saved in a user-specific text file (username_history.txt).

Error Handling and Messaging: The application provides error messages for invalid inputs, out-of-stock situations, and successful operation confirmations using Tkinter's messagebox.

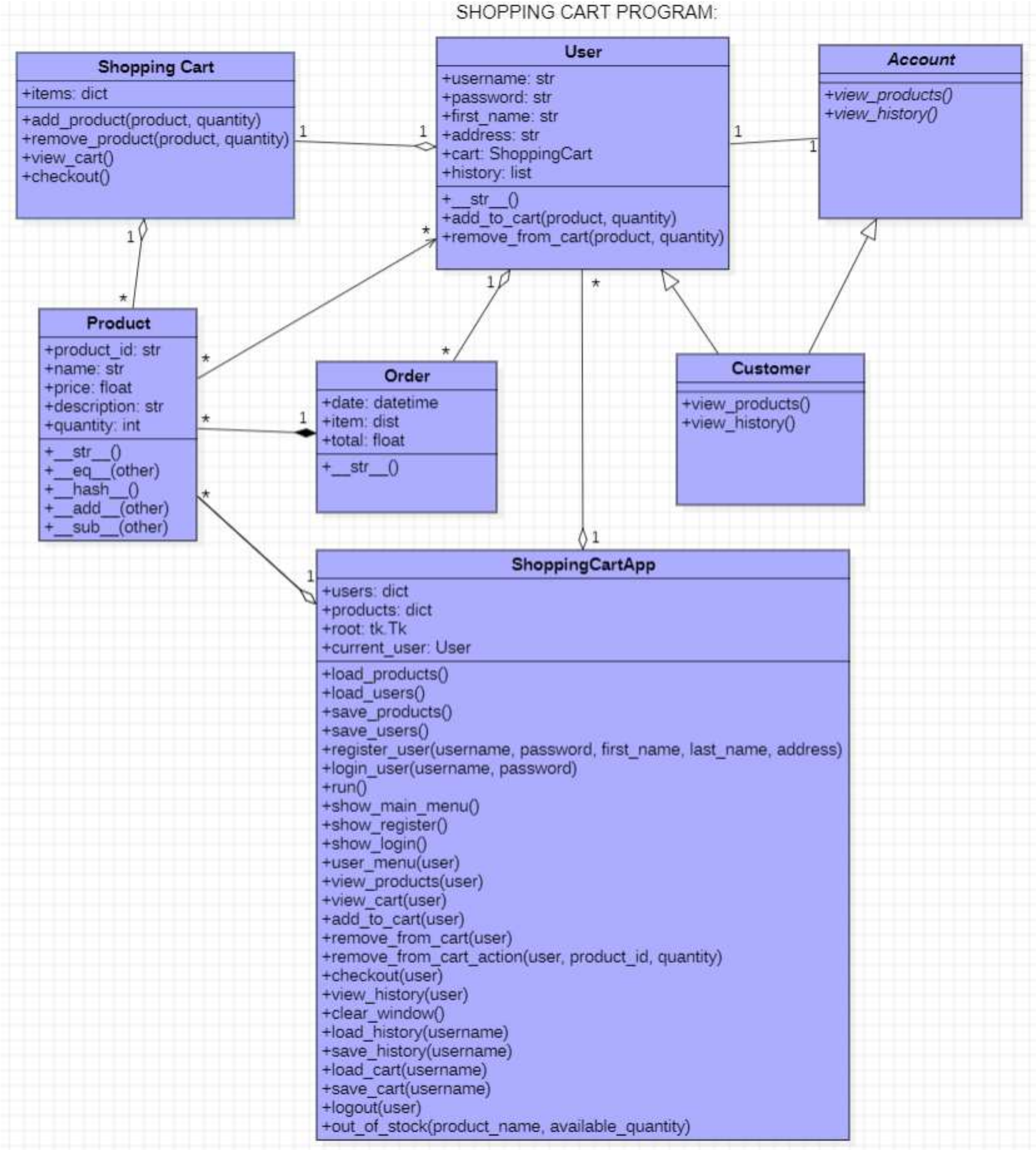
Persistence: User data, cart contents, and purchase history are saved between sessions, ensuring continuity across multiple uses of the application. Data is stored in text files (users.txt, username_cart.txt, username_history.txt) and loaded upon application startup.

GUI Design: The GUI is user-friendly, featuring a main menu for navigation, clear prompts for actions like registration and login, and intuitive buttons for adding/removing products and checking out.

FLOW OF OUR PROJECT:



Class Diagram:



Application involves following OOP concepts:

1. Inheritance:

- Customer class inherits from both User and Account.

- User and Product classes do not show explicit inheritance, but they are likely inheriting from Python's base object class implicitly.

2. Association:

- User has an association with ShoppingCart and Order through its attributes (cart and history).

- ShoppingCart contains Product objects in its items.

3. Method Overriding:

- Customer class overrides the abstract methods view_products and view_history from the Account abstract class.

4. Operator Overloading:

- Product class overloads the __eq__, __hash__, __add__, and __sub__ operators.

5. Abstract Classes:

- Account is an abstract class with abstract methods view_products and view_history.

6. Exception Handling:

- Exception handling is used in methods like load_products, load_users, load_history, load_cart, save_products, save_users, save_history, save_cart with try-except blocks to handle file-related errors.

- ShoppingCart class handles exceptions when adding or removing products (e.g., checking stock availability and cart emptiness).

Challenging Part:

1) Making sure that the data stayed the same across different parts of the application was difficult. For example, when a product's stock level changed because of a purchase, this change needed to be shown correctly in all places, like in the cart and product list.

2) Managing the reading from and writing to multiple text files (for products, users, carts, and history) was complex also History control for every user was very challenging.

3) Making sure the application could smoothly handle errors, like missing files or wrong data formats, was hard. It was important to have strong error handling to manage file read/write errors and data problems to prevent the app from crashing.

4) Additionally, We faced challenges in creating the GUI for my program. Making sure that the GUI responded smoothly to user actions and displayed information accurately took a lot of effort and troubleshooting.

New thing learnt in the Project:

We learnt concept of filing more learned in this program was how to implement persistent data storage. This involved saving and loading user data, shopping cart contents, and purchase history to and from text files.

Another thing we learned was how to create a more advanced GUI. Like how to Design a multi-screen interface, including registration, login, product list, cart, and checkout screens. Handle user input and navigation between different parts of the application. Implement dynamic elements such as list boxes with scrollbars to display lists of products and cart items.

Individual Contributions of each Group Member in the Project:

CS23104 did the following parts:

1) PRODUCT CLASS
2) SHOPPINGCART CLASS
3) ORDER CLASS
4) SHOPPINGCARTAPP
5) GUI

CS23110 did the following parts:

1) USER CLASS
2) CUSTOMER
3) ACCOUNT CLASS
4) SHOPPINGCARTAPP
5) GUI

Future Recommendations:

- **Wishlist and Save for Later:**

  Wishlist Feature: Allow users to add products to a wish-list so they can save items they are interested in for future purchases. This can help increase customer engagement and encourage return visits.

  Save for Later: Provide an option for users to move items from their cart to a "Save for Later" section. This helps users manage their current purchases while keeping track of items they may want to buy in the future.

- ### Product Reviews and Ratings:

  Review System: Allow users to leave reviews and ratings for products they purchase. This can help other customers make informed decisions and improve overall customer satisfaction.

  Moderation Tools: Include moderation tools for administrators to manage and approve reviews, ensuring the platform maintains quality and prevents spam.

- ### Enhanced Search and Filtering Options:

  Advanced Search: Implement an advanced search feature that allows users to search for products based on various criteria such as price range, category, brand, and ratings.

  Filter Options: Provide filtering options on product listing pages to help users quickly find products that match their preferences, improving the shopping experience and making it easier to navigate large inventories.

- ### ALSO;

- Stock update option can also be added for the administration purpose.
- Administrative account setup where they can handle the administration of application.
- We can also use some external libraries of python for colorful outputs.

List of References:

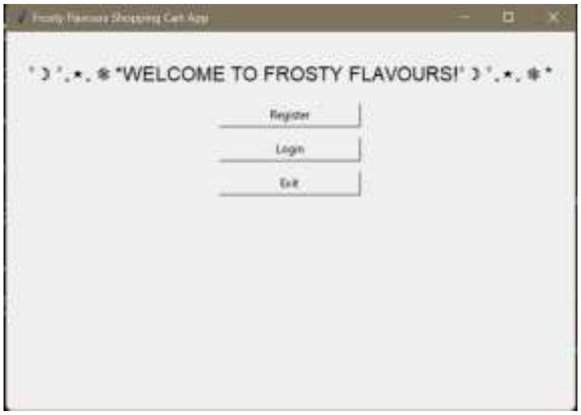
- **REFERENCE # 01:** For developing code in file **users.txt** we took help from **CHATGPT** for avoiding multiple accounts of same username and password and implemented the use above mentioned functions of (try and except) as well as raise ValueError.

- **REFRENCE # 03:** For developing code in displaying history for users we took help from **COPILOT** extension from Visual Studio and implemented the use of mentioned function of **(import)**.

- **REFRENCE # 03:** For designing in display of products by using scroll bar of GUI list and application name we took help from CHATGPT.

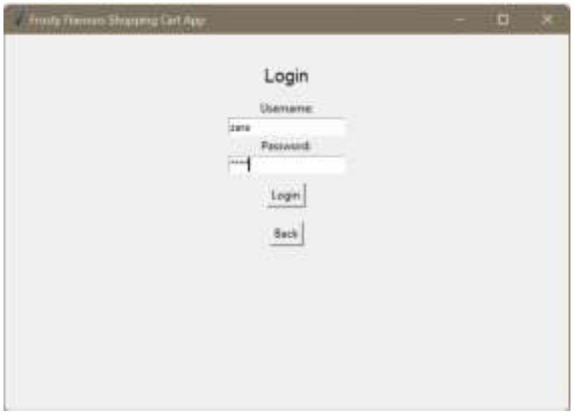
- **REFRENCE # 04:** For GUI code in displaying different errors we use Chatgpt and different YouTube sources. And to create effective Interface of our program

Application Snapshots:



˚ ☽ ˚.⋆. ❄ ˚WELCOME TO FROSTY FLAVOURS!˚ ☽ ˚.⋆. ❄ ˚

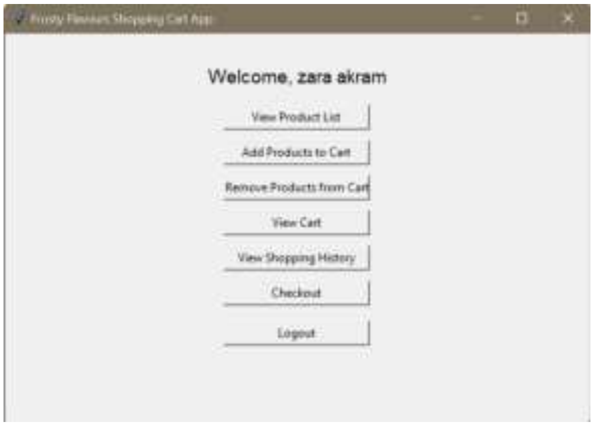
Figure 1 STARTING THE APPLICATION



Username: zara

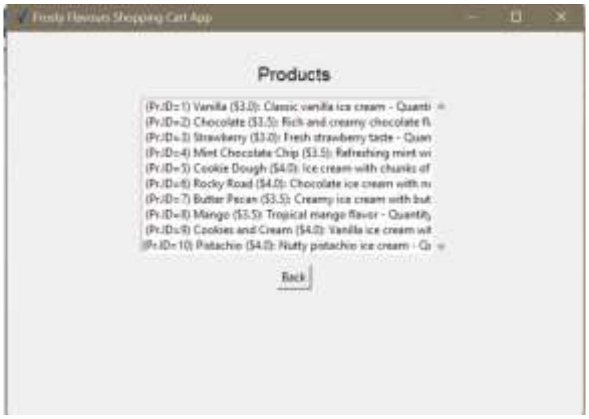
Password: ****

Figure 2 LOGGING IN



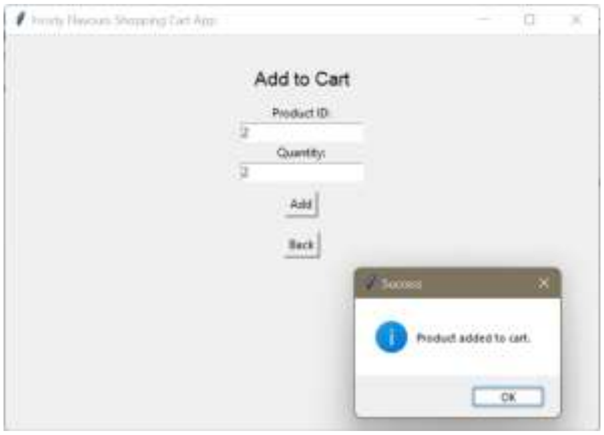
Welcome, zara akram

Figure 3 DISPLAY OF MENU



```
(Pr.ID=1) Vanilla ($3.0): Classic vanilla ice cream - Quanti
(Pr.ID=2) Chocolate ($3.5): Rich and creamy chocolate fl
(Pr.ID=3) Strawberry ($3.0): Fresh strawberry taste - Quan
(Pr.ID=4) Mint Chocolate Chip ($3.5): Refreshing mint wi
(Pr.ID=5) Cookie Dough ($4.0): Ice cream with chunks of
(Pr.ID=6) Rocky Road ($4.0): Chocolate ice cream with m
(Pr.ID=7) Butter Pecan ($3.5): Creamy ice cream with but
(Pr.ID=8) Mango ($3.5): Tropical mango flavor - Quantity
(Pr.ID=9) Cookies and Cream ($4.0): Vanilla ice cream wit
(Pr.ID=10) Pistachio ($4.0): Nutty pistachio ice cream - Q
```

Figure 4 DISPLAY OF PRODUCT

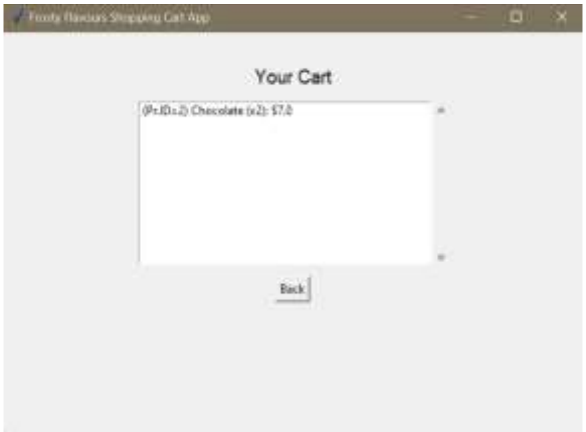


Product ID: 2

Quantity: 2

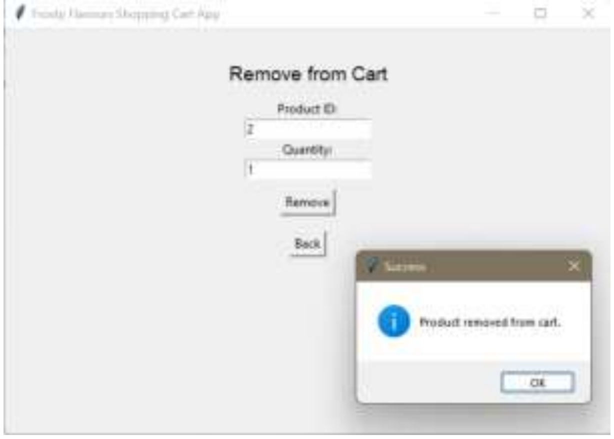
Product added to cart.

Figure 5 ADDING PRODUCTS TO CART



```
(Pr.ID=2) Chocolate (x2): $7.0
```

Figure 6 VIEWING CART



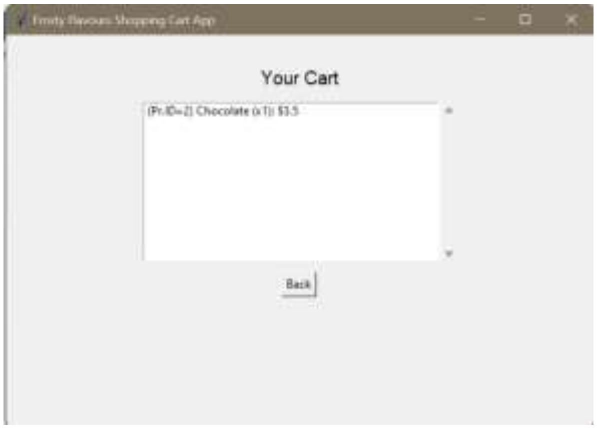
Remove from Cart

Product ID: 2

Quantity: 1

Product removed from cart.

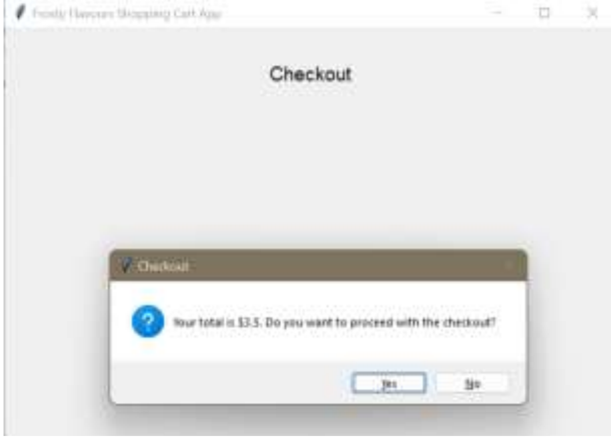
Figure 7 REMOVING PRODUCT FROM CART



Your Cart

(Pr.ID=2) Chocolate (x1): $3.5

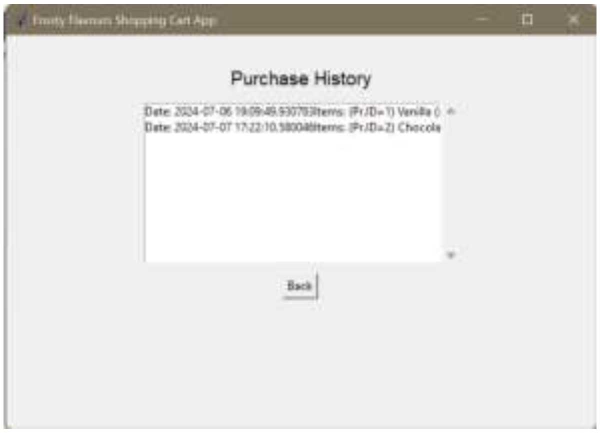
Figure 8 AGAIN VIEWING THE CART



Checkout

Your total is $3.5. Do you want to proceed with the checkout?

Figure 9 CHECKING OUT



Purchase History

```
Date: 2024-07-06 19:09:49.930793Items: (Pr.ID=1) Vanilla (
Date: 2024-07-07 17:22:10.580048Items: (Pr.ID=2) Chocola
```

Figure 10 VIEWING PURCHASING HISTORY

CONCLUSION:

Our “Frosty Flavors” application is a user-friendly, object-oriented Python program that simulates an online shopping platform. It allows users to register, log in, view products, and manage a shopping cart through a graphical interface. The program supports adding and removing products from the cart, viewing the cart's contents, and checking out with a purchase summary. It incorporates advanced object-oriented concepts like inheritance, method overriding, abstract classes, and operator overloading. Additionally, the application ensures data persistence by saving user information, cart contents, and purchase history to files. Exception handling is implemented to provide a robust and smooth user experience.