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	**SHAMEEN GHYAS**	**CS-002**
S2	**NABEEHA ASHAR**	**CS-005**
S3	**MUTAHIR AHMED**	**CS-032**

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

CONTENTS:

OBJECT-ORIENTED PROGRAMMING PROJECT

CEP REPORT

TERM PROJECT TITLE:

ONLINE SHOPPING CART **("VIRSA FOREVER")**

PROBLEM DESCRIPTION:

Online shopping cart 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 allows the users to view history of their past purchases. The main objective of our shopping cart application is to enable users to easily browse products, add items to their cart, and complete their purchase online. It aims to provide a user-friendly, secure, and efficient shopping experience.

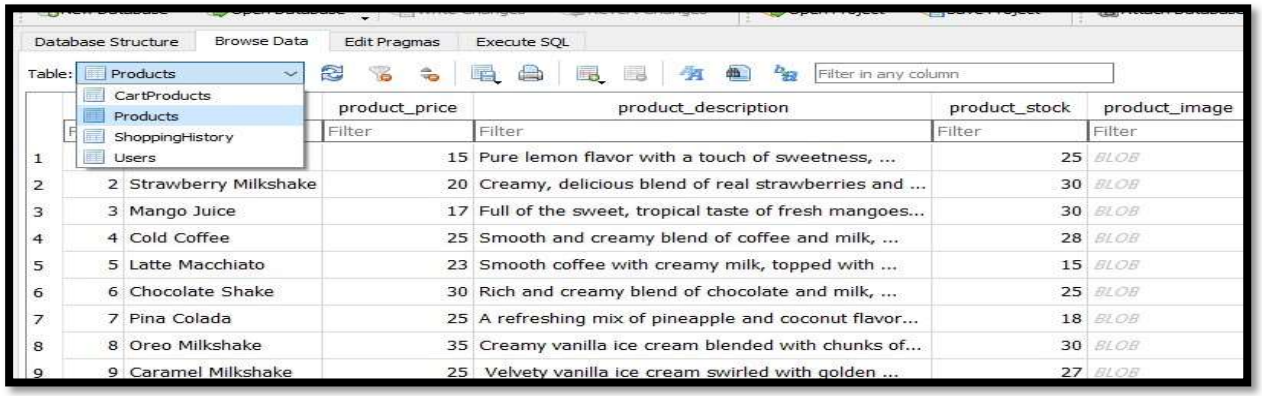
DISTINGUISHING FEATURES OF OUR PROJECT:

ROBUST:

Our shopping application functions efficiently in terms of making accounts, saving data for users and adding and removing products to cart. Cart and Shopping history is maintained for every user that can be displayed whenever user wants along with date of purchase and total price and quantity. Our application also allows the users to logout whenever they want. Moreover, it also allows the user to save carts for later use.

DATA HANDLING USING DATABASE:

We use SQLite as our database. SQLite enables efficient data management with its lightweight design and support for standard SQL commands, facilitating seamless storage and retrieval of essential project data. This allows us to store data for every user regarding their accounts, in managing products for our application, in saving Cart Product for every user and managing their shopping history by date and time.



		product_price	product_description	product_stock	product_image
		Filter	Filter	Filter	Filter
1		15	Pure lemon flavor with a touch of sweetness, ...	25	BLOB
2	2 Strawberry Milkshake	20	Creamy, delicious blend of real strawberries and ...	30	BLOB
3	3 Mango Juice	17	Full of the sweet, tropical taste of fresh mangoes...	30	BLOB
4	4 Cold Coffee	25	Smooth and creamy blend of coffee and milk, ...	28	BLOB
5	5 Latte Macchiato	23	Smooth coffee with creamy milk, topped with ...	15	BLOB
6	6 Chocolate Shake	30	Rich and creamy blend of chocolate and milk, ...	25	BLOB
7	7 Pina Colada	25	A refreshing mix of pineapple and coconut flavor...	18	BLOB
8	8 Oreo Milkshake	35	Creamy vanilla ice cream blended with chunks of...	30	BLOB
9	9 Caramel Milkshake	25	Velvety vanilla ice cream swirled with golden ...	27	BLOB

Figure 1Product table from database

WEB INTERFACE:

Our shopping cart application features a robust web interface built using Flask, a Python based web-framework. This interface enables the users to seamlessly interact with our application. We use HTML, CSS and bootstrap for web designing to provide a visually appealing and user-friendly experience. We have not used PHP for any additional functionality instead, opting for a pure Python approach.

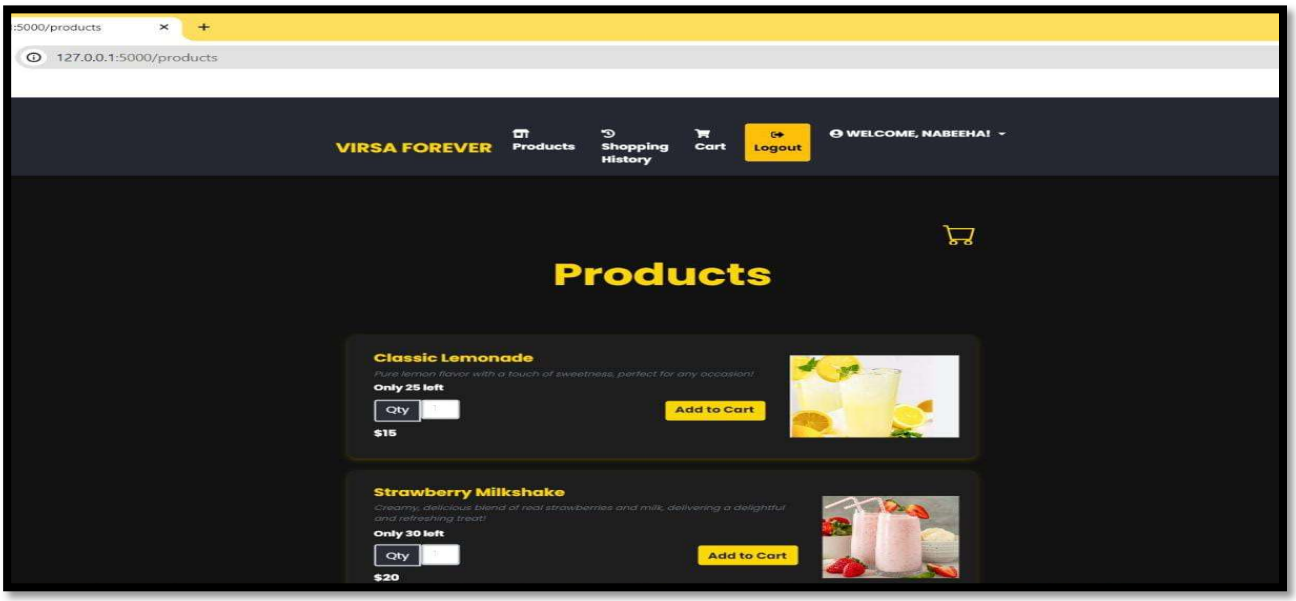


Figure 2 Product page

LOGIN AND SIGNUP:

Our application allows the user to login their accounts with their respective usernames and passwords if their account exist, otherwise user have to create account using signup option.

PAYMENT USING CREDIT CARD AND CHECKOUT PROCESS:

When user decides to checkout, all the products are enlisted along with the total price and quantity and user will proceed to payment options using credit cards. The Checkout will update the user shopping history.

PRODUCT DISPLAY:

Our application displays the product along with their prices and quantities. There are more than 10 products for the users to choose from. User can add as many products as they want from the list and they can also update their quantity after adding to cart. User can also save their carts for later use.

RESPONSIVE WEBSITE:

A responsive website is a website that adapts its layout, design, and content to fit different screen sizes and devices, such as Desktop computers, Laptops, Tablets (e.g., iPads, Android tablets), Smartphones (e.g., iPhones, Android phones). Our website is responsive as it uses various techniques, including flexible grids

and layouts, images that scale with the layout, Media queries (CSS) to apply different styles based on screen size. So, this approach ensures that our website is easy to navigate and read on any device, optimized for user experience regardless of screen size or device type. In short, a responsive website provides an optimal viewing experience across various devices, making it a crucial aspect of modern web design and development.

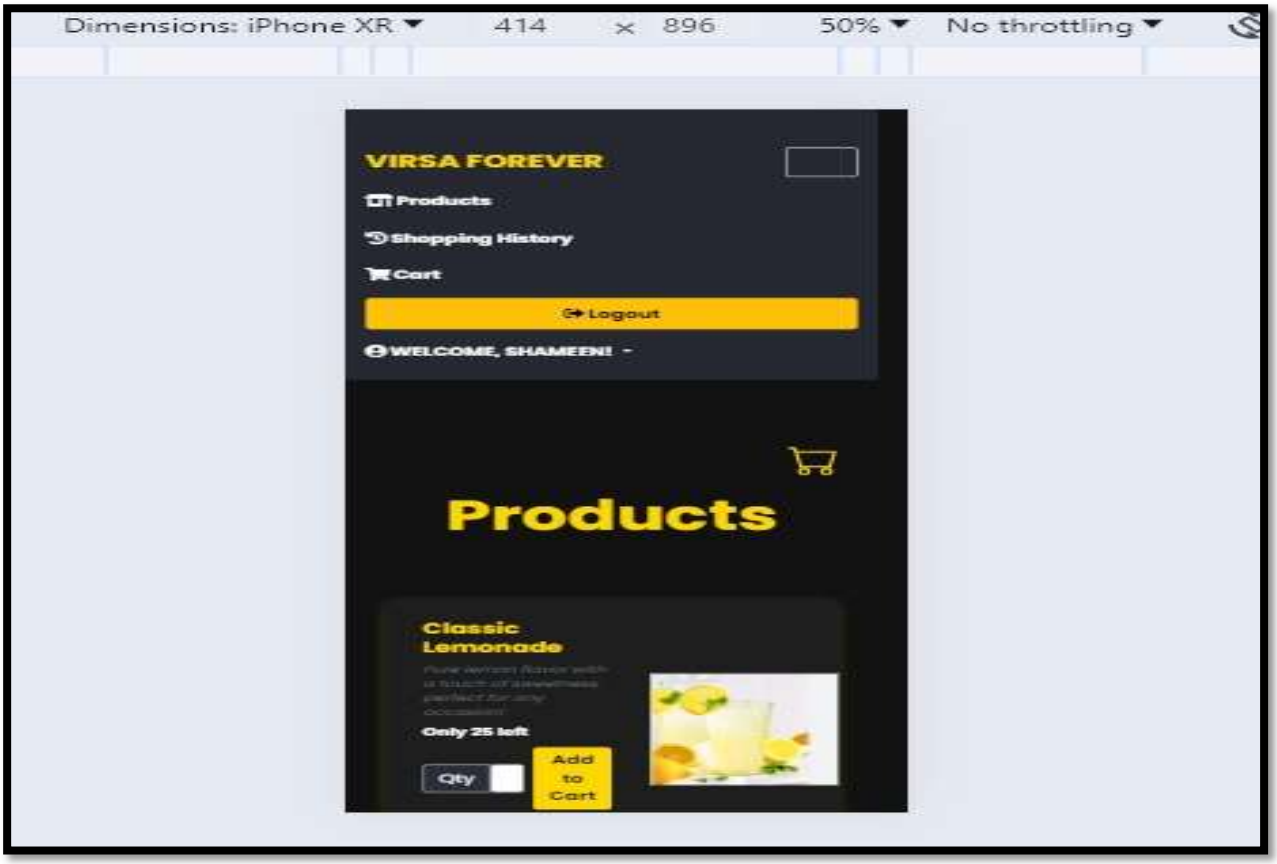


Figure 3 Website displayed on Phones

The figure 3 shows that if user runs this website on mobile it is displayed like this so it is readable and can easily be viewed

Figure 4 By clicking the square shape drop-down menu appears

In figure 4 it can be seen that if user clicks that square- type shape then the drop down menu appears.

ERROR HANDLING:

In this application messages are handled accurately. FOR EXAMPLE:

If user try to login the account without signup it will display the message “Invalid credentials” and if user does not fill all the required fields then the error message will be displayed “Please fill out the field”. Similarly, it will also show error if user try to signup by using same username.

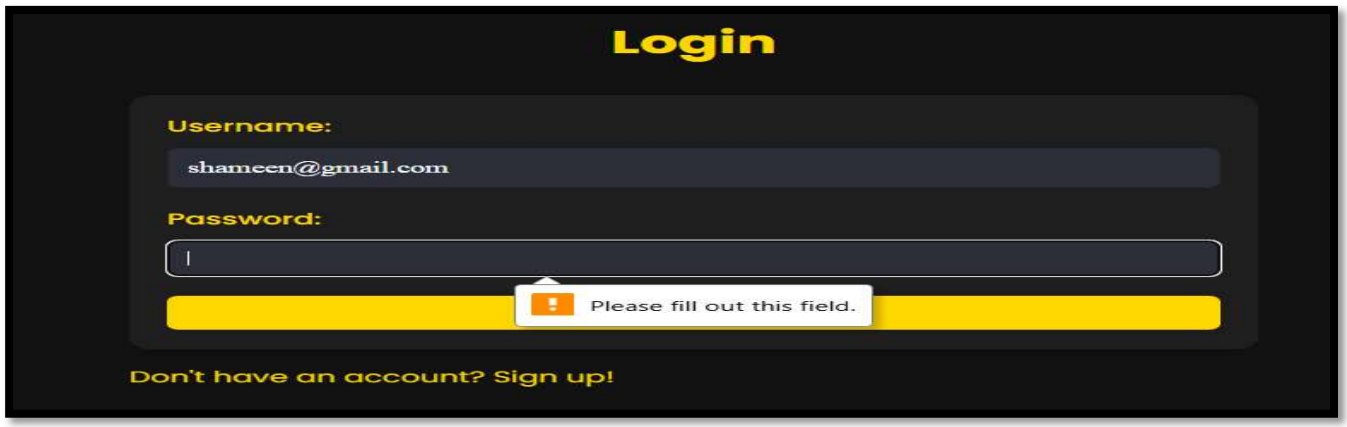


Figure 5error message if user does not fill all the required field

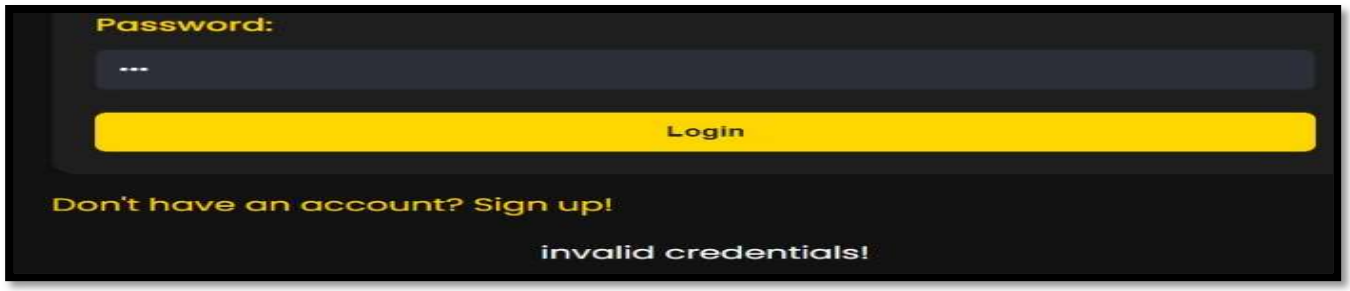
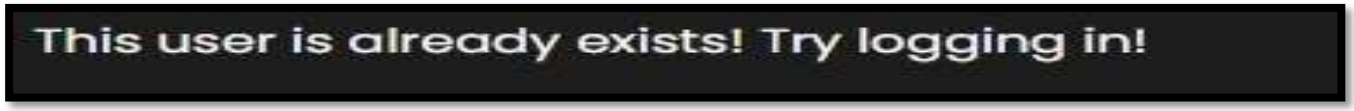


Figure 6 If user does not enter anything correctly

If user tries to make two accounts (admin and customer) with same username error message is displayed.



In our application, if the admin attempts to remove or update a product when there are no products in the database, the system displays a message indicating that “There are no products in the database”.

After adding products to cart the users cannot add value greater than quantity and they cannot add value in negative otherwise the message will be “Please enter value greater than 1”.

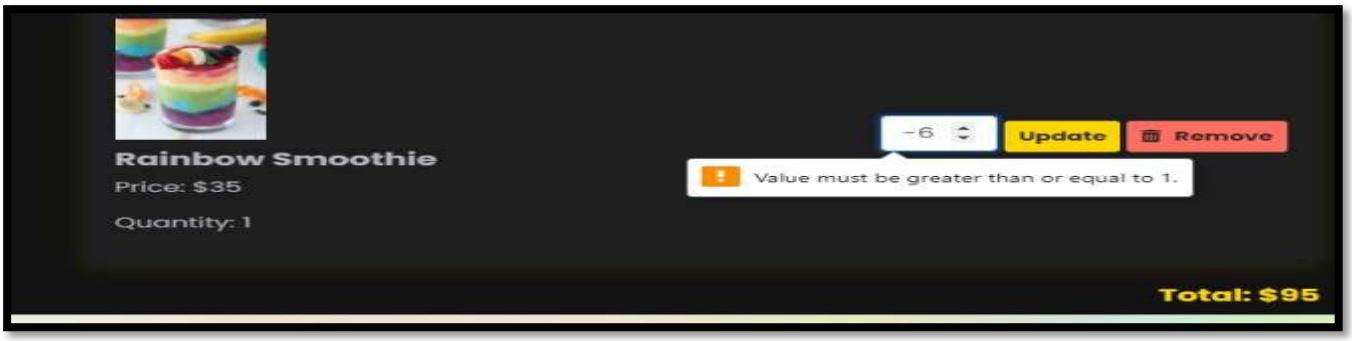


Figure 7 If user enters a negative value

If the Admin add or update more products to database then the message will be displayed like “Product is added or updated to database”.

If Admin tries to update the product name with incorrect name error message is displayed.

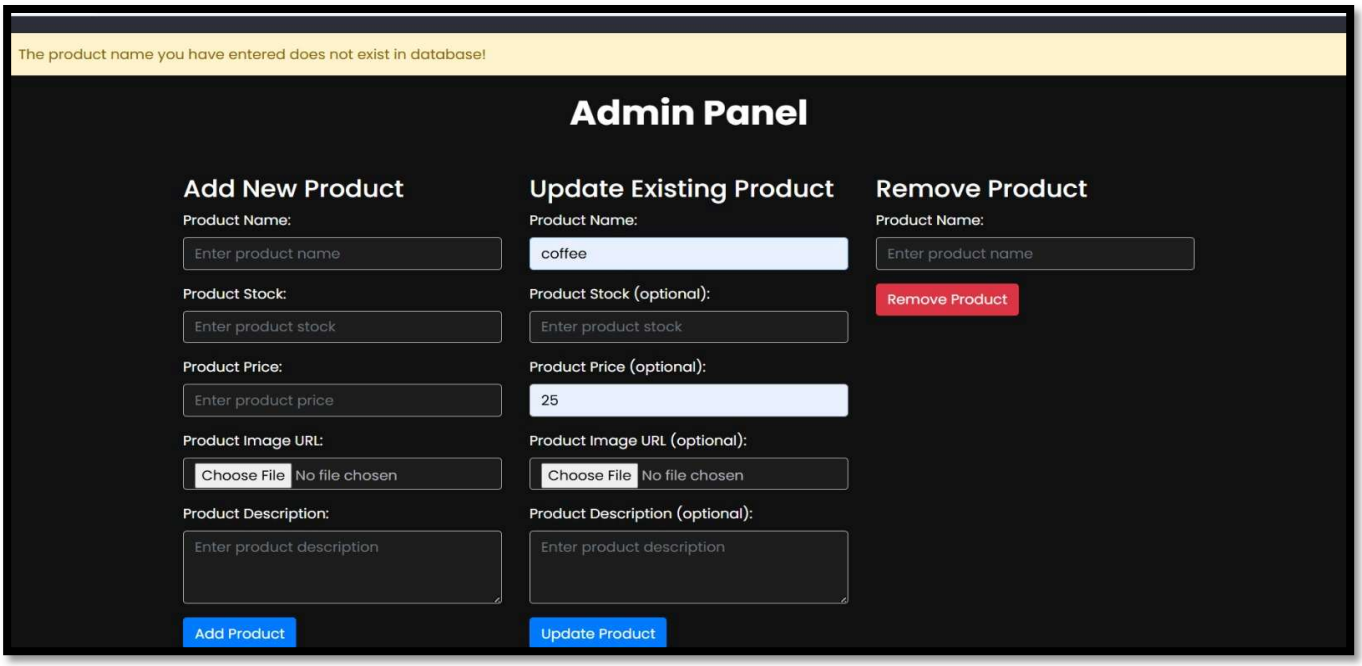


Figure 8 if admin enter product name wrong

Similarly, all messages like the product is added or remove from cart is displayed accordingly.

ADMIN ACCOUNT:

The Admin account can add new products to the database. He can also update the existing products, price quantity images and remove the product. If Admin removes the product then it will also be removed from the customer shopping cart.

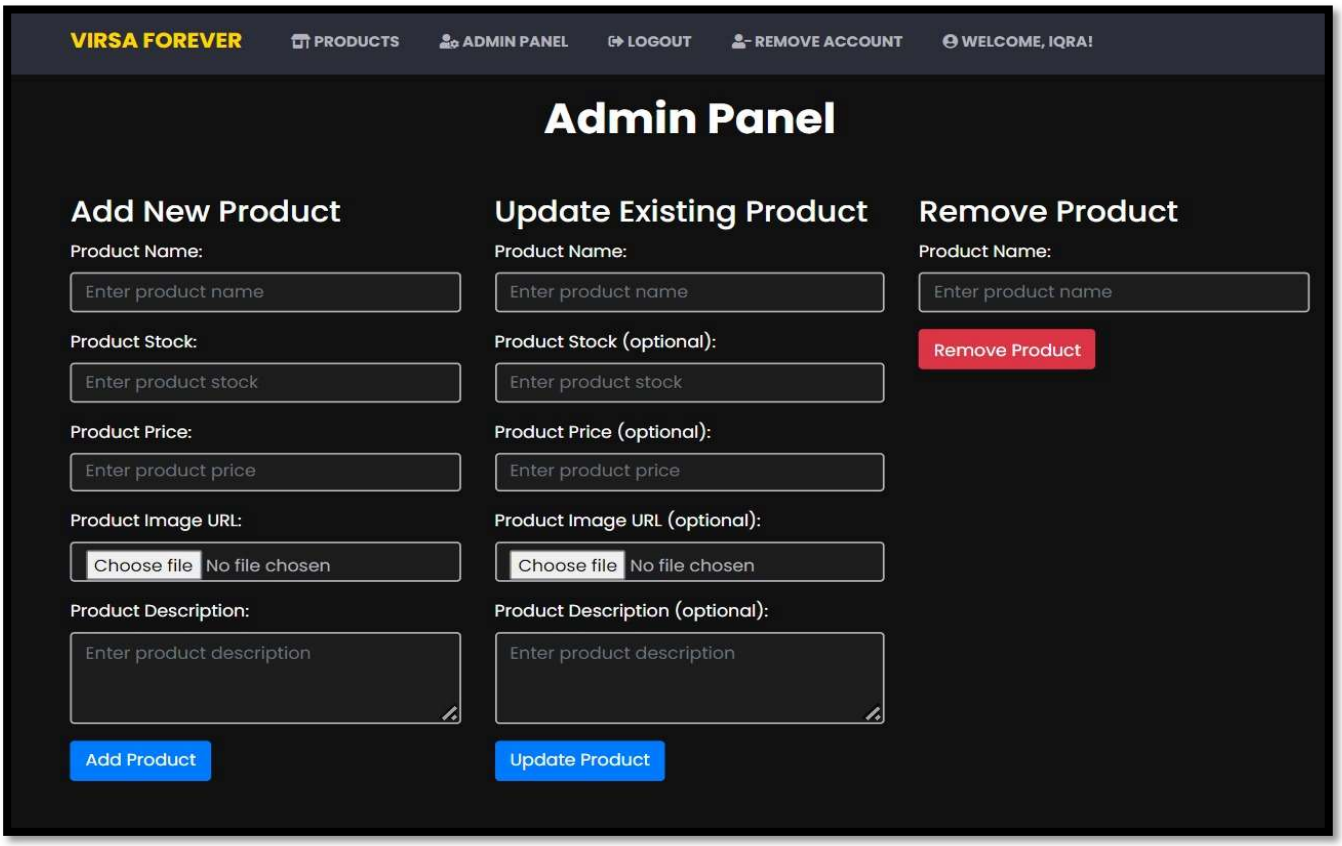
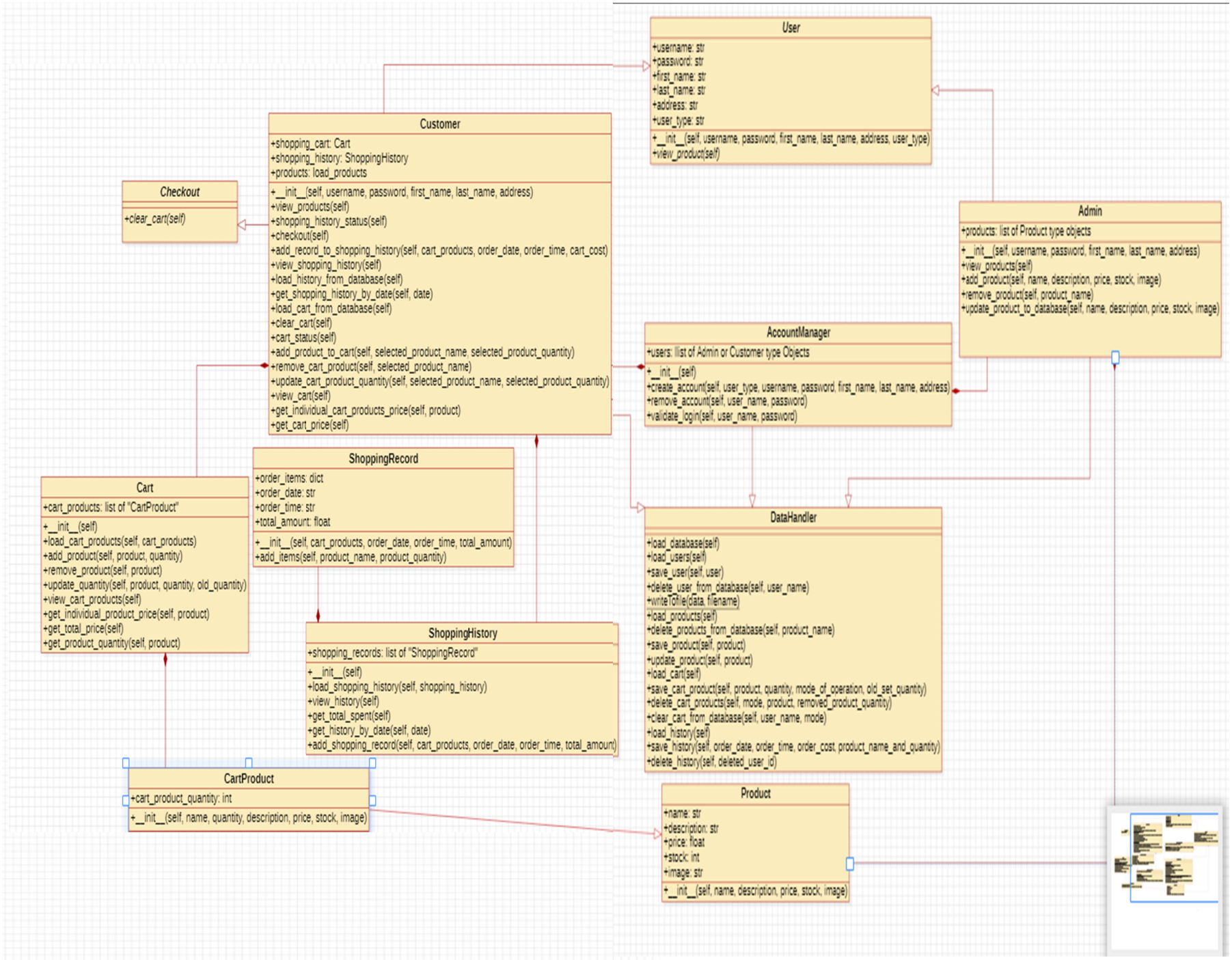


Figure 9 Admin Panel for updating products

CLASS DIAGRAM:



EXPLANATION OF CLASS DIAGRAM:

This class diagram represents the structure and relationships of the key components in our system. It illustrates both abstract and concrete classes, detailing their attributes, operations, and how they interact with one another. The diagram serves as a blueprint for understanding the design and functionality of the system.

CLASS USER:

The class User is an abstract class which defines the common attributes and behaviors that all user types in the system should have. It acts as a blueprint for specific types of users, like customers or administrators, making sure they have the same basic structure and features.

CLASS ADMIN:

The class Admin inherits from class User and class DataHandler. The Admin class represents administrative user in the system. It extends the User abstract class, inheriting its attributes and methods. Moreover, it adds advanced tools from the DataHandler class. This means the Admin class has everything it needs to manage products, including the ability to add new products, remove old ones, and update existing ones. This makes it a central hub for all administrative tasks related to products. Furthermore, Admin class is associated with class Product and class AccountManager via composition. This enables the Admin class to

- Login and Signup accounts easily

- Remove accounts

- Manage products, overseeing multiple Product instances

CLASS CUSTOMER:

This class also inherits from class User, class DataHandler and class checkout, providing extra functionalities to class. This class is associated with class Cart, class shopping history and class account manager via composition This composition relation allows the customer class to encapsulate these classes, the Customer class contains instances of these classes, hiding their complexity and providing a simpler interface. Moreover, the Customer class can access the data and behavior of these classes, enabling it to perform various customer-related tasks.

This class allows the customers to

- Login and Signup accounts easily

- Remove accounts

- View product

- Add products to Cart

-Remove products from cart

-Update their carts

- Save their carts for later access without checking out

- Checkout

- View shopping history by date

CLASS ACCOUNT MANAGER:

The class AccountManager is in charge of user accounts. It stores a list of customer and admin accounts, offering a simple way to manage their accounts, all in one place. It enables the class to

- keep track of user accounts.

-add, change, or remove accounts in an organized way.

CLASS DATA HANDLER:

The DataHandler class is the brain behind the system's database. It's responsible for storing, managing, and retrieving all the data that makes the system work. This includes:

- Saving and loading user information, like their accounts and settings

- Adding, updating, and deleting products from the database

- Managing shopping carts, including saving and deleting them

- Keeping track of user history, like their past purchases and activity

In essence, DataHandler is the central hub that ensures all data is properly stored, updated, and retrieved, making it a crucial part of the system's functionality.

CLASS PRODUCT:

The Product class represents an individual product in the system. It stores the information about the product including:

- Product name

- Price

- Description

- Stock level (availability) and Image

The Product class is also connected to the Admin class through a composition relationship, which means that Admin can manage and access Product information.

CLASS CARTPRODUCT:

The CartProduct class inherits from class Product which enables it to access the product information. It represents an individual product in the cart along with its quantity.

CLASS CART:

The Cart class represents a customer's shopping cart. It has a list of CartProduct instances, which means it is composed of multiple CartProduct objects. This class is also associated with the Customer class, as a customer owns a cart. The Cart class provides the following functionalities:

- Load cart products: retrieves the list of products in the cart

- Add product: adds a new product to the cart

- Remove product: deletes a product from the cart

- Update quantity: changes the quantity of a product in the cart

- View cart: displays the products of the cart along with quantities.

- Get individual product price: retrieves the price of a specific product in the cart

- Get total price: calculates the total cost of all products in the cart

In essence, the Cart class manages the products a customer has added to their cart, and provides methods to manipulate and view the cart's products.

CLASS SHOPPING RECORD:

The ShoppingRecord class represents a single shopping transaction. It stores the following information:

- A dictionary of product name-product quantity pairs, which represents the items purchased and their respective quantities

- The date and time of the order

- The total amount paid for the purchase

In short, the ShoppingRecord class acts as a receipt or a record of a single purchase, including the items bought, the date and time of purchase, and the total cost.

CLASS SHOPPING HISTORY:

The ShoppingHistory class represents a collection of a customer's past purchases. It stores a list of

ShoppingRecord objects, each representing a single shopping event. The ShoppingHistory class is associated with the Customer class, as a customer has a shopping history. Additionally, it is composed of multiple ShoppingRecord objects, representing individual shopping transactions

The Shopping History class provides the ability to:

- display the details of past shopping transactions

- display the total amount spent across all shopping transactions

- retrieve Shopping records for a specific date or date range

CLASS CHECKOUT:

The class checkout is an abstract class having only one abstract method. This class is inherited by class customer, providing implementation for the abstract method.

In short, our class diagram demonstrates a well-designed structure that prioritizes high encapsulation and data hiding. By encapsulating sensitive data and behaviour within specialized classes, we have successfully hidden the complexities of the system. This design ensures that data is protected from external interference and misuse, and that each component has a clear and specific responsibility. The abstraction and composition relationships between components further enhance the system's modularity and scalability. Overall, our design has achieved a reliable and maintainable structure that will support the needs of our e-commerce system."

METHOD OVERRIDING:

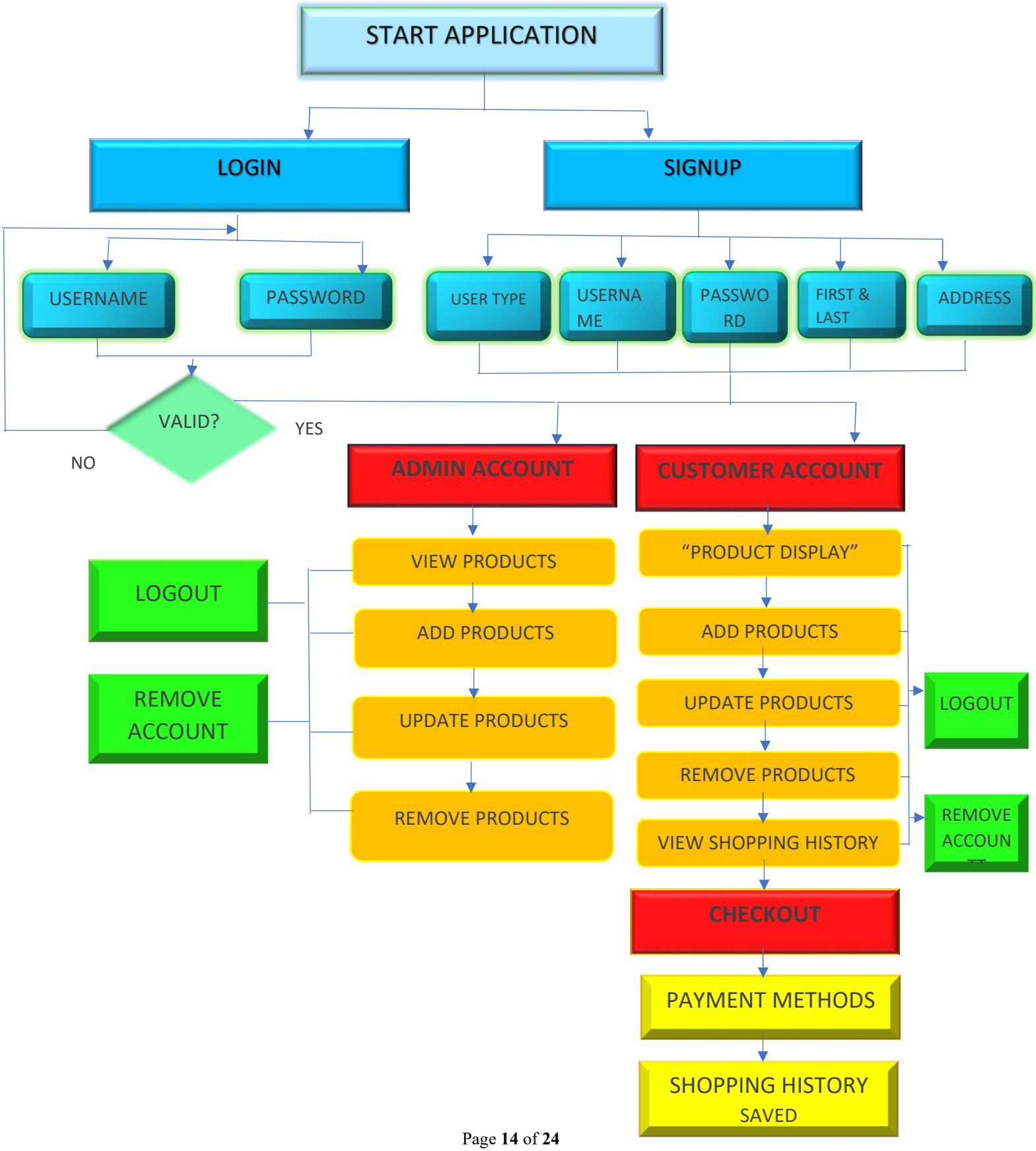
In our application, we have implemented a scalable class hierarchy by utilizing method overriding, specifically in the classes CartProduct, Admin and Customer that inherit from parent class. Each subclass (CartProduct, Admin, and Customer) is customizing its initialization process by overriding the _init_ method from its parent class.

This allows each subclass to

- set its own specific attributes and behavior

- inherit the common attributes and behavior from its parent class.

FLOW OF PROJECT:



INDIVIDUAL CONTRIBUTION OF EACH MEMBER:

Each member in our group played the crucial role in shopping cart application. All members decided the structure and relations between classes in Class Diagram and also contributing in debugging and testing the code.

Mutahir Ahmed: He developed a robust database framework, making functions for efficient data management using DataHandler class. He also skillfully designed the web interface, utilizing Flask, HTML, CSS, and Bootstrap to create an intuitive and visually appealing platform. Additionally, he implemented account management features, ensuring secure user authentication and authorization. He also assisted in every database functionality into various features, demonstrating his versatility and technical expertise.

Shameen Ghyas : She excelled in crafting a seamless checkout experience. Her work on administrative features enabled smooth system management, and her contributions to shopping history and report generation provided valuable insights. She was responsible for ensuring a seamless user experience, carefully designing and testing the interface to ensure ease of use. She also developed a user-friendly interface for managing orders, and her attention to detail ensured a polished user experience.

Nabeeha Ashar: She designed a comprehensive customer management system, effectively handling user accounts and shopping records. Her work on cart functionality was user-friendly, and her contributions to report generation provided actionable insights. She also implemented a robust search function, allowing users to quickly find products, and her work on user profiling enabled personalized recommendations, enhancing the overall user experience.

Each and every member contributed equally in terms of debugging, in writing efficient code and main code.

NEW THINGS LEARNT IN PYTHON:

DATABASE MANAGEMENT:

We learned how to design and implement a database using SQLite, creating tables, inserting, updating and retrieving data using fundamental SQL commands and implemented in our project and we also learned about the execution of database commands in Python. By creating this, the data was managed effectively.

WEB INTERFACE:

We also learned about the basics of web development using Flask, a Python based web framework. We used HTML templates to render dynamic content and we learned the basics of HTML and CSS. By using Flask to handle the backend logic and HTML, CSS and Bootstrap for frontend design and layout ,we manage to make a user friendly web interface.

EXCEPTION HANDLING AND OTHER CONCEPTS:

In our code we use exception handling using try-except blocks in classes. This helps us in handling errors. We make the use of static methods and learn more about the use of decorators. These things enhance our code's functionality and robustness.

RELATIONS AMONG CLASSES AND USING IN CODE:

We gained a deeper understanding of object-oriented programming by learning about relationships among classes, specifically composition and association. By leveraging class diagrams to visualize these relationships, we were able to clarify complex connections between classes, identify dependencies and hierarchies and write more organized, efficient, and maintainable code. In essence, understanding class relationships and implementing them in our code using composition and association enabled us to create a robust, scalable, and well-structured program.

ENCAPSULATION:

We implemented encapsulation in our project by defining classes that encapsulated related data and methods. We use database to store and manage data, leveraging its built-in data protection features. In essence, we encapsulated data and behavior within classes, using the database as a secure container to store data. This approach ensured data integrity and consistency, while also promoting modular, reusable, and maintainable code. For example, our Product class encapsulated product-related data and methods, while the database managed the storage and retrieval of that data.

MOST CHALLENGING PART:

The most challenging aspect of our shopping cart application was designing and implementing the database. We had to ensure seamless data flow across various components including Products, User accounts and Shopping history management. We overcame this by creating a Data Handler class, which encapsulated numerous functions to efficiently manage database interactions. By abstracting the database logic into Data Handler class, we successfully tackled database challenges and established a solid foundation for our application. We also faced challenges in deciding the relations between classes and we ended up making most relations as composition and association. We have also implemented web design using HTML, CSS so we also find that part a bit difficult but that was handled after exploring different design approaches.

FUTURE EXPANSIONS:

Our project fulfils every basic requirement that includes add or remove products from cart view cart, view list of products, view shopping history, Admin account etc. We would like to add:

CUSTOMER REVIEWS AND RATINGS: which will allow the user to leave reviews and ratings for products and display them on product pages.

SEARCH FUNCTIONALITY: This allows the users to search for products by name, description or category by implementing a search bar functionality.

LIST OF REFERENCES:

1) In class exercises provided by teacher
2) Internet and ChatGPT for definitions and explanations
3) All handouts provided by teacher

TEST CASE RUN SCREENSHOTS:

1) **THIS SHOWS THE WEB PAGE FOR LOGIN AND SIGNUP PAGE. USER WILL CREATE ACCOUNT USING SIGNUP AND IT SHOULD BE KEPT IN MIND THAT THE USERNAME MUST BE UNIQUE. TWO ACCOUNTS (ADMIN AND CUSTOMER) CAN'T BE WITH SAME USERNAME**

   ADMIN: To create an account the user type will be filled as Admin and user can use the website but if the account exist user will login with username and password.

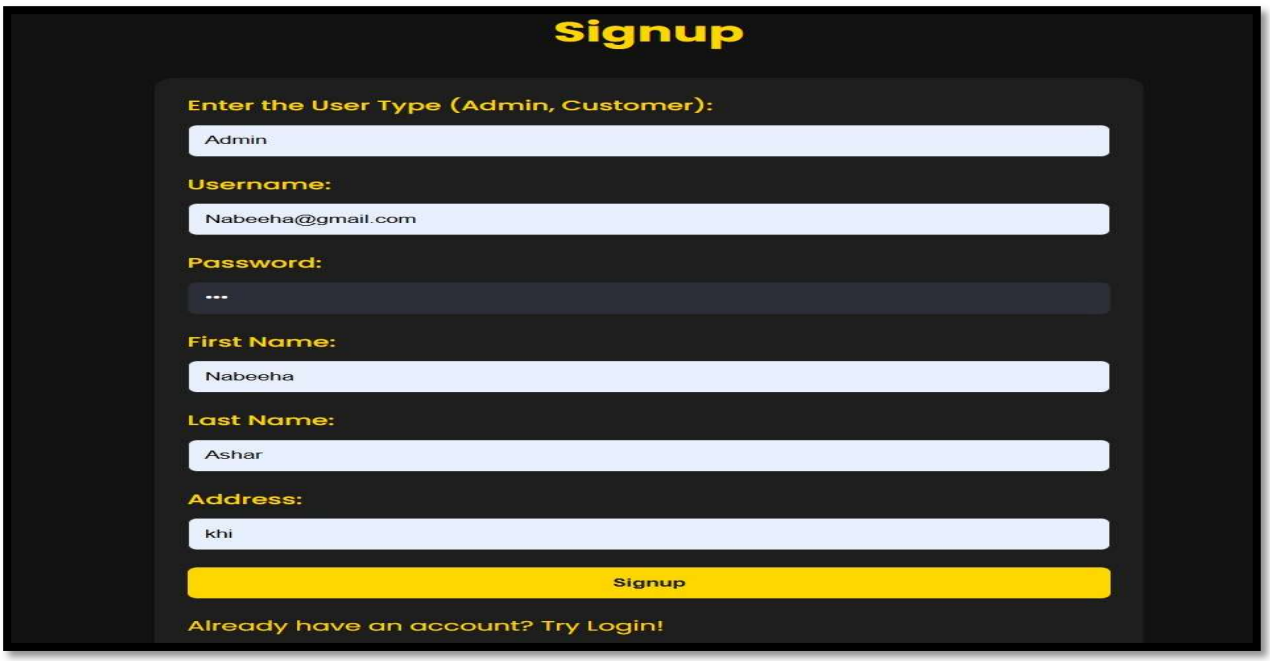


Figure 10 Sign up page

CUSTOMER: The customer can create account just like admin but the user type will be filled as "customer". If their account exists the customer will login to their respective accounts.

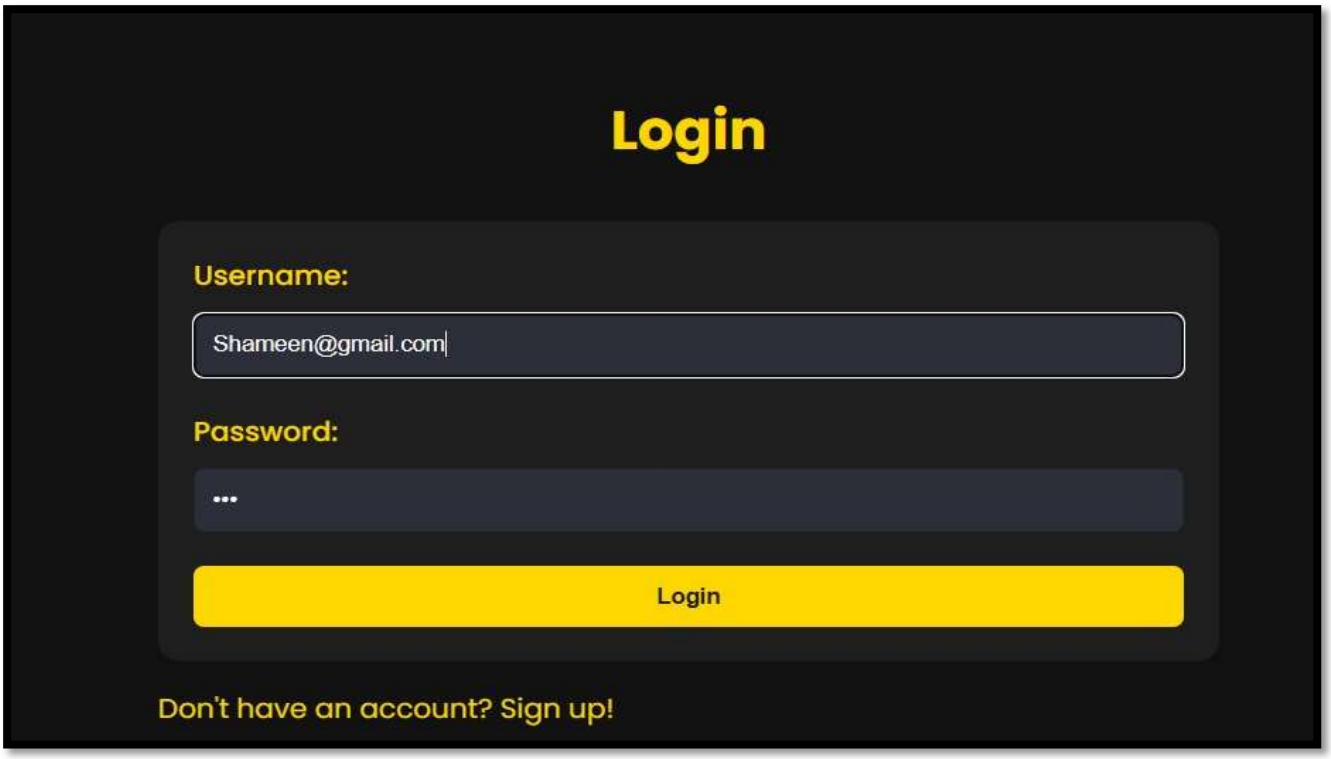
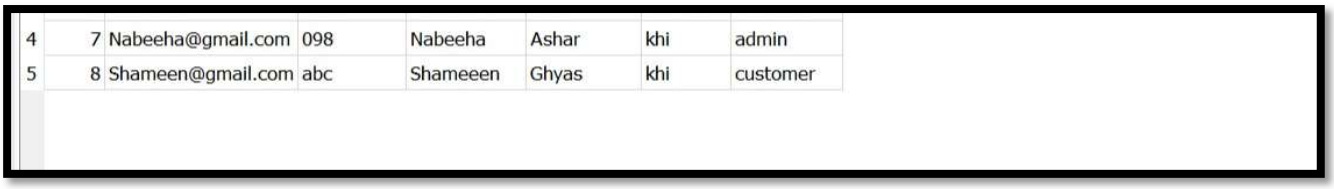


Figure 11 Login Page

DATABASE: The database table is also being updated simultaneously for admin and customer.



4	7	Nabeeha@gmail.com	098	Nabeeha	Ashar	khi	admin
5	8	Shameen@gmail.com	abc	Shameeen	Ghyas	khi	customer

Figure 12 Data base for Users

2) THIS SHOWS THE PRODUCT DISPLAY FOR USERS:

CUSTOMER: can add products to Cart from Products webpage. User can also remove products from cart or update the quantity of products later.

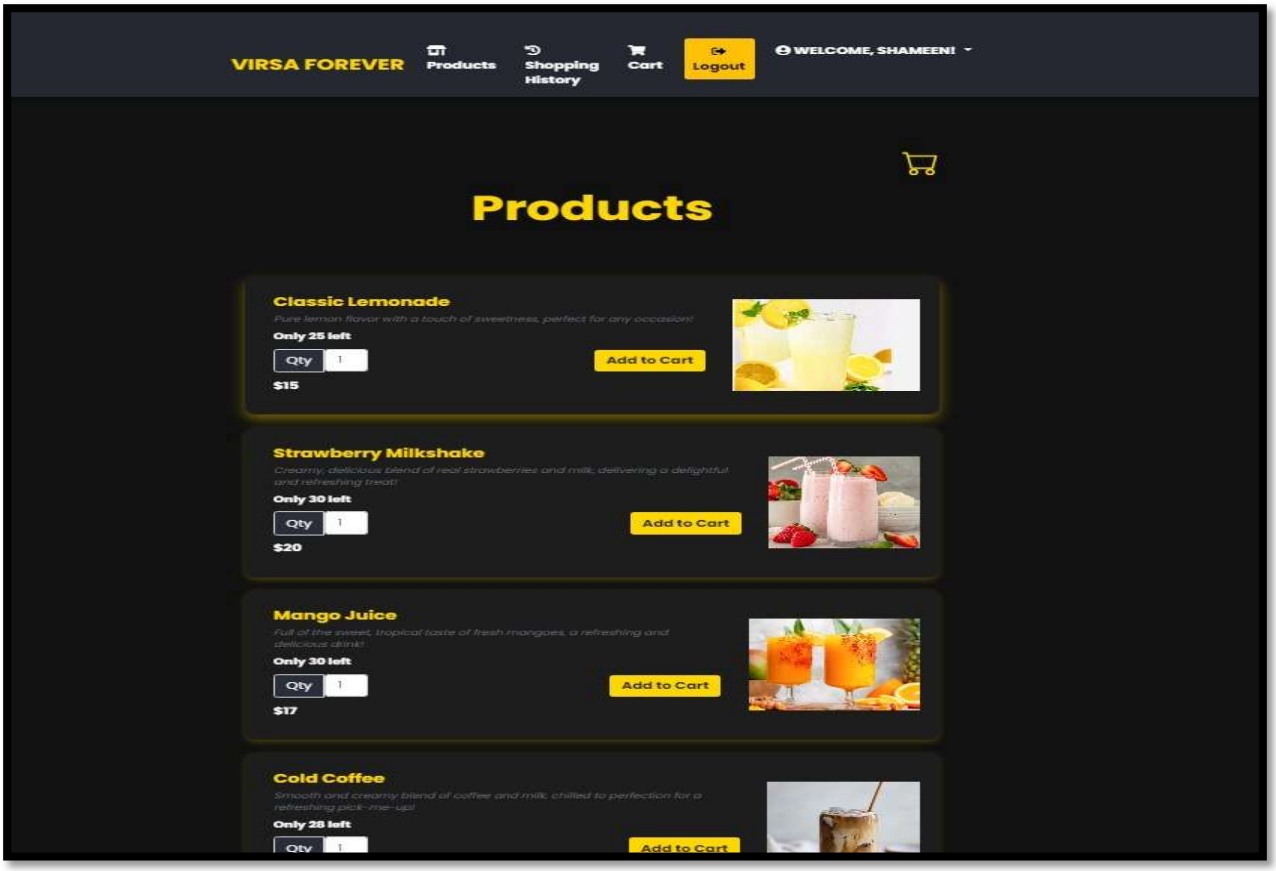
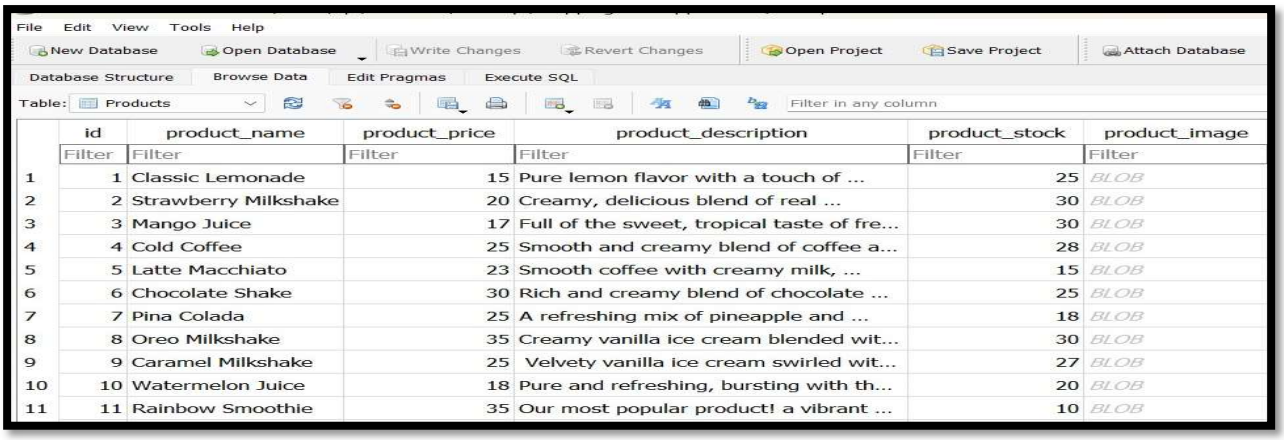


Figure 13 Product Page

DATABASE: The database table Products contains all the product names prices and stock.



File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: Products — Filter in any column

	id	product_name	product_price	product_description	product_stock	product_image
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Classic Lemonade	15	Pure lemon flavor with a touch of ...	25	BLOB
2	2	Strawberry Milkshake	20	Creamy, delicious blend of real ...	30	BLOB
3	3	Mango Juice	17	Full of the sweet, tropical taste of fre...	30	BLOB
4	4	Cold Coffee	25	Smooth and creamy blend of coffee a...	28	BLOB
5	5	Latte Macchiato	23	Smooth coffee with creamy milk, ...	15	BLOB
6	6	Chocolate Shake	30	Rich and creamy blend of chocolate ...	25	BLOB
7	7	Pina Colada	25	A refreshing mix of pineapple and ...	18	BLOB
8	8	Oreo Milkshake	35	Creamy vanilla ice cream blended wit...	30	BLOB
9	9	Caramel Milkshake	25	Velvety vanilla ice cream swirled wit...	27	BLOB
10	10	Watermelon Juice	18	Pure and refreshing, bursting with th...	20	BLOB
11	11	Rainbow Smoothie	35	Our most popular product! a vibrant ...	10	BLOB

Figure 14 Product Table in data base

3) SHOPPING HISTORY AND SHOPPING CART :

CUSTOMER: As customer add products to cart it is stored in CartProducts table in database and it is also added in the Shopping cart:

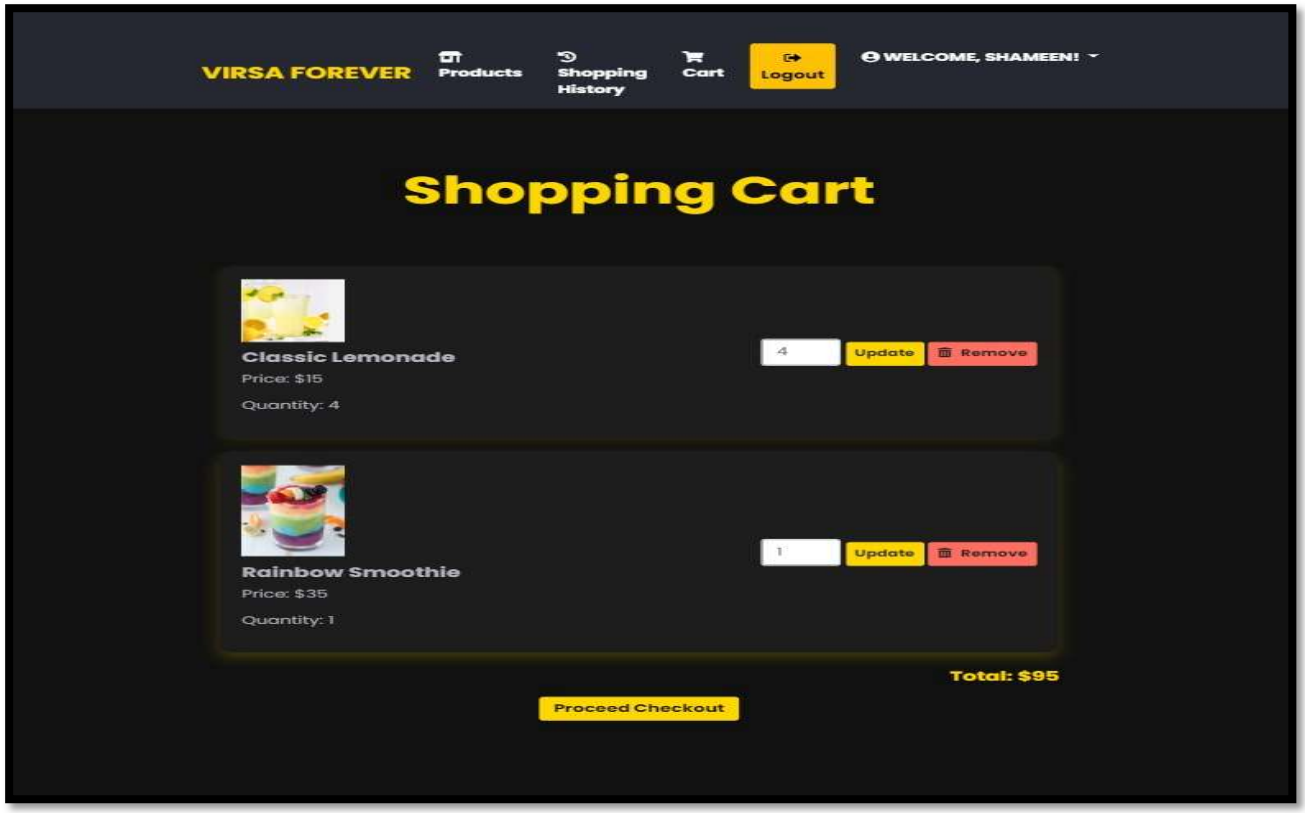


Figure 15 Shopping Cart Page

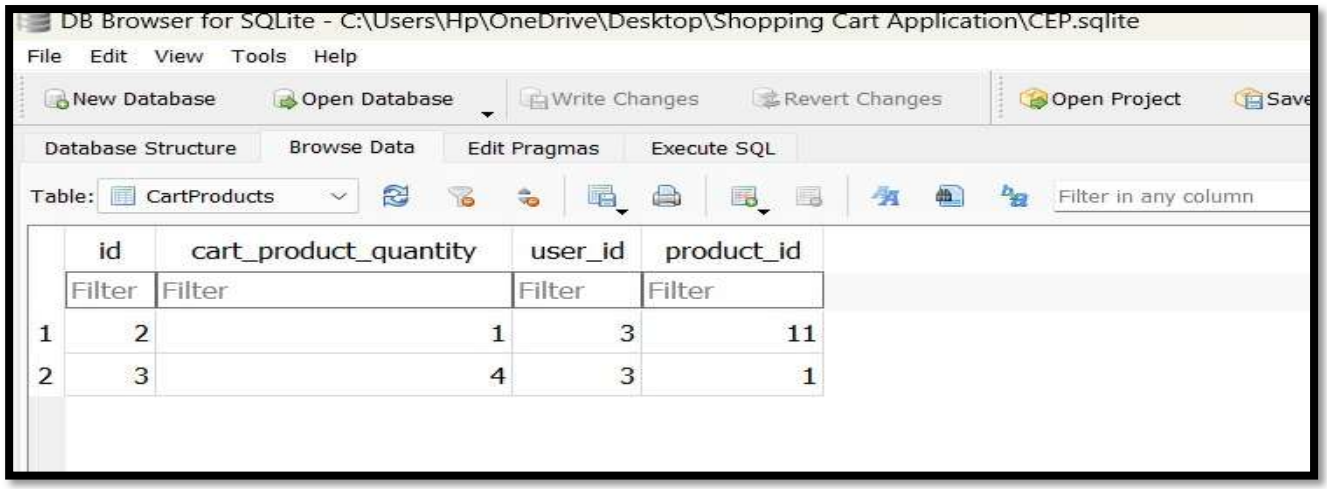


Figure 16 CartProduct table in database

CUSTOMER: As user checkout from cart the history is maintained in Shopping History and user can view their history by date and time also:

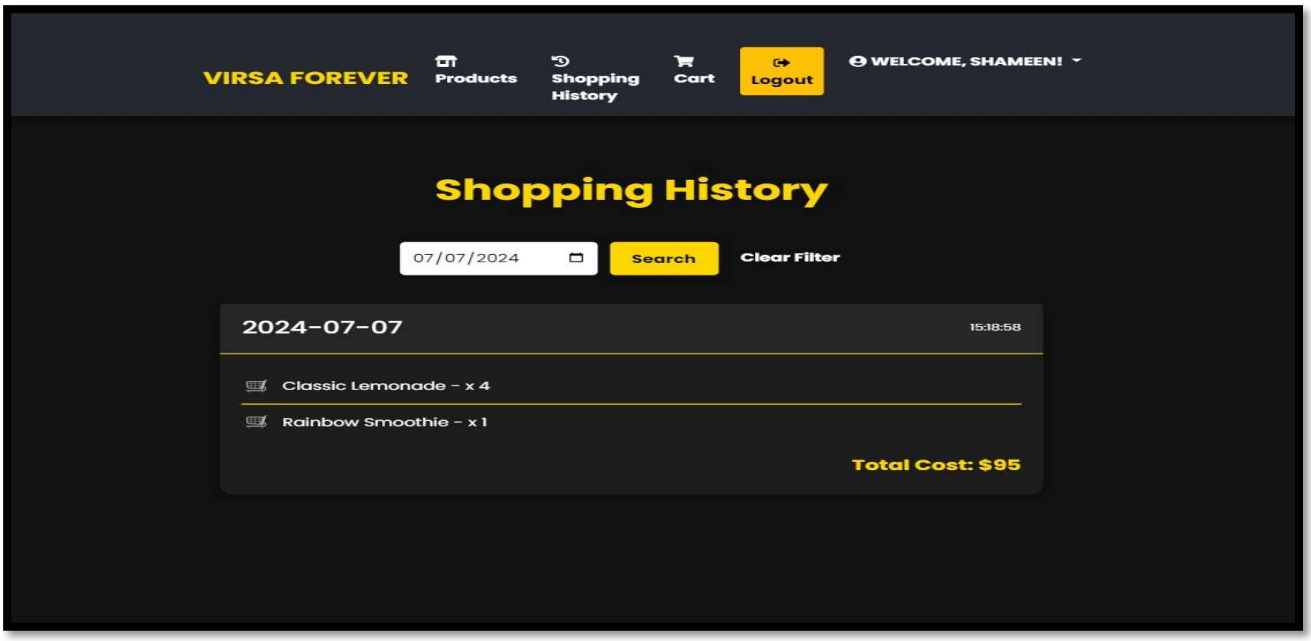


Figure 17 Shopping History Page

DATABASE: The user's history is stored in database in a table "ShoppingHistory".

4	4	2024-07-07	15:18:58	95	Classic ...	4	3
5	5	2024-07-07	15:18:58	95	Rainbow ...	1	3

Figure 18 Shopping History Table in Database

FOR ADMIN: There is Admin Panel for user to add products, remove products and update products.

4) FUNCTIONALITY OF SHOPPING CART:

Shopping cart is saved for every user. When a user buys number of products and leaves the cart without checking out so the products remain in his cart till the next time he logged- in.

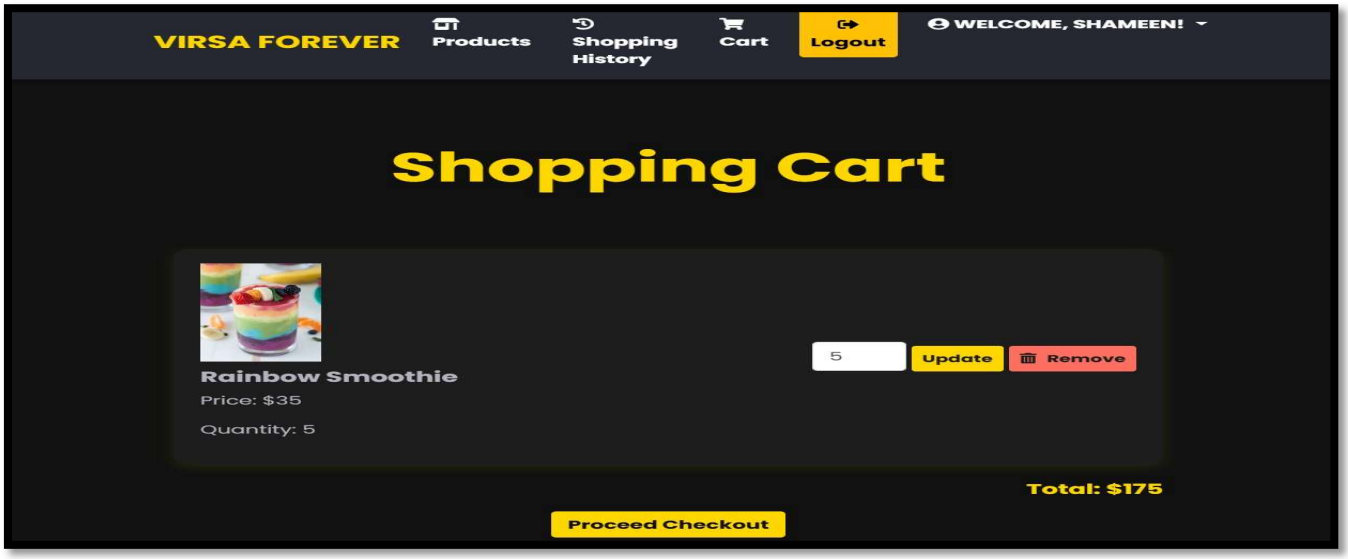


Figure 19 Customer buys numbers of products and products are stored

If another customer buys the same product and proceeds checkout the stock will be updated.

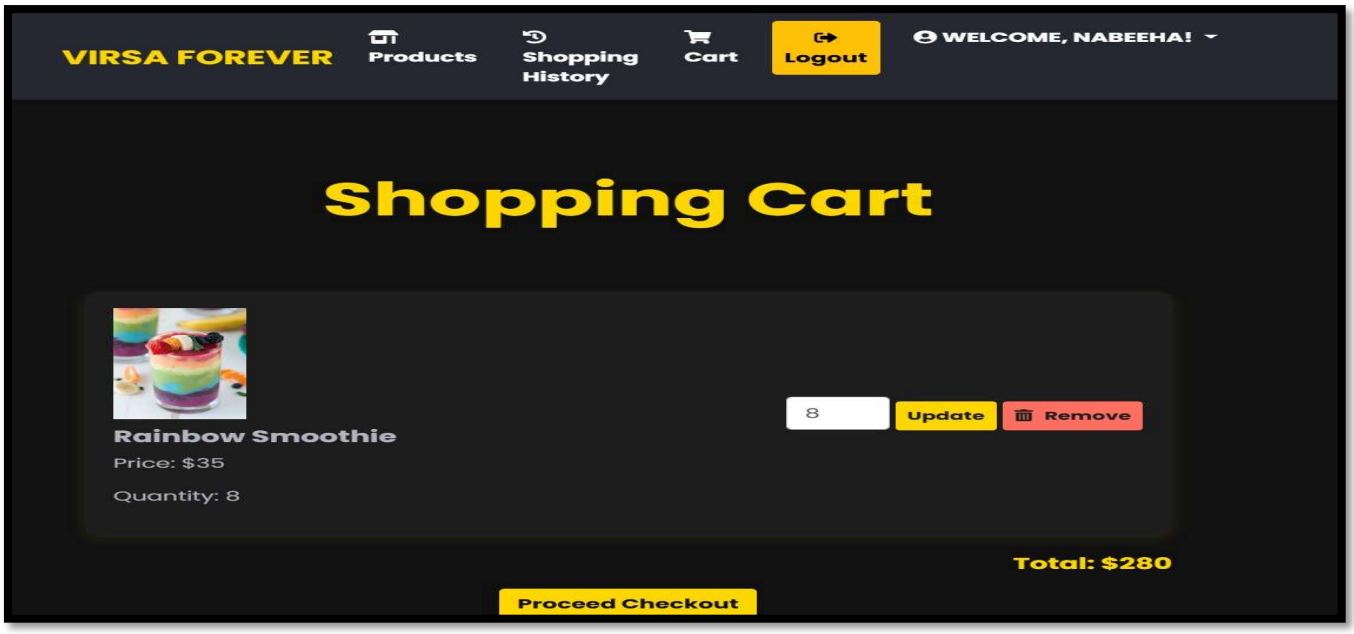


Figure 20 Another user buys same product and proceeds checkout

The saved cart of first user gets updated according to the remaining product stock.

Figure 21 The saved cart of fisrt customer gets updated

5) **PAYMENT METHODS: Users can purchase items by Payment methods and then they can Checkout finally.**

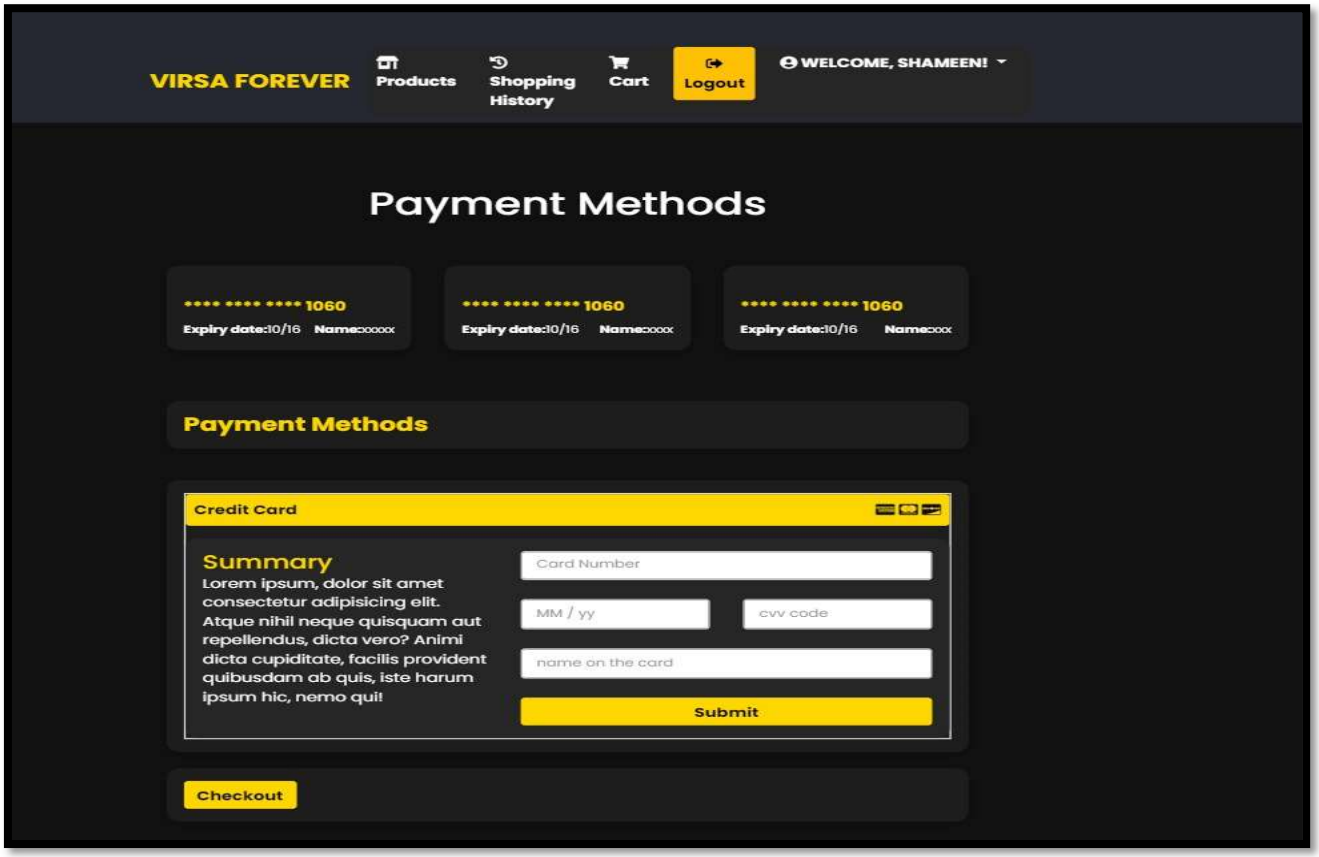


Figure 22 Payment Methods Page