

**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	Rida Noor Qasmi	CS-004
S2	Asra Siddiqui	CS-006
S3	Ashan Baig	CS-043

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

Table of Contents

Problem Description.....	3
Minimum Required Features:.....	3
Installation Guide.....	4
Distinguishing Features of our Project.....	4
1. Object-Oriented Programming Approached.....	4
2. Bootstrap for Frontend design (Web Interface).....	4
3. Jinja Templating with Flask.....	5
4. Flask for Backend.....	5
5. Admin Panel.....	5
6. Email Notifications.....	6
a. Order confirmation emails.....	6
b. Feedback emails.....	7
7. Custom Web Pages for Error Handling.....	7
8. Detailed User Profiles.....	8
9. User Authentication.....	9
10. Cart Management.....	9
Flow of our Project.....	11
Class Diagram.....	11
Flow of Program with Classes interaction.....	11
Challenging part of our Program.....	12
- Frontend and Backend Coordination.....	12
- Email Functionality.....	12
- Database Management with Txt Files.....	12
- State Management for the Shopping Cart.....	12
What did we learn while Working on this project?.....	13
Individual Contributions of each Group Member.....	14
- Member 1: Asra (Class User + Update Product).....	14
- Member 2: Ashan (Class Product + logic building).....	14
- Member 3: Rida (Class Route, App, & Mail).....	14
Future Expansions.....	15
List of References.....	15

Problem Description

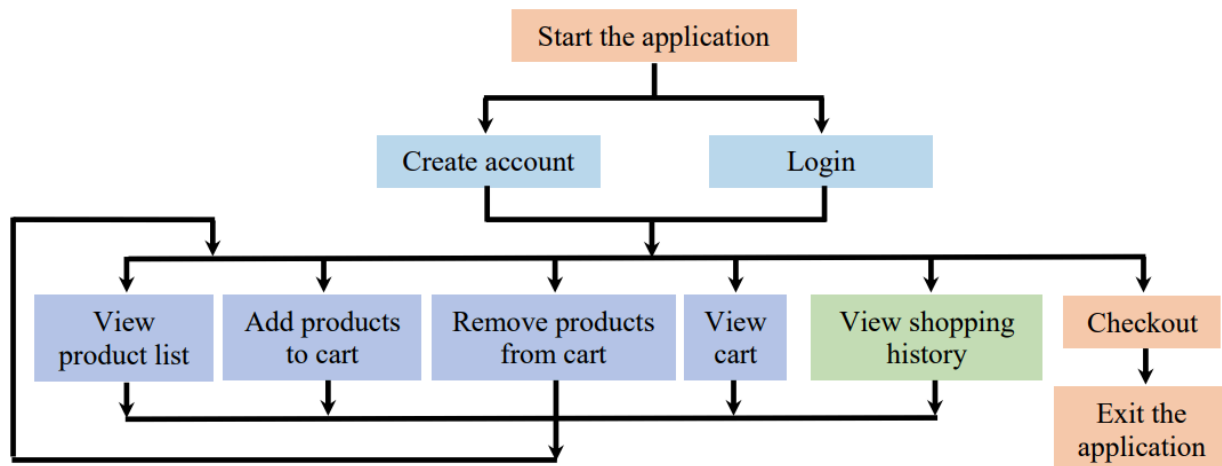
An online shopping cart is a virtual shopping trolley, where shoppers can put all of their want-to-buy products in, review to make adjustments in quantity, product attributes, etc., and remove it before or during the checkout if they change their mind. The application also allows users to view the history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

Minimum Required Features:

Your application should have the following features.

1. Store and display a product list with price and other relevant information. Use fixed products; there should be at least 10 products for the user to choose from.
2. Allow users to create an account with some basic information like username, password, first and last names, address, etc.
3. Allow the account holder to log in to the respective account and add to the cart as many products as needed from the available list of products.
4. When a user checks out, the items in the cart are transferred to his/her shopping history. For simplicity, the payment process is omitted.
5. Maintain shopping history for each user with date, items purchased, total bill, and other relevant information.
6. Each account holder should be able to
 - View the list of products and their relevant information from the virtual shopping shelves.
 - Add products to the cart.
 - Remove products from the cart.
 - View the cart with, product names, prices, quantities, total price, etc.
 - Checkout
 - View shopping history

The following flow diagram depicts the overall working of the application:



Installation Guide:

To run our program, please import the following prompt in your terminal:

- pip install flask
- pip install flask_mail

Make sure your internet connection is turned ON.

Use valid email IDs with @gmail.com.

You can't use '@' other than with @gmail.com and any special characters.

After installing these, copy the following link from your terminal to your browser:

```
http://127.0.0.1:5000
```

Distinguishing Features of our Project

1. Object-Oriented Programming Approached

Our online shopping cart application stands out due to its well-structured implementation of Object-Oriented Programming (OOP) principles, including encapsulation, inheritance, polymorphism, abstraction, composition, and association. We organized our code into classes like **'MyApp'**, **'User'**, and **'Product'**, which made everything easier to manage and more modular. We used inheritance and polymorphism to reuse code efficiently, like when the **'User'** inherits from the abstract class **'User_abstract'**.

Composition and association are evident as **'MyApp'** composes instances of **'User'**, **'Product'**, and **'Mail_service'**, while **'Routes'** associates with **'MyApp'** to define routes. This OOP approach made our code more organized, reusable, and easier to maintain and expand, which is essential for our online shopping application.

2. Bootstrap for Frontend design (Web Interface)

This is the most exciting feature of our program. We have implemented a bootstrapped web interface made of HTML and CSS. Bootstrap allowed us to design a professional-looking GUI with minimal effort, incorporating modern layouts, navigation bars, and responsive design elements. This made our online shopping site not only functional but also attractive and user-friendly. Overall, the bootstrapped interface pulled off the whole application with its clean and well-organized design that is also responsive on all devices if published.

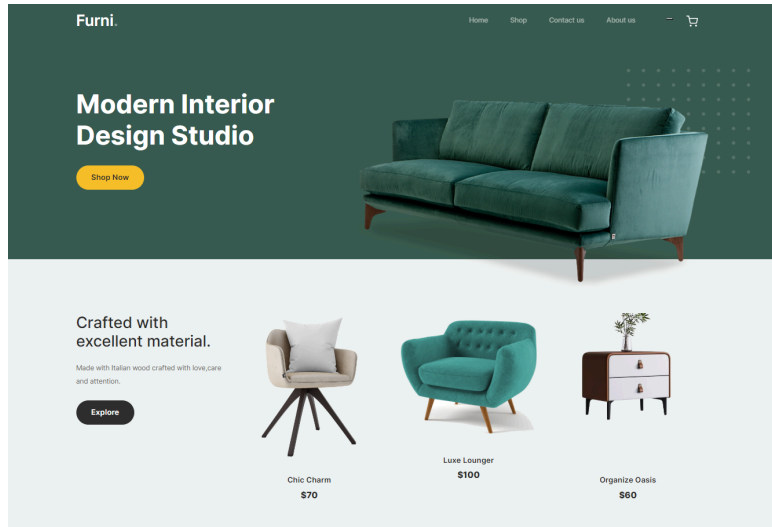


Fig 1.1 - The frontend interface

3. Jinja Templating with Flask

Integrating the Bootstrap site wasn't enough; we had to use Jinja Templating as it allows us to dynamically insert data into the web pages, and use loops and conditions to control what gets displayed. For example, we used Jinja to display the product details from our Python code directly onto our HTML pages. When a user adds a product to the cart, Jinja helps update the cart page with the new item and its details. This made our site more interactive and user-friendly. Using Jinja templating made managing and displaying the data easier, making our development process much simpler and more efficient.

4. Flask for Backend

Besides using Jinja templating to manage data between Python and HTML, we relied entirely on Flask (Python) for our website's backend. Flask helped us handle user interactions, process information from forms, and smoothly track user history. It made setting up different pages and functionalities—like user login, product management, and shopping cart—simple and efficient.

5. Admin Panel

The fifth distinguishing feature of our program is the admin panel. We have a dedicated page for admin operations where the admin can add more products along with their pictures (PNG file), quantity, and prices. Admin also has the leverage to remove shop items. Once done, the updated product will be displayed on the shop page.

Please login with the following credentials to use this feature;

- **Email:** admin@gmail.com
- **Password:** admin123
- Make sure the image file is in PNG mode.

Go to the highlighted option to add or remove your desired products:

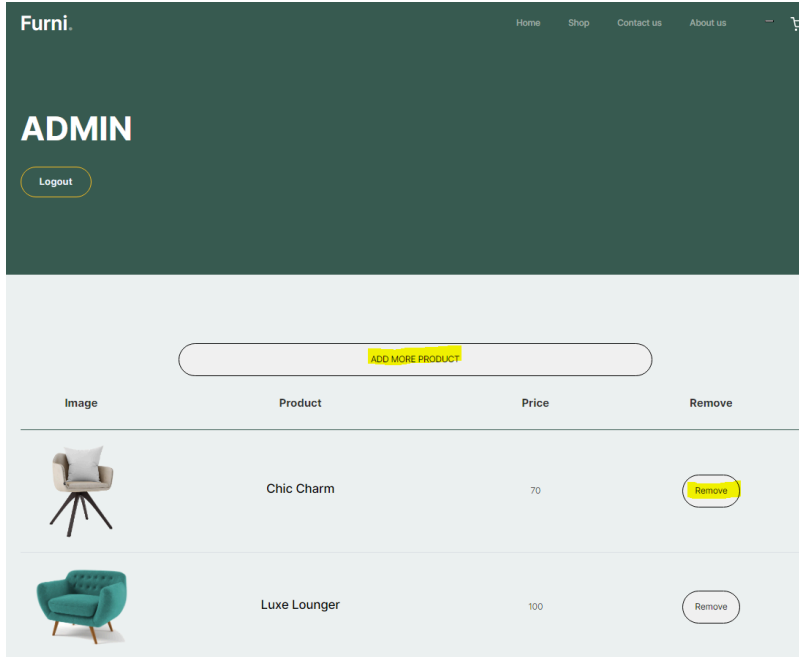


Fig 1.2 - Admin Panel

6. Email Notifications

Our program can also send emails upon order confirmations and feedback handling using the Flask Mail feature. The whole email-sending procedure is done through the class '**Mail Service**'.

a. Order confirmation emails

Upon completing an order, our application automatically generates and sends a detailed confirmation email to the customer that includes order details, total cost, etc. The **cart_mail()** method within '**Mail Service**' compiles this information into a structured email message and sends it to the customer's provided email address.

Note: Please put in your valid Email ID to receive the email.

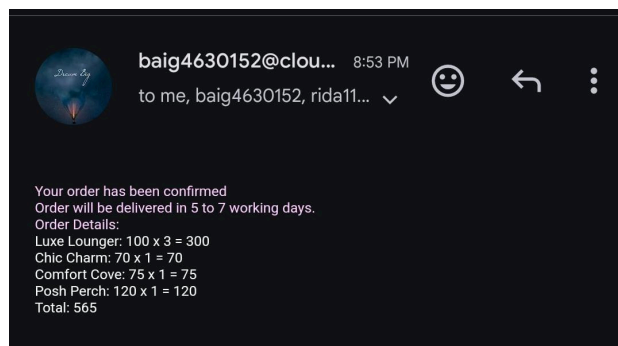


Fig 1.3 - Confirmation email

b. Feedback emails

Our program also enables users to submit feedback via a contact form on our website. When a user submits feedback, the **contact()** method in ‘**Mail Service**’ captures their message and sends an acknowledgment email confirming receipt.

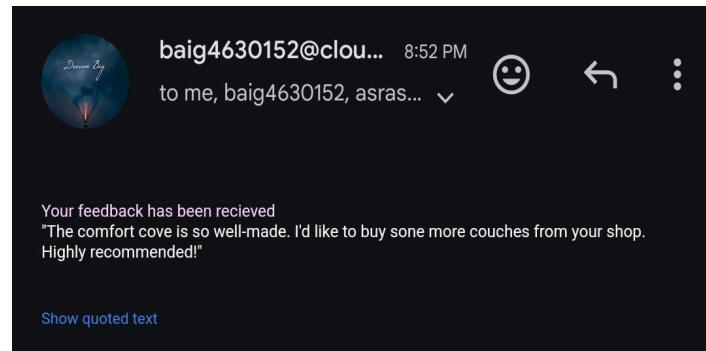


Fig 1.4 - Feedback email

7. Custom Web Pages for Error Handling

For error handling, we have custom web pages to display the error message. These pages are designed to display clear and user-friendly error messages whenever a user encounters a problem or an exception occurs.

Here's how errors have been handled;

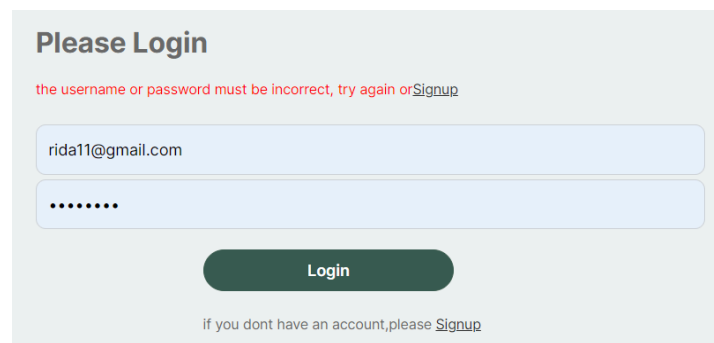


Fig 1.5 - Error handling (1)

Please Login

Please Login or [Signup](#) Before proceeding to checkout

rida11@gmail.com

Login

if you dont have an account,please [Signup](#)

Fig 1.6 - Error handling (2)

8. Detailed User Profiles

In our application, the user's detailed profile and purchase history are displayed on the **profile.html** page. Upon access, the application reads the user's information and purchase history from a text file associated with their email address. The user's purchase history includes dates, product indices, quantities, and total costs. These details are parsed and organized using Python within the 'User' class.

Here's how the user's data is displayed;




<div> <div>123</div> <div>rida11@gmail.com</div> <div>124567890</div> </div>				
Date of Purchasing: 2024-07-06 17:34:46				
Image	Product	Price	Quantity	Total cost
	Sleek Storage	100	700	70000
Grand Total:70000				
Date of Purchasing: 2024-07-06 20:53:24				
Image	Product	Price	Quantity	Total cost
	Luxe Lounger	100	3	300
	Chic Charm	70	1	

Fig 1.7 - User's history

To see your history, please refer to the following web page after purchasing;
You can also log out of your current account by clicking on the logout button.

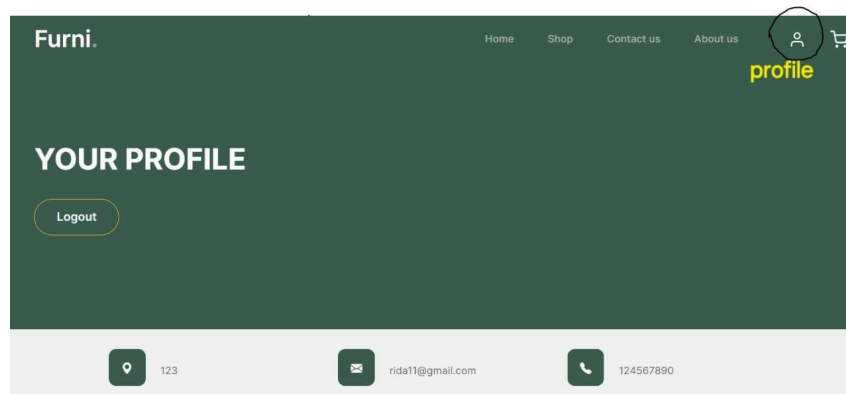


Fig 1.8 - Profile webpage

9. User Authentication

We are authenticating the user by checking the email ID and password. If the user already exists, our program will redirect the user to the login page, else the user will be asked to sign up first. In case of invalid inputs or errors, the user will be displayed with error messages written on the webpage.

Note:

- Your email ID should must end with '@gmail.com'.
- Don't use any special characters.

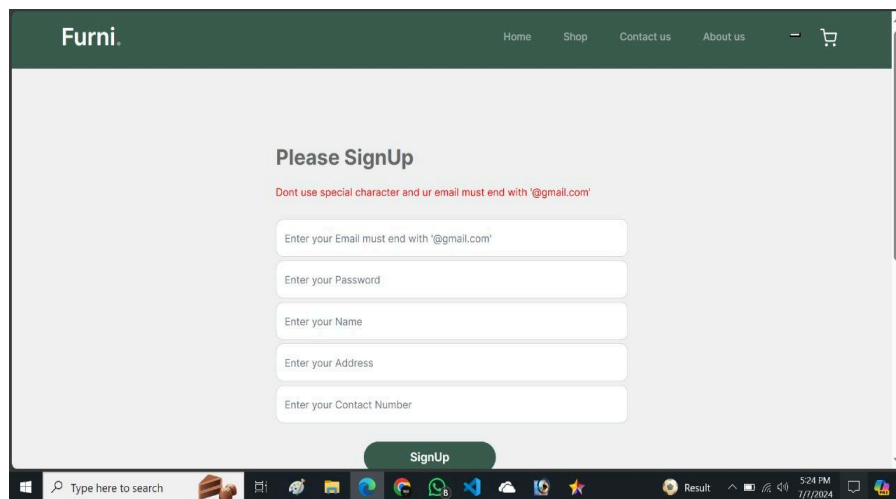


Fig 1.9 - Invalid input handling

10. Cart Management

This is another distinguishing feature of our program that has seamlessly been done through Flask (Python) and Jinja Templating to display on HTML pages.

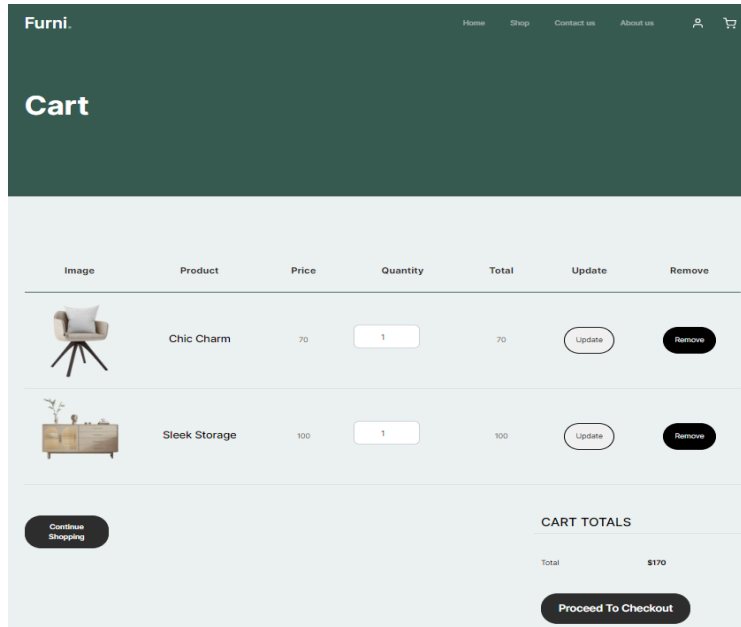


Fig 1.10 - Cart Management

The cart management in the provided code is handled by the Product class:

- Adding Products to Cart:

Products are added to the cart when users click on the '+' button.

- Viewing Cart:

The cart method displays the cart contents, calculating subtotal and grand totals based on the purchase.

- Removing Products:

Products can be removed while also updating subtotal and grand total accordingly.

- Updating Cart:

Quantity updates are managed by **update_cart**, adjusting totals based on new quantities. However, our product stock is unlimited, we're not managing inventory. Thus, you can add as many quantities as you want.

- Checkout:

The **check_out** method validates user login and presents the checkout page with cart details.

To see your cart and use its various functionalities, try these:

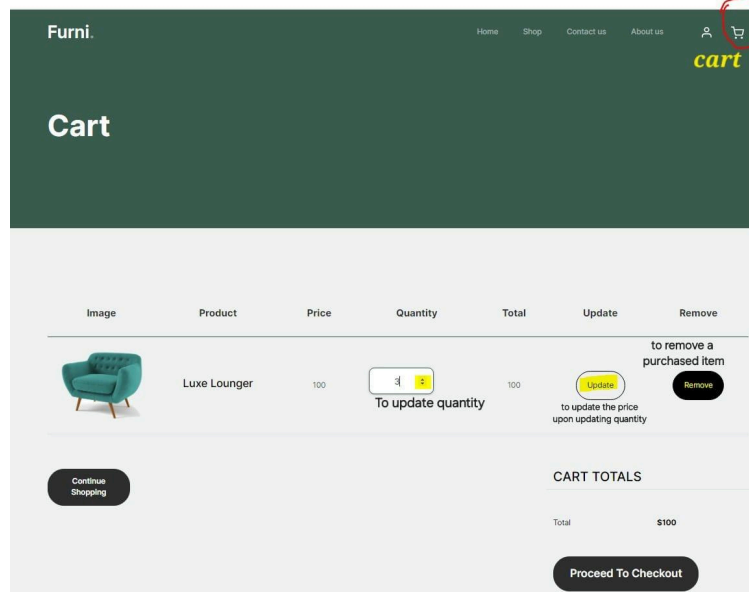


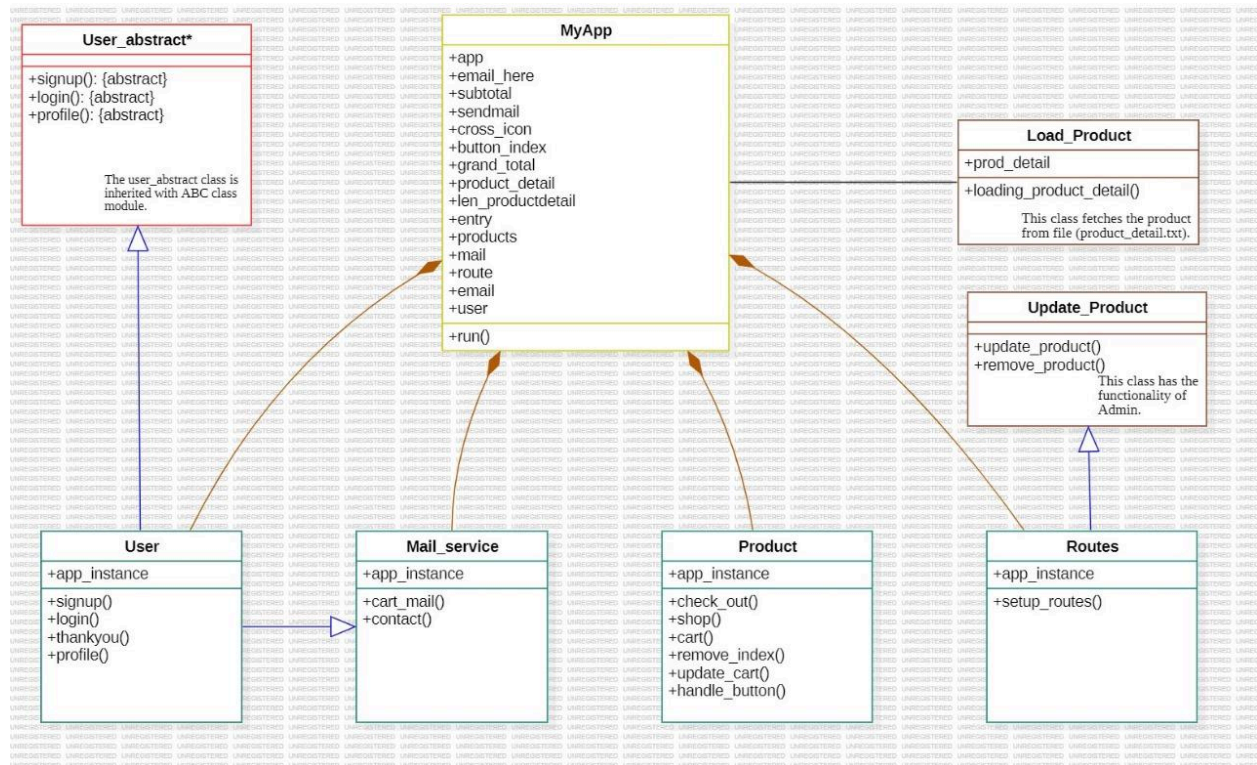
Fig 1.11 - Cart Webpage

Note:

- Do click the update button after updating quantities otherwise, prices won't get updated.
- To increase/decrease the quantity, either input the number or use the increment buttons on the right.

Flow of our Project

Class Diagram



Flow of Program with Classes interaction

a. Loading Products:

The '**Load_Product**' class is responsible for loading product details from a text file into a list, prod_detail, which includes information like product name, price, and initial quantity. This list is then used throughout the application.

b. Main Application Initialization:

'MyApp': This is the central class that initializes the Flask application and configures email settings. It creates instances of other key classes (User, Product, Mail_service, Routes) and sets up the main product list.

c. Setup Routing:

'Routes' define URL paths for different pages like home, shop, cart, checkout, user actions, and admin panel, linking them to appropriate view functions.

d. User Authentication and Profile:

'User' handles user signup, login, profile management, and order history. Users can log in, view profile details, and manage their accounts.

e. **Product Management:**

'Update_Product' is basically the admin panel that allows adding new products to the product_detail.txt file, ensuring the product list is up-to-date. The **'remove_product()'** function helps remove the item from the shop, giving leverage to the admin.

f. **Product Display and Cart Management:**

Users can browse products on the shop page, add items to their cart, and view the cart's contents. The **'Product'** class handles these actions and updates the cart's total dynamically based on user input.

g. **Checkout and Email Confirmation:**

During checkout, the Product class calculates the total cost and prompts the user to confirm their purchase. The **'Mail_service'** class sends a confirmation email with the order details to the user and other specified recipients.

Challenging part of our Program

Though developing the online shopping system was fun and a great learning experience, we also faced numerous challenges. A few of them are discussed below;

1. Frontend and Backend Coordination

It was a challenge to make the front end and back end work together smoothly. We had to make sure data flowed well between them, updating the cart in real time and keeping everything in sync.

2. Email Functionality

Adding email features for order confirmation and feedback was tough. We had to learn the Flask Mail feature and spent a lot of time implementing and troubleshooting the errors while integrating. We also needed to set up secure email settings and make sure emails were sent reliably so users always got their notifications.

3. Database Management with Txt Files

Unlike through designated databases, our program's data is efficiently managed with Txt files and it has its own difficulties. Managing read and write operations, ensuring data consistency, and handling concurrent access were significant challenges compared to using a more structured database system.

4. State Management for the Shopping Cart

This was undoubtedly the most challenging part of the program as managing the state of the shopping cart was very tricky, particularly in updating item quantities and handling removals without losing data.

Ensuring that the cart maintained its state across different user sessions and pages required careful implementation of session management and data handling logic from our side.

What did we learn while Working on this project?

Here are a few things we learned while building our online shopping cart;

a. OOP principles

We learned how to organize our code using classes and objects, which helped us manage different parts of our project more clearly.

b. Flask Framework

Working with Flask taught us how to build web applications efficiently, handling routing, request handling, and integrating various components like templates and databases.

c. Jinja Templating

Using Jinja templates in Flask enabled us to create dynamic HTML pages by embedding Python code. This allowed us to reuse templates and easily pass data from our backend to the frontend.

d. Bootstrap for Front-end development

Bootstrap made it simpler to design our website's appearance, ensuring it looked good and worked well on phones and computers alike.

e. Data Handling

We figured out how to store and use information effectively, like managing what users put in their shopping carts or keeping track of their orders.

f. Debugging and Problem Solving

We enhanced our skills in identifying and fixing errors by debugging, analyzing code flow, and applying solutions for smooth functionality.

g. Exception Handling

Implementing exception handling improved our application's robustness by efficiently managing errors so our website didn't crash unexpectedly, giving users a better experience overall.

h. Teamwork Cooperation

Working together as a team taught us how to communicate better, share tasks, and support each other's efforts to build a great responsive shopping system together.

Individual Contributions of each Group Member

The project is built on mutual decisions and each group member has effectively contributed his/her part.

- Member 1: Asra (Class User + Update Product)

Asra focused on user features and product management:

1. Login: Handled user authentication.
2. Sign Up: Managed user registration.
3. Showing History: Displayed transaction records.
4. Thank You Page: Designed a responsive confirmation page.
5. Add and Remove More Product (Admin Panel): Implemented product addition and removal for admins.

- Member 2: Ashan (Class Product + logic building)

Ashan enhanced the shopping experience and managed products:

1. Check-Out: Implemented secure checkout.
2. Shop: Developed interactive shopping pages.
3. Cart: Created a dynamic cart system.
4. Remove Product: Added functionality to remove items.
5. Add Quantity: Enabled quantity adjustments.
6. Update Button: Implemented real-time updates.
7. Logic building and error solving in most cases.
8. Filing: Integrated dynamic file handling method using Txt files for the data flow.

- Member 3: Rida (Class Route, App, & Mail)

Rida coordinated application flow and communication:

1. Route Handling: Defined and managed the flow of the application.
2. App: Established relationships between different classes.
3. Sending Mail: Implemented email functionalities for notifications and contact forms.
4. Designing UI: Contributed to UI design decisions, including theme integration and photo selection.
5. Project Report: Created the CEP report for project representation.

Future Expansions

The following expansions can be done in the future:

- Database Integration:

For the latter, we can upgrade to a relational database (e.g., SQLite, MongoDB) for storing user details, product data, and transactions.

- User Authentication and Security:

Additionally, we can integrate security tools with bcrypt or passlib for password hashing and introduce email-based password reset for user convenience and safety.

- Order Management:

We can implement another system to track orders, manage status updates (e.g., pending, confirmed, shipped), and send detailed order confirmations via email.

- Admin Panel Enhancements:

We can later add more administrative tools for inventory management, order processing, and user account management for improved control.

- Advanced Search and Filtering:

Implementing search and filtering options to help users easily find products based on various criteria.

List of References

The following learning sources helped us build this website interface of our line shopping cart system:

- Apna College [YouTube Channel]: <https://youtube.com/@apnacollege>
- Code with Harry [YouTube Channel]: <https://youtube.com/@codewithharry>

