

E-COMMERCE APP DEVELOPMENT

PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION

In this phase we are going to document the final look of our e-commerce app. This includes the outline of the e-commerce app's objective, design thinking, development phases (1 and 2), the platforms layout, features we have added till now and technical implementation details.

As a part of this documentation, we are adding details about the updated frontend and backend.

As instructed, we have used html and CSS for frontend, flask for backend and SQLite for database purposes.

PROBLEM AND OBJECTIVE:

Crafts have been an integral part of daily life in villages, towns, courts, and religious establishments. There are approximately 70 lakh handicraft artisans in the country. As a largely unorganized sector, handicrafts face problems such as a paucity of professional infrastructure such as work sheds, storage space, shipping, and packing facilities. Due to their low

education, artisans often cannot identify potential new markets for their products, nor do they understand the requirements for interacting with these markets.

This project is to build an artisanal e-commerce platform. The goal is to connect skilled artisans with global audience, showcasing their handmade products and providing features like secure shopping carts, payment gateways and an intuitive checkout process. This involves designing of e-commerce platform, implementing necessary features, and ensuring a seamless user experience.

The foremost objective is to provide local artisans with an easy-to-use online platform wherein they can register and sell/promote their products.

PROJECT OVERVIEW

PROJECT NAME:

ARHA - CONNECTING ARTISANS

PROJECT DESCRIPTION:

ARHA - CONNECTING ARTISANS is an e-commerce platform that connects artisans and customers. It provides artisans with a platform to showcase and sell their handcrafted products while allowing customers to discover unique and authentic artisanal items.

DESIGN THINKING:

♣ Platform layout:

The layout consists of product categories, individual product pages, shopping cart, payment, and a smooth checkout process.

- ♠ The homepage has products, product categories, sign in, log in, shopping cart etc....
- ♠ The product categories page has various categories of handmade products. For example, it has categories such as bamboo works, rattan works, pottery, etc....
- ♠ Individual product pages are the detailed pages about a particular product which includes description, pictures, price, seller info and many more.
- ♠ Shopping cart has all the products that have been added by the customer. It has the image, name, product details, quantity, total price and buying options. The user can also delete a product from the cart.
- ♠ Checkout processes begin when customers place an order in the online store and proceed to payment.

- ♠ Payment can be made after the customer has ordered the product. The user has the option to pay with card.

♣ Product Showcase:

We have database tables for registration, user, customer order, categories, products, brands. We have used SQLite3 for database.

- ♠ The registration table has details such as user name, email, password, country, city, contact, address, zip code, profile, account creation date.
- ♠ The customer order table has invoice number, status of the order, customer id., date ordered, other orders made by the customer.
- ♠ The add products table is made for sellers. This has the name, price, discount, stock, colors, description, category, and images of the products.
- ♠ There are separate tables for brands and category too.

♣ User Authentication:

Flask_login is a flask extension that enables user authentication. Flask_login uses cookie-based authentication. When a client login via his credentials, Flask creates a session containing a user

id and then sends the session id to the user via a cookie, using which he/she can log in and out as and when required.

♣ Shopping cart:

The shopping cart enables the users to add products to the cart, update quantities and remove items, calculate the total cost, proceed to checkout.

♣ Checkout process:

The checkout process helps the user to place the order, review order details, get the pdf of the invoice and enables the user to pay with a card.

♣ Payment Integration:

To facilitate transactions, the app integrates with secure payment gateway stripe.

♣ User Experience:

HTML, CSS has been used to create a visually appealing user interface. It offers a user-friendly navigation and search features.

LOGO OF THE APP:



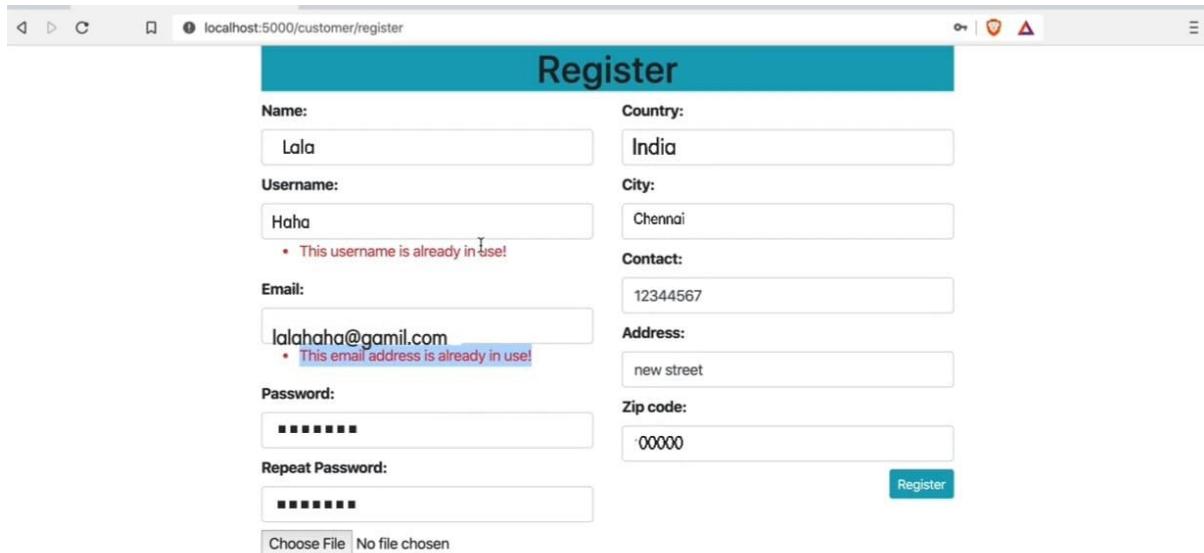
FRONTEND:

The Front end of our app has been designed using html and CSS. The various pages available in our app are showcased below:

As the app gets opened the user will be prompted to register. This is basic process as all apps do in the beginning. The registration page of our app is:

A screenshot of a web browser showing the registration page at localhost:5000/customer/register. The page has a teal header bar with the word "Register". Below the header are two columns of form fields. The left column contains: "Name:" with an input field, "Username:" with an input field, "Email:" with an input field, "Password:" with an input field containing a single character 'I', and "Repeat Password:" with an input field. The right column contains: "Country:" with an input field, "City:" with an input field, "Contact:" with an input field, "Address:" with an input field, and "Zip code:" with an input field. At the bottom left is a "Choose File" button with "No file chosen" next to it. At the bottom right is a teal "Register" button.

If the customer uses an email or username which is already taken then he/she will be warned of that error.

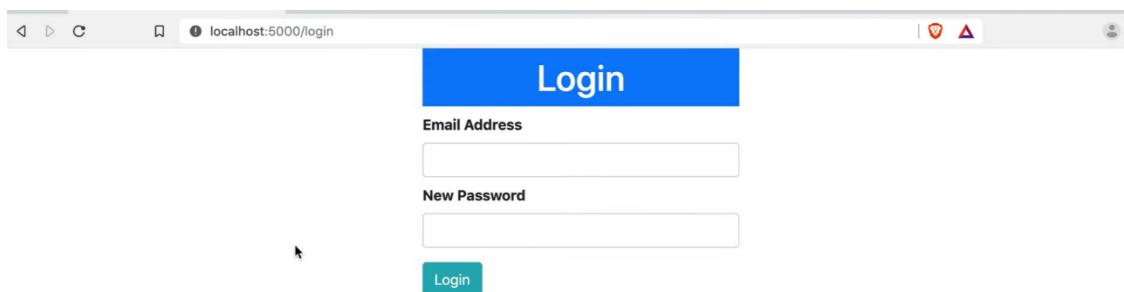


A screenshot of a web browser window titled "localhost:5000/customer/register". The page has a teal header bar with the word "Register". Below it are several input fields:

- Name: Lala
- Country: India
- Username: Haha
- City: Chennai
- Email: lalahaha@gmail.com (with a red error message: "This email address is already in use!")
- Contact: 12344567
- Address: new street
- Zip code: 00000
- Password: (redacted)
- Repeat Password: (redacted)

At the bottom left is a "Choose File" button with the text "No file chosen". On the right side, there is a teal "Register" button.

Once the user has registered in our app, he/she will be prompted to login. The login page is:

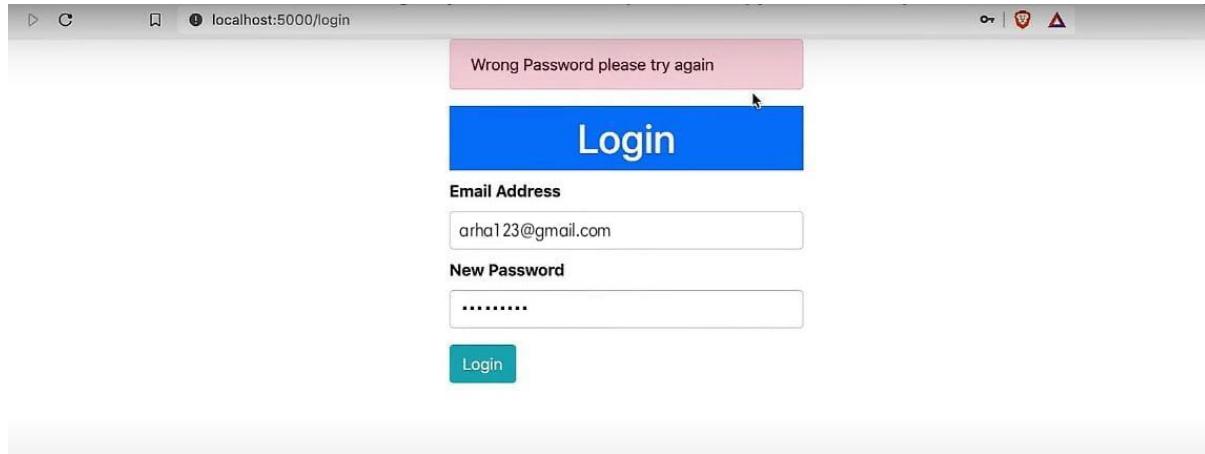


A screenshot of a web browser window titled "localhost:5000/login". The page has a blue header bar with the word "Login". Below it are two input fields:

- Email Address: (empty)
- New Password: (empty)

At the bottom center is a blue "Login" button.

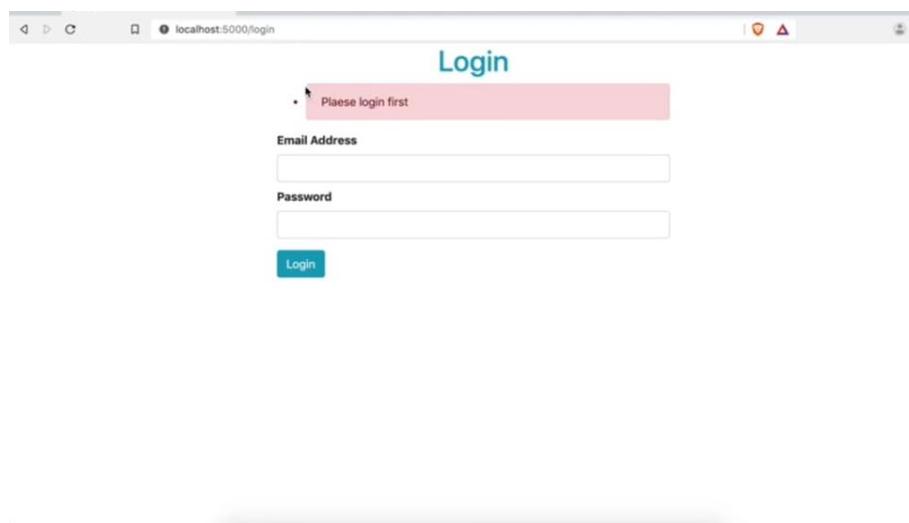
If the user logs in properly then the user will be redirected to the home page else the user will be shown an error.



A screenshot of a web browser window titled "localhost:5000/login". The page features a blue "Login" button at the top. Below it are two input fields: "Email Address" containing "arha123@gmail.com" and "New Password" containing ".....". A pink error message box is displayed above the "Login" button, stating "Wrong Password please try again".

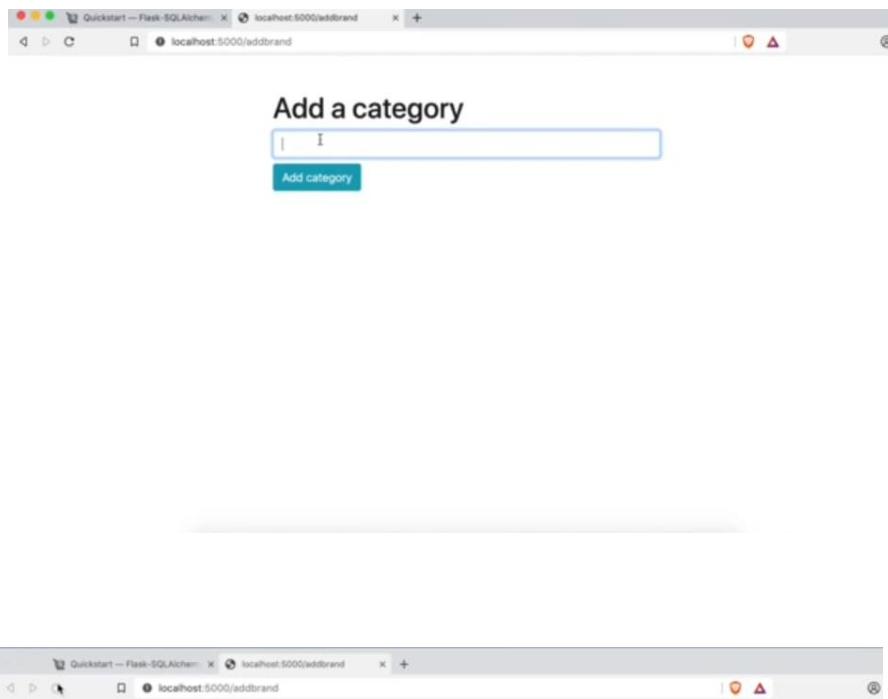
Once the user has completed a successful login the user will be redirected to the home page.

If the user is a seller/Merchant then he/she has to login as an admin. This enables the seller to add brands, categories, products to the platform.

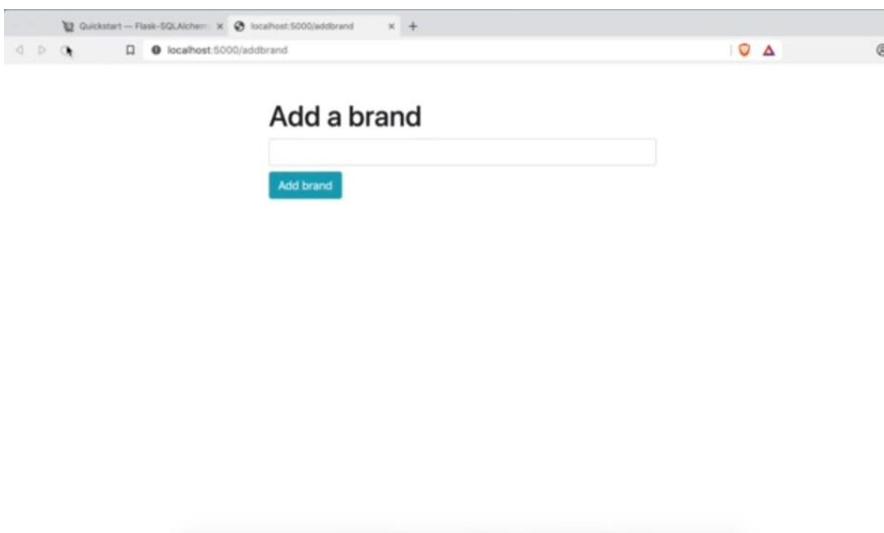


A screenshot of a web browser window titled "localhost:5000/login". The page features a teal "Login" button at the top. Below it are two input fields: "Email Address" and "Password". A pink success message box is displayed above the "Login" button, stating "Please login first".

Once the seller has logged in, he can add brands, categories, products.



The screenshot shows a web browser window with the title "Quickstart — Flask-SQLAlchemy" and the URL "localhost:5000/addbrand". The page has a light gray header with the text "Add a category". Below the header is a single input field with a placeholder "I" and a blue "Add category" button below it. The browser interface includes standard window controls (minimize, maximize, close) and a toolbar with icons for back, forward, and search.



The screenshot shows a web browser window with the title "Quickstart — Flask-SQLAlchemy" and the URL "localhost:5000/addbrand". The page has a light gray header with the text "Add a brand". Below the header is a single input field and a blue "Add brand" button below it. The browser interface includes standard window controls and a toolbar with icons for back, forward, and search.

Once the seller adds an item to the brands/categories it will be added to database.

The seller can add the products by just filling up the add products form.

A screenshot of a web browser window titled "localhost:5000/addproduct". The page has a teal header bar with the title "Add a Product". Below the header are several input fields:

- name**: A text input field containing "Add product name".
- Price**: A text input field containing "Add product price".
- Discount**: A text input field containing "Add product discount".
- Stock**: A text input field containing "Add product stock".
- Add a brand**: A dropdown menu currently set to "Select A brand".
- Add a Category**: A dropdown menu currently set to "Add a category".
- Colors**: A text input field containing "Add product colors".
- Description**: A large text area containing "Add product detail".

A screenshot of a web browser window titled "localhost:5000/addproduct". The page has a teal header bar with the title "Add a Product". Below the header are several input fields:

- Stock**: A text input field containing "0".
- Colors**: A text input field containing "Add product colors".
- Description**: A large text area containing "Add product detail".
- Image 1**, **Image 2**, **Image 3**: Three separate input fields for file uploads, each showing "Choose file No fi...osen".

At the bottom center is a green "Add Product" button.

The seller can also edit/delete the product, brands, categories added by him/her. All these activities can be carried out in the admin page itself.

A screenshot of a web browser window titled "localhost:5000/admin". The page displays a table of products with columns: Sr, Product, Price, Discount, Brand, Image, Edit, and Delete. There are three rows of data:

Sr	Product	Price	Discount	Brand	Image	Edit	Delete
1	Handmade cot	1200.00	20 %			<button>Edit</button>	<button>Delete</button>
2	earthern pot	800.05	15 %			<button>Edit</button>	<button>Delete</button>
3	Jute bag	1099.99	0 %			<button>Edit</button>	<button>Delete</button>

This page shows brands maintained by admin. These brands can be updated by just pressing the edit button.

A screenshot of a web browser window titled "localhost:5000/updatebrand/3". The page has a heading "Add a brand" and a text input field. Below the input field are two buttons: "Update brand" (blue) and "Cancel" (yellow).

The categories page maintained by the admin(seller) is shown below:

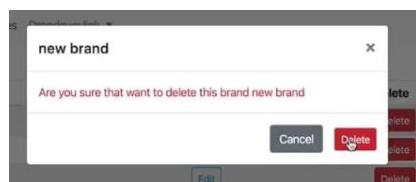
Sr	Name	Edit	Delete
1	Bamboo	<button>Edit</button>	<button>Delete</button>
2	Rattan	<button>Edit</button>	<button>Delete</button>
3	Jute	<button>Edit</button>	<button>Delete</button>
4	Furniture	<button>Edit</button>	<button>Delete</button>

The admin can upgrade a category by just pressing edit button.

Add a category

Update category Cancel

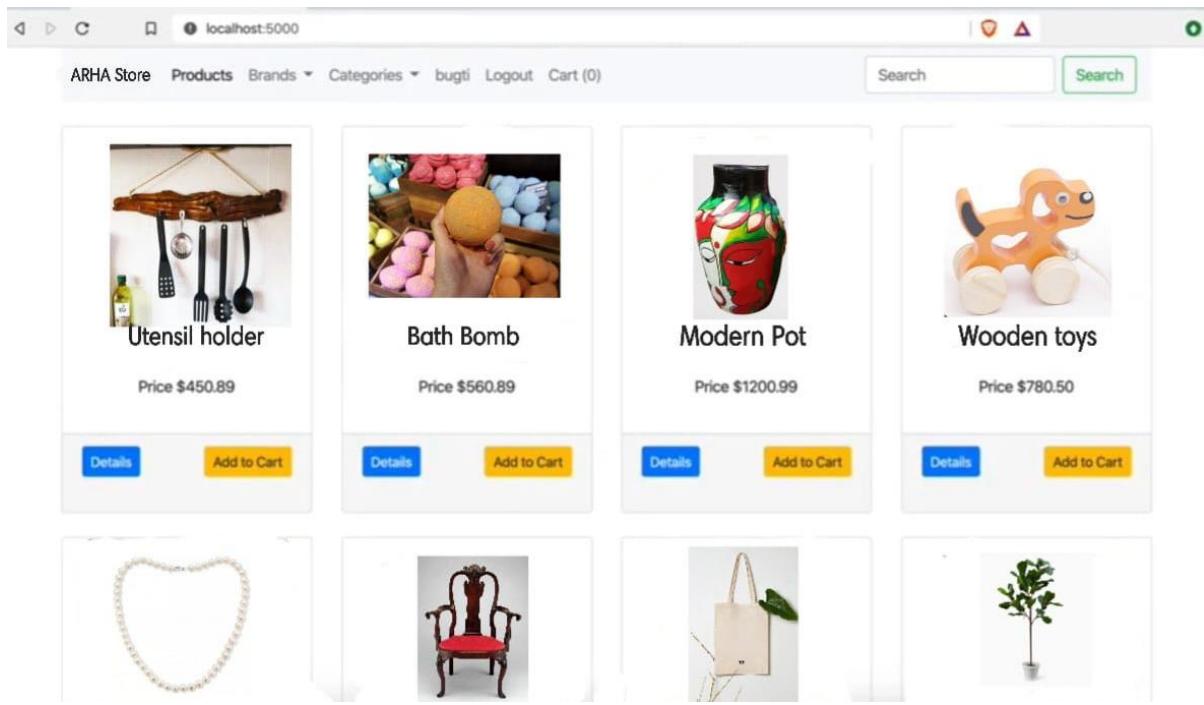
If the admin wishes to delete a category or brand, he/she can use the delete button in the respective pages.



These are user end features that a seller can make use of for adding, editing, or deleting a product/brand/category.

Next, let us see the user features such as home page, products page, individual products page, shopping cart, payment page, checkout page.

The Home page of the app has navigation bar to product, categories, and so on. The home page looks like this:



The individual product page has add to cart option and a summarized description about the product.

Product name: Modern Pot

Product price: \$450.89

Discount: 10 %

Product description

The Small Of Soil are extremely modern, and the novelty, it will enhance the beauty of your house. Both of them are colorful, unique and elegant. It is made by experienced artisans. NOTE as this product is hand-crafted there might be a slight variation in shape or design which is normal. Hence make the product unique. These Terracotta Vases are ideal to be used as decorative dÃ©cor pieces or flower. Cannot be used to put flowers with water due to the hand painting.

Add to Cart Quantity: 1 Colors: Blue

The user can add items to the cart by pressing add to cart button and can choose the colors, quantity.

The shopping cart page looks like this:

Sr	Image	Name	Color	Price	Quantity	Discount	Subtotal	Update	Delete
1		Modern pot	Colors: Golden	\$1200.99	1		\$1200.99	<button>Update</button>	<button>Delete</button>
2		Wodden toy	Colors: Blue	\$450.89	2	10 % is 45.09	\$856.78	<button>Update</button>	<button>Delete</button>

Order now Tax: \$123.46 Grand total: \$2181.14 Clear cart

When the user touches the order now button, the user proceeds to the payment and checkout process.

The orders page will be like this:

The screenshot shows a web browser window with the URL `localhost:5000/orders/4d59e3624c`. The page title is "ARHA Store". The navigation bar includes links for "Products", "Brands", "Categories", "jamal", "Logout", and "Cart (0)". There is a search bar with a "Search" button. A green success message box displays "Your order has been sent successfully". Below it, order details are listed: Inoice: 4d59e3624c, Status: Pending, Customer name: jamal, Customer email: admin@site.com, and Customer contact: 2345325. A table shows the order items:

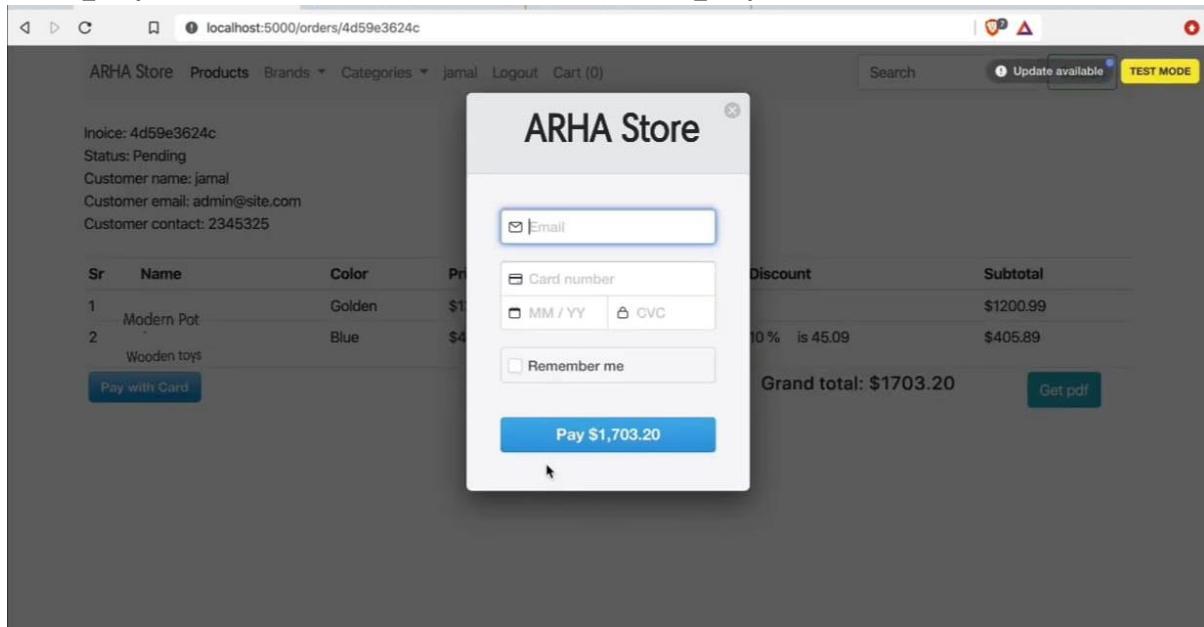
Sr	Name	Color	Price	Quantity	Discount	Subtotal
1	Modern Pot	Golden	\$1200.99	1		\$1200.99
2	Wooden toy	Blue	\$450.89	1	10 % is 45.09	\$405.89

Total Tax: \$96.41 Grand total: \$1703.20

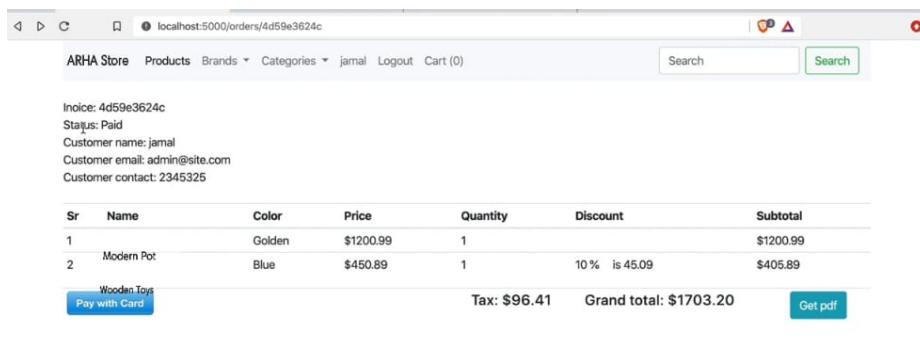
Buttons at the bottom include "Pay with Card" and "Get pdf".

The user has placed an order but still the status is pending because the user has not yet paid. To proceed

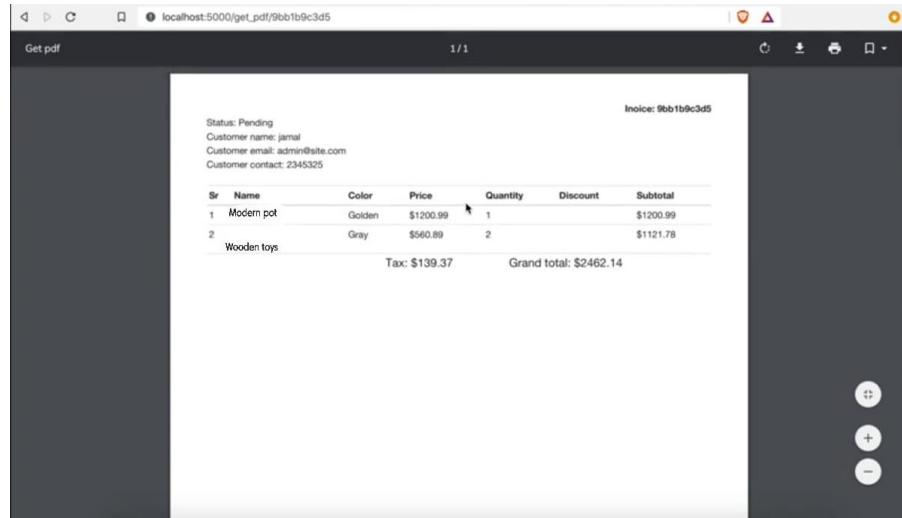
for payment the user must click pay with card.



Then the user must fill the details to continue the payment. After payment the status is paid.



The user can get the pdf of the invoice generated by clicking get pdf.



Thus, the app comes out with a smooth payment and checkout process.

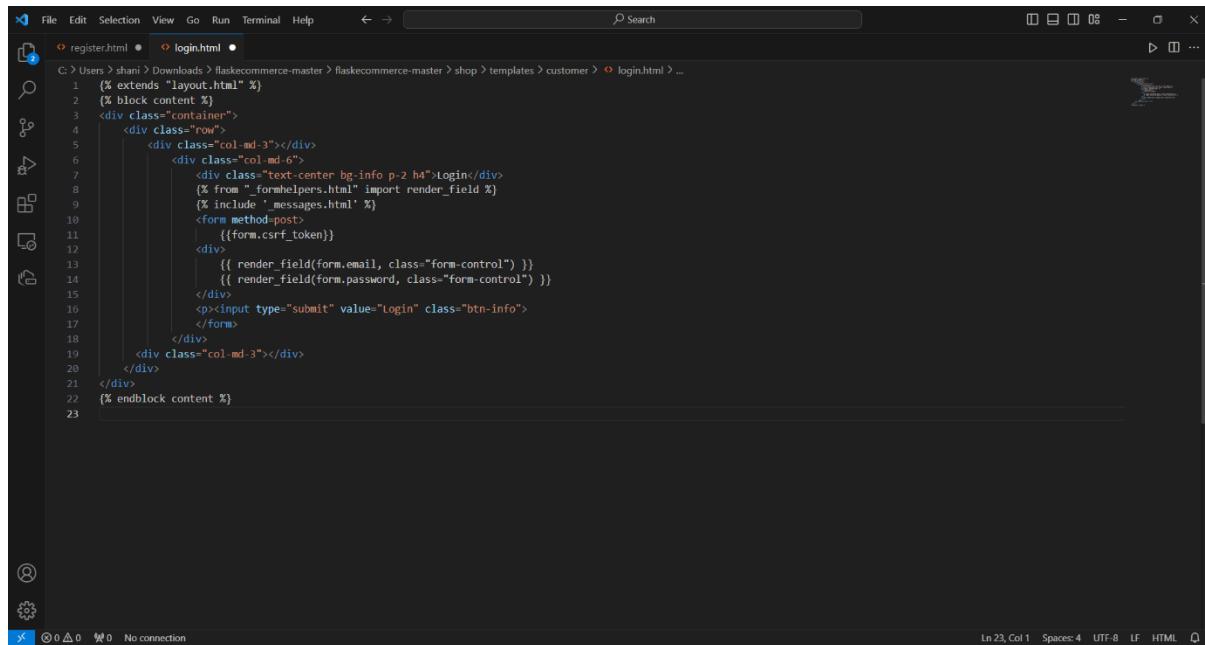
The frontend coding using html and CSS are shown below:

Register.html

```
C:\> Users > shani > Downloads > flaskcommerce-master > flaskcommerce-master > shop > templates > customer > register.html > div.container
1  {% extends "layout.html" %} 
2  {% block content %} 
3  {% from "_formhelpers.html" import render_field %} 
4  {% endblock %} 
5  <div class="container">
6      <div class="row">
7          <div class="col-md-2"></div>
8          <div class="col-md-8">
9              <h1 class="bg-info text-center">Register</h1>
10             </div>
11             <div class="col-md-2"></div>
12         </div>
13         <form action="" method="post">
14             {{ form.csrf_token }}
15             <div class="row">
16                 <div class="col-md-2"></div>
17                 <div class="col-md-4">
18                     {{ render_field(form.name, class="form-control")}}
19                     {{ render_field(form.username, class="form-control")}}
20                     {{ render_field(form.email, class="form-control")}}
21                     {{ render_field(form.password, class="form-control")}}
22                     {{ render_field(form.confirm, class="form-control")}}
23                     {{ form.profile() }}
24                 </div>
25                 <div class="col-md-4">
26                     {{ render_field(form.country, class="form-control")}}
27                     {{ render_field(form.city, class="form-control")}}
28                     {{ render_field(form.contact, class="form-control")}}
29                     {{ render_field(form.address, class="form-control")}}
30                     {{ render_field(form.zipcode, class="form-control")}}
31                     {{ form.submit(class="btn btn-sm btn-info float-right") }}
32                 </div>
33                 <div class="col-md-2"></div>
34             </div>
35         </form>
36     </div>
37     {% endblock content %}
```

CSRF tokens are used to validate the user's registration requests. This prevents the same mail id or user name to be registered again.

Login.html



The screenshot shows a code editor window with the file 'login.html' open. The code is a template for a login page, extending from 'layout.html'. It includes a form for email and password input, and a CSRF token field. The code is written in Jinja2 syntax.

```
C:\> Users > shani > Downloads > flaskcommerce-master > flaskcommerce-master > shop > templates > customer > login.html > ...
1  {% extends "layout.html" %} 
2  {% block content %} 
3  <div class="container">
4      <div class="row">
5          <div class="col-md-3"></div>
6          <div class="col-md-6">
7              <div class="text-center bg-info p-2 h4">Login</div>
8              {% from "formhelpers.html" import render_field %}
9              {% include '_messages.html' %}
10             <form method="post">
11                 {{form.csrf_token}}
12                 <div>
13                     {{ render_field(form.email, class="form-control") }}
14                     {{ render_field(form.password, class="form-control") }}
15                 </div>
16                 <p><input type="submit" value="Login" class="btn-info"></p>
17             </form>
18         </div>
19     </div>
20 </div>
21 {% endblock content %}
```

The login page is also designed to use CSRF token. This prevents the user from logging in using an unregistered mail or a wrong password.

ADMIN:

Registration.html

The screenshot shows a code editor with two tabs open: 'register.html' and 'login.html'. Both files are located in the 'templates/admin' directory. The code uses Bootstrap's grid system and Jinja2 templating syntax to create registration and login forms. It includes CSRF tokens and form validation.

```
C:\> Users > shani > Downloads > flaskcommerce-master > flaskcommerce-master > templates > admin > register.html > ...
1  {% extends "layout.html" %}
2  {% block content %}
3  <div class="container">
4      <div class="row">
5          <div class="col-md-6">
6              <div class="text-center bg-info p-2 h4">Register User</div>
7              {% from "_formhelpers.html" import render_field %}
8              <form method="post">
9                  {{ form.csrf_token }}
10                 {{ render_field(form.name, class="form-control") }}
11                 {{ render_field(form.username, class="form-control") }}
12                 {{ render_field(form.email, class="form-control") }}
13                 {{ render_field(form.password, class="form-control") }}
14                 {{ render_field(form.confirm, class="form-control") }}
15
16
17             </div>
18             <p><input type="submit" value="Register" class="btn-info"></p>
19         </form>
20     </div>
21  </div>
22 </div>
23 </div>
24 </div>
25 {% endblock content %}
```

Ln 25, Col 23 Spaces: 4 UTF-8 LF HTML

Login.html

The screenshot shows a code editor with the 'login.html' file open. This file is also located in the 'templates/admin' directory. It follows a similar structure to the registration page, using Bootstrap and Jinja2 to handle user login.

```
C:\> Users > shani > Downloads > flaskcommerce-master > flaskcommerce-master > templates > admin > login.html > ...
1  {% extends "layout.html" %}
2  {% block content %}
3  <div class="container">
4      <div class="row">
5          <div class="col-md-3">
6              <div class="col-md-6">
7                  <div class="text-center bg-info p-2 h4">Login</div>
8                  {% from "_formhelpers.html" import render_field %}
9                  {% include '_messages.html' %}
10                 <form method="post">
11                     {{form.csrf_token}}
12                     <div>
13                         {{ render_field(form.email, class="form-control") }}
14                         {{ render_field(form.password, class="form-control") }}]
15                     </div>
16                     <p><input type="submit" value="Login" class="btn-info"></p>
17                 </form>
18             </div>
19             <div class="col-md-3"></div>
20         </div>
21     </div>
22 {% endblock content %}
```

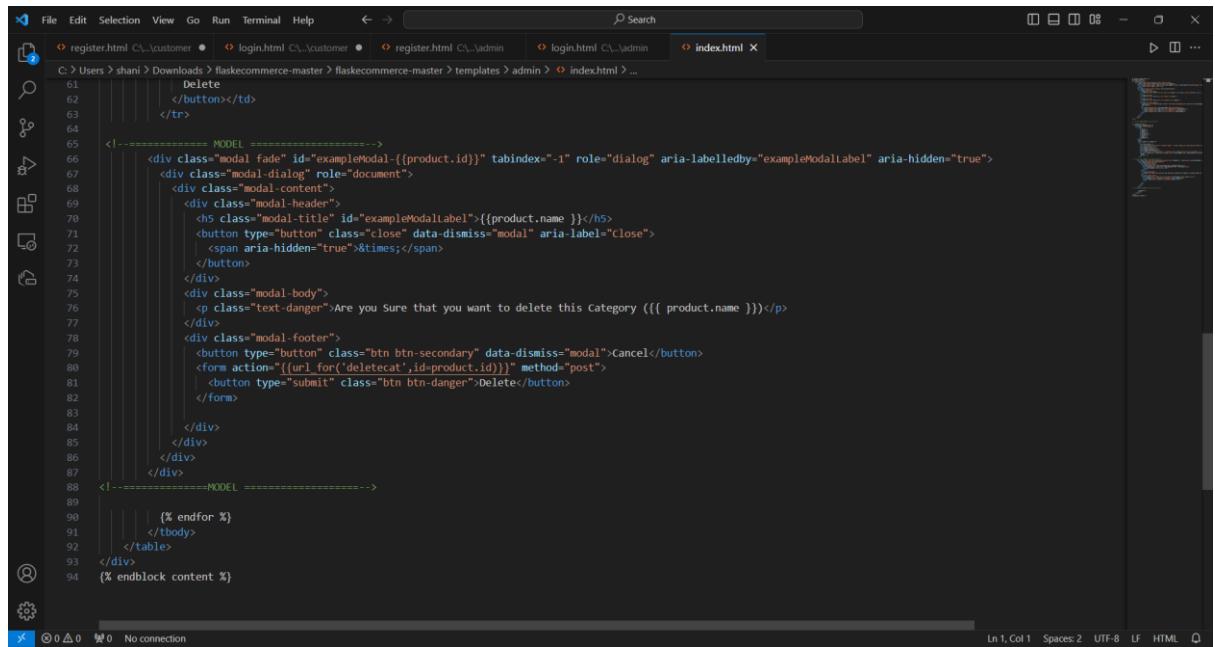
Ln 14, Col 76 Spaces: 4 UTF-8 LF HTML

The registration and login page of admin also uses the CSRF tokens to avoid CSRF attacks as the app uses cookies to store the user details.

Index.html

```
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > admin > index.html ...
1  [% extends 'layout.html' %]
2  [% block content %]
3  <div class="container">
4      <nav class="navbar navbar-expand-lg navbar-light bg-light">
5          <a class="navbar-brand" href="{{url_for('admin')}}">Navbar</a>
6          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-
7              <span class="navbar-toggler-icon"></span>
8          </button>
9          <div class="collapse navbar-collapse" id="navbarNavDropdown">
10             <ul class="navbar-nav">
11                 <li class="nav-item active">
12                     <a class="nav-link" href="{{url_for('admin')}}">product <span class="sr-only">(current)</span></a>
13                 </li>
14                 <li class="nav-item">
15                     <a class="nav-link" href="{{url_for('brands')}}">brand</a>
16                 </li>
17                 <li class="nav-item">
18                     <a class="nav-link" href="{{url_for('categories')}}">category</a>
19                 </li>
20                 <li class="nav-item dropdown">
21                     <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
22                         Add products
23                     </a>
24                     <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
25                         <a class="dropdown-item" href="{{url_for('addproduct')}}">Add product</a>
26                         <a class="dropdown-item" href="{{url_for('addbrand')}}">Add brand</a>
27                         <a class="dropdown-item" href="{{url_for('addcat')}}">Add category</a>
28                     </div>
29                 </li>
30             </ul>
31         </div>
32     </div>
33 </div>
34 <!-- END NAVBAR -->
35
36
37 <div class="container">
38     [% include '_messages.html' %]
39     <table class="table table-sm">
40         <thead>
41             <th>Sr</th>
42             <th>Image</th>
43             <th>Name</th>
44             <th>Price</th>
45             <th>Discount</th>
46             <th>Brand</th>
47             <th>Edit</th>
48             <th>Delete</th>
49         </thead>
50         <tbody>
51             [% for product in products %]
52                 <tr>
53                     <td>{{loop.index}}</td>
54                     <td></td>
55                     <td>{{product.name}}</td>
56                     <td>{{product.price}}</td>
57                     <td>{{product.discount}}</td>
58                     <td>{{product.brand.name}}</td>
59                     <td><a href="{{url_for('updateproduct', id=product.id)}}>Edit </a></td>
60                     <td><button type="button" class="btn btn-danger btn-sm" data-toggle="modal" data-target="#exampleModal-{{product.id}}">
61                         Delete
62                     </button></td>
63                 </tr>
64             [%-- MODEL --%]
65             <div class="modal fade" id="exampleModal-{{product.id}}" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
66                 <div class="modal-dialog" role="document">
67                     <div class="modal-content">
68                         <div class="modal-header">
69                             <h5 class="modal-title" id="exampleModalLabel">{{product.name}}</h5>
```

```
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > admin > index.html ...
34
35 <!-- END NAVBAR -->
36
37 <div class="container">
38     [% include '_messages.html' %]
39     <table class="table table-sm">
40         <thead>
41             <th>Sr</th>
42             <th>Image</th>
43             <th>Name</th>
44             <th>Price</th>
45             <th>Discount</th>
46             <th>Brand</th>
47             <th>Edit</th>
48             <th>Delete</th>
49         </thead>
50         <tbody>
51             [% for product in products %]
52                 <tr>
53                     <td>{{loop.index}}</td>
54                     <td></td>
55                     <td>{{product.name}}</td>
56                     <td>{{product.price}}</td>
57                     <td>{{product.discount}}</td>
58                     <td>{{product.brand.name}}</td>
59                     <td><a href="{{url_for('updateproduct', id=product.id)}}>Edit </a></td>
60                     <td><button type="button" class="btn btn-danger btn-sm" data-toggle="modal" data-target="#exampleModal-{{product.id}}">
61                         Delete
62                     </button></td>
63                 </tr>
64             [%-- MODEL --%]
65             <div class="modal fade" id="exampleModal-{{product.id}}" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
66                 <div class="modal-dialog" role="document">
67                     <div class="modal-content">
68                         <div class="modal-header">
69                             <h5 class="modal-title" id="exampleModalLabel">{{product.name}}</h5>
```



The screenshot shows a code editor with a dark theme. The file being edited is a Python template (HTML-like) for a modal dialog. The code includes logic for deleting products or categories. It features a modal header with a title and a close button, a modal body with a confirmation message, and a modal footer with cancel and delete buttons. The code uses Bootstrap classes for styling.

```
File Edit Selection View Go Run Terminal Help ← → ⌘ Search
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > admin > index.html > ...
61     Delete
62     </button></td>
63   </tr>
64
65   <!--===== MODEL =====-->
66   <div class="modal fade" id="exampleModal-{{product.id}}" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
67     <div class="modal-dialog" role="document">
68       <div class="modal-content">
69         <div class="modal-header">
70           <h5 class="modal-title" id="exampleModalLabel">{{product.name}}</h5>
71           <button type="button" class="close" data-dismiss="modal" aria-label="Close">
72             <span aria-hidden="true">&times;
```

This is the coding for admin page from which the admin can navigate to pages with features of add/edit/delete products/categories and perform these activities.

Products:

Add product.html

```
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > products > addproduct.html > ...
1  [% extends "layout.html" %]
2  [% block content %]
3
4  <div class="container">
5      <nav class="navbar navbar-expand-lg navbar-light bg-light">
6          <a class="navbar-brand" href="#">{% url_for('admin') %}{% url_for('admin') %}>product <span class="sr-only">(current)</span></a>
14                 </li>
15                 <li class="nav-item">
16                     <a class="nav-link" href="#">{% url_for('brands') %}>brand</a>
17                 </li>
18                 <li class="nav-item">
19                     <a class="nav-link" href="#">{% url_for('categories') %}>catgory</a>
20                 </li>
21                 <li class="nav-item dropdown">
22                     <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
23                         Add products
24                     </a>
25                     <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
26                         <a class="dropdown-item" href="#">{% url_for('addproduct') %}>Add product</a>
27                         <a class="dropdown-item" href="#">{% url_for('addbrand') %}>Add brand</a>
28                         <a class="dropdown-item" href="#">{% url_for('addcat') %}>Add category</a>
29                     </div>
30                 </li>
31             </ul>
32         </div>
33     </nav>
34 </div>
35
36 <!-------END NAVBAR ----->
37
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF HTML

```
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > products > addproduct.html > ...
38  <div class="container">
39      <div class="row">
40          <div class="col-md-1"></div>
41          <div class="col-md-10">
42              <div class="text-center bg-info p-2 h4">Add a product</div>
43              <% from "&_formhelpers.html" import render_field %>
44              <form method="post" enctype="multipart/form-data">
45                  <div>
46                      {{ render_field(form.name, class="form-control") }}
47                      {{ render_field(form.price, class="form-control") }}
48                      {{ render_field(form.discount, class="form-control") }}
49                      {{ render_field(form.stock, class="form-control") }}
50                      <label for="brand">Add a brand:</label>
51                      <select name="brand" id="brand" class="form-control" required>
52                          {% if getproduct %}
53                              <option value="{{getproduct.brand_id}}>{{getproduct.brand.name}}</option>
54                          {% for brand in brands %}
55                              <option value="{{brand.id}}>{{brand.name}}</option>
56                          {% endfor %}
57                          {% else %}
58                              <option value=""> Select A brand</option>
59                          {% for brand in brands %}
60                              <option value="{{brand.id}}>{{brand.name}}</option>
61                          {% endfor %}
62                          {% endif %}
63                      </select>
64
65                      <label for="category">Add a category:</label>
66                      <select name="category" id="category" class="form-control" required>
67                          {% if getproduct %}
68                              <option value="{{getproduct.category_id}}>{{getproduct.category.name}}</option>
69                          {% for category in categories %}
70                              <option value="{{category.id}}>{{category.name}}</option>
71                          {% endfor %}
72                          {% else %}
73                              <option value=""> Select A category</option>
74                          {% for category in categories %}
75
```

Ln 1, Col 1 Spaces: 2 UTF-8 LF HTML

```

 76     {% endif %}
 77     {% endif %}
 78   
```

```

 79   {{ render_field(form.colors, class="form-control") }}
 80   {{ render_field(form.description, class="form-control", rows="10") }}
```

```

 81   
```

```

 82   
```

```

 83   
```

```

 84   
```

```

 85   
```

```

 86   
```

```

 87   
```

```

 88   
```

```

 89   
```

```

 90   
```

```

 91   
```

```

 92   
```

```

 93   
```

```

 94   
```

```

 95   
```

```

 96   
```

```

 97   
```

```

 98   
```

```

 99   
```

```

100   
```

```

101   
```

```

102   
```

```

103   
```

```

104   
```

```

105   
```

```

106   
```

```

107   
```

```

108   
```

```

109   
```

```

110   
```

This is the coding for adding products.

Add brands.html

```

 1  [% extends "layout.html" %]
 2  [% block content %]
 3
 4   
```

```

 5   
```

```

 6   
```

```

 7   
```

```

 8   
```

```

 9   
```

```

10   
```

```

11   
```

```

12   
```

```

13   
```

```

14   
```

```

15   
```

```

16   
```

```

17   
```

```

18   
```

```

19   
```

```

20   
```

```

21   
```

```

22   
```

```

23   
```

```

24   
```

```

25   
```

```

26   
```

```

27   
```

```

28   
```

```

29   
```

```

30   
```

```

31   
```

```

32   
```

```

33   
```

```

34   
```

```

35   
```

```

36   
```

```

37   
```

Screenshot of a code editor showing the file `addbrand.html`. The code is a Jinja template for adding or updating brands. It includes sections for updating a brand and adding a new brand, each with input fields for name and a submit button. The code uses Bootstrap classes like `col-md-3`, `form-control`, and `btn btn-info`.

```

38 <div class="container">
39     <div class="row">
40         <div class="col-md-6">
41             (% include 'messages.html' %)
42             (% if brands %)
43                 <div class="text-center p-2 h3 mb-3">{% if updatebrand %} Update {%- else %}Add a {%- endif %} brand</div>
44                 <form method="post">
45                     <div action="" method="post">
46                         (% if updatebrand %)
47                             <input type="text" name="brand" class="form-control" value="{{updatebrand.name}}">
48                             <input type="submit" value="Update brand" class="btn btn-info mt-3">
49                         (% else %)
50                             <input type="text" name="brand" class="form-control">
51                             <input type="submit" value="Add a brand" class="btn btn-info mt-3">
52                         (% endif %)
53                     </div>
54                 </form>
55             (% else %)
56                 <div class="text-center p-2 h3 mb-3">{%- if updatecat %}Upade {%- else %}Add a {%- endif %}category</div>
57                 <form method="post">
58                     <div action="" method="post">
59                         (% if updatecat %)
60                             <input type="text" name="category" class="form-control" value="{{updatecat.name}}">
61                             <input type="submit" value="Update category" class="btn btn-info mt-3">
62                         (% else %)
63                             <input type="text" name="category" class="form-control">
64                             <input type="submit" value="Add a category" class="btn btn-info mt-3">
65                         (% endif %)
66                     </div>
67                 </form>
68             </div>
69         </div>
70     </div>
71 </div>
72 </div>
73 </div>
74 
```

Ln 1, Col 28 Spaces: 2 UTF-8 LF HTML

Carts.html

Screenshot of a code editor showing the file `carts.html`. This is a Jinja template for displaying a shopping cart. It extends the `layout.html` base template and includes a `messages.html` block. The main content is a table with columns for Sr, Image, Name, Color, Price, Quantity, Discount, Subtotal, Update, and Delete. Each row represents a product in the session's shopping cart. A form is provided for updating the quantity.

```

1 [% extends 'layout.html' %]
2 [% block content %]
3     [% include 'navbar.html' %]
4     <div class="container mt-4">
5         (% include 'messages.html' %)
6         <div class="row">
7             <div class="col-md-12">
8                 <table class="table table-sm">
9                     <thead>
10                         <th>Sr</th>
11                         <th>Image</th>
12                         <th>Name</th>
13                         <th>Color</th>
14                         <th>Price</th>
15                         <th>Quantity</th>
16                         <th>Discount</th>
17                         <th>Subtotal</th>
18                         <th>Update</th>
19                         <th>Delete</th>
20                     </thead>
21                     <tbody>
22                         (% for key, product in session['Shoppingcart'].items() %)
23                             (% set discount = (product.discount/100) * product.price|float %)
24                             <tr>
25                                 <td>{{loop.index}}</td>
26                                 <td></td>
27                                 <td>{{product.name}}</td>
28                                 <form action="{{url_for('updatecart', code=key)}}" method="post">
29                                     <td>
30                                         (% set colors = product.colors.split(',') %)
31                                         <label for="colors">Colors: </label>
32                                         <select name="color" id="color">
33                                             <option value="{{product.color}}" style="display: none;">{{product.color|capitalize}}</option>
34                                             (% for color in colors %)
35                                                 <option value="{{color}}">{{color|capitalize}}</option>
36                                             (% endfor %)
37                                     </td>
38                                 </form>
39                             </tr>
40                         (% endfor %)
41                     </tbody>
42                 </table>
43             </div>
44         </div>
45     </div>
46 
```

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML

```

38     </select>
39   </td>
40   <td>${"%,.2f".format(product.price)}</td>
41   <td><input type="number" name="quantity" min="1" max="10" value="{{product.quantity}}"></td>
42   (% if product.discount %)
43   <td>{{product.discount}} %&brnbsp; is {{"%.2f".format(discount)}}</td>
44   (% else %)
45   <td>/</td>
46   (% endif %)
47   (% set subtotal = product.quantity|int * product.price|float %)
48   <td>{{"%.2f".format(subtotal - discount|round(1,'floor'))}}</td>
49   <td><button type="submit" class="btn btn-sm btn-info">Update</button></td>
50 </form>
51   <td> <a href="{{url_for('deleteitem', id=key)}}>Delete</a></td>
52 </tr>
53   (% endfor %)
54 </tbody>
55 </table>
56 <table class="table table-sm">
57   <tr>
58     <td> <a href="{{url_for('get_order')}}" class="btn btn-success">Order now </a> </td>
59     <td width="35%"></td>
60     <td> <h5>Tax: ${{tax}}</h5></td>
61     <td> <h3>Grand total: ${{grandtotal}}</h3> </td>
62     <td> <a href="{{url_for('clearcart')}}" class="btn btn-danger btn-sm float-right mr-4">Clear cart</a> </td>
63   </tr>
64 </table>
65 </div>
66 </div>
67 </div>
68 </div>
69 (% endblock content %)

```

Ln 1, Col 1 | Spaces: 4 | UTF-8 | LF | HTML | ⚡

This is the coding to display cart. This page has the items added to cart by the customers. From this page the customer can clear the cart items or can move to the payment and checkout process by ordering the items.

Index.html

```

1  [% extends 'layout.html' %]
2  [% block content %]
3  [% include 'navbar.html' %]
4  <div class="container">
5    <% if brand %>
6      <% for b in brand.items %>
7        <div class="col-md-3 mt-4">
8          <div class="card">
9            
10           <div class="card-body">
11             <% if b.discount > 0 %>
12               <h5 style="text-shadow: 1px 2px 2px #000; color: #f00; transform: rotate(-15deg); position: absolute; top: 23%; left: 25%; font-weight: 600;">Dis
13             <% endif %>
14             <h5 class="text-center">{{b.name}}</h5>
15             <p class="text-center">Price ${{b.price}}</p>
16           </div>
17           <div class="card-footer">
18             <a href="{{url_for('single_page', id=b.id)}}>Details</a>
19             <form action="{{url_for('AddCart')}}" method="post">
20               <input type="hidden" name="product_id" value="{{(b.id)}}>
21               <button type="submit" class="btn btn-sm btn-warning float-right">Add to Cart</button>
22               <input type="hidden" name="quantity" value="1" min="1" max="20">
23               (% set colors = b.colors.split(',') %)
24               <select name="colors" id="colors" style="visibility: hidden;">
25                 (% for color in colors %)
26                   <option value="{{color}}>{{color}}</option>
27                 (% set col = color.split(':') %)
28                   <option value="{{col[0]}}>{{col[0]} | capitalize }}</option>
29                 (% endfor %)
30               </select>
31             </form>
32           </div>
33         </div>
34       <% endfor %>
35     </div>
36   </div>

```

Ln 1, Col 1 | Spaces: 2 | UTF-8 | LF | HTML | ⚡

```
File Edit Selection View Go Run Terminal Help ⏎ → ⏎ Search
C:\Users\shani>Downloads>flaskecommerce-master>flaskecommerce-master>templates>products> index.html ...
38     <div class="row mt-4">
39         <div class="col text-center">
40             {% if brand.has_prev %}
41                 <a href="{{url_for('get_brand', id=get_brand.id, page=brand.prev_num)}}>{{page_num}}</a>
42             {% endif %}
43             {% if brand.total > 8 %}
44             {% for page_num in brand.iter_pages(left_edge=1, right_edge=2, left_current=1,right_current=2) %}
45                 {% if page_num %}
46                     {% if brand.page == page_num %}
47                         <a href="{{url_for('get_brand', id=get_brand.id, page=page_num)}}>{{page_num}}</a>
48                     {% else %}
49                         <a href="{{url_for('get_brand', id=get_brand.id, page=page_num)}}>{{page_num}}</a>
50                     {% endif %}
51                 {% else %}
52                     ...
53                 {% endif %}
54             {% endif %}
55             {% endif %}
56             {% if brand.has_next %}
57                 <a href="{{url_for('get_brand', id=get_brand.id, page=brand.next_num)}}>{{page_num}}</a>
58             {% endif %}
59         </div>
60     <!--===== END OF BRANDS FORLOOP =====-->
61     <div class="row">
62         {% elif get_cat_prod %}
63         {% for get_cat in get_cat_prod.items %}
64             <div class="col-md-3 mt-4">
65                 <div class="card">
66                     
67                     <div class="card-body">
68                         {% if get_cat.discount > 0 %}
69                             <div style="text-shadow: 1px 2px 2px #000; color: red; transform: rotate(-15deg); position: absolute; top: 23%; left: 25%; font-weight: bold;"> Discount {{get_cat.discount}}% </div>
70                         {% endif %}
71                         <h5 class="text-center">{{get_cat.name}}</h5>
72                         <p class="text-center">Price ${{get_cat.price}}</p>
73             </div>
74         </div>
75     </div>
76     <div class="card-footer">
77         <a href="{{url_for('single_page', id=get_cat.id)}}>Details</a>
78         <form action="{{url_for('AddCart')}}" method="post">
79             <input type="hidden" name="product_id" value="{{get_cat.id}}>
80             <button type="submit" class="btn btn-warning float-right">Add to Cart</button>
81             <input type="hidden" name="quantity" value="1" min="1" max="20">
82             {% set colors = get_cat.colors.split(',') %}
83             <select name="colors" id="colors" style="visibility: hidden;">
84                 {% for color in colors %}
85                     <option value="{{color}}>{{color}}</option>
86                 {% endfor %}
87             </select>
88         </form>
89     </div>
90     </div>
91     </div>
92     {% endfor %}
93 </div>
94
95     <div class="row mt-4">
96         <div class="col text-center">
97             {% if get_cat_prod.has_prev %}
98                 <a href="{{url_for('get_category', id=get_cat.id, page=get_cat_prod.prev_num)}}>{{page_num}}</a>
99             {% endif %}
100             {% if get_cat_prod.total > 8 %}
101             {% for page_num in get_cat_prod.iter_pages(left_edge=1, right_edge=2, left_current=1,right_current=2) %}
102                 {% if page_num %}
103                     {% if get_cat_prod.page == page_num %}
104                         <a href="{{url_for('get_category', id=get_cat.id, page=page_num)}}>{{page_num}}</a>
105                     {% else %}
106                         <a href="{{url_for('get_category', id=get_cat.id, page=page_num)}}>{{page_num}}</a>
107                     {% endif %}
108                 {% else %}
109                     ...
110             </div>
111         </div>
112     <!--===== END OF CATEGORIES FORLOOP =====-->
113     <div class="row mt-4">
114         <div class="col text-center">
115             {% if login_form.errors %}
116                 <div style="background-color: #f8d7da; border: 1px solid #ffccbc; padding: 5px; margin-bottom: 10px;">
117                     <strong>Please correct the error(s) below:</strong>
118                     <ul style="list-style-type: none; padding-left: 0; margin: 0; font-size: small;">
119                         <li>{{error}}</li>
120                     </ul>
121                 </div>
122             {% endif %}
123             <form action="{{url_for('login')}}" method="post">
124                 <input type="text" name="username" placeholder="Username" required="required" style="width: 100%; height: 40px; border: 1px solid #ccc; border-radius: 5px; margin-bottom: 10px;">
125                 <input type="password" name="password" placeholder="Password" required="required" style="width: 100%; height: 40px; border: 1px solid #ccc; border-radius: 5px; margin-bottom: 10px;">
126                 <input type="checkbox" name="remember_me" checked="checked" style="margin-bottom: 10px;"> Remember Me
127                 <input type="submit" value="Login" style="width: 100%; height: 40px; background-color: #ff9800; color: white; border: none; border-radius: 5px; font-weight: bold; font-size: inherit; cursor: pointer;">
128             </form>
129         </div>
130     </div>
131 
```

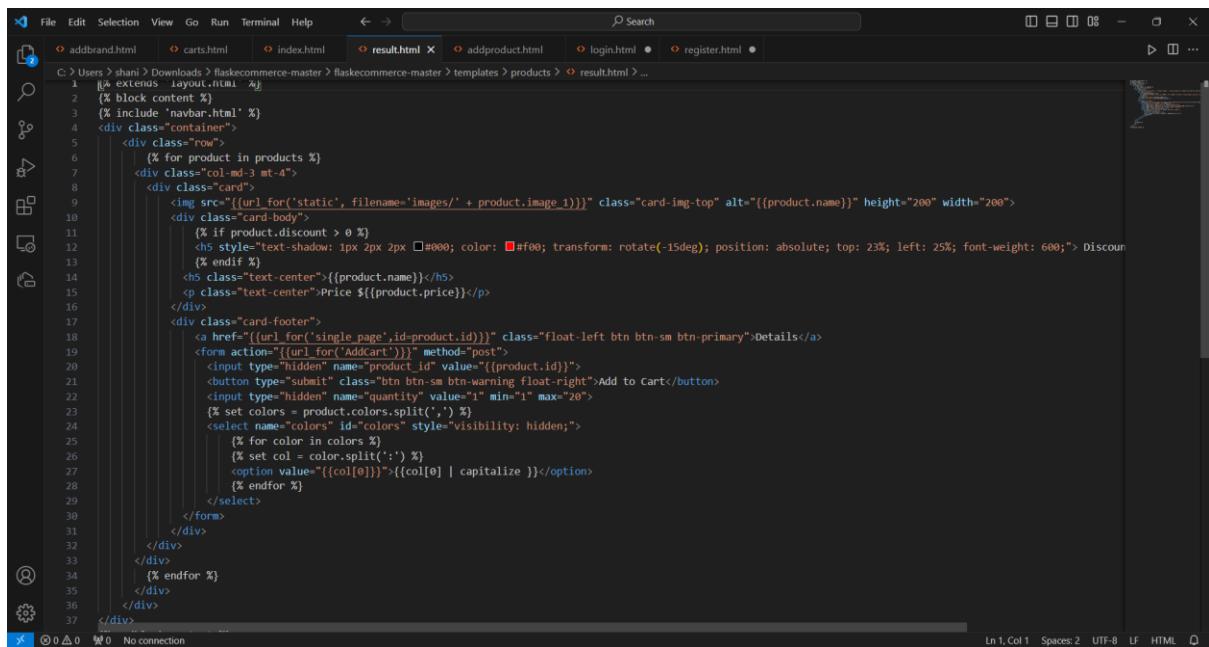
```
File Edit Selection View Go Run Terminal Help ⏎ → ⏎ Search
C:\Users\shani>Downloads>flaskecommerce-master>flaskecommerce-master>templates>products> index.html ...
74         </div>
75     <div class="card-footer">
76         <a href="{{url_for('single_page', id=get_cat.id)}}>Details</a>
77         <form action="{{url_for('AddCart')}}" method="post">
78             <input type="hidden" name="product_id" value="{{get_cat.id}}>
79             <button type="submit" class="btn btn-warning float-right">Add to Cart</button>
80             <input type="hidden" name="quantity" value="1" min="1" max="20">
81             {% set colors = get_cat.colors.split(',') %}
82             <select name="colors" id="colors" style="visibility: hidden;">
83                 {% for color in colors %}
84                     <option value="{{color}}>{{color}}</option>
85                 {% endfor %}
86             </select>
87         </form>
88     </div>
89     </div>
90     </div>
91     </div>
92     {% endfor %}
93 </div>
94
95     <div class="row mt-4">
96         <div class="col text-center">
97             {% if get_cat_prod.has_prev %}
98                 <a href="{{url_for('get_category', id=get_cat.id, page=get_cat_prod.prev_num)}}>{{page_num}}</a>
99             {% endif %}
100             {% if get_cat_prod.total > 8 %}
101             {% for page_num in get_cat_prod.iter_pages(left_edge=1, right_edge=2, left_current=1,right_current=2) %}
102                 {% if page_num %}
103                     {% if get_cat_prod.page == page_num %}
104                         <a href="{{url_for('get_category', id=get_cat.id, page=page_num)}}>{{page_num}}</a>
105                     {% else %}
106                         <a href="{{url_for('get_category', id=get_cat.id, page=page_num)}}>{{page_num}}</a>
107                     {% endif %}
108                 {% else %}
109                     ...
110             </div>
111         </div>
112     <!--===== END OF CATEGORIES FORLOOP =====-->
113     <div class="row mt-4">
114         <div class="col text-center">
115             {% if login_form.errors %}
116                 <div style="background-color: #f8d7da; border: 1px solid #ffccbc; padding: 5px; margin-bottom: 10px;">
117                     <strong>Please correct the error(s) below:</strong>
118                     <ul style="list-style-type: none; padding-left: 0; margin: 0; font-size: small;">
119                         <li>{{error}}</li>
120                     </ul>
121                 </div>
122             {% endif %}
123             <form action="{{url_for('login')}}" method="post">
124                 <input type="text" name="username" placeholder="Username" required="required" style="width: 100%; height: 40px; border: 1px solid #ccc; border-radius: 5px; margin-bottom: 10px;">
125                 <input type="password" name="password" placeholder="Password" required="required" style="width: 100%; height: 40px; border: 1px solid #ccc; border-radius: 5px; margin-bottom: 10px;">
126                 <input type="checkbox" name="remember_me" checked="checked" style="margin-bottom: 10px;"> Remember Me
127                 <input type="submit" value="Login" style="width: 100%; height: 40px; background-color: #ff9800; color: white; border: none; border-radius: 5px; font-weight: bold; font-size: inherit; cursor: pointer;">
128             </form>
129         </div>
130     </div>
131 
```

```
File Edit Selection View Go Run Terminal Help ⏎ → Search
C:\Users\shani\Downloads>flaskcommerce-master>flaskcommerce-master>templates>products> index.html ...
112     {% endif %}
113     {% endif %}
114     {% if get_cat_prod.has_next %}
115         <a href="{{url_for('home', page=get_cat_prod.next_num)}}>{{product.name}}</a>
116     {% endif %}
117     </div>
118 </div>
119     <!-- ===== END ELSE IF =====-->
120     <div class="row">
121         {% else %}
122             {% for product in products.items %}
123                 <div class="col-md-3 mt-4">
124                     <div class="card">
125                         
126                         <div class="card-body">
127                             {% if product.discount > 0 %}
128                                 <div style="text-shadow: 1px 2px 2px #0000; color: #f00; transform: rotate(-15deg); position: absolute; top: 23%; left: 25%; font-weight: 600; margin-left: -15px; margin-top: -10px; z-index: 1;>{{product.discount}}%</div>
129                             {% endif %}
130                             <h5 class="text-center">{{product.name}}</h5>
131                             <p class="text-center">Price {{product.price}}</p>
132                         </div>
133                         <div class="card-footer">
134                             <a href="{{url_for('single_page', id=product.id)}}" class="float-left btn btn-sm btn-primary">Details</a>
135                             <form action="{{url_for('Addcart')}}" method="post">
136                                 <input type="hidden" name="product_id" value="{{product.id}}"/>
137                                 <button type="submit" class="btn btn-sm btn-warning float-right">Add to Cart</button>
138                                 <input type="hidden" name="quantity" value="1" min="1" max="20">
139                                 {% set colors = product.colors.split(',') %}
140                                 <select name="colors" id="colors" style="visibility: hidden;">
141                                     {% for color in colors %}
142                                         <option value="{{color}}>{{color | capitalize}}</option>
143                                     {% endfor %}
144                                 </select>
145                             </form>
146                         </div>
147                     </div>
148                 </div>
149             {% for product in products.items %}
150                 <div class="col-md-3 mt-4">
151                     <div class="card">
152                         
153                         <div class="card-body">
154                             <div class="row mt-4">
155                                 <div class="col text-center">
156                                     {% if products.has_prev %}
157                                         <a href="{{url_for('home', page=products.prev_num)}}>{{product.name}}</a>
158                                     {% endif %}
159                                     {% if products.total > 8 %}
160                                         <div style="text-align: center; margin-top: 10px; margin-bottom: 10px; border: 1px solid #ccc; padding: 5px; display: inline-block; width: fit-content; border-radius: 10px; background-color: #f0f0f0; font-size: 0.9em; font-weight: bold; text-decoration: none; color: inherit; transition: all 0.3s ease; position: relative; z-index: 1;>
161                                         <div style="position: absolute; left: 0; top: 0; width: 100%; height: 100%; background-color: white; border-radius: 10px; z-index: 2; transition: all 0.3s ease; opacity: 0; transform: scale(0.9);></div>
162                                         <a href="{{url_for('home', page=page_num)}}>{{page_num}}</a>
163                                         {% else %}
164                                         <a href="{{url_for('home', page=page_num)}}>{{page_num}}</a>
165                                         {% endif %}
166                                         {% else %}
167                                         ...
168                                         {% endif %}
169                                         {% endfor %}
170                                         {% endif %}
171                                         {% if products.has_next %}
172                                         <a href="{{url_for('home', page=products.next_num)}}>{{product.name}}</a>
173                                         {% endif %}
174                                     </div>
175                                 </div>
176                             </div>
177                         </div>
178                     </div>
179                 </div>
180             {% endblock content %}
```

```
File Edit Selection View Go Run Terminal Help ⏎ → Search
C:\Users\shani\Downloads>flaskcommerce-master>flaskcommerce-master>templates>products> index.html ...
146             </div>
147         </div>
148     </div>
149     {% endif %}
150   </div>
151 
152   <div class="row mt-4">
153     <div class="col text-center">
154       {% if products.has_prev %}
155           <a href="{{url_for('home', page=products.prev_num)}}>{{product.name}}</a>
156       {% endif %}
157       {% if products.total > 8 %}
158           <div style="text-align: center; margin-top: 10px; margin-bottom: 10px; border: 1px solid #ccc; padding: 5px; display: inline-block; width: fit-content; border-radius: 10px; background-color: #f0f0f0; font-size: 0.9em; font-weight: bold; text-decoration: none; color: inherit; transition: all 0.3s ease; position: relative; z-index: 1;>
159             <div style="position: absolute; left: 0; top: 0; width: 100%; height: 100%; background-color: white; border-radius: 10px; z-index: 2; transition: all 0.3s ease; opacity: 0; transform: scale(0.9);></div>
160             <a href="{{url_for('home', page=page_num)}}>{{page_num}}</a>
161             {% else %}
162             <a href="{{url_for('home', page=page_num)}}>{{page_num}}</a>
163             {% endif %}
164             {% else %}
165             ...
166             {% endif %}
167             {% endfor %}
168             {% endif %}
169             {% endif %}
170             {% if products.has_next %}
171             <a href="{{url_for('home', page=products.next_num)}}>{{product.name}}</a>
172             {% endif %}
173         </div>
174     </div>
175   </div>
176 
177   <% endif %>
178 </div>
179 <% endblock content %>
```

This is the coding for all products page. The customer can view all the available products in this page. If the customer clicks any product, then the user will be redirected to the individual products page.

Result.html

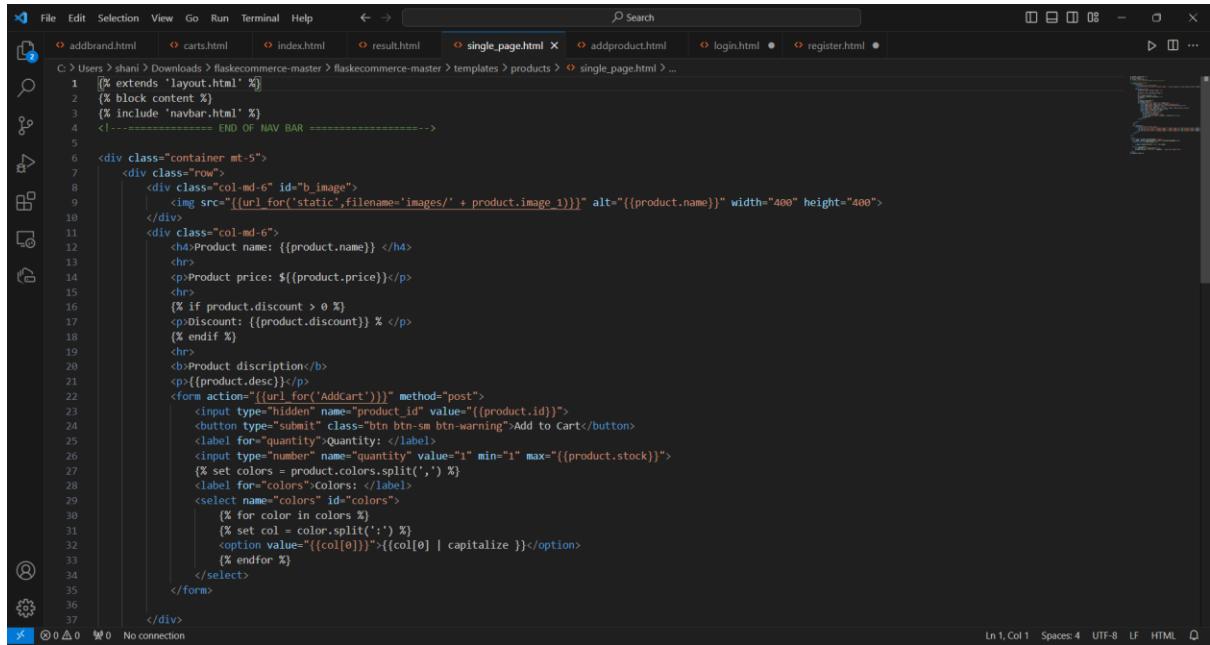


The screenshot shows a code editor window with the file 'result.html' open. The code is a Jinja template for displaying products. It includes a header section with a logo and navigation links, followed by a main content area where products are listed in a grid. Each product card displays the product name, price, and a 'Details' link. A form is present for adding the product to the cart, including fields for quantity and color selection. The code uses Bootstrap classes for styling.

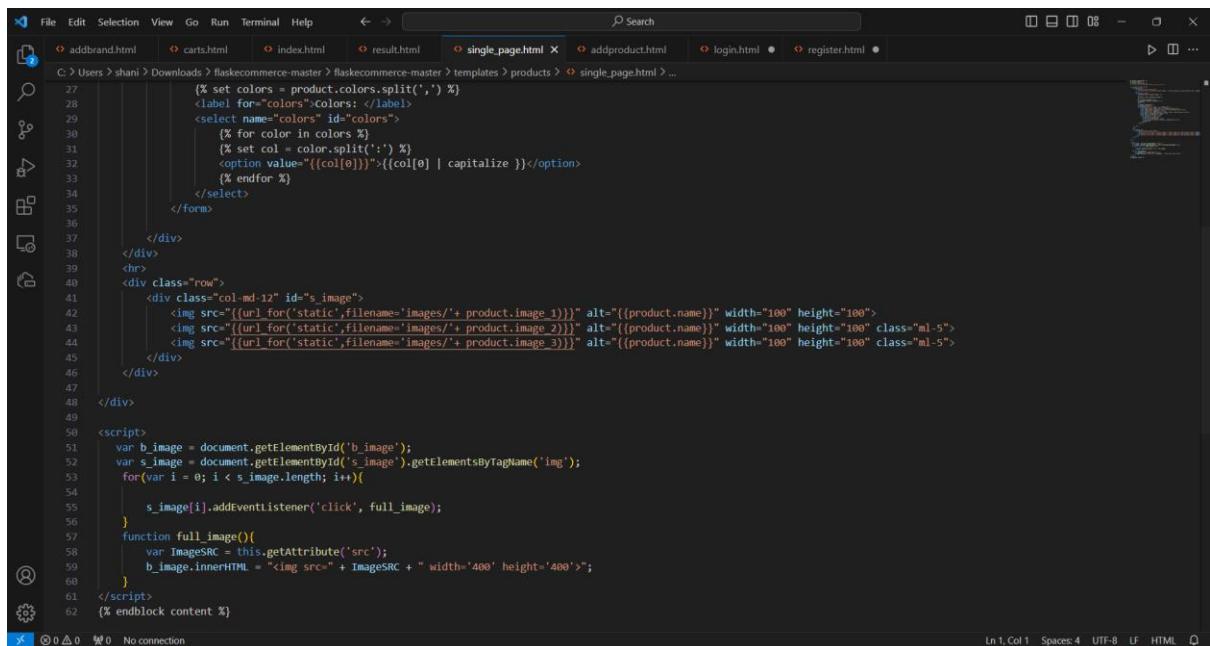
```
1  {% extends 'layout.html' %} 2  {% block content %} 3  {% include 'navbar.html' %} 4  <div class="container"> 5    <div class="row"> 6      {% for product in products %} 7        <div class="col-md-3 mt-4"> 8          <div class="card"> 9             10           <div class="card-body"> 11             {% if product.discount > 0 %} 12               <h5 style="text-shadow: 1px 2px 2px #000; color: #f00; transform: rotate(-15deg); position: absolute; top: 23%; left: 25%; font-weight: 600;"> Discount! 13             {% endif %} 14             <h5>{{product.name}}</h5> 15             <p>Price {{product.price}}</p> 16           </div> 17           <div class="card-footer"> 18             <a href="{{url_for('single_page', id=product.id)}}" class="float-left btn btn-sm btn-primary">Details</a> 19             <form action="{{url_for('AddCart')}}" method="post"> 20               <input type="hidden" name="product_id" value="{{product.id}}"> 21               <button type="submit" class="btn btn-sm btn-warning float-right">Add to Cart</button> 22               <input type="hidden" name="quantity" value="1" min="1" max="20"> 23               {% set colors = product.colors.split(',') %} 24               <select name="colors" id="colors" style="visibility: hidden;"> 25                 {% for color in colors %} 26                   {% set col = color.split(':') %} 27                   <option value="{{col[0]}}>{{col[0]} | capitalize }}</option> 28                 {% endfor %} 29               </select> 30             </form> 31           </div> 32         </div> 33       <% endfor %> 34     </div> 35   </div> 36 </div> 37 </div>
```

This is the coding for individual product page. This has the details about a particular product the user to add the product to the cart by clicking add to cart button provided there.

Single result.html



```
C:\> Users > shani > Downloads > flaskcommerce-master > flaskcommerce-master > templates > products > single_page.html ...  
1  {% extends 'layout.html' %}  
2  {% block content %}  
3  {% include 'navbar.html' %}  
4  <!--===== END OF NAV BAR =====-->  
5  
6  <div class="container mt-5">  
7    <div class="row">  
8      <div class="col-md-6" id="b_image">  
9          
10     </div>  
11     <div class="col-md-6">  
12       <h4>Product name: {{product.name}}</h4>  
13       <hr>  
14       <p>Product price: ${{product.price}}</p>  
15       <hr>  
16       {% if product.discount > 0 %}  
17         <p>Discount: {{product.discount}} %</p>  
18       {% endif %}  
19       <hr>  
20       <b>Product description</b>  
21       <p>{{product.desc}}</p>  
22       <form action="{{url_for('AddCart')}}" method="post">  
23         <input type="hidden" name="product_id" value="{{product.id}}>  
24         <button type="submit" class="btn btn-sm btn-warning">Add to Cart</button>  
25         <label for="quantity">Quantity: </label>  
26         <input type="number" name="quantity" value="1" min="1" max="{{(product.stock)}}>  
27         {% set colors = product.colors.split(',') %}  
28         <label for="colors">Colors: </label>  
29         <select name="colors" id="colors">  
30           {% for color in colors %}  
31             {% set col = color.split(':') %}  
32             <option value="{{col[0]}}>{{col[0]} | capitalize }}</option>  
33           {% endfor %}  
34         </select>  
35       </form>  
36     </div>  
37   </div>  
38 </div>  
39 <hr>  
40 <div class="row">  
41   <div class="col-md-12" id="s_image">  
42       
43       
44       
45   </div>  
46 </div>  
47 </div>  
48 </div>  
49 <script>  
50   var b_image = document.getElementById('b_image');  
51   var s_image = document.getElementById('s_image').getElementsByTagName('img');  
52   for(var i = 0; i < s_image.length; i++){  
53     s_image[i].addEventListener('click', full_image);  
54   }  
55   function full_image(){  
56     var ImageSRC = this.getAttribute('src');  
57     b_image.innerHTML = "img src=" + ImageSRC + " width='400' height='400'>";  
58   }  
59 </script>  
60 <% endblock content %>
```



```
C:\> Users > shani > Downloads > flaskcommerce-master > flaskcommerce-master > templates > products > single_page.html ...  
27  {% set colors = product.colors.split(',') %}  
28  <label for="colors">Colors: </label>  
29  <select name="colors" id="colors">  
30    {% for color in colors %}  
31      {% set col = color.split(':') %}  
32      <option value="{{col[0]}}>{{col[0]} | capitalize }}</option>  
33    {% endfor %}  
34  </select>  
35 </form>  
36 </div>  
37 </div>  
38 <hr>  
39 <div class="row">  
40   <div class="col-md-12" id="s_image">  
41       
42       
43       
44   </div>  
45 </div>  
46 </div>  
47 </div>  
48 </div>  
49 <script>  
50   var b_image = document.getElementById('b_image');  
51   var s_image = document.getElementById('s_image').getElementsByTagName('img');  
52   for(var i = 0; i < s_image.length; i++){  
53     s_image[i].addEventListener('click', full_image);  
54   }  
55   function full_image(){  
56     var ImageSRC = this.getAttribute('src');  
57     b_image.innerHTML = "img src=" + ImageSRC + " width='400' height='400'>";  
58   }  
59 </script>  
60 <% endblock content %>
```

This is the individual page with description and other necessary details. Once the user clicks a particular

product he will navigate to the individual products page. The page has a summarized detail about the product. To view a detailed description of the product this page has been designed.

Order.html

```

File Edit Selection View Go Run Terminal Help ← → Search
C:\Users\shani\Downloads\flaskecommerce-master\flaskecommerce-master\templates\customer>order.html ...
1 [% extends 'layout.html' %]
2 [% block content %]
3 [% include 'navbar.html' %]
4 <div class="container mt-4">
5   [% include '_messages.html' %]
6   <div class="row">
7     <div class="col-md-12">
8       invoice: {{orders.invoice}}
9       <br>
10      status: {{orders.status}}
11      <br>
12      Customer name: {{customer.name}}
13      <br>
14      Customer email: {{customer.email}}
15      <br>
16      Customer contact: {{customer.contact}}
17      <br>
18      <br>
19      <table class="table table-sm">
20        <thead>
21          <th>Sr</th>
22          <th>Name</th>
23          <th>Color</th>
24          <th>Price</th>
25          <th>Quantity</th>
26          <th>Discount</th>
27          <th>Subtotal</th>
28        </thead>
29        <tbody>
30          [% for key, product in orders.orders.items() %]
31            [% set discount = (product.discount/100) * product.price|float %]
32            <tr>
33              <td>{{loop.index}}</td>
34              <td>{{product.name}}</td>
35              <form action="{{url_for('updatecart', code=key)}}" method="post">
36                <td>
37                  {{product.color|capitalize}}
38                </td>
39                <td>${{"%.2f" | format(product.price)}}</td>
40                <td>{{product.quantity}}</td>
41                [% if product.discount %]
42                  <td>{{product.discount}} % &nbsp; is ${{"%.2f" | format(discount)}}</td>
43                [% else %]
44                  <td>/</td>
45                [% endif %]
46                [% set subtotal = product.quantity|int * product.price|float %]
47                <td>${{"%.2f" | format(subtotal - discount|round(1,'floor')) }}</td>
48              </form>
49            </tr>
50          [% endfor %]
51        </tbody>
52      </table>
53      <table class="table table-sm">
54        <tr>
55          <td>
56            [% if orders.status =='Paid' %]
57            [% else %]
58              <form action="{{url_for('payment')}}" method="POST">
59                [% set amount = grandTotal.replace('.','') %]
60                <input type="hidden" name="amount" value="{{(amount)}}>
61                <input type="hidden" name="invoice" value="{{(orders.invoice)}}>
62                <script src="https://checkout.stripe.com/checkout.js">
63                  class="stripe-button"
64                  data-key="pk_test_MaIxTYQ1sv5Uh6nK19wPd00eqL0qzsl"
65                  data-name="{{customers.name}}"
66                  data-description="myshop purchase"
67                  data-amount="{{(amount) }}"
68                  data-currency="usd"
69                </script>
70              </form>
71            [% endif %]
72          </td>
73        <td width="35%></td>
74      </tr>

```

```

File Edit Selection View Go Run Terminal Help ← → Search
C:\Users\shani\Downloads\flaskecommerce-master\flaskecommerce-master\templates\customer>order.html ...
38      </td>
39      <td>${{"%.2f" | format(product.price)}}</td>
40      <td>{{product.quantity}}</td>
41      [% if product.discount %]
42        <td>{{product.discount}} % &nbsp; is ${{"%.2f" | format(discount)}}</td>
43      [% else %]
44        <td>/</td>
45      [% endif %]
46      [% set subtotal = product.quantity|int * product.price|float %]
47      <td>${{"%.2f" | format(subtotal - discount|round(1,'floor')) }}</td>
48    </form>
49  </tr>
50  </tbody>
51 </table>
52 <table class="table table-sm">
53   <tr>
54     <td>
55       [% if orders.status =='Paid' %]
56       [% else %]
57         <form action="{{url_for('payment')}}" method="POST">
58           [% set amount = grandTotal.replace('.','') %]
59           <input type="hidden" name="amount" value="{{(amount)}}>
60           <input type="hidden" name="invoice" value="{{(orders.invoice)}}>
61           <script src="https://checkout.stripe.com/checkout.js">
62             class="stripe-button"
63             data-key="pk_test_MaIxTYQ1sv5Uh6nK19wPd00eqL0qzsl"
64             data-name="{{customers.name}}"
65             data-description="myshop purchase"
66             data-amount="{{(amount) }}"
67             data-currency="usd"
68           </script>
69         </form>
70       [% endif %]
71     </td>
72   <td width="35%></td>
73 </tr>

```

```

File Edit Selection View Go Run Terminal Help ← → Search
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > customer > order.html ...
54
55
56
57     <td>
58         {% if orders.status == 'Paid' %}
59         {% else %}
60             <form action="{{url_for('payment')}}" method="POST">
61                 {% set amount = grandTotal.replace('.', ',') %}
62                 <input type="hidden" name="amount" value="{{(amount)}}>
63                 <input type="hidden" name="invoice" value="{{(orders.invoice)}}>
64                 <script src="https://checkout.stripe.com/checkout.js"
65                     class="stripe-button"
66                     data-key="pk_test_MaIjXtYQ1vsvUhdkIK9wPd00qgLoqzsl"
67                     data-name="{{customers.name}}"
68                     data-description="myshop purchase"
69                     data-amount="{{(amount)}}">
70                     data-currency="usd">
71                 </script>
72             </form>
73         {% endif %}
74     </td>
75     <td width="35%></td>
76     <td> Tax: {{(tax)}}</td>
77     <td> Grand total: {{(grandTotal)}}</td>
78
79     <td>
80         <form action="{{url_for('get_pdf', invoice=orders.invoice)}}" method="post">
81             <button type="submit" class="btn btn-info">Get pdf</button>
82         </form>
83     </td>
84 </tr>
85 </div>
86 </div>
87 </div>
88 {% endblock content %}

```

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML

This is the orders page coding. Once the customer has ordered the items in the cart he will proceed to the payment and checkout process.

Pdf.html

To receive the invoice that is the bill of the items we have the pdf facility.

```

File Edit Selection View Go Run Terminal Help ← → Search
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > customer > pdf.html ...
1  <!DOCTYPE html>
2  <html lang="en">
3      <!-- Required meta tags -->
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6
7      <!-- Bootstrap CSS -->
8      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5Paxtktu6ug5TOeNv6gBiFe" crossorigin="anonymous">
9
10
11      <title>Get pdf</title>
12  </head>
13  <body>
14
15      <div class="container mt-4">
16          {% include '_messages.html' %}
17          <div class="row">
18              <div class="col-md-12">
19                  <b style="float: right;">Invoice: {{(orders.invoice)}} </b>
20                  <br>
21                  Status: {{(orders.status)}}
22                  <br>
23                  Customer name: {{customer.name}}
24                  <br>
25                  Customer email: {{customer.email}}
26                  <br>
27                  Customer contact: {{customer.contact}}
28                  <br>
29                  <br>
30                  <table class="table table-sm">
31                      <thead>
32                          <th>Sr</th>
33                          <th>Name</th>
34                          <th>Color</th>
35                          <th>Price</th>
36                          <th>Quantity</th>
37                          <th>Discount</th>

```

Ln 1, Col 1 Spaces: 4 UTF-8 LF HTML

The screenshot shows a code editor window with the file 'pdf.html' open. The code is a Jinja2 template for generating a PDF. It includes two tables: one for individual items and another for a summary. The code uses Bootstrap classes like 'table' and 'table-sm'. It also includes form elements for updating the cart. The code editor has a dark theme and shows line numbers from 30 to 65.

```
C:\> Users > shani > Downloads > flaskecommerce-master > flaskecommerce-master > templates > customer > pdf.html > ...
30     <table class="table table-sm">
31         <thead>
32             <tr>
33                 <th>Sr</th>
34                 <th>Name</th>
35                 <th>Color</th>
36                 <th>Price</th>
37                 <th>Quantity</th>
38                 <th>Discount</th>
39                 <th>Subtotal</th>
40             </thead>
41             <tbody>
42                 {% for key, product in orders.orders.items() %}
43                     {% set discount = (product.discount/100) * product.price|float %}
44                     <tr>
45                         <td>{{loop.index}}</td>
46                         <td>{{product.name}}</td>
47                         <form action="{{url_for('updatecart', code=key)}}" method="post">
48                             <td>
49                                 {{(product.color|capitalize)}}
50                             </td>
51                             <td>${{"%.2f"|format(product.price)}}</td>
52                             <td>{{product.quantity}}</td>
53                             {% if product.discount %}
54                             <td>${{(product.discount) }} &nbsp; is ${{("%.2f"|format(discount))}}</td>
55                             {% else %}
56                             <td>/</td>
57                             {% endif %}
58                             {% set subtotal = product.quantity|int * product.price|float %}
59                             <td>${{"%.2f"|format(subtotal - discount|round(1,'floor')) }}</td>
60                         </form>
61                         </tr>
62                     {% endfor %}
63                 </tbody>
64             </table>
65             <table class="table table-sm">
66                 <tr>
67                     <td>
```

This screenshot shows the same 'pdf.html' file as above, but with more content. It includes a table with a 35% width column, and two h3 elements for 'Tax' and 'Grand total'. At the bottom, there is a comment about optional JavaScript and three script tags for jQuery, Popper.js, and Bootstrap. The code editor shows line numbers from 47 to 81.

```
47         <td>
48             {{(product.color|capitalize)}}
49         </td>
50         <td>${{"%.2f"|format(product.price)}}</td>
51         <td>{{product.quantity}}</td>
52         {% if product.discount %}
53         <td>${{(product.discount) }} &nbsp; is ${{("%.2f"|format(discount))}}</td>
54         {% else %}
55         <td>/</td>
56         {% endif %}
57         {% set subtotal = product.quantity|int * product.price|float %}
58         <td>${{"%.2f"|format(subtotal - discount|round(1,'floor')) }}</td>
59     </form>
60     </tr>
61     {% endfor %}
62   </tbody>
63   <table class="table table-sm">
64     <tr>
65       <td width="35%"></td>
66       <td>h3>Tax: ${{tax}}</h3></td>
67       <td>h3>Grand total: ${{grandtotal}}</h3> </td>
68     </tr>
69   </table>
70   </div>
71 </div>
72 </div>
73 </div>
74 <!-- Optional JavaScript -->
75 <!-- jQuery first, then Popper.js, then Bootstrap JS -->
76 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa489bE2poT4MyKh5vZFSRp08IEjwBVKU7imGAV0wj1yYfoR5J0z+n" crossorigin="anonymous">
77 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-o6e9R9vH/E4pIjlwsNlZl9W2xXr2b/wu/8L7G3UOGdXn9WpGQGmGZBhs/d/2" >
78 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uudj003uM83njUU041h7YwaYdiqfktj0Uod8GCEx13og8ifwB" >
79 </body>
80 </html>
```

This is the coding to format and style the pdf generated.

Thankyou.html

<h1>Thank you shopping with us! </h1>

This is the coding to generate a thank you message. The output is this.

Thank you shopping with us!

Backend:

Flask program:

Init.py



The screenshot shows a Windows Notepad window titled "_init_.py - Notepad". The window contains Python code for a Flask application. The code imports various modules from the Flask ecosystem, including Flask, SQLAlchemy, Bcrypt, and UploadSet. It sets up a SQLite database at 'test.db', defines a secret key, and configures file uploads to a 'static/images' directory. It initializes a Flask app, sets up a database connection, and creates a search and migration context. It also configures a login manager and defines routes for products, admin, carts, and customers. The code ends with a copyright notice and a license statement.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_uploads import UploadSet, configure_uploads, IMAGES, patch_request_class
import os
from flask_msearch import Search
from flask_login import LoginManager
from flask_migrate import Migrate
basedir = os.path.abspath(os.path.dirname(__file__))
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SECRET_KEY'] = 'hfouewhfoiwefoquw'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['UPLOADED_PHOTOS_DEST'] = os.path.join(basedir, 'static/images')
photos = UploadSet('photos', IMAGES)
configure_uploads(app, photos)
patch_request_class(app)
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
search = Search()
search.init_app(app)
migrate = Migrate(app, db)
with app.app_context():
    if db.engine.url.drivername == "sqlite":
        migrate.init_app(app, db, render_as_batch=True)
    else:
        migrate.init_app(app, db)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'customerLogin'
login_manager.needs_refresh_message_category = 'danger'
login_manager.login_message = u"Please login first"
from shop.products import routes
from shop.admin import routes
from shop.carts import carts
from shop.customers import routes

Copyright © 2018, All rights reserved.
This software is licensed under the MIT License.

```

Run.py

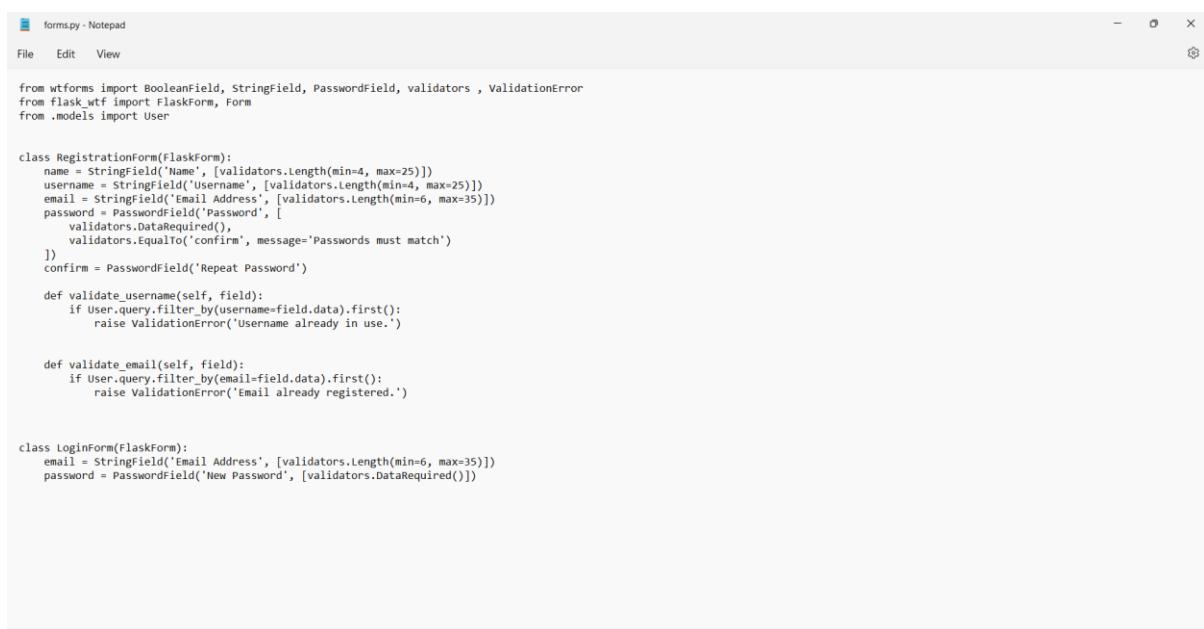
```
from shop import app  
  
if __name__ == "__main__":  
  
    app.run(debug=True)
```

We have created a folder named shop which has all the coding related to the flask program so that the flask coding can be made smaller.

Shop:

Admin:

Forms.py:



A screenshot of a Windows Notepad window titled "forms.py - Notepad". The window shows Python code for defining two forms: RegistrationForm and LoginForm. The RegistrationForm includes fields for name, username, email, and password, with validation rules like length checks and password confirmation. The LoginForm includes fields for email and password. The code uses Flask-WTF and SQLAlchemy models.

```
forms.py - Notepad  
File Edit View  
  
from wtforms import BooleanField, StringField, PasswordField, validators , ValidationError  
from flask_wtf import FlaskForm, Form  
from .models import User  
  
class RegistrationForm(FlaskForm):  
    name = StringField('Name', [validators.Length(min=4, max=25)])  
    username = StringField('Username', [validators.Length(min=4, max=25)])  
    email = StringField('Email Address', [validators.Length(min=6, max=35)])  
    password = PasswordField('Password', [  
        validators.DataRequired(),  
        validators.EqualTo('confirm', message='Passwords must match')  
    ])  
    confirm = PasswordField('Repeat Password')  
  
    def validate_username(self, field):  
        if User.query.filter_by(username=field.data).first():  
            raise ValidationError('Username already in use.')  
  
    def validate_email(self, field):  
        if User.query.filter_by(email=field.data).first():  
            raise ValidationError('Email already registered.')  
  
class LoginForm(FlaskForm):  
    email = StringField('Email Address', [validators.Length(min=6, max=35)])  
    password = PasswordField('New Password', [validators.DataRequired()])
```

Models.py

```
models.py - Notepad
File Edit View
from shop import db
from datetime import datetime

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50),unique=False, nullable=False)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(180),unique=False, nullable=False)
    profile = db.Column(db.String(180), unique=False, nullable=False,default='profile.jpg')

    def __repr__(self):
        return '<User %r>' % self.username

db.create_all()

Ln 1, Col 1 100% Unix (LF) UTF-8
```

Routes.py

```
*routes.py - Notepad
File Edit View
from flask import render_template,session, request,redirect,url_for,flash
from shop import app,db,bcrypt
from .forms import RegistrationForm,LoginForm
from .models import User
from shop.products.models import Addproduct,Category,Brand

@app.route('/admin')
def admin():
    products = Addproduct.query.all()
    return render_template('admin/index.html', title='Admin page',products=products)

@app.route('/brands')
def brands():
    brands = Brand.query.order_by(Brand.id.desc()).all()
    return render_template('admin/brand.html', title='brands',brands=brands)

@app.route('/categories')
def categories():
    categories = Category.query.order_by(Category.id.desc()).all()
    return render_template('admin/brand.html', title='categories',categories=categories)

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        hash_password = bcrypt.generate_password_hash(form.password.data)
        user = User(name=form.name.data,username=form.username.data, email=form.email.data,
                    password=hash_password)
        db.session.add(user)
        flash(f'welcome {form.name.data} Thanks for registering','success')
        db.session.commit()
    return redirect(url_for('login'))
    return render_template('admin/register.html',title='Register user', form=form)

@app.route('/login', methods=['GET','POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            session['user_id'] = user.id
            flash('Login successful', 'success')
            return redirect(url_for('home'))
        else:
            flash('Invalid credentials', 'danger')
```

```

routes.py - Notepad
File Edit View
@app.route('/admin')
def admin():
    products = Addproduct.query.all()
    return render_template('admin/index.html', title='Admin page', products=products)

@app.route('/brands')
def brands():
    brands = Brand.query.order_by(Brand.id.desc()).all()
    return render_template('admin/brand.html', title='brands', brands=brands)

@app.route('/categories')
def categories():
    categories = Category.query.order_by(Category.id.desc()).all()
    return render_template('admin/brand.html', title='categories', categories=categories)

@app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        hash_password = bcrypt.generate_password_hash(form.password.data)
        user = User(name=form.name.data, username=form.username.data, email=form.email.data,
                    password=hash_password)
        db.session.add(user)
        flash(f'welcome {form.name.data} Thanks for registering', 'success')
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('admin/register.html', title='Register user', form=form)

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            session['email'] = form.email.data
            flash(f'welcome {form.email.data} you are logedin now', 'success')
            return redirect(url_for('admin'))
        else:
            flash('wrong email and password', 'success')
            return redirect(url_for('login'))
    return render_template('admin/login.html', title='Login page', form=form)

```

Ln 35, Col 1 100% Unix (LF) UTF-8

Carts:

Carts.py:

```

carts.py - Notepad
File Edit View
from flask import render_template, session, request, redirect, url_for, flash, current_app
from shop import db, app
from shop.products.models import Addproduct
from shop.products.routes import brands, categories
import json

def MagerDicts(dict1, dict2):
    if isinstance(dict1, list) and isinstance(dict2, list):
        return dict1 + dict2
    if isinstance(dict1, dict) and isinstance(dict2, dict):
        return dict(list(dict1.items()) + list(dict2.items()))

@app.route('/addcart', methods=['POST'])
def AddCart():
    try:
        product_id = request.form.get('product_id')
        quantity = int(request.form.get('quantity'))
        color = request.form.get('colors')
        product = Addproduct.query.filter_by(id=product_id).first()

        if request.method == "POST":
            DictItems = (product_id: {'name': product.name, 'price': float(product.price), 'discount': product.discount, 'color': color, 'quantity': quantity, 'image': product.image_1,
            'colors': product.colors})
            if 'Shoppingcart' in session:
                print(session['Shoppingcart'])
                if product_id in session['Shoppingcart']:
                    for key, item in session['Shoppingcart'].items():
                        if int(key) == int(product_id):
                            session.modified = True
                            item['quantity'] += 1
                else:
                    session['Shoppingcart'] = MagerDicts(session['Shoppingcart'], DictItems)
                    return redirect(request.referrer)
            else:
                session['Shoppingcart'] = DictItems
                return redirect(request.referrer)
        except Exception as e:
            print(e)
    finally:

```

Ln 89, Col 1 100% Unix (LF) UTF-8

* carts.py - Notepad

```

File Edit View
@app.route('/carts')
def getCart():
    if 'Shoppingcart' not in session or len(session['Shoppingcart']) <= 0:
        return redirect(url_for('home'))
    subtotal = 0
    grandtotal = 0
    for key,product in session['Shoppingcart'].items():
        discount = (product['discount']/100) * float(product['price'])
        subtotal += float(product['price']) * int(product['quantity'])
        subtotal -= discount
        tax = ("%.2f" % float(subtotal)))
        grandtotal = float("%.2f" % (1.06 * subtotal))
    return render_template('products.carts.html',tax=tax, grandtotal=grandtotal,brands=brands(),categories=categories())
@app.route('/updatecart</int:code>', methods=['POST'])
def updatecart(code):
    if 'Shoppingcart' not in session or len(session['Shoppingcart']) <= 0:
        return redirect(url_for('home'))
    if request.method == "POST":
        quantity = request.form.get('quantity')
        color = request.form.get('color')
        try:
            session.modified = True
            for key , item in session['Shoppingcart'].items():
                if str(key) == code:
                    item['quantity'] = quantity
                    item['color'] = color
                    flash('Item is updated!')
                    return redirect(url_for('getCart'))
        except Exception as e:
            print(e)
            return redirect(url_for('getCart'))
@app.route('/deleteitem</int:id>')
def deleteitem(id):
    if 'Shoppingcart' not in session or len(session['Shoppingcart']) <= 0:
        return redirect(url_for('home'))
    try:
        session.modified = True
        for key , item in session['Shoppingcart'].items():

```

Ln 89, Col 1 100% Unix (LF) UTF-8

```

@app.route('/clearcart')
def clearcart():
    try:
        session.pop('Shoppingcart', None)
        return redirect(url_for('home'))
    except Exception as e:
        print(e)

```

Ln 5, Col 12 100% Unix (LF) UTF-8

Customers:

Forms.py

forms.py - Notepad

```

File Edit View
from wtforms import Form, StringField, TextAreaField, PasswordField, SubmitField, validators, ValidationError
from flask_wtf.file import FileRequired, FileAllowed, FileField
from flask_wtf import FlaskForm
from .model import Register

class CustomerRegisterForm(FlaskForm):
    name = StringField('Name: ')
    username = StringField('Username: ', [validators.DataRequired()])
    email = StringField('Email: ', [validators.Email(), validators.DataRequired()])
    password = PasswordField('Password: ', [validators.DataRequired()], validators.EqualTo('confirm', message=' Both password must match! '))
    confirm = PasswordField('Repeat Password: ', [validators.DataRequired()])
    country = StringField('Country: ', [validators.DataRequired()])
    city = StringField('City: ', [validators.DataRequired()])
    contact = StringField('Contact: ', [validators.DataRequired()])
    address = StringField('Address: ', [validators.DataRequired()])
    zipcode = StringField('Zip code: ', [validators.DataRequired()])
    profile = FileField('Profile', validators=[FileAllowed(['jpg','png','jpeg','gif']), 'Image only please!'])
    submit = SubmitField('Register')

    def validate_username(self, username):
        if Register.query.filter_by(username=username.data).first():
            raise ValidationError("This username is already in use!")

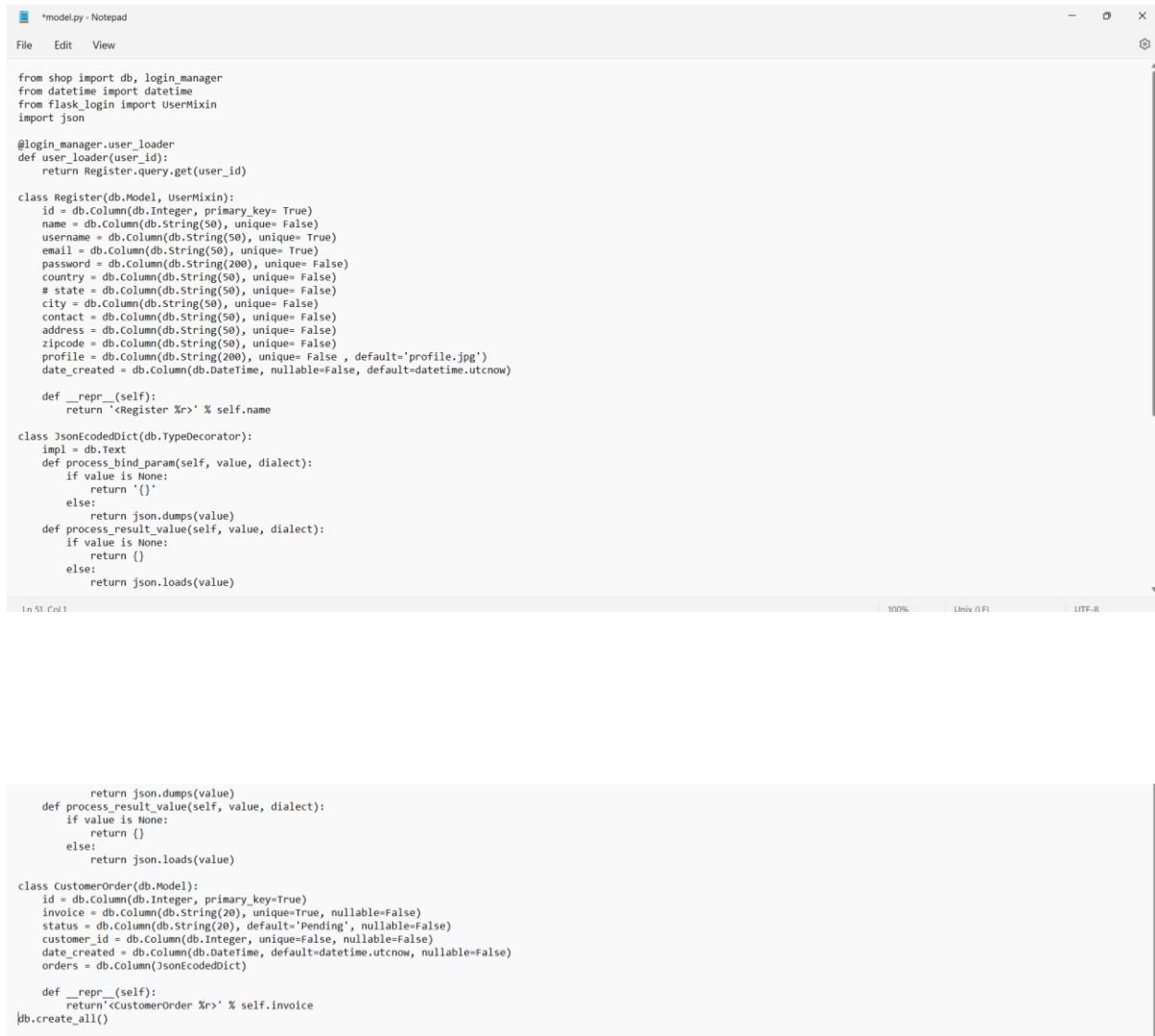
    def validate_email(self, email):
        if Register.query.filter_by(email=email.data).first():
            raise ValidationError("This email address is already in use!")

class CustomerLoginForm(FlaskForm):
    email = StringField('Email: ', [validators.Email(), validators.DataRequired()])
    password = PasswordField('Password: ', [validators.DataRequired()])

```

Ln 1, Col 1 100% Unix (LF) UTF-8

Models.py



```
*model.py - Notepad
File Edit View
from shop import db, login_manager
from datetime import datetime
from flask_login import UserMixin
import json

@login_manager.user_loader
def user_loader(user_id):
    return Register.query.get(user_id)

class Register(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), unique=False)
    username = db.Column(db.String(50), unique=True)
    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(200), unique=False)
    country = db.Column(db.String(50), unique=False)
    state = db.Column(db.String(50), unique=False)
    city = db.Column(db.String(50), unique=False)
    contact = db.Column(db.String(50), unique=False)
    address = db.Column(db.String(50), unique=False)
    zipcode = db.Column(db.String(50), unique=False)
    profile = db.Column(db.String(200), unique=False, default='profile.jpg')
    date_created = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

    def __repr__(self):
        return '<Register %r>' % self.name

class JsonEncodedDict(db.TypeDecorator):
    impl = db.Text
    def process_bind_param(self, value, dialect):
        if value is None:
            return '{}'
        else:
            return json.dumps(value)
    def process_result_value(self, value, dialect):
        if value is None:
            return {}
        else:
            return json.loads(value)

class CustomerOrder(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    invoice = db.Column(db.String(20), unique=True, nullable=False)
    status = db.Column(db.String(20), default='Pending', nullable=False)
    customer_id = db.Column(db.Integer, unique=False, nullable=False)
    date_created = db.Column(db.DateTime, default=datetime.utcnow, nullable=False)
    orders = db.Column(JsonEncodedDict)

    def __repr__(self):
        return '<CustomerOrder %r>' % self.invoice
    db.create_all()
```

Routes.py

```
routes.py - Notepad
File Edit View
from flask import render_template, session, request, redirect, url_for, flash, current_app, make_response
from flask_login import login_required, current_user, logout_user, login_user
from shop import app, db, photos, search, bcrypt, login_manager
from .forms import CustomerRegisterForm, CustomerLoginForm
from .model import Register, CustomerOrder
import secrets
import os
import json
import pdfkit
import stripe
publishable_key = 'pk_test_MaILxTYQ15v5Uh6MKI9wPdD00qgL0QZSL'
stripe.api_key = 'sk_test_9JlhVB6qwjcRdyzjdwgIo0Dt00N55uxBwy'
@app.route('/payment', methods=['POST'])
def payment():
    invoice = request.get('invoice')
    amount = request.form.get('amount')
    customer = stripe.Customer.create(
        email=request.form['stripeEmail'],
        source=request.form['stripeToken'],
    )
    charge = stripe.Charge.create(
        customer=customer.id,
        description='Myshop',
        amount=amount,
        currency='usd',
    )
    orders = CustomerOrder.query.filter_by(customer_id = current_user.id, invoice=invoice).order_by(CustomerOrder.id.desc()).first()
    orders.status = 'Paid'
    db.session.commit()
    return redirect(url_for('thanks'))
@app.route('/thanks')
def thanks():
    return render_template('customer/thank.html')
@app.route('/customer/register', methods=['GET', 'POST'])
def customer_register():
    Ln 1 Col 1
    100% Unix (LF) UTF-8
```

```
routes.py - Notepad
File Edit View
@app.route('/customer/register', methods=['GET', 'POST'])
def customer_register():
    form = CustomerRegisterForm()
    if form.validate_on_submit():
        hash_password = bcrypt.generate_password_hash(form.password.data)
        register = Register(name=form.name.data, username=form.username.data, email=form.email.data, password=hash_password, country=form.country.data, city=form.city.data, contact=form.contact.data, address=form.address.data, zipcode=form.zipcode.data)
        db.session.add(register)
        flash(f'Welcome {form.name.data} Thank you for registering', 'success')
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('customer/register.html', form=form)

@app.route('/customer/login', methods=['GET', 'POST'])
def customer_login():
    form = CustomerLoginForm()
    if form.validate_on_submit():
        user = Register.query.filter_by(email=form.email.data).first()
        if user and bcrypt.check_password_hash(user.password, form.password.data):
            login_user(user)
            flash('You are login now!', 'success')
            next = request.args.get('next')
            return redirect(next or url_for('home'))
            flash('Incorrect email and password', 'danger')
            return redirect(url_for('customerLogin'))
    return render_template('customer/login.html', form=form)

@app.route('/customer/logout')
def customer_logout():
    logout_user()
    return redirect(url_for('home'))

def updateShoppingCart():
    for key, shopping in session['Shoppingcart'].items():
        session.modified = True
        del shopping['image']
        del shopping['colors']
Ln 1 Col 1
    100% Unix (LF) UTF-8
```

```
routes.py - Notepad
File Edit View
del shopping['image']
del shopping['colors']
return updateshoppingcart

@app.route('/getorder')
@login_required
def get_order():
    if current_user.is_authenticated:
        customer_id = current_user.id
        invoice = secrets.token_hex(5)
        updateShoppingCart
        try:
            order = CustomerOrder(invoice=invoice, customer_id=customer_id, orders=session['ShoppingCart'])
            db.session.add(order)
            db.session.commit()
            session.pop('ShoppingCart')
            flash("Your order has been sent successfully", 'success')
            return redirect(url_for('orders', invoice=invoice))
        except Exception as e:
            print(e)
            flash('Something went wrong while getting order', 'danger')
            return redirect(url_for('getCart'))
    else:
        return render_template('customer/login.html')

@app.route('/orders/<invoice>')
@login_required
def orders(invoice):
    if current_user.is_authenticated:
        grandTotal = 0
        subtotal = 0
        customer_id = current_user.id
        customer = Register.query.filter_by(id=customer_id).first()
        orders = CustomerOrder.query.filter_by(customer_id=customer_id, invoice=invoice).order_by(CustomerOrder.id.desc()).first()
        for key, product in orders.orders.items():
            discount = (product['discount']/100) * float(product['price'])
            subtotal += float(product['price']) * int(product['quantity'])
            subtotal -= discount
            tax = ("%.2f" % (.06 * float(subtotal)))
            grandTotal = ("%.2f" % (1.06 * float(subtotal)))
    else:
        return redirect(url_for('customerLogin'))
    return render_template('customer/order.html', invoice=invoice, tax=tax, subTotal=subTotal, grandTotal=grandTotal, customer=customer, orders=orders)

Ln 1, Col 1
100% Unix (LF) UTF-8
```

```
routes.py - Notepad
File Edit View
grandTotal = 0
subtotal = 0
customer_id = current_user.id
customer = Register.query.filter_by(id=customer_id).first()
orders = CustomerOrder.query.filter_by(customer_id=customer_id, invoice=invoice).order_by(CustomerOrder.id.desc()).first()
for key, product in orders.orders.items():
    discount = (product['discount']/100) * float(product['price'])
    subtotal += float(product['price']) * int(product['quantity'])
    subtotal -= discount
    tax = ("%.2f" % (.06 * float(subtotal)))
    grandTotal = ("%.2f" % (1.06 * float(subtotal)))

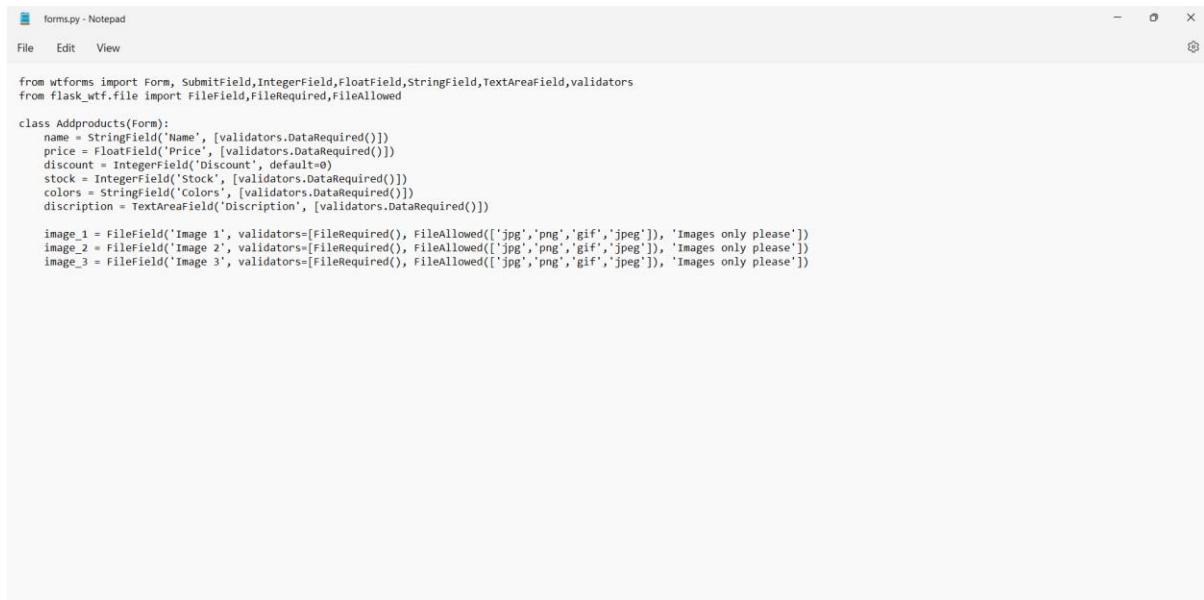
else:
    return redirect(url_for('customerLogin'))
return render_template('customer/order.html', invoice=invoice, tax=tax, subTotal=subTotal, grandTotal=grandTotal, customer=customer, orders=orders)

@app.route('/get_pdf/<invoice>', methods=['POST'])
@login_required
def get_pdf(invoice):
    if current_user.is_authenticated:
        grandTotal = 0
        subtotal = 0
        customer_id = current_user.id
        if request.method == "POST":
            customer = Register.query.filter_by(id=customer_id).first()
            orders = CustomerOrder.query.filter_by(customer_id=customer_id, invoice=invoice).order_by(CustomerOrder.id.desc()).first()
            for key, product in orders.orders.items():
                discount = (product['discount']/100) * float(product['price'])
                subtotal += float(product['price']) * int(product['quantity'])
                subtotal -= discount
                tax = ("%.2f" % (.06 * float(subtotal)))
                grandTotal = float("%.2f" % (1.06 * subtotal))

            rendered = render_template('customer/pdf.html', invoice=invoice, tax=tax, grandTotal=grandTotal, customer=customer, orders=orders)
            pdf = pdfkit.from_string(rendered, False)
            response = make_response(pdf)
            response.headers['Content-Type'] = 'application/pdf'
            response.headers['Content-Disposition'] = 'inline; filename=' + invoice + '.pdf'
            return response
        return request(url_for('orders'))
```

Products:

Forms.py



```
forms.py - Notepad
File Edit View
from wtforms import Form, SubmitField, IntegerField, FloatField, StringField, TextAreaField, validators
from flask_wtf.file import FileField, FileRequired, FileAllowed

class Addproducts(Form):
    name = StringField('Name', [validators.DataRequired()])
    price = FloatField('Price', [validators.DataRequired()])
    discount = IntegerField('Discount', default=0)
    stock = IntegerField('Stock', [validators.DataRequired()])
    colors = StringField('Colors', [validators.DataRequired()])
    description = TextAreaField('Description', [validators.DataRequired()])

    image_1 = FileField('Image 1', validators=[FileRequired(), FileAllowed(['jpg', 'png', 'gif', 'jpeg']), 'Images only please'])
    image_2 = FileField('Image 2', validators=[FileRequired(), FileAllowed(['jpg', 'png', 'gif', 'jpeg']), 'Images only please'])
    image_3 = FileField('Image 3', validators=[FileRequired(), FileAllowed(['jpg', 'png', 'gif', 'jpeg']), 'Images only please'])
```

Models.py



```
*models.py - Notepad
File Edit View
from shop import db
from datetime import datetime
class Addproduct(db.Model):
    __searchable__ = ['name', 'desc']
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
    price = db.Column(db.Numeric(10,2), nullable=False)
    discount = db.Column(db.Integer, default=0)
    stock = db.Column(db.Integer, nullable=False)
    colors = db.Column(db.Text, nullable=False)
    desc = db.Column(db.Text, nullable=False)
    pub_date = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)

    category_id = db.Column(db.Integer, db.ForeignKey('category.id'), nullable=False)
    category = db.relationship('Category', backref=db.backref('categories', lazy=True))

    brand_id = db.Column(db.Integer, db.ForeignKey('brand.id'), nullable=False)
    brand = db.relationship('Brand', backref=db.backref('brands', lazy=True))

    image_1 = db.Column(db.String(150), nullable=False, default='image1.jpg')
    image_2 = db.Column(db.String(150), nullable=False, default='image2.jpg')
    image_3 = db.Column(db.String(150), nullable=False, default='image3.jpg')

    def __repr__(self):
        return '<Post %r>' % self.name

class Brand(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(30), unique=True, nullable=False)

    def __repr__(self):
        return '<Brand %r>' % self.name

class Category(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(30), unique=True, nullable=False)

    def __repr__(self):
        return '<Category %r>' % self.name

db.create_all()
```

Routes.py

```
routes.py - Notepad
File Edit View

from flask import render_template, session, request, redirect, url_for, flash, current_app
from shop import app, db, photos, search
from .models import Category, Brand, Addproduct
from .forms import Addproducts
import secrets
import os

def brands():
    brands = Brand.query.join(Addproduct, (Brand.id == Addproduct.brand_id)).all()
    return brands

def categories():
    categories = Category.query.join(Addproduct, (Category.id == Addproduct.category_id)).all()
    return categories

@app.route('/')
def home():
    page = request.args.get('page', 1, type=int)
    products = Addproduct.query.filter(Addproduct.stock > 0).order_by(Addproduct.id.desc()).paginate(page=page, per_page=8)
    return render_template('products/index.html', products=products, brands=brands(), categories=categories())

@app.route('/result')
def result():
    searchword = request.args.get('q')
    products = Addproduct.query.msearch(searchword, fields=['name', 'desc'], limit=6)
    return render_template('products/result.html', products=products, brands=brands(), categories=categories())

@app.route('/product<int:id>')
def single_page(id):
    product = Addproduct.query.get_or_404(id)
    return render_template('products/single_page.html', product=product, brands=brands(), categories=categories())

@app.route('/brand<int:id>')
def get_brand(id):
    if not brand(id):
        Ln 1, Col 1
        100% Unix (LF) UTF-8
```

```
routes.py - Notepad
File Edit View

@app.route('/brand<int:id>')
def get_brand(id):
    page = request.args.get('page', 1, type=int)
    get_brand = Brand.query.filter_by(id=id).first_or_404()
    brand = Addproduct.query.filter_by(brand=get_brand).paginate(page=page, per_page=8)
    return render_template('products/index.html', brand=brand, brands=brands(), categories=categories(), get_brand=get_brand)

@app.route('/categories<int:id>')
def get_category(id):
    page = request.args.get('page', 1, type=int)
    get_cat = Category.query.filter_by(id=id).first_or_404()
    get_cat_prod = Addproduct.query.filter_by(category=get_cat).paginate(page=page, per_page=8)
    return render_template('products/index.html', get_cat_prod=get_cat_prod, brands=brands(), categories=categories(), get_cat=get_cat)

@app.route('/addbrand', methods=['GET', 'POST'])
def addbrand():
    if request.method == "POST":
        getbrand = request.form.get('brand')
        brand = Brand(name=getbrand)
        db.session.add(brand)
        flash(f'The brand {getbrand} was added to your database', 'success')
        db.session.commit()
    return render_template('products/addbrand.html', title='Add brand', brands=brands())

@app.route('/updatebrand<int:id>', methods=['GET', 'POST'])
def updatebrand(id):
    if 'email' not in session:
        flash('Login first please', 'danger')
        return redirect(url_for('login'))
    updatebrand = Brand.query.get_or_404(id)
    brand = request.form.get('brand')
    if request.method == "POST":
        updatebrand.name = brand
        flash(f'The brand {updatebrand.name} was changed to {brand}', 'success')
        db.session.commit()
    Ln 1, Col 1
    100% Unix (LF) UTF-8
```



The screenshot shows a Notepad window titled "routes.py - Notepad". The code is Python code for a Flask application. It includes routes for adding, updating, and deleting categories. The code uses SQLAlchemy's session to interact with a database named "test.db". It also uses the "flash" function to show success or error messages to the user.

```
routes.py - Notepad
File Edit View
@app.route('/addcat',methods=['GET','POST'])
def addcat():
    if request.method == "POST":
        getcat = request.form.get('category')
        category = Category(name=getcat)
        db.session.add(category)
        flash(f'The brand {getcat} was added to your database','success')
        db.session.commit()
        return redirect(url_for('addcat'))
    return render_template('products/addbrand.html', title='Add category')

@app.route('/updatecat/<int:id>',methods=['GET','POST'])
def updatecat(id):
    if 'email' not in session:
        flash('Login first please','danger')
        return redirect(url_for('login'))
    updatecat = Category.query.get_or_404(id)
    category = request.form.get('category')
    if request.method == "POST":
        updatecat.name = category
        flash(f'The category {updatecat.name} was changed to {category}','success')
        db.session.commit()
        return redirect(url_for('categories'))
    category = updatecat.name
    return render_template('products/addbrand.html', title='Update cat',updatecat=updatecat)

@app.route('/deletecat/<int:id>', methods=['GET','POST'])
def deletecat(id):
    category = Category.query.get_or_404(id)
    if request.method=="POST":
        db.session.delete(category)
        flash(f'The brand {category.name} was deleted from your database',"success")
        db.session.commit()
        return redirect(url_for('admin'))
    flash(f'The brand {category.name} can't be deleted from your database',"warning")
    return redirect(url_for('admin'))

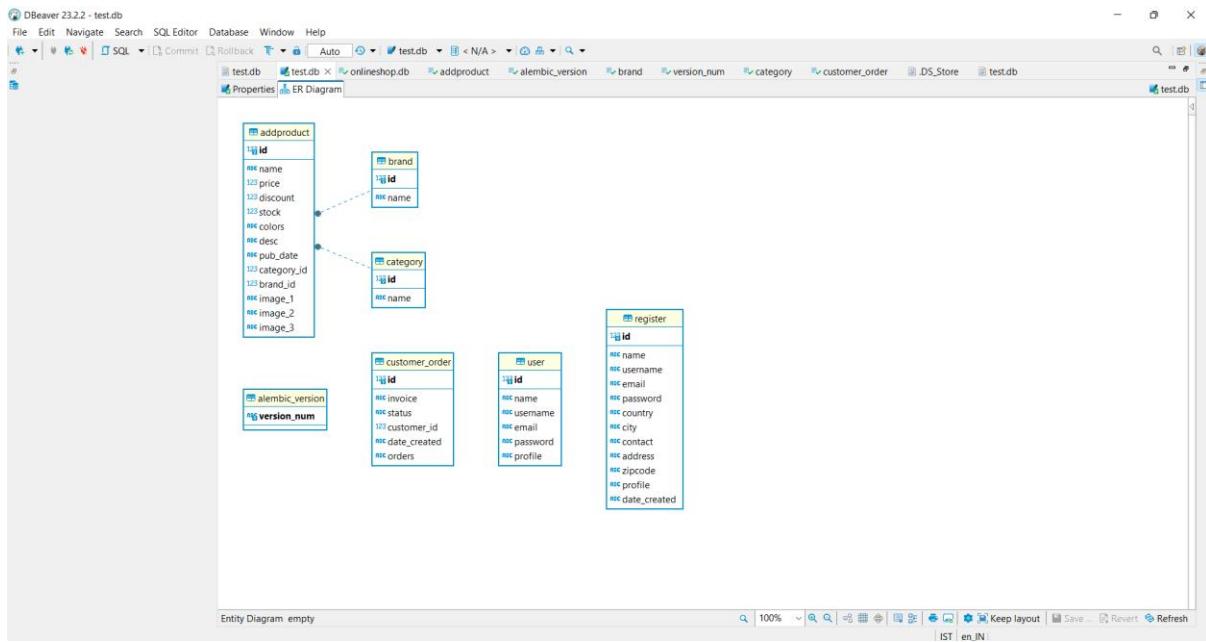
Ln 1, Col 1 100% Unix (LF) UTF-8
```

These are the coding in the shop folder.

Database:

As said, we have used SQLite for database which we have named test.db.

The entity relationship diagram is shown below



The various databases handled are shown below:

Database for products:

DBeaver 23.2.2 - addproduct

File Edit Navigate Search SQL Editor Database Window Help

test.db test.db onlineshop.db addproduct alembic_version brand version_num category customer_order .DS_Store test.db

Properties Data ER Diagram

Grid Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	price	discount	stock	colors	desc	pub_date	category_id	brand_id	image_1	image_2
1	Modem pot	1,200.99	10	15	gray,white,black,pink	100% handmade product with quality and Assurance	2023-09-16 19:48:54.706965	2	492392ca1d80c65f5fc3.jpg	2	fb152ad7428856f61a92.jpg	942930069478205ca.jpeg
2	tote bag	1,150.99	0	20	gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	1	1b6b45a578914e011de8.jpg	1	b7f02a703d918008306.jpg	fcd9f839069478205ca.jpeg
3	Handmade jewel	1,950.99	10	20	gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	3	1a8f02a703d918008306.jpg	2	87fcbab43ca157a1407.jpg	87fcbab43ca157a1407.jpg
4	Armchair	880.5	15	20	gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	2	040f3c7cb7d97ac4ee5e.jpeg	2	040f3c7cb7d97ac4ee5e.jpeg	040f3c7cb7d97ac4ee5e.jpeg
5	Wooden toys	980.5	0	20	pink,gray,white,black,blue	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	4	87fcbab43ca157a1407.jpg	3	87fcbab43ca157a1407.jpg	87fcbab43ca157a1407.jpg
6	Bath bomb	780.5	10	20	blue,gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	1	040f3c7cb7d97ac4ee5e.jpeg	2	040f3c7cb7d97ac4ee5e.jpeg	040f3c7cb7d97ac4ee5e.jpeg
7	Utensil holder	780.5	5	20	blue,gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	2	fb152ad7428856f61a92.jpg	3	c8cd10e2fd79892d1f111.jpg	fb152ad7428856f61a92.jpg
8	Wire craft plant	780.5	5	20	blue,gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	2	3d	3d	b93a85f9a61f2a7856d8.png	b93a85f9a61f2a7856d8.png
9	Earthen pot	780.5	15	25	blue,gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	1	5d	5d	b659045271d0ce49390.jpe	b659045271d0ce49390.jpe
10	leather bag	1,200.99	0	25	golden,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	7	7d	7d	dc900c380ba7f227ef0.png	dc900c380ba7f227ef0.png
11	Candle	560.89	0	25	gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	6	7d	7d	09788fcb99480faa7.png	09788fcb99480faa7.png
12	Scarf	450.89	10	10	blue,gray,white,black	100% handmade product with quality and Assurance	2023-09-16 20:09:14.716623	7	1d	1d	09788fcb99480faa7.png	09788fcb99480faa7.png

Record

Database for brands:

The screenshot shows the DBeaver interface with the database 'brand' selected. The left sidebar displays the schema tree for the 'brand' table, including columns like id, name, price, discount, stock, desc, colors, and pub_date. The main pane shows a grid of data for the 'brand' table, with 7 rows and 2 columns. The data is as follows:

ID	Name
1	KK Brand
2	Sony
3	New Brand
4	HP
5	M Brand
6	samsung
7	#Brand

Database for categories:

The screenshot shows the DBeaver interface with the database 'category' selected. The left sidebar displays the schema tree for the 'category' table, including columns like desc, pub_date, category_id, brand_id, and image fields. The main pane shows a grid of data for the 'category' table, with 7 rows and 2 columns. The data is as follows:

ID	Name
1	Toys
2	Pots
3	Minatures
4	Leather
5	Jewels
6	Bath
7	Legs

Database for customer orders:

The screenshot shows the DBBeaver interface with the 'customer_order' table selected. The table has columns: id, invoice, status, customer_id, date_created, and orders. The data grid contains 8 rows of order information, each with a unique ID, invoice number, status (Pending or Paid), customer ID, creation date, and a JSON string for the order details.

	id	invoice	status	customer_id	date_created	orders
1	5cb3282fd9	Pending	1	2020-05-05 19:08:39.240926	"10": {"color": "golden", "colors": "golden:white:black", "discount": 0, "image": "b85904527fd0ceae93}	
2	a6e8a7bf3b	Pending	2	2020-05-06 11:31:17.614707	"10": {"color": "golden", "colors": "golden:white:black", "discount": 0, "image": "b85904527fd0ceae93}	
3	2cd1da95380	Pending	2	2020-05-06 11:31:27.139262	"11": {"color": "gray", "colors": "gray:white:black", "discount": 0, "image": "dc900c380ba7f2227ef0pr}	
4	09fefc3d3c	Pending	2	2020-05-06 11:36:27.669339	"10": {"color": "golden", "colors": "golden:white:black", "discount": 0, "image": "b85904527fd0ceae93}	
5	9b6b119c3d5	Pending	2	2020-05-06 11:39:05.303821	"10": {"color": "golden", "colors": "golden:white:black", "discount": 0, "image": "b85904527fd0ceae93}	
6	31b572e7ef	Pending	2	2020-05-11 18:06:29.579892	"10": {"color": "golden", "colors": "golden:white:black", "discount": 0, "image": "b85904527fd0ceae93}	
7	ebdbbf601b	Paid	2	2020-05-11 18:42:51.527684	"10": {"color": "golden", "discount": 0, "name": "Rado watch", "price": 1200.99, "quantity": 1}, "12": {"12": {"color": "blue", "discount": 10, "name": "Apple watch", "price": 450.89, "quantity": 1}}	
8	347bcdf796	Paid	2	2020-05-11 18:56:24.106484	"12": {"color": "blue", "discount": 10, "name": "Apple watch", "price": 450.89, "quantity": 1}}	

Database for registration and login:

The screenshot shows the DBBeaver interface with the 'register' table selected. The table has columns: id, name, username, email, password, country, city, contact, address, zipcode, profile, and date_created. The data grid contains 2 rows of user registration information, including names, usernames, emails, hashed passwords, and location details.

	id	name	username	email	password	country	city	contact	address	zipcode	profile	date_created
1	1	HAMA	BUGTI	arha123@gmail.co	\$2b\$12\$/hdxLPfEKKSyyy/BO42ekCPmJAvit67auWV9PR5	India	Chennai	2345325	new street	3468	profile.jpg	2020-04-04 15:29:17.876467
2	2	LALA	HANA	lala@gmail.com	\$2b\$12\$/7pk/28Lzg4hbG8rOmloTuYk4UhCtmJlt63CA	India	Trichy	2345325	old street	3468	profile.jpg	2020-04-04 16:49:08.912593

	<code>id</code>	<code>name</code>	<code>username</code>	<code>email</code>	<code>password</code>	<code>profile</code>
1	1	Lala	lala	lala@gmail.com	\$2b\$12\$HtMMinn6Z0rLBcngqvIUoEKGjsTQOY6ujKPLNtfreDoWkKSHm	profile.jpg
2	2	Hana	lala	hana@gmail.com	\$2b\$12\$6vJ7vr29OWbWnD/YdA2njT8jLyf8pM6ZbiPNRWHjbfHHS	profile.jpg
3	3	Arha	Arha123	arha123@gmail.com	\$2b\$12\$1TBBC2MUHOVPCy1DUUJGU8LYYg/vzIq8raJwbeCWGf5Puuf7e	profile.jpg
4	4	Hana	bugti	hanabugti@gmail.com	\$2b\$12\$sdubuBajeCkB6XESLIVR9uaSxDhtbRhUomWLsy3cGo7jKnlyGm	profile.jpg

Cloud deployment:

The flask app developed can be deployed in cloud by following these steps:

1. Build a app by using Python Flask.
2. Build and test a Docker container (Postman is an alternative to Docker).
3. Push the container to the IBM Cloud Registry.
4. Deploy and manage the application by using Code Engine.
5. Be enabled for OpenAPI 3.0.

This is the entire final design, development, and deployment of our simple and user friendly E-Commerce app ARHA.