Mini Project: Shopping Cart System

Brendan Badhe [230962061]

Talluri Srilekha [230962056]

Manipal Institute of Technology, MAHE

CSE 2241: Database Systems Lab

Dr. U. Dinesh Acharya

April 2025

Abstract

This project implements a comprehensive Shopping Cart System using a relational database and a web application interface. The system is designed to simulate an e-commerce platform, allowing users to browse products, manage shopping carts, and place orders. The backend is powered by Oracle Database, with tables and procedures handling user authentication, product inventory, shopping cart operations, and order management. The web application, built with Flask, provides a user-friendly interface for functionalities such as user registration, login, product browsing, cart management, and order placement. Key features include dynamic stock updates, order history tracking, and secure session management. This project demonstrates the integration of database design principles, SQL procedures, and web development to create a functional and scalable e-commerce solution.

Problem Statement

Design and implement a shopping cart system that lets shoppers collect items into a shopping cart and purchase together. You should check for the availability of the item and deal with unavailable items as you feel appropriate. The solution must address both data requirements and functional requirements as outlined below:

Data Requirements

1. User Management

- Maintain user information, including unique usernames, email addresses, passwords, and account creation timestamps.
- Ensure secure storage and retrieval of user credentials.

2. Product Catalog

- Store product details such as name, description, price, stock quantity, category, and brand.
- Support categorization of products into predefined categories (e.g., Electronics, Clothing, Books) and brands.

3. Shopping Cart

- Maintain a shopping cart for each user, storing the items added by the user along with their quantities.
- Ensure that the cart reflects real-time stock availability.

4. Order Management

- Record order details, including the user who placed the order, the shipping address, and the order date.
- Track individual items in each order, including product details, quantity, and price.

5. Database Integrity

- Enforce referential integrity between tables (e.g., users, products, orders, and shopping carts).
- Use constraints to ensure data validity (e.g., non-negative stock quantities, valid prices).

Functional Requirements

6. User Authentication

- Allow users to sign up with unique usernames and email addresses.
- Provide a secure login mechanism to authenticate users.

7. Product Browsing

- Display a list of available products with details such as name, description, price, and stock status.
- Indicate whether a product is out of stock or already in the user's cart.

8. Shopping Cart Operations

- Enable users to add products to their cart, increment, or decrement quantities, and remove items.
- Prevent users from adding more items than are available in stock.
- Dynamically update stock levels when items are added or removed from cart.

9. Checkout and Order Placement

- Allow users to review their cart and provide a shipping address during checkout.
- Validate the availability of items before finalizing the order.
- Deduct purchased quantities from the product stock and clear the user's cart after order placement.

10. Order History

- Provide users with a history of their past orders, including order details, timestamps, and total amounts.

- Allow users to review detailed information about individual orders.

11. Error Handling

- Manage scenarios such as invalid login credentials, mismatched passwords during signup, and attempts to add out-of-stock items to the cart.
- Display appropriate error messages to guide users.

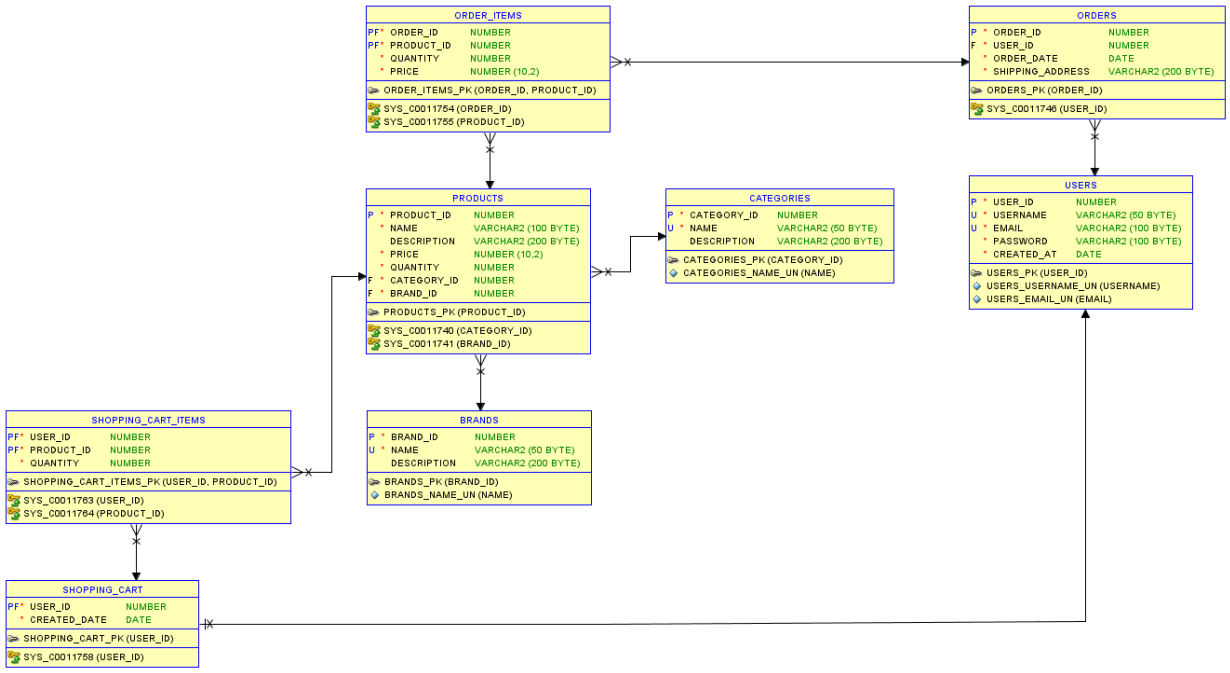
12. Scalability and Maintainability

- Ensure the system can manage multiple users and large product catalogs efficiently.
- Use modular database design and reusable stored procedures for core operations like adding to the cart, placing orders, and calculating totals.

ER Diagram & Relational Tables

ER Diagram

Fig: Entity – Relationship Diagram



Relational Tables

Normal Form Analysis

First Normal Form (1NF)

- Each table has a primary key.
- All columns contain atomic values (no repeating groups or arrays).
- All attributes depend on the primary key.

Second Normal Form (2NF)

- It meets 1NF requirements.
- All non-key attributes are fully functionally dependent on the primary key.

Third Normal Form (3NF)

- It meets 2NF requirements.
- There are no transitive dependencies where non-key attributes depend on other non-key attributes.

Boyce-Codd Normal Form (BCNF)

- It meets 3NF requirements.
- For every functional dependency X -> Y, X is a super key.

Fourth Normal Form (4NF)

- It meets BCNF requirements.
- No multi-valued dependencies.

Fifth Normal Form (5NF)

- It meets 4NF requirements.
- All join dependencies are consequences of candidate keys.

The database is normalized to 5NF, which is sufficient for most practical applications, the industry standard minimum requirement is 3NF. It is well-designed with no redundancy, no partial dependencies, and no multi-valued dependencies. It is efficient and adheres to best practices for relational database design.

Tables

BRANDS

BRAND_ID	NAME	DESCRIPTION
1	Apple	Premium tech brand
2	Nike	Sports and fashion brand
3	Samsung	Electronics and appliances
4	Penguin Random House	Leading book publisher

CATEGORIES

CATEGORY_ID	NAME	DESCRIPTION
1	Electronics	Devices and gadgets
2	Clothing	Apparel and accessories
3	Books	Educational and fictional books

ORDERS

ORDER_ID	USER_ID	ORDER_DATE	SHIPPING_ADDRESS
1	1	16-04-2025	123 Main St, New York, NY
2	2	16-04-2025	456 Elm St, Los Angeles, CA
3	3	16-04-2025	789 Oak St, Chicago, IL

ORDER_ITEMS

ORDER_ID	PRODUCT_ID	QUANTITY	PRICE
1	1	1	999.99
1	3	2	120.5
2	2	1	899.99
3	5	3	49.99

PRODUCTS

PRODUCT_ID	NAME	DESCRIPTION	PRICE	QUANTITY	CATEGORY_ID	BRAND_ID
1	iPhone 15	Latest Apple smartphone	999.99	50	1	1
2	Galaxy S23	Samsung flagship phone	899.99	40	1	3
3	Nike Running Shoes	Comfortable sports shoes	120.5	100	2	2
4	MacBook Pro	Apple laptop with M3 chip	1999	30	1	1
5	Python Programming	Guide to Python development	49.99	200	3	4

SHOPPING_CART

USER_ID	CREATED_DATE
1	16-04-2025
2	16-04-2025
3	16-04-2025

SHOPPING_CART_ITEMS

USER_ID	PRODUCT_ID	QUANTITY
1	4	1
1	2	1
2	3	2
3	5	1

USERS

USER_ID	USERNAME	EMAIL	PASSWORD	CREATED_AT
1	john_doe	john@example.com	password1	16-04-2025
2	jane_smith	jane@example.com	password2	16-04-2025
3	alex_jones	alex@example.com	password3	16-04-2025

DDL Commands

Users

```
create table users (
   user_id    number primary key,
   username   varchar2(50) not null unique,
   email      varchar2(100) not null unique,
   password   varchar2(100) not null,
   created_at date default sysdate not null
);
```

Categories

```
create table categories (
   category_id number primary key,
   name        varchar2(50) not null unique,
   description varchar2(200)
);
```

Brands

```
create table brands (
   brand_id    number primary key,
   name        varchar2(50) not null unique,
   description varchar2(200)
);
```

Products

```
create table products (
   product_id  number primary key,
   name        varchar2(100) not null,
   description varchar2(200),
   price       number(10,2) not null check ( price > 0 ),
   quantity    number not null check ( quantity >= 0 ),
   category_id number not null,
   brand_id    number not null,
   foreign key ( category_id )
      references categories ( category_id )
         on delete cascade,
   foreign key ( brand_id )
      references brands ( brand_id )
         on delete cascade
);
```

Orders

```
create table orders (
   order_id         number primary key,
   user_id          number not null,
   order_date       date default sysdate not null,
   shipping_address varchar2(200) not null,
   foreign key ( user_id )
      references users ( user_id )
         on delete cascade
);
```

Order_Items

```
create table order_items (
   order_id   number not null,
   product_id number not null,
   quantity   number not null check ( quantity > 0 ),
   price      number(10,2) not null check ( price > 0 ),
   primary key ( order_id,
                 product_id ),
   foreign key ( order_id )
      references orders ( order_id )
         on delete cascade,
   foreign key ( product_id )
      references products ( product_id )
         on delete cascade
);
```

Shopping_Cart

```
create table shopping_cart (
   user_id      number primary key,
   created_date date default sysdate not null,
   foreign key ( user_id )
      references users ( user_id )
         on delete cascade
);
```

Shopping_Cart_Items

```
create table shopping_cart_items (
   user_id    number not null,
   product_id number not null,
   quantity   number not null,
   primary key ( user_id,
                 product_id ),
   foreign key ( user_id )
      references shopping_cart ( user_id )
         on delete cascade,
   foreign key ( product_id )
      references products ( product_id )
         on delete cascade
);
```

Sequences

```
create sequence users_seq start with 1 increment by 1 nocache;
create sequence categories_seq start with 1 increment by 1 nocache;
create sequence brands_seq start with 1 increment by 1 nocache;
create sequence products_seq start with 1 increment by 1 nocache;
create sequence orders_seq start with 1 increment by 1 nocache;
```

List of SQL Queries

Populating the database

```sql
insert into users (
   user_id,
   username,
   email,
   password
) values ( users_seq.nextval,
           'john_doe',
           'john@example.com',
           'password1' );

insert into users (
   user_id,
   username,
   email,
   password
) values ( users_seq.nextval,
           'jane_smith',
           'jane@example.com',
           'password2' );

insert into users (
   user_id,
   username,
   email,
   password
) values ( users_seq.nextval,
           'alex_jones',
           'alex@example.com',
           'password3' );

insert into categories (
   category_id,
   name,
   description
) values ( categories_seq.nextval,
           'Electronics',
           'Devices and gadgets' );

insert into categories (
   category_id,
   name,
```

```sql
   description
) values ( categories_seq.nextval,
           'Clothing',
           'Apparel and accessories' );

insert into categories (
   category_id,
   name,
   description
) values ( categories_seq.nextval,
           'Books',
           'Educational and fictional books' );

insert into brands (
   brand_id,
   name,
   description
) values ( brands_seq.nextval,
           'Apple',
           'Premium tech brand' );

insert into brands (
   brand_id,
   name,
   description
) values ( brands_seq.nextval,
           'Nike',
           'Sports and fashion brand' );

insert into brands (
   brand_id,
   name,
   description
) values ( brands_seq.nextval,
           'Samsung',
           'Electronics and appliances' );

insert into brands (
   brand_id,
   name,
   description
) values ( brands_seq.nextval,
           'Penguin Random House',
           'Leading book publisher' );
```

```sql
insert into products (
   product_id,
   name,
   description,
   price,
   quantity,
   category_id,
   brand_id
) values ( products_seq.nextval,
           'iPhone 15',
           'Latest Apple smartphone',
           999.99,
           50,
           1,
           1 );

insert into products (
   product_id,
   name,
   description,
   price,
   quantity,
   category_id,
   brand_id
) values ( products_seq.nextval,
           'Galaxy S23',
           'Samsung flagship phone',
           899.99,
           40,
           1,
           3 );

insert into products (
   product_id,
   name,
   description,
   price,
   quantity,
   category_id,
   brand_id
) values ( products_seq.nextval,
           'Nike Running Shoes',
           'Comfortable sports shoes',
           120.50,
           100,
```

```
            2,
            2 );

insert into products (
   product_id,
   name,
   description,
   price,
   quantity,
   category_id,
   brand_id
) values ( products_seq.nextval,
            'MacBook Pro',
            'Apple laptop with M3 chip',
            1999.00,
            30,
            1,
            1 );

insert into products (
   product_id,
   name,
   description,
   price,
   quantity,
   category_id,
   brand_id
) values ( products_seq.nextval,
            'Python Programming',
            'Guide to Python development',
            49.99,
            200,
            3,
            4 );

insert into orders (
   order_id,
   user_id,
   shipping_address
) values ( orders_seq.nextval,
            1,
            '123 Main St, New York, NY' );

insert into orders (
   order_id,
```

```sql
   user_id,
   shipping_address
) values ( orders_seq.nextval,
           2,
           '456 Elm St, Los Angeles, CA' );

insert into orders (
   order_id,
   user_id,
   shipping_address
) values ( orders_seq.nextval,
           3,
           '789 Oak St, Chicago, IL' );

insert into order_items (
   order_id,
   product_id,
   quantity,
   price
) values ( 1,
           1,
           1,
           999.99 );

insert into order_items (
   order_id,
   product_id,
   quantity,
   price
) values ( 1,
           3,
           2,
           120.50 );

insert into order_items (
   order_id,
   product_id,
   quantity,
   price
) values ( 2,
           2,
           1,
           899.99 );

insert into order_items (
```

```sql
   order_id,
   product_id,
   quantity,
   price
) values ( 3,
           5,
           3,
           49.99 );

insert into shopping_cart ( user_id ) values ( 1 );
insert into shopping_cart ( user_id ) values ( 2 );
insert into shopping_cart ( user_id ) values ( 3 );

insert into shopping_cart_items (
   user_id,
   product_id,
   quantity
) values ( 1,
           4,
           1 );

insert into shopping_cart_items (
   user_id,
   product_id,
   quantity
) values ( 1,
           2,
           1 );

insert into shopping_cart_items (
   user_id,
   product_id,
   quantity
) values ( 2,
           3,
           2 );

insert into shopping_cart_items (
   user_id,
   product_id,
   quantity
) values ( 3,
           5,
           1 );
```

User Authentication

```
select user_id
  from users
 where username = :1
   and password = :2
```

User Registration

```
   insert into users (
      user_id,
      username,
      email,
      password
   ) values ( users_seq.nextval,
              :1,
              :2,
              :3 )
```

Retrieve all products

```
select product_id,
       name,
       description,
       price,
       quantity
  from products
```

Check if product is in user's cart

```
select product_id
  from shopping_cart_items
 where user_id = :1
```

Retrieve items from user's cart

```
select p.product_id,
       p.name,
       p.description,
       p.price,
       sci.quantity,
       p.quantity
  from shopping_cart_items sci
  join products p
on sci.product_id = p.product_id
 where sci.user_id = :1
```

Retrieve order history of user sorted by latest first

```
select o.order_id,
       o.order_date,
       o.shipping_address,
       sum(oi.quantity * oi.price) as total
  from orders o
  join order_items oi
on o.order_id = oi.order_id
 where o.user_id = :1
 group by o.order_id,
          o.order_date,
          o.shipping_address
 order by o.order_date desc
```

Retrieve details of a specific order

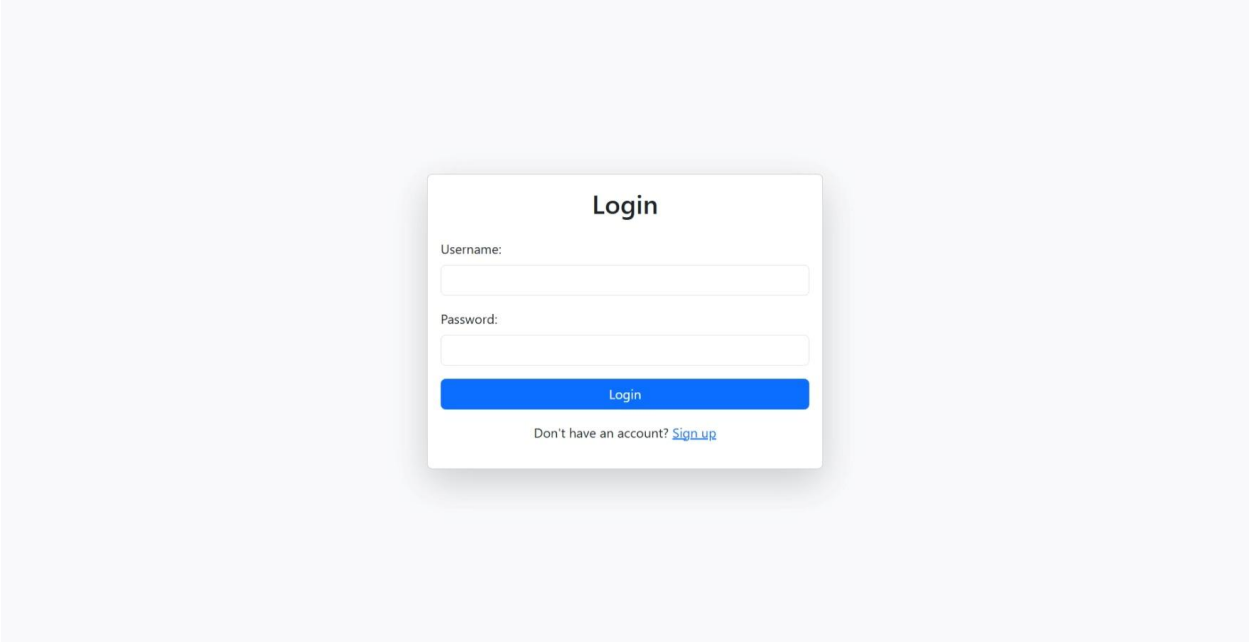
```
select p.name,
       p.description,
       oi.price,
       oi.quantity,
       ( oi.price * oi.quantity ) as total
  from order_items oi
  join products p
on oi.product_id = p.product_id
 where oi.order_id = :1
   and exists (
   select 1
     from orders o
    where o.order_id = :2
      and o.user_id = :3
)
```

Note: Additional SQL Queries are a part of PL/SQL blocks used.

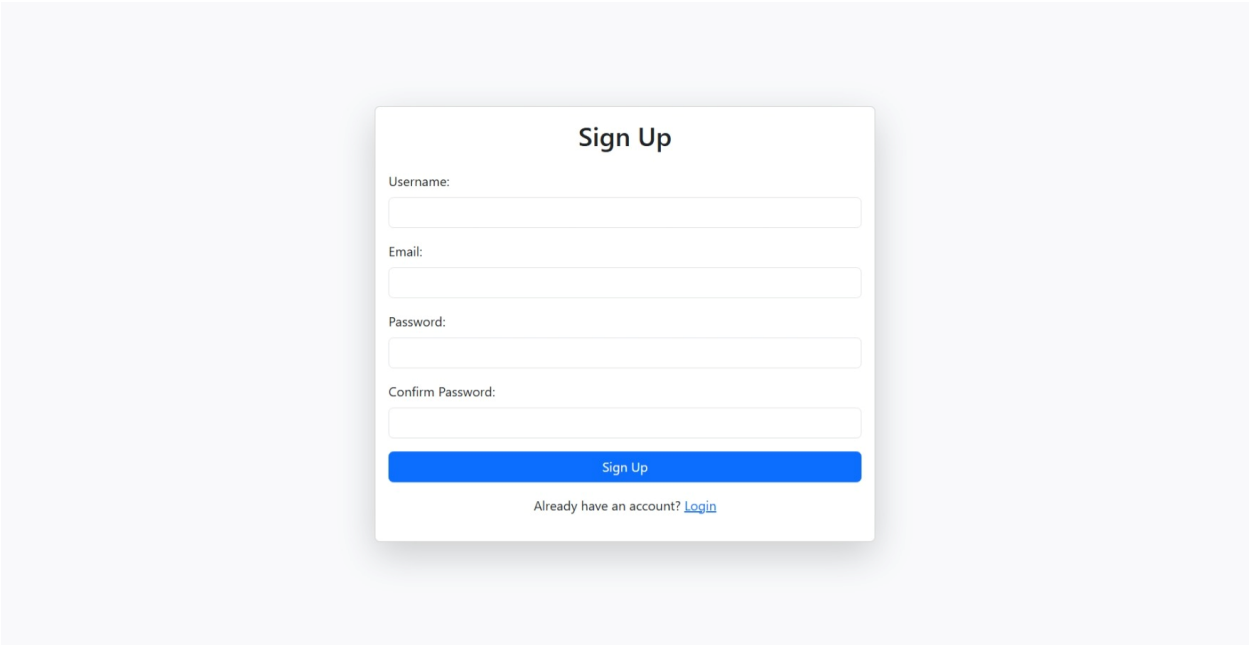
UI Design

To build a dynamic, responsive and user-friendly interface, HTML is used to define the structure of the page, Jinja2 templating is used to reflect dynamic updates and changes in the database by interacting with the Flask back-end, and Bootstrap is used for styling and making the website responsive. The application is secure and well designed, it incorporates input validation in the front-end as well as the back end, there is a system of safety-checks throughout the application to ensure data consistency and positive user interactions. The website's responsive UI dynamically scales to accommodate usage on all types of devices. It is a server-side application, where all the logic is managed through the Flask back-end that interacts with the database. The front-end is only for user interaction, this separation enhances the security of the application and further highlights the database since changes are reflected only if changes have occurred within the database.

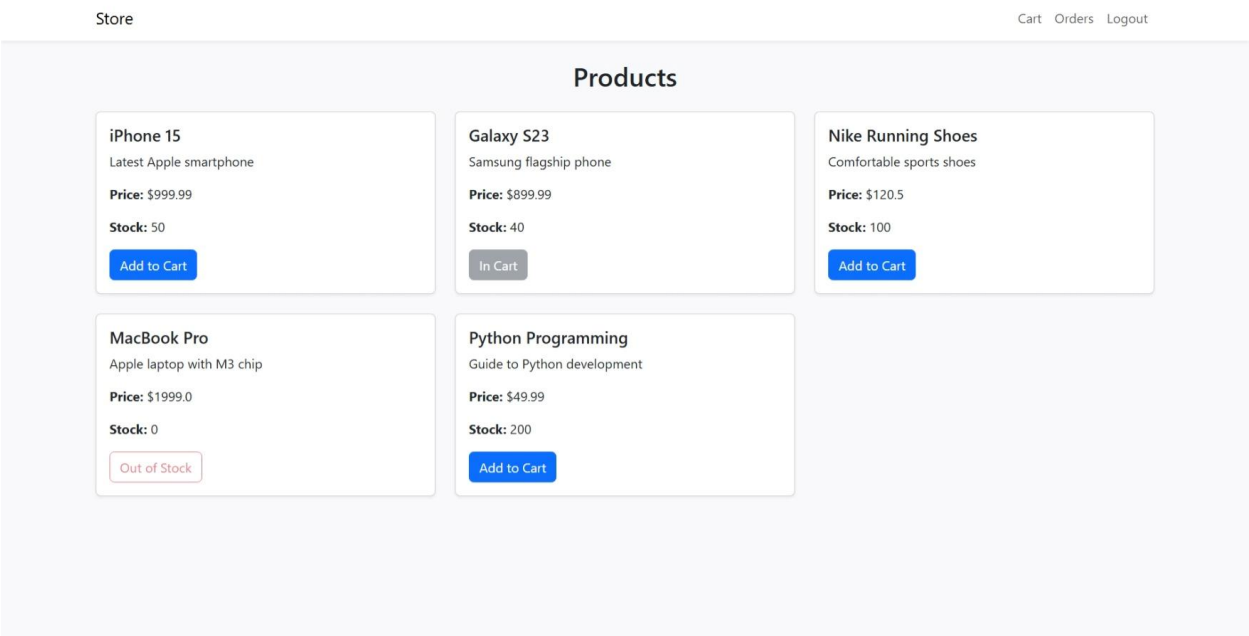
Login



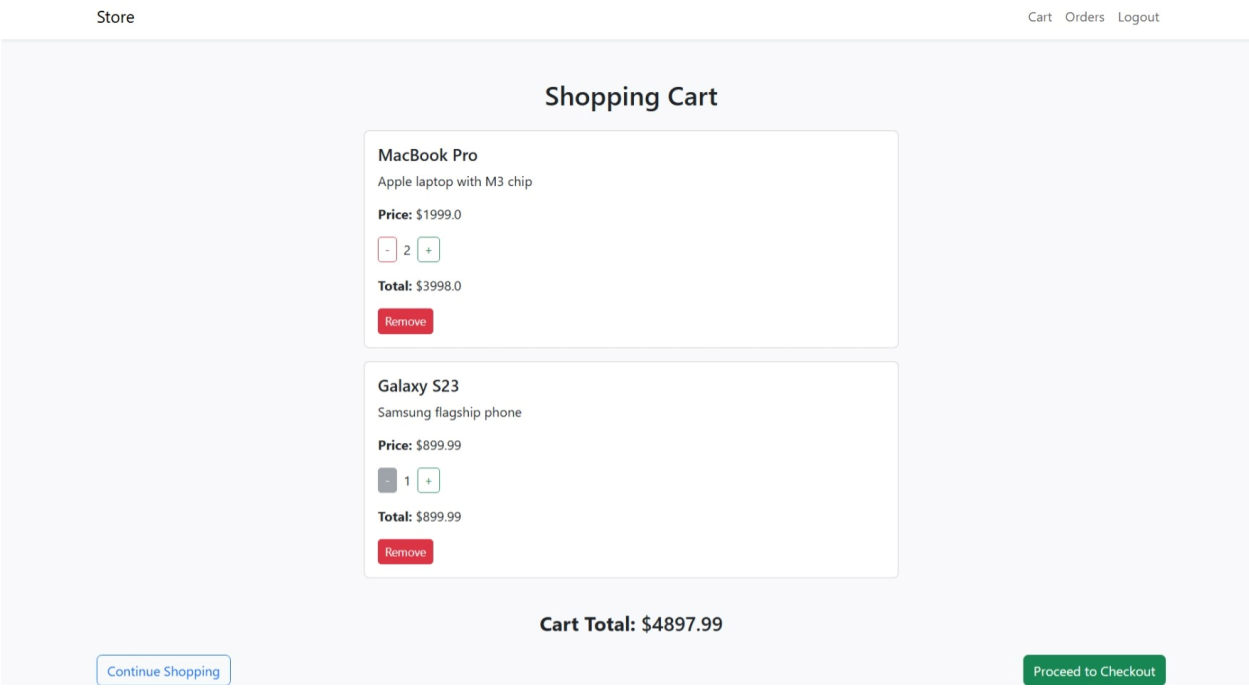
Sign Up



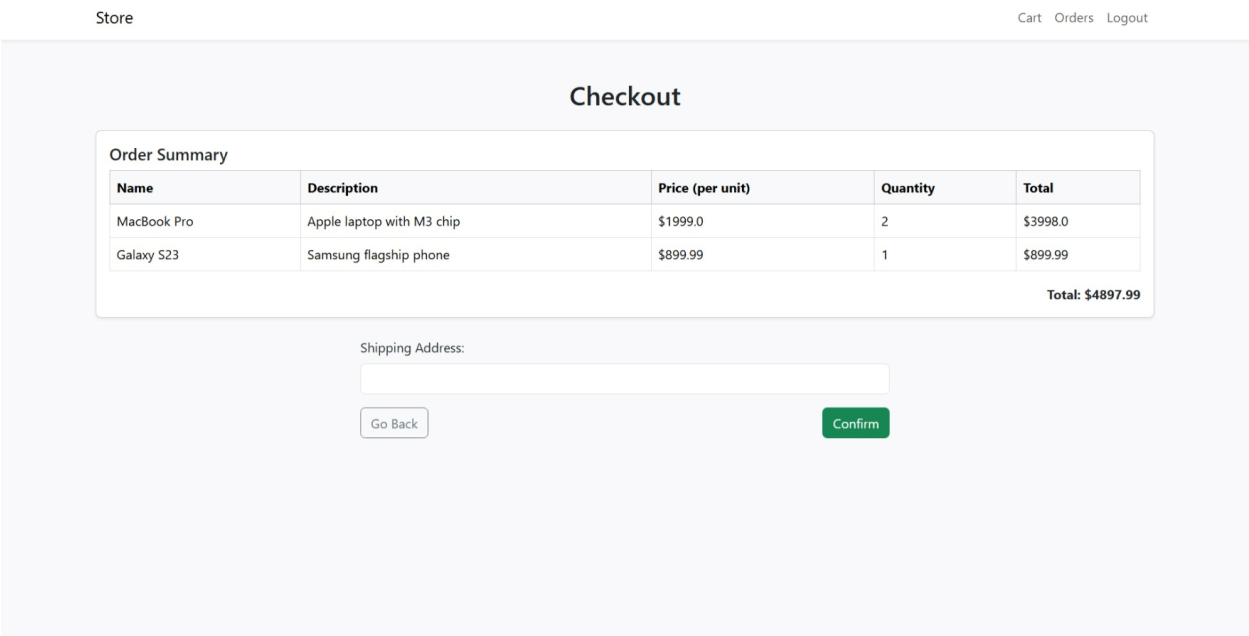
Products



Cart



Checkout



Orders

Store

Cart Orders Logout

Orders

Order History

ID	Timestamp	Shipping Address	Total	Details
1	2025-04-15 15:58:34	123 Main St, New York, NY	$1240.99	View Details

Continue Shopping

Order Details

Store

Cart Orders Logout

Order Details

Order Summary #1

Name	Description	Price (per unit)	Quantity	Total
iPhone 15	Latest Apple smartphone	$999.99	1	$999.99
Nike Running Shoes	Comfortable sports shoes	$120.5	2	$241

Total: $1240.99

Go Back

PL/SQL

Triggers

Create a cart for a user who has successfully signed up

```
create or replace trigger create_cart_after_user after
   insert on users
   for each row
begin
   insert into shopping_cart ( user_id ) values ( :new.user_id );
end;
/
```

Procedures

Add item to cart

```
create or replace procedure add_to_cart (
   p_user_id    in number,
   p_product_id in number
) as
begin
   insert into shopping_cart_items (
      user_id,
      product_id,
      quantity
   ) values ( p_user_id,
              p_product_id,
              1 );

   update products
      set
      quantity = quantity - 1
    where product_id = p_product_id
      and quantity > 0;
end;
/
```

Increment quantity of item in cart

```
create or replace procedure increment_quantity (
   p_user_id    in number,
   p_product_id in number
) as
begin
   update shopping_cart_items
      set
      quantity = quantity + 1
    where user_id = p_user_id
      and product_id = p_product_id;

   update products
      set
      quantity = quantity - 1
    where product_id = p_product_id
      and quantity > 0;
end;
/
```

Decrement quantity of item in cart

```
create or replace procedure decrement_quantity (
   p_user_id    in number,
   p_product_id in number
) as
begin
   update shopping_cart_items
      set
      quantity = quantity - 1
    where user_id = p_user_id
      and product_id = p_product_id
      and quantity > 1;

   update products
      set
      quantity = quantity + 1
    where product_id = p_product_id;
end;
/
```

Remove item from cart

```
create or replace procedure remove_from_cart (
   p_user_id    in number,
   p_product_id in number
) as
   v_quantity number;
begin
   select quantity
     into v_quantity
     from shopping_cart_items
    where user_id = p_user_id
      and product_id = p_product_id;

   delete from shopping_cart_items
    where user_id = p_user_id
      and product_id = p_product_id;

   update products
      set
      quantity = quantity + v_quantity
    where product_id = p_product_id;
end;
/
```

Place order and clear cart

```oracle
create or replace procedure place_order (
   p_user_id          in number,
   p_shipping_address in varchar2
) as
   v_order_id number;
begin
   insert into orders (
      order_id,
      user_id,
      shipping_address
   ) values ( orders_seq.nextval,
              p_user_id,
              p_shipping_address ) returning order_id into v_order_id;

   insert into order_items (
      order_id,
      product_id,
      quantity,
      price
   )
      select v_order_id,
             sci.product_id,
             sci.quantity,
             p.price
        from shopping_cart_items sci
        join products p
      on sci.product_id = p.product_id
       where sci.user_id = p_user_id;

   delete from shopping_cart_items
    where user_id = p_user_id;
end;
/
```

Functions

Calculate total cost of items in cart

```
create or replace function get_cart_total (
   p_user_id in number
) return number is
   v_total number := 0;
begin
   select sum(sci.quantity * p.price)
     into v_total
     from shopping_cart_items sci
     join products p
   on sci.product_id = p.product_id
    where sci.user_id = p_user_id;
   return nvl(
      v_total,
      0
   );
end;
/
```

Calculate total cost of items in order

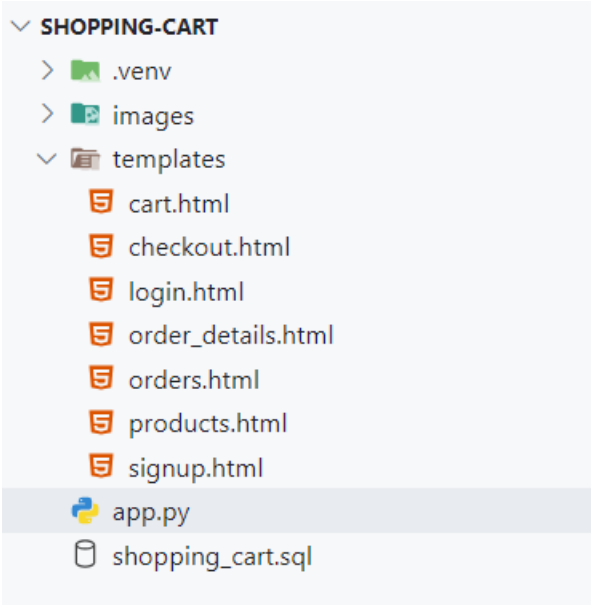
```
create or replace function get_order_total (
   p_order_id in number
) return number is
   v_total number := 0;
begin
   select sum(oi.quantity * oi.price)
     into v_total
     from order_items oi
    where oi.order_id = p_order_id;

   return nvl(
      v_total,
      0
   );
end;
/
```

Code for functional design

The following code is for functional design that includes, database connectivity, PL/SQL Procedure/ Function call, and data access. The Flask back-end is responsible for interacting with the database and the templates created using HTML, Bootstrap and Jinja2 are responsible for rendering the data.

File Structure



Flask Backend

app.py

```
import os

import oracledb
from flask import Flask, redirect, render_template, request, session

app = Flask(__name__)
app.secret_key = os.getenv("FLASK_SECRET_KEY", "default_secret_key")

conn = oracledb.connect(
    user="brendanbadhe", password="bren1234", dsn="localhost:1521/XEPDB1"
)
```

```python
@app.route("/")
def home():
    return redirect("/login")


@app.route("/login", methods=["GET", "POST"])
def login():
    error = None
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]

        cursor = conn.cursor()
        cursor.execute(
            "SELECT user_id FROM users WHERE username = :1 AND password =
:2",
            (username, password),
        )
        user = cursor.fetchone()
        cursor.close()

        if user:
            session["user_id"] = user[0]
            return redirect("/products")
        else:
            error = "Invalid credentials. Try again."

    return render_template("login.html", error=error)


@app.route("/signup", methods=["GET", "POST"])
def signup():
    error = None
    if request.method == "POST":
        username = request.form["username"]
        email = request.form["email"]
        password = request.form["password"]
        confirm_password = request.form["confirm_password"]

        if password != confirm_password:
            error = "Passwords do not match."
        else:
            cursor = conn.cursor()
            try:
                cursor.execute(
```

```
                    "INSERT INTO users (user_id, username, email, password)
"
                    "VALUES (users_seq.NEXTVAL, :1, :2, :3)",
                    (username, email, password),
                )
                conn.commit()
                return redirect("/login")
            except oracledb.IntegrityError:
                error = "Username or email already exists."
            finally:
                cursor.close()

    return render_template("signup.html", error=error)


@app.route("/products")
def products():
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    cursor.execute(
        "SELECT product_id, name, description, price, quantity FROM
products"
    )
    items = cursor.fetchall()

    cursor.execute(
        "SELECT product_id FROM shopping_cart_items WHERE user_id = :1",
        (user_id,),
    )
    cart_items = {row[0] for row in cursor.fetchall()}
    cursor.close()
    return render_template("products.html", products=items,
cart_items=cart_items)


@app.route("/add_to_cart/<int:product_id>", methods=["POST"])
def add_to_cart(product_id):
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]
```

```
    cursor = conn.cursor()
    try:
        cursor.callproc("add_to_cart", [user_id, product_id])
        conn.commit()
    finally:
        cursor.close()

    return redirect("/products")


@app.route("/cart")
def cart():
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    try:
        cursor.execute(
            "SELECT p.product_id, p.name, p.description, p.price,
sci.quantity, p.quantity "
            "FROM shopping_cart_items sci "
            "JOIN products p ON sci.product_id = p.product_id "
            "WHERE sci.user_id = :1",
            (user_id,),
        )
        cart_items = cursor.fetchall()
        cart_total = cursor.callfunc("get_cart_total", oracledb.NUMBER,
[user_id])
    finally:
        cursor.close()
    return render_template("cart.html", cart_items=cart_items,
cart_total=cart_total)


@app.route("/increment_quantity/<int:product_id>", methods=["POST"])
def increment_quantity(product_id):
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
```

```
    try:
        cursor.callproc("increment_quantity", [user_id, product_id])
        conn.commit()
    finally:
        cursor.close()

    return redirect("/cart")


@app.route("/decrement_quantity/<int:product_id>", methods=["POST"])
def decrement_quantity(product_id):
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    try:
        cursor.callproc("decrement_quantity", [user_id, product_id])
        conn.commit()
    finally:
        cursor.close()

    return redirect("/cart")


@app.route("/remove_from_cart/<int:product_id>", methods=["POST"])
def remove_from_cart(product_id):
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    try:
        cursor.callproc("remove_from_cart", [user_id, product_id])
        conn.commit()
    finally:
        cursor.close()

    return redirect("/cart")


@app.route("/checkout")
def checkout():
```

```
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    try:
        cursor.execute(
            "SELECT p.product_id, p.name, p.description, p.price,
sci.quantity, p.quantity "
            "FROM shopping_cart_items sci "
            "JOIN products p ON sci.product_id = p.product_id "
            "WHERE sci.user_id = :1",
            (user_id,),
        )
        cart_items = cursor.fetchall()
        cart_total = cursor.callfunc("get_cart_total", oracledb.NUMBER,
[user_id])
    finally:
        cursor.close()
    return render_template(
        "checkout.html", cart_items=cart_items, cart_total=cart_total
    )


@app.route("/place_order", methods=["POST"])
def place_order():
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]
    shipping_address = request.form.get("shipping_address")

    if not shipping_address:
        return redirect("/cart")

    cursor = conn.cursor()
    try:
        cursor.callproc("place_order", [user_id, shipping_address])
        conn.commit()
    finally:
        cursor.close()

    return redirect("/orders")
```

```
@app.route("/orders")
def orders():
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    try:
        cursor.execute(
            "SELECT o.order_id, o.order_date, o.shipping_address,
SUM(oi.quantity * oi.price) "
            "AS total FROM orders o "
            "JOIN order_items oi ON o.order_id = oi.order_id "
            "WHERE o.user_id = :1 GROUP BY o.order_id, o.order_date,
o.shipping_address "
            "ORDER BY o.order_date DESC",
            (user_id,),
        )
        order_history = cursor.fetchall()
    finally:
        cursor.close()

    return render_template("orders.html", orders=order_history)


@app.route("/order_details/<int:order_id>")
def order_details(order_id):
    if "user_id" not in session:
        return redirect("/login")

    user_id = session["user_id"]

    cursor = conn.cursor()
    try:
        cursor.execute(
            "SELECT p.name, p.description, oi.price, oi.quantity, (oi.price
* oi.quantity) "
            "AS total FROM order_items oi JOIN products p ON oi.product_id =
p.product_id "
            "WHERE oi.order_id = :1 "
            "AND EXISTS ( SELECT 1 FROM orders o WHERE o.order_id = :2 AND
o.user_id = :3 )",
            (order_id, order_id, user_id),
        )
```

```
        order_items = cursor.fetchall()
        order_total = cursor.callfunc("get_order_total", oracledb.NUMBER,
[order_id])
    finally:
        cursor.close()

    if not order_items:
        return redirect("/orders")

    return render_template(
        "order_details.html",
        order_items=order_items,
        order_id=order_id,
        order_total=order_total,
    )


@app.route("/logout")
def logout():
    session.pop("user_id", None)
    return redirect("/login")


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

Templates

cart.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Shopping Cart</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
        integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light">
```

```
    <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
        <div class="container">
            <a class="navbar-brand" href="/products">Store</a>
            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/orders">Orders</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/logout">Logout</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container mt-5">
        <h2 class="text-center mb-4">Shopping Cart</h2>
        {% if cart_items %}
        <div class="d-flex flex-column align-items-center">
            {% for item in cart_items %}
            <div class="card mb-3 w-50 mx-auto">
                <div class="card-body">
                    <h5 class="card-title">{{ item[1] }}</h5>
                    <p class="card-text">{{ item[2] }}</p>
                    <p class="card-text"><strong>Price:</strong> ${{ item[3]
}}</p>
                    <div class="d-flex align-items-center mb-3">
                        <form action="/decrement_quantity/{{ item[0] }}"
method="post" class="me-2">
                            <button type="submit"
                                class="btn btn-sm {% if item[4] <= 1 %}btn-
secondary{% else %}btn-outline-danger{% endif %}"
                                title="decrement_quantity" {% if item[4] <=1
%} disabled {% endif %}>
                                -
                            </button>
```

```
                        </form>
                        <span>{{ item[4] }}</span>
                        <form action="/increment_quantity/{{ item[0] }}"
method="post" class="ms-2">
                            <button type="submit"
                                class="btn btn-sm {% if item[5] == 0 %}btn-
secondary{% else %}btn-outline-success{% endif %}"
                                title="increment_quantity" {% if item[5]==0
%} disabled {% endif %}>
                                +
                            </button>
                        </form>
                    </div>
                    <p class="card-text"><strong>Total:</strong> ${{ item[3]
* item[4] }}</p>
                    <form action="/remove_from_cart/{{ item[0] }}"
method="post">
                        <button type="submit" class="btn btn-danger btn-sm">
                            Remove
                        </button>
                    </form>
                </div>
            </div>
            {% endfor %}
        </div>
        <div class="mt-4 text-center">
            <h4><strong>Cart Total:</strong> ${{ cart_total }}</h4>
        </div>
        <div class="d-flex justify-content-between mt-4">
            <a href="/products" class="btn btn-outline-primary">
                Continue Shopping
            </a>
            <a href="/checkout" class="btn btn-success">
                Proceed to Checkout
            </a>
        </div>
        {% else %}
        <div class="alert alert-primary text-center" role="alert">
            Your cart is empty. <a href="/products" class="alert-link">Start
shopping now!</a>
        </div>
        {% endif %}
    </div>
```

```
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
        crossorigin="anonymous"></script>
</body>

</html>
```

checkout.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Checkout</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light">
    <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
        <div class="container">
            <a class="navbar-brand" href="/products">Store</a>
            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/orders">Orders</a>
                    </li>
                    <li class="nav-item">
```

```
                        <a class="nav-link" href="/logout">Logout</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container mt-5">
        <h2 class="text-center mb-4">Checkout</h2>
        <div class="card shadow-sm mb-4">
            <div class="card-body">
                <h5 class="card-title">Order Summary</h5>
                <table class="table table-bordered">
                    <thead class="table-light">
                        <tr>
                            <th>Name</th>
                            <th>Description</th>
                            <th>Price (per unit)</th>
                            <th>Quantity</th>
                            <th>Total</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for item in cart_items %}
                        <tr>
                            <td>{{ item[1] }}</td>
                            <td>{{ item[2] }}</td>
                            <td>${{ item[3] }}</td>
                            <td>{{ item[4] }}</td>
                            <td>${{ item[3] * item[4] }}</td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
                <div class="text-end">
                    <strong>Total: ${{ cart_total }}</strong>
                </div>
            </div>
        </div>

        <form action="/place_order" method="post" class="w-50 mx-auto">
            <div class="mb-3">
                <label for="shipping_address" class="form-label">Shipping
Address:</label>
```

```
                <input type="text" id="shipping_address"
name="shipping_address" class="form-control" required />
            </div>
            <div class="d-flex justify-content-between">
                <a href="/cart" class="btn btn-outline-secondary">Go
Back</a>
                <button type="submit" class="btn btn-
success">Confirm</button>
            </div>
        </form>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"
        integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
        crossorigin="anonymous"></script>
</body>

</html>
```

login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Login</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
        integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light d-flex justify-content-center align-items-center vh-
100">
    <div class="container-fluid">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card shadow-lg">
```

```
                    <div class="card-body">
                        <h2 class="text-center mb-4">Login</h2>
                        <form method="post">
                            <div class="mb-3">
                                <label class="form-label">Username:</label>
                                <input type="text" name="username"
class="form-control" title="username" required />
                            </div>
                            <div class="mb-3">
                                <label class="form-label">Password:</label>
                                <input type="password" name="password"
class="form-control" title="password" required />
                            </div>
                            {% if error %}
                            <div class="alert alert-danger alert-dismissible
fade show" role="alert">
                                {{ error }}
                                <button type="button" class="btn-close"
data-bs-dismiss="alert"
                                        aria-label="Close"></button>
                            </div>
                            {% endif %}
                            <button type="submit" class="btn btn-primary w-
100">
                                Login
                            </button>
                        </form>
                        <p class="text-center mt-3">
                            Don't have an account? <a href="/signup">Sign
up</a>
                        </p>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"
        integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
        crossorigin="anonymous"></script>
</body>

</html>
```

order_details.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Order Details</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
    integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light">
  <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
    <div class="container">
      <a class="navbar-brand" href="/products">Store</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav"
        aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          <li class="nav-item">
            <a class="nav-link" href="/cart">Cart</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/orders">Orders</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/logout">Logout</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

  <div class="container mt-5">
```

```html
    <h2 class="text-center mb-4">Order Details</h2>
    <div class="card shadow-sm mb-4">
      <div class="card-body">
        <h5 class="card-title">Order Summary #{{order_id}}</h5>
        <table class="table table-bordered">
          <thead class="table-light">
            <tr>
              <th>Name</th>
              <th>Description</th>
              <th>Price (per unit)</th>
              <th>Quantity</th>
              <th>Total</th>
            </tr>
          </thead>
          <tbody>
            {% for item in order_items %}
            <tr>
              <td>{{ item[0] }}</td>
              <td>{{ item[1] }}</td>
              <td>${{ item[2] }}</td>
              <td>{{ item[3] }}</td>
              <td>${{ item[4] }}</td>
            </tr>
            {% endfor %}
          </tbody>
        </table>
        <div class="text-end">
          <strong>Total: ${{ order_total }}</strong>
        </div>
      </div>
    </div>
    <div class="text-center mt-4">
      <a href="/orders" class="btn btn-outline-secondary">Go Back</a>
    </div>
  </div>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
    crossorigin="anonymous"></script>
</body>

</html>
```

orders.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Orders</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
    integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light">
  <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
    <div class="container">
      <a class="navbar-brand" href="/products">Store</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav"
        aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          <li class="nav-item">
            <a class="nav-link" href="/cart">Cart</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/orders">Orders</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="/logout">Logout</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

  <div class="container mt-5">
    <h2 class="text-center mb-4">Orders</h2>
```

```
    {% if orders %}
    <div class="card shadow-sm mb-4">
      <div class="card-body">
        <h5 class="card-title">Order History</h5>
        <table class="table table-bordered">
          <thead class="table-light">
            <tr>
              <th>ID</th>
              <th>Timestamp</th>
              <th>Shipping Address</th>
              <th>Total</th>
              <th>Details</th>
            </tr>
          </thead>
          <tbody>
            {% for order in orders %}
            <tr>
              <td>{{ order[0] }}</td>
              <td>{{ order[1] }}</td>
              <td>{{ order[2] }}</td>
              <td>${{ order[3] }}</td>
              <td>
                <a href="/order_details/{{ order[0] }}">View Details</a>
              </td>
            </tr>
            {% endfor %}
          </tbody>
        </table>
      </div>
    </div>
    <div class="text-center mt-4">
      <a href="/products" class="btn btn-outline-primary">Continue
Shopping</a>
    </div>
    {% else %}
    <div class="alert alert-primary text-center" role="alert">
      You have no orders yet.
      <a href="/products" class="alert-link">Start shopping now!</a>
    </div>
    {% endif %}
  </div>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"
```

```
    integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
    crossorigin="anonymous"></script>
</body>

</html>
```

products.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Products</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
        integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light">
    <nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm">
        <div class="container">
            <a class="navbar-brand" href="/products">Store</a>
            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/orders">Orders</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/logout">Logout</a>
                    </li>
                </ul>
```

```
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <h2 class="mb-4 text-center">Products</h2>
        <div class="row">
            {% for product in products %}
            <div class="col-md-4">
                <div class="card mb-4 shadow-sm">
                    <div class="card-body">
                        <h5 class="card-title">{{ product[1] }}</h5>
                        <p class="card-text">{{ product[2] }}</p>
                        <p class="card-text"><strong>Price:</strong> ${{
product[3] }}</p>
                        <p class="card-text"><strong>Stock:</strong> {{
product[4] }}</p>
                        <form action="/add_to_cart/{{ product[0] }}"
method="post">
                            <button type="submit"
                                class="btn {% if product[4] == 0 %} btn-
outline-danger {% elif product[0] in cart_items %} btn-secondary {% else %}
btn-primary {% endif %}"
                                {% if product[4]==0 or product[0] in
cart_items %} disabled {% endif %}>
                                {% if product[4] == 0 %}
                                Out of Stock
                                {% elif product[0] in cart_items %}
                                In Cart
                                {% else %}
                                Add to Cart
                                {% endif %}
                            </button>
                        </form>
                    </div>
                </div>
            </div>
            {% endfor %}
        </div>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"
        integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
```

```
        crossorigin="anonymous"></script>
</body>

</html>
```

signup.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Sign Up</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.cs
s" rel="stylesheet"
        integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous" />
</head>

<body class="bg-light d-flex justify-content-center align-items-center vh-
100">
    <div class="container-fluid">
        <div class="row justify-content-center">
            <div class="col-md-5">
                <div class="card shadow-lg">
                    <div class="card-body">
                        <h2 class="text-center mb-4">Sign Up</h2>
                        <form method="post">
                            <div class="mb-3">
                                <label for="username" class="form-
label">Username:</label>
                                <input type="text" id="username"
name="username" class="form-control" required />
                            </div>
                            <div class="mb-3">
                                <label for="email" class="form-
label">Email:</label>
                                <input type="email" id="email" name="email"
class="form-control" required />
                            </div>
                            <div class="mb-3">
```

```
                                <label for="password" class="form-
label">Password:</label>
                                <input id="password" type="password"
name="password" class="form-control" required />
                            </div>
                            <div class="mb-3">
                                <label for="confirm_password" class="form-
label">Confirm Password:</label>
                                <input type="password" id="confirm_password"
name="confirm_password"
                                        class="form-control" required />
                            </div>
                            {% if error %}
                            <div class="alert alert-danger alert-dismissible
fade show" role="alert">
                                {{ error }}
                                <button type="button" class="btn-close"
data-bs-dismiss="alert"
                                        aria-label="Close"></button>
                            </div>
                            {% endif %}
                            <button type="submit" class="btn btn-primary w-
100">
                                Sign Up
                            </button>
                        </form>
                        <p class="text-center mt-3">
                            Already have an account? <a
href="/login">Login</a>
                        </p>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"
        integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
        crossorigin="anonymous"></script>
</body>

</html>
```

References

1. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts (6th ed.). McGraw-Hill.
2. Database Systems. Manipal Institute of Technology, MAHE.
3. Database Systems Lab. Manipal Institute of Technology, MAHE.
4. Bootstrap (n.d.). Bootstrap Documentation. Get Started with Bootstrap. https://getbootstrap.com/docs/5.3/
5. *Oracle (n.d.). Oracle Database Documentation. Oracle Database 21c. https://docs.oracle.com/en/database/oracle/oracle-database/21/*
6. Pallets (2010). Flask Documentation. Flask Documentation. https://flask.palletsprojects.com/en/stable/
7. Pallets (2007). Jinja Documentation. Jinja. https://jinja.palletsprojects.com/en/stable/
8. Python (2025). Python Documentation. 3.13.3 Documentation. https://docs.python.org/3/index.html
9. GitHub. (2025). GitHub Copilot. https://github.com/