E-COMMERCE APPLICATION

1. Setup and Configuration:

Set up your development environment by creating a new directory for your project and initializing a virtual environment.

Install Flask and any necessary extensions, e.g., Flask-WTF for forms.

Configure your Flask application and define your secret key and database connection.

2. User Registration and Authentication:

Create data models for users, products, and orders. You can use SQLAlchemy as an ORM.

Implement user registration and login forms using Flask-WTF.

Create routes and views for user registration, login, and logout.

Implement secure password hashing (e.g., bcrypt) for user passwords.

Use Flask-Login to manage user sessions.

3. Product Listings:

Set up a product database (in this example, you can use Python dictionaries).

Create routes to display product listings.

4. Shopping Cart:

Implement a shopping cart mechanism using session storage.

Allow users to add/remove items from the cart.

Calculate and display the total cost of items in the cart.

5. Checkout Process:

Create a checkout form using Flask-WTF.

Implement a route and view for the checkout process.

Verify user authentication before proceeding with the checkout.

For a simplified example, simulate the payment process.

6. HTML Templates:

Create HTML templates for user registration, login, product listings, shopping cart, and checkout pages using Jinja2 templating.

7. Styling and Frontend:

Apply CSS styles to your HTML templates for a better user interface.

Optionally, use JavaScript to enhance the client-side interactivity (e.g., updating the shopping cart without page reload).

8. Testing:

Thoroughly test your e-commerce platform's functionality.

Consider using testing frameworks like pytest for automated testing.

9. Deployment:

Deploy your e-commerce platform on a web server or cloud hosting service (e.g., Heroku, AWS, DigitalOcean).

10. Security and Data Storage:

Enhance security with secure password hashing (e.g., bcrypt).

Implement security measures for payments, such as SSL encryption and integrating with real payment gateways.

Replace the in-memory data store with a real database (e.g., PostgreSQL, MySQL) for production use.

11. Scaling and Additional Features:

As your e-commerce platform grows, consider adding features like user profiles, order history, product reviews, and product search.

Please note that this is a simplified guide, and building a fully functional e-commerce platform involves a lot of work. It's essential to stay updated on best practices for security and user data protection. Additionally, if your e-commerce platform is going to handle real payments, it's advisable to engage with professionals for payment processing and security audits.

**HTML code:login**

<!DOCTYPE html>

<html>

<head>

<title>Login</title>

</head>

<body>

<h1>Please Log In</h1>

<form method="POST" action="/login">

<label for="username">Username:</label>

<input type="text" name="username" id="username" required>

<br>

<label for="password">Password:</label>

<input type="password" name="password" id="password" required>

<br>

<input type="submit" value="Log In">

</form>

</body>

</html>

**HTML code:Register**

<!DOCTYPE html>

<html>

<head>

<title>Register</title>

</head>

<body>

<h1>Register for an Account</h1>

<form method="POST" action="/register">

<label for="username">Username:</label>

<input type="text" name="username" id="username" required>

<br>

<label for="password">Password:</label>

<input type="password" name="password" id="password" required>

<br>

<input type="submit" value="Register">

</form>

</body>

</html>

**HTML code:Cart**

<!DOCTYPE html>

<html>

<head>

<title>Shopping Cart</title>

</head>

<body>

<h1>Shopping Cart</h1>

<ul>

{% for item in cart %}

<li>{{ item.product.name }} - ${{ item.product.price }}</li>

{% endfor %}

</ul>

<a href="checkout.html">Checkout</a>

</body>

</html>

**JAVASCRIPT:**

const express = require('express');

const session = require('express-session');

const bodyParser = require('body-parser');

const app = express();

const port = 3000;

// Use sessions for tracking user login state

app.use(session({

secret: 'your-secret-key',

resave: true,

saveUninitialized: false,

}));

app.use(bodyParser.urlencoded({ extended: true }));

// In-memory user and product data (for demonstration)

const users = [];

const products = [

{ id: 1, name: 'Product 1', price: 10 },

{ id: 2, name: 'Product 2', price: 20 },

{ id: 3, name: 'Product 3', price: 30 },

];

// Middleware to check if the user is logged in

const requireLogin = (req, res, next) => {

if (!req.session.user) {

res.redirect('/login');

} else {

next();

}

};

// Home page

app.get('/', (req, res) => {

res.send('Welcome to the e-commerce platform');

});

// User registration

app.get('/register', (req, res) => {

res.send('Register for an account');

});

app.post('/register', (req, res) => {

const { username, password } = req.body;

users.push({ username, password });

res.redirect('/login');

});

// User login

app.get('/login', (req, res) => {

res.send('Please log in');

});

app.post('/login', (req, res) => {

const { username, password } = req.body;

const user = users.find((u) => u.username === username && u.password === password);

if (user) {

req.session.user = user;

res.redirect('/dashboard');

} else {

res.send('Login failed. Check your username and password.');

}

});

// User dashboard

app.get('/dashboard', requireLogin, (req, res) => {

res.send(`Welcome, ${req.session.user.username}! This is your dashboard.`);

});

// Product listings

app.get('/products', (req, res) => {

res.send('Product listings:\n' + products.map(p => `${p.name} - $${p.price}`).join('\n'));

});

// Shopping cart (simplified)

const cart = [];

app.get('/cart', requireLogin, (req, res) => {

res.send('Shopping Cart:\n' + cart.map(item => `${item.product.name} - $${item.product.price}`).join('\n'));

});

app.post('/cart/add/:productId', requireLogin, (req, res) => {

const productId = parseInt(req.params.productId, 10);

const product = products.find(p => p.id === productId);

if (product) {

cart.push({ product });

res.redirect('/cart');

} else {

res.send('Product not found.');

}

});

// Checkout (simplified)

app.get('/checkout', requireLogin, (req, res) => {

const total = cart.reduce((acc, item) => acc + item.product.price, 0);

res.send(`Checkout - Total: $${total}`);

});

app.listen(port, () => {

console.log(`Server is running on port ${port}`);

});

**DATABASE:**

CREATE DATABASE e\_commerce;

CREATE USER e\_commerce\_user WITH PASSWORD 'your\_password';

GRANT CONNECT ON DATABASE TO USER e\_commerce\_user;

GRANT CREATEIN ON SCHEMA PUBLIC TO USER e\_commerce\_user;

GRANT IMPLICIT\_SCHEMA ON DATABASE TO USER e\_commerce\_user;

**OUTLOOK:**

**LOGIN PAGE**



**HOME PAGE**

