1. Commerce Application on IBM Cloud Foundry

**Phase 4-Development part-2**

**Problem Definition:**

Continue building the e-commerce platform by implementing user authentication, shopping cart, and checkout functionality.

**Goal:**

The goal is to connect skilled artisans with global audience,showcasing their handmade products and provide features like secure shopping carts,Payment gateways and an initiative checkout process

**User Authentication:**

1. Setting Up the Project:

* Create a new directory for our project.
* Set up a virtual environment to isolate dependencies.
* Initialize a Flask project in your directory using flask init or flask create.

1. Install Required Packages:

* Required Packages are Flask and SQLite3. Now we install them using pip:
* pip install Flask

1. Database Setup:

* Create a SQLite database for our application to store user data.Due to technical issues we cant’t able to access IBM db2 so we are using SQLITE3.
* Define a schema for the user table, including fields Like pid, name, address, contact, pwd.

4)User Registration:

* Create a registration form in HTML with fields for username, email, and password.

* Create a Flask route to handle user registration.

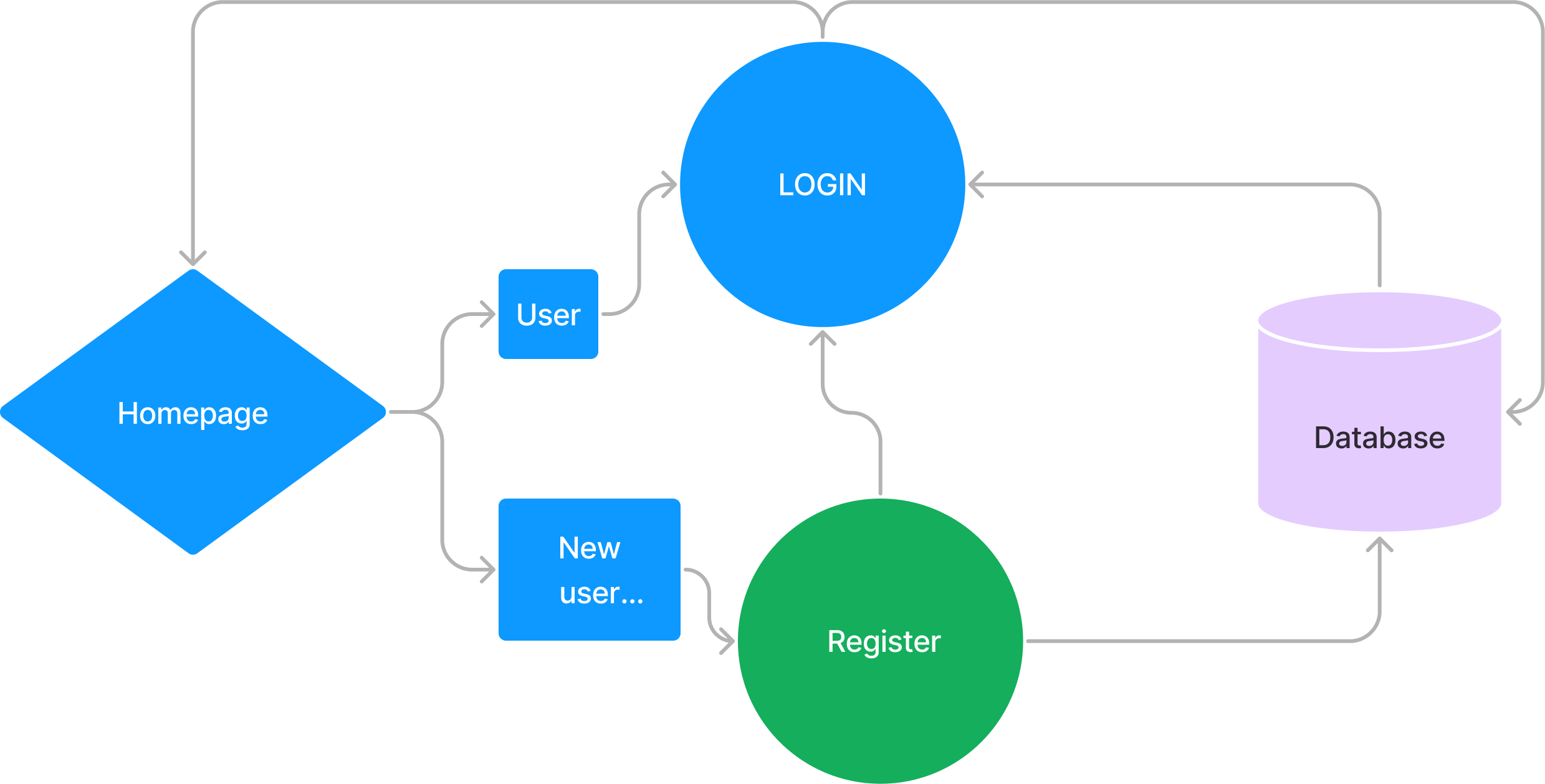
* In the registration route, validate and hash the password before storing the user data in the

database.

5)User Login:

* Design a login form with fields for username and password.
* Implement a route to handle user login.
* Check if the provided credentials match the data stored in the database.
* Use Flask's session to store user information to keep them logged in

**User Flow Diagram:**



**Flask code for User Authentication:**

**from flask import Flask, render\_template, request, flash, redirect, url\_for, session**

**import sqlite3**

**app = Flask(\_\_name\_\_)**

**app.secret\_key = "123"**

**con = sqlite3.connect("MYDB.db")**

**con.execute("create table if not exists customer(pid integer primary key, name text, address text, contact text, mail text, password text)")**

**con.close()**

**@app.route('/')**

**def index():**

**logged\_in = session.get("logged\_in", False)**

**return render\_template('index.html', logged\_in=logged\_in)**

**@app.route('/login', methods=["GET", "POST"])**

**def login():**

**if request.method == 'POST':**

**name = request.form['name']**

**password = request.form['password']**

**con = sqlite3.connect("MYDB.db")**

**con.row\_factory = sqlite3.Row**

**cur = con.cursor()**

**cur.execute("select \* from customer where name=? and password=?", (name, password))**

**data = cur.fetchone()**

**if data:**

**session["name"] = data["name"]**

**session["password"] = data["password"]**

**session["logged\_in"] = True  # Flag for a successful login**

**return redirect(url\_for("index"))**

**else:**

**flash("Username and Password Mismatch", "danger")**

**return render\_template('login&signup.html')**

**@app.route('/register', methods=['GET', 'POST'])**

**def register():**

**if request.method == 'POST':**

**name = request.form['name']**

**mail = request.form['mail']**

**password = request.form['password']**

**cnfpassword = request.form['cnfpassword']**

**address = request.form['address']**

**contact = request.form['contact']**

**if password != cnfpassword:**

**flash("Passwords do not match", "danger")**

**else:**

**con = sqlite3.connect("MYDB.db")**

**cur = con.cursor()**

**cur.execute("insert into customer(name, address, contact, mail, password) values (?, ?, ?, ?, ?)", (name, address, contact, mail, password))**

**con.commit()**

**flash("Registration Successful", "success")**

**con.close()**

**return render\_template('login&signup.html')**

**return render\_template('register.html')**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

**Shopping Cart:**

1. Cart Initialization:

Implement a mechanism to initialize an empty cart for a user (or guest) when they first visit the website or log in.

2)Add to Cart:

Create a route or endpoint to add products to the cart. This should validate the product's availability, update the cart data in the database, and provide feedback to the user.

1. View Cart:

Design a page that displays the contents of the user's cart. Fetch the cart data from the database and display it with product details, quantities, and totals.

1. Update Cart:

Allow users to change the quantity or remove items in their cart. Create routes for updating the cart contents, and update the database accordingly.

1. Calculate Totals:

Implement a function to calculate the cart's total price and update it in real-time as users make changes to their cart.

**Checkout:**

1. Collect Shipping Information: You've implemented a form to collect shipping details from the user, including their shipping address, name, and contact information.

2. Payment Details: You've integrated a payment gateway (e.g., Card, UPI, COD) that allows users to securely provide their payment information, such as credit card details.

3. Order Summary: After shipping and payment details are collected, you display an order summary for the user, listing the products they're purchasing, the total cost, and the shipping address.

4. Place Order: Users have the option to confirm the order, and you have a "Place Order" button that finalizes the transaction.

1. Order Confirmation: After a successful order, you generate an order confirmation page or email receipt to provide the user with a summary of their purchase and order details.

**Conclusion**

In this development phase-4 of the e-commerce platform on IBM Cloud Foundry, we successfully implemented user authentication, a shopping cart, and a checkout process. This allows users to register, log in, add products to their cart, view and update their cart, provide shipping information, enter payment details, review an order summary, place orders, and receive order confirmations. The platform aims to connect skilled artisans with a global audience, showcasing handmade products and providing a secure and user-friendly shopping experience.