# Project Title: Artisanal E-Commerce Platform Development – PHASE 4

## Introduction:

The Artisanal E-Commerce Platform project focuses on implementing key functionalities, including user authentication, shopping cart management, and a seamless checkout process. Below are the details of the implemented modules: checkout.py and auth.py. These modules handle the checkout process and user authentication, respectively.

## Checkout.py:

This module manages the checkout process, allowing users to finalize their purchases. It provides functions to retrieve cart items, clear the cart after checkout, and create orders in the database. The checkout\_items function is responsible for validating user input, clearing the cart, and creating orders based on the user's selections. It ensures a smooth transaction process for users.

"""

Module for handling checkout-related logic.

"""

from datetime import date

from flask import session

from sqlalchemy.sql import text

from website.db import db

def get\_cart\_items(user\_id):

    """

    Retrieve cart items for the user.

    Keyword arguments:

        user\_id (int): User's ID.

    Returns:

        list: List of tuples containing (artisanal\_products\_id, quantity).

    """

    cart\_query = db.session.execute(text(

        "SELECT artisanal\_products\_id, quantity FROM cart WHERE user\_id=:user\_id;"), {"user\_id": user\_id})

    return cart\_query.fetchall()

def clear\_cart(user\_id):

    """

    Remove items from the cart after checkout.

    Keyword arguments:

        user\_id (int): User's ID.

    Returns:

        None

    """

    db.session.execute(text("DELETE FROM cart WHERE user\_id=:user\_id;"), {"user\_id": user\_id})

    db.session.commit()

def create\_order(user\_id, artisanal\_products\_id, first\_name, last\_name, shipping\_address, billing\_address, phone\_number, email, quantity, payment\_method, order\_date):

    """

    Create a new order in the database.

    Keyword arguments:

        user\_id (int): User's ID.

        artisanal\_products\_id (int): artisanal\_products ID.

        first\_name (str): User's first name.

        last\_name (str): User's last name.

        shipping\_address (str): User's shipping address.

        billing\_address (str): User's billing address.

        phone\_number (str): User's phone number.

        email (str): User's email address.

        quantity (int): Quantity of the item.

        payment\_method (str): Payment method used for the order.

        order\_date (date): Order date.

    Returns:

        None

    """

    query = (

        "INSERT INTO orders (user\_id, artisanal\_products\_id, first\_name, last\_name, "

        "shipping\_address, billing\_address, phone\_number, email, "

        "quantity, payment\_method, order\_date) "

        "VALUES (:user\_id, :artisanal\_products\_id, :first\_name, :last\_name, "

        ":shipping\_address, :billing\_address, :phone\_number, "

        ":email, :quantity, :payment\_method, :order\_date);"

    )

    db.session.execute(

        text(query),

        {

            "user\_id": user\_id,

            "artisanal\_products\_id": artisanal\_products\_id,

            "first\_name": first\_name,

            "last\_name": last\_name,

            "shipping\_address": shipping\_address,

            "billing\_address": billing\_address,

            "phone\_number": phone\_number,

            "email": email,

            "quantity": quantity,

            "payment\_method": payment\_method,

            "order\_date": order\_date

        }

    )

    db.session.commit()

def checkout\_items(first\_name, last\_name, shipping\_address, billing\_address, phone\_number, email, payment\_method):

    """

    Checkout items and create orders in the database.

    Keyword arguments:

        first\_name (str): User's first name.

        last\_name (str): User's last name.

        shipping\_address (str): User's shipping address.

        billing\_address (str): User's billing address.

        phone\_number (str): User's phone number.

        email (str): User's email address.

        payment\_method (str): Payment method used for the order.

    Returns:

        None

    """

    user\_id = session.get("user\_id")

    order\_date = date.today()

    if all([

        first\_name,

        last\_name,

        shipping\_address,

        billing\_address,

        phone\_number,

        email,

        payment\_method

    ]):

        cart\_items = get\_cart\_items(user\_id)

        if cart\_items:

            clear\_cart(user\_id)

            for item in cart\_items:

                artisanal\_products\_id, quantity = item

                create\_order(

                    user\_id,

                    artisanal\_products\_id,

                    first\_name,

                    last\_name,

                    shipping\_address,

                    billing\_address,

                    phone\_number,

                    email,

                    quantity,

                    payment\_method,

                    order\_date

                )

## Auth.py:

The authentication module handles user login and registration. It includes functions for user login, logout, and registration. Password requirements are enforced, and the module provides appropriate error messages for failed login attempts and registration errors. It ensures secure user authentication and registration processes.

"""

Module for handling authentication-related logic.

"""

import re

from flask import session, flash

from sqlalchemy.sql import text

from sqlalchemy.exc import IntegrityError

from werkzeug.security import check\_password\_hash, generate\_password\_hash

from website.db import db

def login\_user(username, password):

    """

    Log in a user.

    Keyword arguments:

        username (str): The username.

        password (str): The user's password.

    Returns:

        tuple: A tuple containing the result and a message.

    """

    query = "SELECT id, password FROM users WHERE username=:username;"

    result = db.session.execute(text(query), {"username": username})

    user = result.fetchone()

    if not user:

*# Generic error message for security reasons*

        return "error", "Invalid username or password"

    if check\_password\_hash(user.password, password):

        session["user\_id"] = user.id

        return "success", f"Welcome {username}!"

    return "error", "Invalid username or password"

def logout\_user():

    """Log out a user."""

    del session["user\_id"]

def register\_user(username, password1, password2):

    """

    Register a new user.

    Keyword arguments:

        username (str): The username.

        password1 (str): The user's password.

        password2 (str): The repeated password.

    Returns:

        None

    """

    try:

        hash\_value = generate\_password\_hash(password1)

        reg = "^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])(?=.\*?[#?!@$%^&\*-]).{8,}$"

        if len(username) < 4:

            flash("username must be at least 4 characters long", category="error")

        elif password1 != password2:

            flash("Passwords didn't match", category="error")

        elif not re.match(reg, password1):

            flash(

                "Password requirements: 8+ characters, "

                "1 uppercase, 1 lowercase, 1 digit, 1 special character.",

                category="error"

            )

        else:

            query = (

                "INSERT INTO users (username, password, privileges) "

                "VALUES (:username, :password, :privileges);"

            )

            db.session.execute(

                text(query),

                {"username": username, "password": hash\_value,

                    "privileges": "customer"}

            )

            db.session.commit()

            flash("Account created!", category="success")

    except IntegrityError:

        flash("Username is already taken", category="error")

Conclusion:

The implemented modules provide essential functionalities for the Artisanal E-Commerce Platform. Users can securely log in, register, add items to their cart, and complete the checkout process. The project has made significant progress in enhancing user experience and ensuring data security.