

**Creating a Shopping App Using Python**

Phase-End Project Problem Statement





**Phase-End Project: Creating a Shopping App Using Python**

### Problem Scenario: You have to develop a shopping application or e-commerce application which has login and public login features on the Python platform. The applications that have been developed should also include categories, such as 3–4 for footwear, clothing, electronics, etc. It should be possible to add and update categories in the application. Additionally, it must contain a feature that allows you to add or remove items from your cart. Finally, the program needs to support a variety of payment options, including UPI and debit cards. This should be only backend implementation, and UX/UI and database connectivity are not required.

### Guidelines:

* A welcome message should initially be displayed in the e-commerce application, such as "Welcome to the Demo Marketplace". - Done
* User login and admin login should be created once the code for the welcome message has been written. For the creation of demo login and admin login, demo databases for those should be created for the user and admin verification, and session id creation.
* After databases are created, create admin and user logins. It is necessary to construct a sample product catalog with three to four product categories, such as Boots, Coats, Jackets, and Caps. The product id, name, category id, and price should all be present for each item in the dummy database of the catalog. Both users and administrators can view the catalog.
* User login rights include the ability to view cart contents, add items to carts, and remove items from carts. The user should be able to add items or delete items in the cart using session id, product id, and quantity.
* Next, the program should provide demo payment checkout options, like Net banking, PayPal, UPI, etc. After selecting the payment option, a checkout message like, "Your order is successfully placed" or "You will be shortly redirected to the portal for Unified Payment Interface to make a payment of Rs. 2000", etc., should be displayed.
* Additionally, the admin can only log in using his credentials, and if the admin attempts to log in using another set of credentials, an error notice must appear.
* Admin should not be able to interfere with any of the functions that the user can perform, as discussed above. An error should appear if the admin tries to carry out those tasks.
* Furthermore, using the previously selected attributes, the admin should be able to add new products to the catalog. Additionally, the program needs to make it possible for an existing product to be modified using an admin session id.
* The admin should then have the ability to remove any existing products from the already-generated catalog.
* Lastly, understanding the dynamic demands of the market, the admin should be able to add a new category of product and delete the prevailing category of product from the catalog.
* Users should not be able to take advantage of any of the admin's rights, as described above.

SOURCE CODE

#

#Database Initialization: The script checks if the user and product databases exist. If not, it creates them.

#Classes:

#User: Handles user login, signup, password change, and account deletion.

#Admin: Inherits from User and adds functionalities to manage products and categories.

#ShoppingCart: Manages the cart operations like adding, removing, viewing, and checking out items.

#Main Function: Handles the initial login/signup process and directs users to the appropriate menu based on their role.

#Admin Menu: Admin can add, update, and delete products and categories.

#User Menu: Users can view products, manage their cart, checkout, change their password, and delete their account.

#Cart Management: Users can add or remove items from their cart.

#Checkout: Users can select a payment method and complete their purchase.

#Password Management: Users can change their password.

#Account Deletion: Users can delete their account.

#Error Handling: The script includes basic error handling for invalid inputs.

#Console Clearing: The script clears the console for a fresh start each time.

#This script provides a backend implementation for a shopping application using Python and pandas to manage data in Excel files.

import pandas as pd

import os

import sys

import time

# Constants

USER\_DB = 'user\_db.xlsx'

PRODUCT\_DB = 'product\_db.xlsx'

CATEGORIES = ['Footwear', 'T-shirts', 'Jeans', 'Wallets', 'Bags']

ADMIN\_CREDENTIALS = {'username': 'Admin1', 'password': 'Admin123', 'role': 'admin'}

# Clear console function

def clear\_console():

os.system('cls' if os.name == 'nt' else 'clear')

# Initialize databases if not present

def initialize\_databases():

if not os.path.exists(USER\_DB):

user\_data = pd.DataFrame(columns=['username', 'password', 'role', 'session\_id'])

user\_data = user\_data.\_append(ADMIN\_CREDENTIALS, ignore\_index=True)

user\_data.to\_excel(USER\_DB, index=False)

if not os.path.exists(PRODUCT\_DB):

product\_data = pd.DataFrame(columns=['product\_id', 'name', 'category', 'price'])

product\_data.to\_excel(PRODUCT\_DB, index=False)

# Load databases

def load\_databases():

user\_data = pd.read\_excel(USER\_DB)

product\_data = pd.read\_excel(PRODUCT\_DB)

global CATEGORIES

CATEGORIES = list(set(product\_data['category'].to\_list()))

return user\_data, product\_data

# Save databases

def save\_database\_product(product\_data):

#user\_data.to\_excel(USER\_DB, index=False)

product\_data.to\_excel(PRODUCT\_DB, index=False)

def save\_database\_user(user\_data):

user\_data.to\_excel(USER\_DB, index=False)

#product\_data.to\_excel(PRODUCT\_DB, index=False)

class User:

def \_\_init\_\_(self, username, password, role='user', session\_id=None):

self.username = username

self.password = password

self.role = role

self.session\_id = session\_id

def login(self, user\_data):

user = user\_data[(user\_data['username'] == self.username) & (user\_data['password'] == self.password)]

if not user.empty:

self.session\_id = os.urandom(16).hex()

user\_data.loc[user.index, 'session\_id'] = self.session\_id

return True

return False

def signup(self, user\_data):

if self.username == ADMIN\_CREDENTIALS['username']:

print("Username 'Admin1' is reserved. Please choose another username.")

return user\_data

if self.username in user\_data['username'].values:

print("Username already exists. Please choose another username.")

return user\_data

new\_user = {'username': self.username, 'password': self.password, 'role': self.role, 'session\_id': ''}

user\_data = user\_data.\_append(new\_user, ignore\_index=True)

return user\_data

def change\_password(self, user\_data, new\_password):

user\_data.loc[user\_data['session\_id'] == self.session\_id, 'password'] = new\_password

print("Password changed successfully.")

return user\_data

def delete\_account(self, user\_data):

user\_data = user\_data[user\_data['session\_id'] != self.session\_id]

print("Account deleted successfully.")

return user\_data

class Admin(User):

def \_\_init\_\_(self, username, password):

super().\_\_init\_\_(username, password, role='admin')

def add\_product(self, product\_data, product\_id, name, category, price):

product = product\_data[product\_data['product\_id'] == int(product\_id)]

if product.empty and (category in CATEGORIES):

new\_product = {'product\_id': product\_id, 'name': name, 'category': category, 'price': price}

product\_data = product\_data.\_append(new\_product, ignore\_index=True)

print("Product added successfully.")

else:

print('Product ID Already Exist or Category Does not Exist')

return product\_data

def update\_product(self, product\_data, product\_id, name, category, price):

product\_id = int(product\_id)

product = product\_data[product\_data['product\_id'] == product\_id]

if not product.empty and (category in CATEGORIES):

product\_data.loc[product.index, ['name', 'category', 'price']] = [name, category, price]

print("Product updated successfully.")

else:

print("Product or Category not found.")

return product\_data

def delete\_product(self, product\_data, product\_id):

product\_id = int(product\_id)

product = product\_data[product\_data['product\_id'] == product\_id]

if not product.empty:

product\_data.drop(product\_data.index[(product\_data["product\_id"] == product\_id)],axis=0,inplace=True)

print("Product deleted successfully.")

else:

print("Product not found.")

return product\_data

def add\_category(self, category):

if category not in CATEGORIES:

CATEGORIES.append(category)

print("Category added successfully.")

else:

print("Category already exists.")

def delete\_category(self, category):

if category in CATEGORIES:

CATEGORIES.remove(category)

print("Category deleted successfully.")

else:

print("Category not found.")

class ShoppingCart:

def \_\_init\_\_(self):

self.cart = []

def add\_to\_cart(self, product\_data, product\_id, quantity):

product = product\_data[product\_data['product\_id'] == int(product\_id)]

if not product.empty:

self.cart.append({'product\_id': product\_id, 'quantity': quantity})

print("Product added to cart.")

else:

print("Product not found.")

def remove\_from\_cart(self, product\_id):

self.cart = [item for item in self.cart if item['product\_id'] != (product\_id)]

print("Product removed from cart.")

def view\_cart(self):

print("\nCart Contents:")

for item in self.cart:

print(f"Product ID: {item['product\_id']}, Quantity: {item['quantity']}")

def checkout(self, product\_id):

if self.cart:

print("Select payment method:")

print("1. Card")

print("2. UPI")

print("3. COD")

print("4. Cancel Order and Exit")

choice = input("Enter your choice: ")

if choice == '1':

attempt = 0

while (True):

if attempt >=3:

print("\nMax attempt Crossed. Re-directing to account")

time.sleep(2)

break

card\_number = input('Enter Card Number: ')

if len(card\_number) !=16:

print("Enter Valid 16 digit Card Number!!!")

attempt +=1

pass

else:

try:

card\_number = int(card\_number)

try:

doe = int(input('Enter Expiry Date[MMYY]: '))

cvv = int(input('Enter CVV: '))

print("Your order is successfully placed.")

print("Your order is successfully placed.")

print("Re-directing to account")

time.sleep(3)

self.cart = []

#self.cart = [item for item in self.cart if item['product\_id'] != (product\_id)]

break

except:

print('\nEnter Valid Expiry date and CVV!!!')

attempt +=1

pass

except:

print("\nEnter Valid 16 Digit Card Number!!!")

attempt +=1

pass

elif choice == '2':

upi\_id = input('Enter UPI ID: ')

print("Your order is successfully placed.")

print("Re-directing to account")

time.sleep(3)

self.cart = []

elif choice == '3':

print("Your order is successfully placed.")

print("Your order is successfully placed.")

print("Re-directing to account")

time.sleep(3)

self.cart = []

elif choice == '4':

print("Order Cancelled")

time.sleep(2)

print("\nThank you and Good Bye!!!")

exit()

else:

print("Invalid choice. Please try again.")

else:

print("Cart is empty.")

def main():

for i in range(3):

clear\_console()

time.sleep(1)

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\n\*\*\*\*\*\* Welcome to the Demo Marketplace \*\*\*\*\*\*\*\*\*")

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

time.sleep(1)

initialize\_databases()

user\_data, product\_data = load\_databases()

while True:

print("\n1. Login")

print("2. Sign Up")

print("3. Exit")

choice = input("Enter your choice: ")

if choice == '1':

username = input("Enter username: ")

password = input("Enter password: ")

if username == ADMIN\_CREDENTIALS['username'] and password == ADMIN\_CREDENTIALS['password']:

admin = Admin(username, password)

admin\_menu(admin, product\_data)

else:

user = User(username, password)

if user.login(user\_data):

user\_menu(user, user\_data, product\_data)

else:

print("Invalid credentials. Try again or create a new account.")

elif choice == '2':

username = input("Enter new username: ")

password = input("Enter new password: ")

user = User(username, password)

user\_data = user.signup(user\_data)

elif choice == '3':

print("Thank you and goodbye!")

try:

save\_database\_product(product\_data)

save\_database\_user(user\_data)

except NameError:

pass

sys.exit()

else:

print("Invalid choice. Please try again.")

def admin\_menu(admin, product\_data):

while True:

print("\nAdmin Menu:")

print("1. Add Product")

print("2. Update Product")

print("3. Delete Product")

print("4. Add Category")

print("5. Delete Category")

print("6. View Catalogue")

print("7. View Categories")

print("8. Exit")

choice = input("Enter your choice: ")

if choice == '1':

product\_id = input("Enter product ID: ")

name = input("Enter product name: ")

category = input("Enter product category: ")

price = float(input("Enter product price: "))

product\_data = admin.add\_product(product\_data, product\_id, name, category, price)

elif choice == '2':

product\_id = input("Enter product ID to update: ")

name = input("Enter new product name: ")

category = input("Enter new product category: ")

price = float(input("Enter new product price: "))

product\_data = admin.update\_product(product\_data, product\_id, name, category, price)

elif choice == '3':

product\_id = input("Enter product ID to delete: ")

product\_data = admin.delete\_product(product\_data, product\_id)

elif choice == '4':

category = input("Enter new category: ")

admin.add\_category(category)

elif choice == '5':

category = input("Enter category to delete: ")

admin.delete\_category(category)

elif choice == '6':

print("\nProduct Catalog:")

print(product\_data)

elif choice == '7':

print("\nThe Category")

print(CATEGORIES)

elif choice == '8':

print("Thank you and goodbye!")

try:

save\_database\_product(product\_data)

save\_database\_user(user\_data)

except NameError:

pass

sys.exit()

else:

print("Invalid choice. Please try again.")

def user\_menu(user, user\_data, product\_data):

cart = ShoppingCart()

while True:

print("\nUser Menu:")

print("1. View Products")

print("2. Add to Cart")

print("3. Remove from Cart")

print("4. View Cart")

print("5. Checkout")

print("6. Change Password")

print("7. Delete Account")

print("8. Exit")

choice = input("Enter your choice: ")

if choice == '1':

view\_products(product\_data)

elif choice == '2':

product\_id = input("Enter product ID to add to cart: ")

quantity = int(input("Enter quantity: "))

cart.add\_to\_cart(product\_data, product\_id, quantity)

elif choice == '3':

product\_id = input("Enter product ID to remove from cart: ")

cart.remove\_from\_cart(product\_id)

elif choice == '4':

cart.view\_cart()

elif choice == '5':

cart.checkout(product\_id)

cart.remove\_from\_cart(product\_id)

clear\_console()

elif choice == '6':

new\_password = input("Enter new password: ")

user\_data = user.change\_password(user\_data, new\_password)

elif choice == '7':

user\_data = user.delete\_account(user\_data)

print("Account deleted. Thank you and goodbye!")

try:

save\_database\_product(product\_data)

save\_database\_user(user\_data)

except NameError:

pass

sys.exit()

elif choice == '8':

print("Thank you and goodbye!")

try:

save\_database\_product(product\_data)

save\_database\_user(user\_data)

except NameError:

pass

sys.exit()

else:

print("Invalid choice. Please try again.")

def view\_products(product\_data):

print("\nProduct Catalog:")

print(product\_data)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Data Bases:

User\_db.xlsx



Product\_db.xlsx

