ONLINE SHOPPING MANAGEMENT SYSTEM

An Internship Project at WiseLearnz

Project Period:
July 2024

Submitted by:
Mathi Priyan A M M
Student, Thiagarajar College of Engineering, Madurai

Completed at:
WiseLearnz

Table of Contents

ABSTRACT

This project documentation delves into the development and implementation of a console-based e-shopping application, designed to replicate key functionalities of an e-commerce platform within a console environment. This application addresses core e-commerce features such as user authentication, product catalog management, and a shopping cart system, complete with a checkout module. Additionally, an admin console enables the addition, deletion, and modification of product data, supporting efficient product management and data persistence across sessions. This documentation describes the project from conception to realization, examining the objectives, methodologies, outcomes, and areas for future development. The project aligns with industry standards in security and user interface design while adapting for a console environment, ensuring accessibility and usability without compromising on essential e-commerce features.

1. Introduction:

The **ONLINE SHOPPING MANAGEMENT SYSTEM** is a console-based Python project designed to manage user authentication processes, such as login, registration, password reset, profile updates, and product management for an e-commerce platform. The system ensures secure user access and provides features to manage products and also enables admin management.

2. Functional Requirements:

2.1 User Authentication Features:

- **User Registration System:** New users can register with their details, including security questions.
- **Login System:** User can Log in with either Email or mobile password with their respective password.
- **Forgot Password:** Users can reset their password using security questions and OTP verification.
- **Profile Management:** Users can view and update their profile information.
- **Logout:** Users can securely log out of the system.

2.2 Product Management Features:

- **Add Products:** Admins can add products like televisions, phones, etc., into the system.
- **View Products:** Users can view the list of products added by the admin.
- **Manage Product:** Admins can edit or delete product details.

2.3 Admin Management Features:

- Admin login is verified from the database.
- Admin can manage products, including adding, updating, or deleting them.
- Admin has access to view all registered users and their profile information.
- Admin can change the order status for different users.

3. Non-Functional Requirements:

- **Data Storage:** User data and login details are stored using Python's pickle module in Dat files.
- **Error Logging:** Errors are logged into a file for debugging.
- **Security:** Implements OTP verification and security questions for password resets.

4. System Design:

4.1. Modules Overview:

1. **Login Module:** Manages user login (Email or mobile number), verifies credentials, and allows access to the system.

2. **Registration Module:** Allows users to register by storing their details securely including security questions and OTP verification.

3. **Password Reset Module:** Helps users reset passwords via OTP and security question verification.

4. **Profile Management Module:** Users can update their profile details (name, mobile number, address).

5. **Logout Module:** Allows user logout.

6. **Validation Module:** This is the module where the personal information of a user is validated such as password validation.

7. **Cart Module:** Allows for adding, removing, editing products in your Cart.
8. **Rating/Review Module:** Allows for products ratings and reviews after purchasing the product and also to view ratings. Reviews are categorized as positive, negative and neutral based on the given words.

9. **Wishlist Module:** Allows for adding, removing products in your Wishlist.

10. **Checkout Module:** Allows for processing payment methods (cash on/Online) and ready for delivery.

11. **File read/write module:** Allows for storing and retrieving entered details in a file securely.

12. **Error logging module:** Logs system errors for future references.

13. **Database module:** Interact with the database to store and retrieve products and user's personal information.

5. User Guide:

5.1 User Authentication:

This module allows users to log in, register, and manage their accounts securely.

- **Register:**

1. Select **Register** in Home screen.
2. Enter Your details such as name, email, password, mobile number, address and security questions along with OTP validation.
3. Upon Registration, Log in using valid credentials.

- **Login:**

1. Once Registered, Select **Login.**
2. Enter either email or mobile number with valid password.
3. In case of invalid credentials, you will be prompted an error and re-enter.
4. Once you visited, select **Logout** and yes.

- **Forgot Password:**

1. Click on **Forgot password**.
2. Enter either email or mobile number.
3. Answer the security question you prefer.
4. OTP will be displayed, enter it.
5. You will be allowed to reset the password.

- **Profile Management:**

1. Click on **update profile Information.**
2. Enter your email/mobile and password to view your current profile.
3. You will be prompted to update profile.
4. Now reset the personal information and select yes.

5.2 Cart Management:

This module allows users to add products to their cart and manage them before proceeding to checkout.

- **Add to Cart:**

1. While browsing products, click on "Add to Cart" for any product you wish to purchase with Quantity.
2. The selected product will be added to your cart.

- **View Cart:**

1. Navigate to the **Cart** section to view all items you've added.
2. From here, you can update the quantity or remove items.

- **Proceed to Checkout:**

1. After reviewing your cart, click "Proceed to Checkout" to complete your purchase.

5.3 Wishlist Management:

The Wishlist module allows users to save products for future reference without adding them to the cart.

- **Add to Wishlist:**

1. On the product page, click on **Add to Wishlist**.
2. The product will be saved to your Wishlist for later.

- **View Wishlist**:

1. Navigate to the **Wishlist** section to view all saved items.
2. You can move items from the **Wishlist** to the cart or remove them.

5.4 Checkout:

This module handles the final purchase of products in the cart.

- **Shipping Details**:

1. Once you are ready to check out, you will be prompted to enter your shipping details.
2. Enter your address, contact number, and preferred shipping method.

- **Payment:**

1. After confirming the shipping details, you will be directed to the payment page.
2. Select your payment method (Online/Cash on).
3. After completing the payment, a confirmation message will be displayed with your order number.

5.5 Admin Management:

This module allows administrators to manage the product catalog, including adding, updating, and deleting products.

- **Admin Login:**

1. Admin can login through normal user login but with their given admin credentials.
2. Once entered, you will be in admin panel.

Manage Products:

1. Add new products by entering product details such as name, category, price, and stock availability.
2. Update existing products or delete products from the catalogue.

- **Change Order status:**

1. Admin can change the order status (Order Processing, Packaging, In-Transit, out of delivery, Delivered) for the respective users.
2. It will be reflected in every user panel.

- **View Users:**

1. Admin can view Registered Users.
2. Admin can view login and logout details with timestamp.

6. Database Structure:

6.1 User Details:

- Stored in User Details.dat file and also in a Database.
- Fields: Name, Email, Mobile Number, Password, Address, Security Answers.

6.2 Product Details:

- Stored in admin_products table.
- **Fields:** ID, Name, Price, Quantity, Availability, Color, Description, No_of_ratings, Total_ratings, Average_rating.

6.3 Login/Logout Records:

- Stored in Login_details.dat and Logout_details.dat file.
- **Fields:** Email/Mobile number, Login/Logout timestamp.

6.4 Password Changes Records:

- Stored in password_changes.dat file.
- **Fields:** Email/Mobile number, timestamp.

6.5 Profile Updates Records:

- Stored in Profile_updates.dat file.
- Fields: Email/Mobile number, timestamp.

7. Error Logging:

All errors occurring during execution are logged in the Error log.txt file with timestamps for debugging purposes.

8. Technologies Used:

Programming Language:

- Python.

Libraries:

- **SQL:** Used to interact with relational databases, manage user accounts, product inventory, and transaction details.
- **Pickle:** Employed for serializing and deserializing Python objects, such as user data and session information.
- **Datetime:** Utilized for recording timestamps for user activities like login, logout, and password changes.
- **Tabulate:** Used for displaying data in table format, providing organized, readable output in the console.

Database:

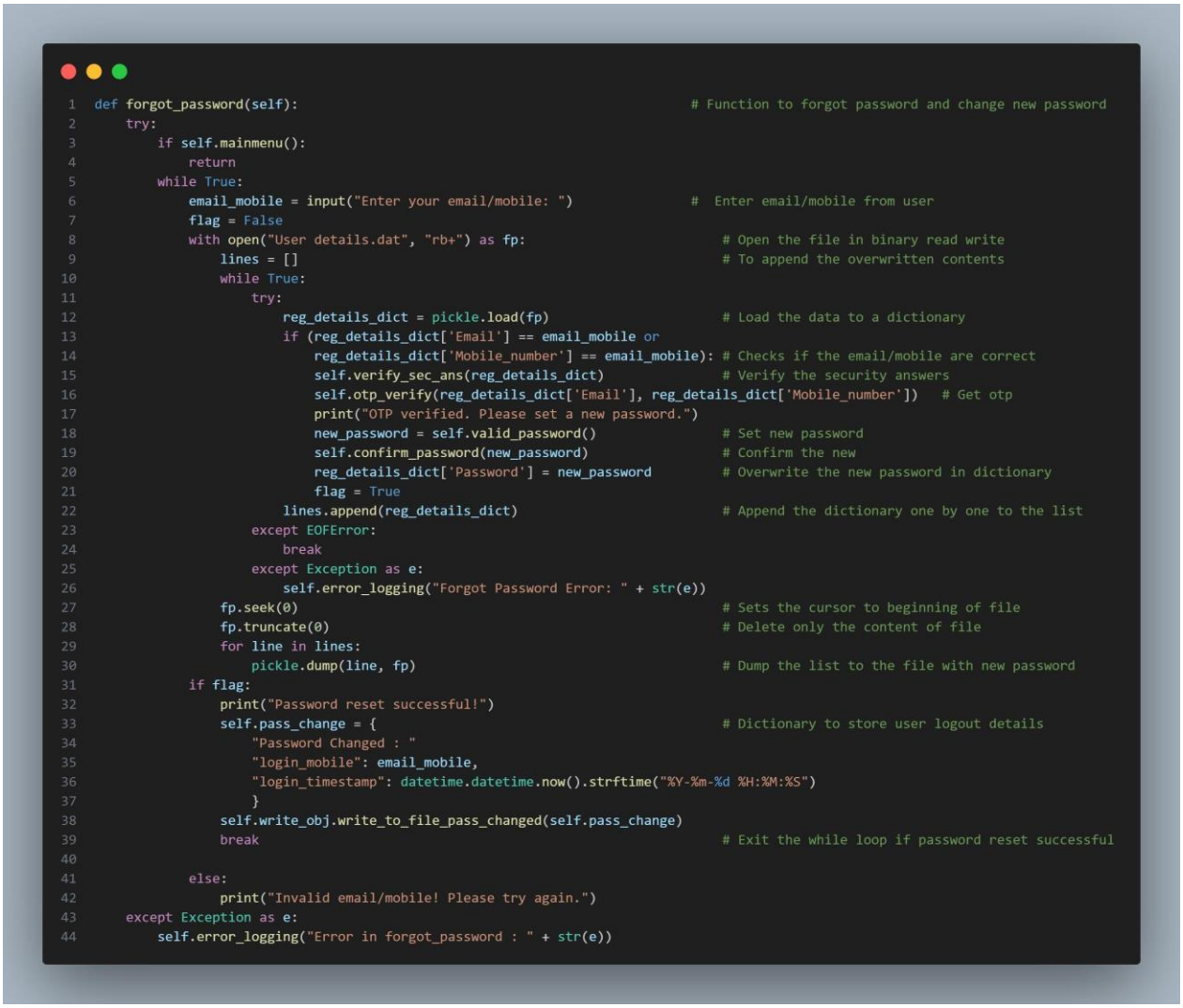
- **MySQL:** The relational database system used to store and manage structured data, including user credentials, product information, and purchase history.

Other Tools:

- **Text Files and Databases:** Used for lightweight storage of login details, user profiles, and password change logs in .Dat files.

9. Sample Code Snippets:

Forgot Password function:



```python
def forgot_password(self):                                          # Function to forgot password and change new password
    try:
        if self.mainmenu():
            return
        while True:
            email_mobile = input("Enter your email/mobile: ")       #  Enter email/mobile from user
            flag = False
            with open("User details.dat", "rb+") as fp:                 # Open the file in binary read write
                lines = []                                              # To append the overwritten contents
                while True:
                    try:
                        reg_details_dict = pickle.load(fp)              # Load the data to a dictionary
                        if (reg_details_dict['Email'] == email_mobile or
                            reg_details_dict['Mobile_number'] == email_mobile): # Checks if the email/mobile are correct
                            self.verify_sec_ans(reg_details_dict)       # Verify the security answers
                            self.otp_verify(reg_details_dict['Email'], reg_details_dict['Mobile_number'])   # Get otp
                            print("OTP verified. Please set a new password.")
                            new_password = self.valid_password()        # Set new password
                            self.confirm_password(new_password)         # Confirm the new
                            reg_details_dict['Password'] = new_password # Overwrite the new password in dictionary
                            flag = True
                        lines.append(reg_details_dict)                  # Append the dictionary one by one to the list
                    except EOFError:
                        break
                    except Exception as e:
                        self.error_logging("Forgot Password Error: " + str(e))
                fp.seek(0)                                              # Sets the cursor to beginning of file
                fp.truncate(0)                                          # Delete only the content of file
                for line in lines:
                    pickle.dump(line, fp)                               # Dump the list to the file with new password
            if flag:
                print("Password reset successful!")
                self.pass_change = {                                    # Dictionary to store user logout details
                    "Password Changed : "
                    "login_mobile": email_mobile,
                    "login_timestamp": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                    }
                self.write_obj.write_to_file_pass_changed(self.pass_change)
                break                                                   # Exit the while loop if password reset successful

            else:
                print("Invalid email/mobile! Please try again.")
    except Exception as e:
        self.error_logging("Error in forgot_password : " + str(e))
```

Payment Process function:

```python
def process_payment(self,user_email):       # Function to make payment for the product
    try :
        if self.cart_obj.view_cart(user_email):     # Call function to view products in cart
            while True:
                try :
                    payment=input("Do you want to purchase items in the cart (yes/press any key to continue) : ").lower()   # again ask user wants to purchase
                    if payment=="yes":              # If user wish to purchase
                        while True:
                            prod_id_no=self.go_back_obj.get_input("Enter the IDs : (enter 'back' to go back) : ").split()   # Asks ids to purchase
                            if prod_id_no is None:      # To go back previous menu
                                return
                            query_check="SELECT cart_ID FROM CARTDETAILS WHERE USER_EMAIL=%s"   # query to select ids in the cart
                            self.cursor.execute(query_check,(user_email,))
                            contents=self.cursor.fetchall()
                            lst=[str(content[0]) for content in contents]                       # Store the ids in cart  in list format
                            flag=False
                            while not flag:                                         # checks if the entered id is in the id columnn in cart
                                for check in prod_id_no:
                                    if check not in lst:
                                        flag=True
                                break
                            if flag:                                                # If user enters invalid id
                                print("Please enter items in your cart!...")
                            if not flag:                                            # If user enters valid id in the cart
                                break
                        flag_quan = False
                        for quan_check in prod_id_no:                               # Loop thorugh each id
                            query_check_quan = "SELECT quantity FROM PRODUCTS WHERE ID=%s"  # query to select quantity from the products
                            self.cursor.execute(query_check_quan, (quan_check,))
                            quantity = self.cursor.fetchone()
                            if quantity and quantity[0] == 0:                       # checks if quantity is zero
                                print("Sorry! ID:", quan_check, "Product out of stock...")
                                flag_quan = True
                        if flag_quan:                                               # if zero go to start
                            continue
                        print("\nProduct selected...")                              # if not proceed
                        while True:
                            cash_on_online=input("\n1. Cash on delivery\n2. Online payment\n3. Exit\nEnter Your choice : ") # asks user payment way
                            if cash_on_online=="1":
                                if self.cash_on(user_email,prod_id_no):         # Call function to go on cash on delivery
                                    return
                            elif cash_on_online=="2":
                                if self.online_pay(user_email,prod_id_no):      # Call function to go on online payment
                                    return
                            elif cash_on_online=="3":                           # Exits from the payment
                                print("Exiting...")
                                break
                            else:                                               # If user enters invalid choice
                                print("Invalid choice...")
                    else:                                                       # If user does not wants to purchase
                        print("Returning back...")
                        break
                except :
                    break
    except Exception as e:
        self.write_file_obj.error_logging("Error in process payment :  "+str(e))        # To log errors
def cash_on(self,user_email,prod_id_no):                                        # Function to make cash on delivery
    try  :
        print("==Cash On Delivery==")
        total_price=self.display(user_email,prod_id_no)                         # Call function to display total cost in the cart
        user_ch=input("Purchase (yes/press any key to continue) : ").lower()    # Ask users to purchase
        if user_ch=="yes":                                                      # if yes
            print("Thankyou! Your order has been confirmed...")
            self.update_transaction(user_email,prod_id_no,total_price)          # call function to update transaction in table
            self.update_products(prod_id_no)                                    # call function to update products quantity column
            return True
        else:                                                                   # If no to purchase
            print("Returning!...")
            return False
    except Exception as e:
        self.write_file_obj.error_logging("Error in cash on :  "+str(e))        # To log errors
def online_pay(self,user_email,prod_id_no):                                     # Function to make online payment
    try :
        print("==Online Payment==")
        total_price=self.display(user_email,prod_id_no)                         # Call function to display total cost in the cart
        while True:
            user_ch=input("Purchase (yes/press any key to continue) : ").lower()    # Ask users to purchase
            if user_ch=="yes":
                upi_=self.upi_id()                                              # Ask upi id from user
                if upi_=="back":                                                # If go back
                    continue
                print("Thankyou! Your order has been confirmed...")
                self.update_transaction(user_email,prod_id_no,total_price)      # call function to update transaction in table
                self.update_products(prod_id_no)                                # call function to update products quantity column
                return True
            else:
                print("Returning!...")
                return False
    except Exception as e:
        self.write_file_obj.error_logging("Error in online pay :  "+str(e))     # To log errors
```

10. Conclusion:

- This project successfully demonstrates the implementation of a **User Authentication and Product Management System** with key features such as user login, registration, password management, product handling and admin management. The system is designed for an e-commerce platform, ensuring secure access and smooth product management.