**README file: Online Shopping Application**

**Installation and Getting Started**

This is my Online Shopping Application for electronics, an e-commerce platform built with Python. Please follow the steps below to install and get started with the application.

1. *Ensure Python 3.x is installed on your system.*

*2. Clone or download the application from our GitHub repository:* [*https://github.com/ashudes/Secure\_Software\_Development/tree/main*](https://github.com/ashudes/Secure_Software_Development/tree/main)

*3. Navigate to the application directory in your command line interface (CLI) and execute:*

*```bash*

*cli.py*

*4. The CLI will guide you through the registration process, product browsing, and other features.*

The application is designed to be intuitive and user-friendly, providing a simple command line interface (CLI) experience.

**Overview**

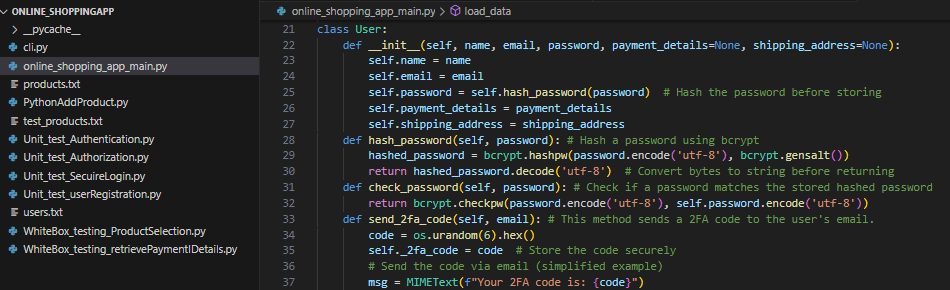
The Online Shopping Application is an Object-Oriented Program (OOP) system that simulates a real-world e-commerce environment. It allows users to explore a wide range of products, make selections, and manage their shopping carts with ease. The application has a robust user authentication system, product management capabilities, and secure payment processing with two-factor authentication (2FA) for enhanced security.

My platform is built with security and user experience at its core, ensuring that every interaction is secure and every transaction is processed with the utmost care.

As a simple e-commerce application, I used an in-built .txt python dictionary. This is one of the easiest methods where strings are saved in a text file and then accessed for the application from other modules (Kong et al., 2020).

**Code Organization**

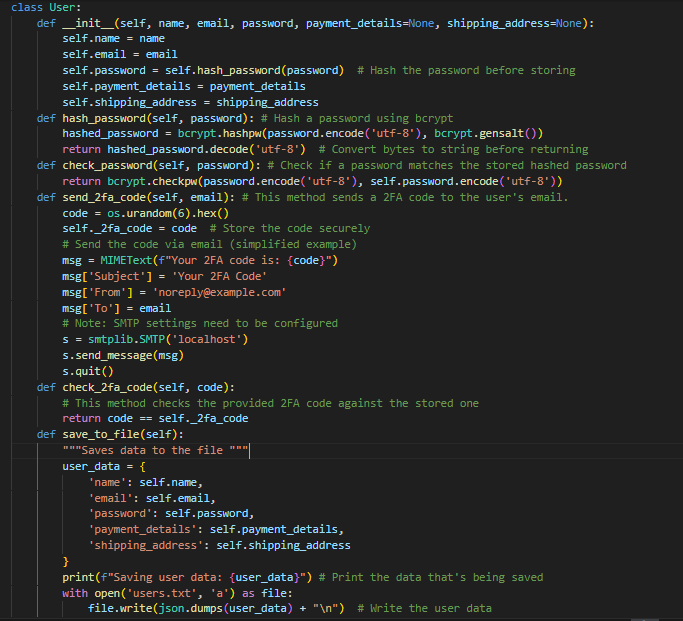
The codebase is structured following the principles of object-oriented programming, which promotes code reusability and logical organization. Classes were created in the online\_shopping\_app.py file while the CLI was separately created in the cli.py. For a smooth automated testing, I also included the unit test and white box testing codes in the same directory.

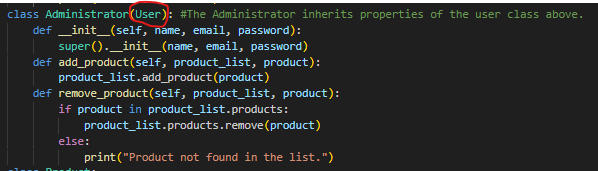


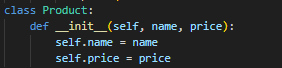
Here’s an overview of the key Classes:

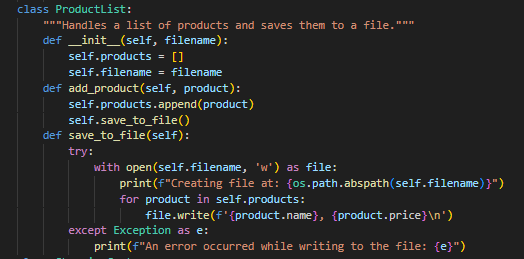
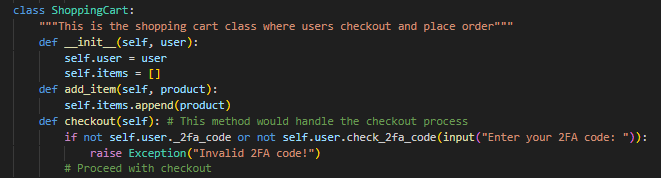
*online\_shopping\_app\_main.py*

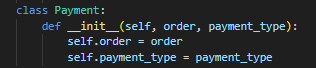
* User: Manages user information, authentication, and session management.

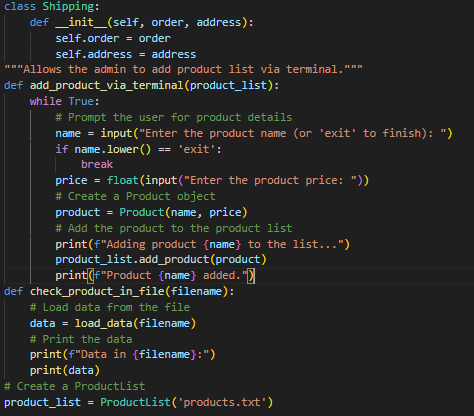
(GeeksforGeeks. (2022)).

* Administrator: Inherits from User, with additional privileges for managing the product inventory.
* Product: Represents individual items available for purchase, encapsulating details such as name and price.



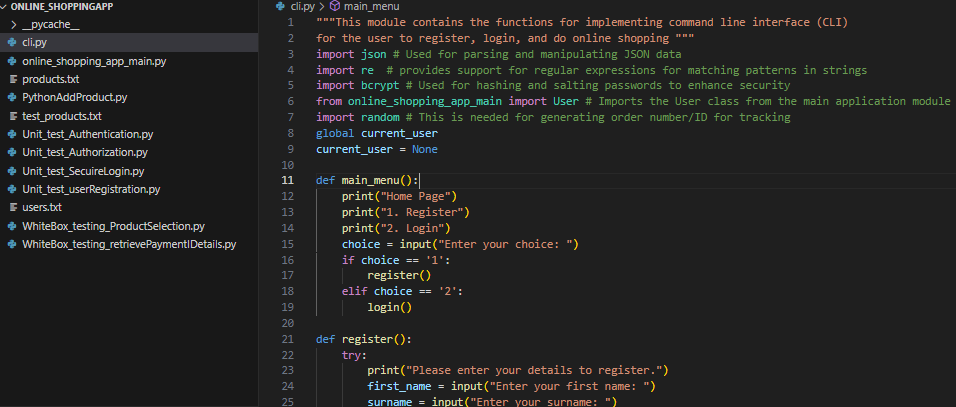
* ProductList: A collection of Product objects, providing methods to add and remove items from the inventory.(GeeksforGeeks. (2022)).
* ShoppingCart: Tracks user-selected products, quantities, and the total cost, offering a seamless checkout experience.
* Order: Processes and finalizes the user’s purchase, interfacing with the ShoppingCart to complete the transaction.
* Payment: Handles payment authorization, processing, and secure storage of payment details.



* Shipping: Manages logistics related to the delivery of orders, including address verification and tracking.

*cli.py*

This application works from command line interface and the cli.py file facilitates the flow between different modules and ensuring a smooth user journey from start to finish (Shehmir et al.,2024).

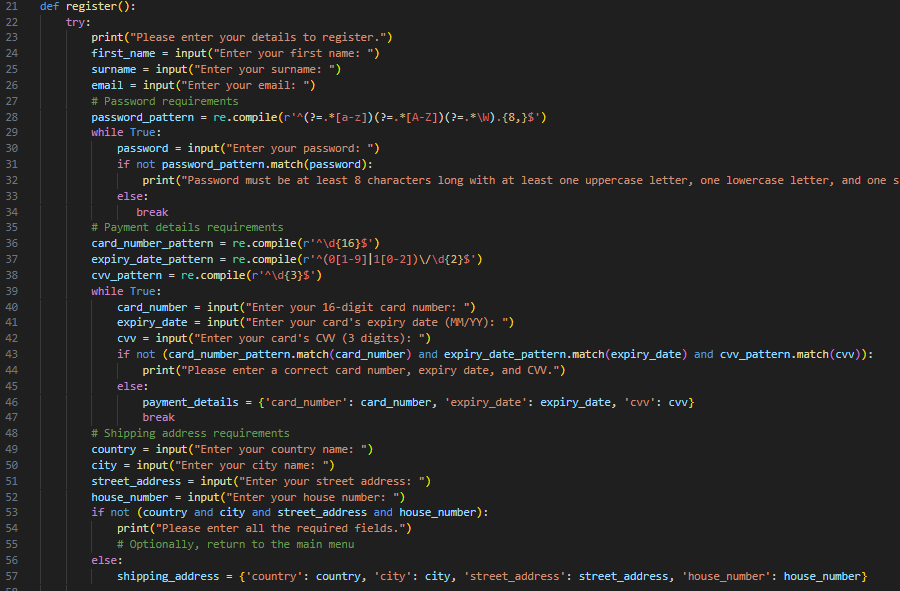


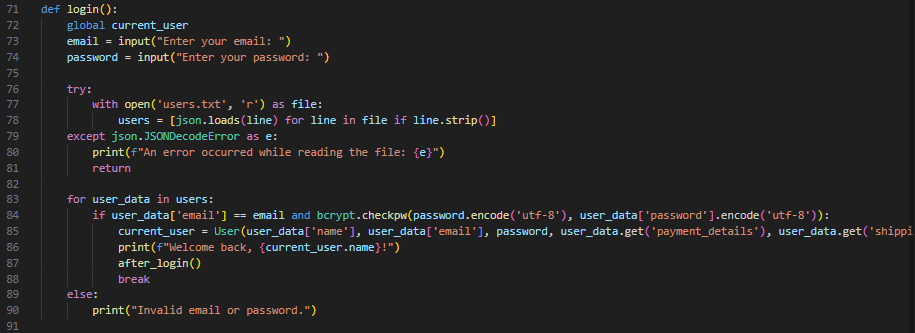
**Application Functions**

**User Authentication**

The journey begins with a secure login system. New users are invited to create an account by providing personal details, which are then encrypted using advanced hashing techniques. I used bcrypt for its reliable password hashing and salting capabilities, ensuring that user credentials are stored securely and are impervious to common security threats (Nielson & Monson, 2019).

Existing users can log in with their credentials, which are verified against the encrypted data stored on the system. This two-step process—hashing on account creation and verification on login—provides a secure authentication framework that protects user data ‌(GeeksforGeeks, 2022).

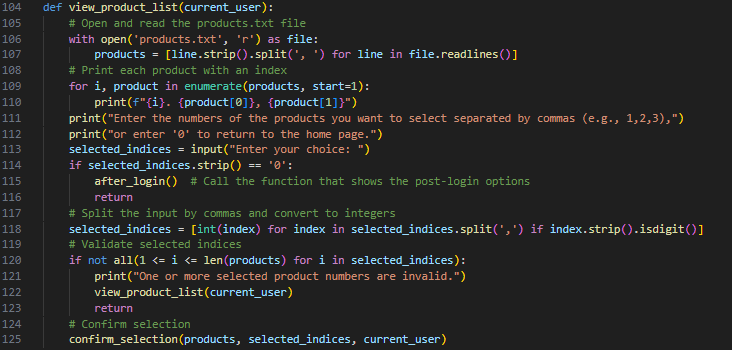




(Stack Overflow, n.d.; GeeksforGeeks, 2022).

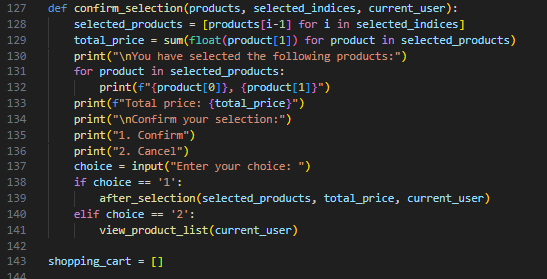
**Viewing Products**

The application dynamically generates a product catalog from the products.txt file, presenting users with an up-to-date list of items complete with descriptions and pricing. This feature allows users to browse our extensive product range and make informed decisions on their purchases.



(Stack Overflow. (n.d.)).

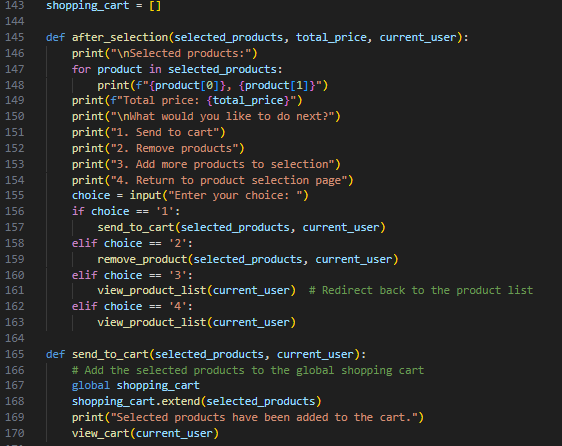
**Selecting Products**

Users can select products for purchase by entering the corresponding product numbers. The application then validates the user’s input, ensuring that only available products are added to the shopping cart. This validation step is crucial for maintaining the integrity of the user’s selection process.

(GeeksforGeeks, 2022).

**Shopping Cart**

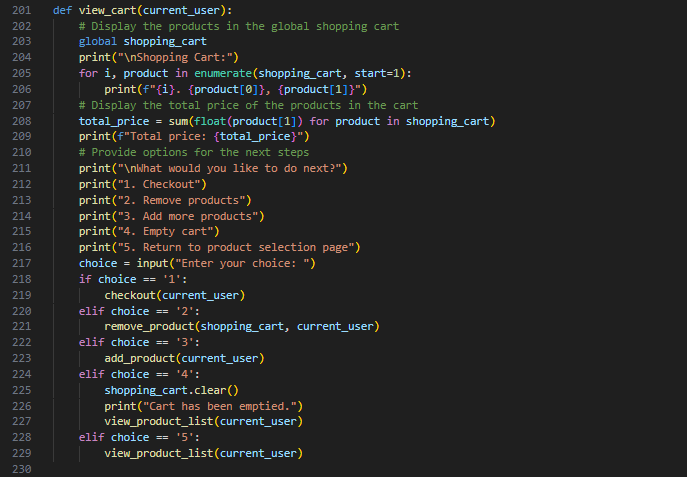
Once selections are made, users can review their shopping cart, which provides a summary of selected items, quantities, and the total price. The shopping cart is designed to give users full control over their selections, allowing them to modify quantities or remove items before proceeding to checkout.

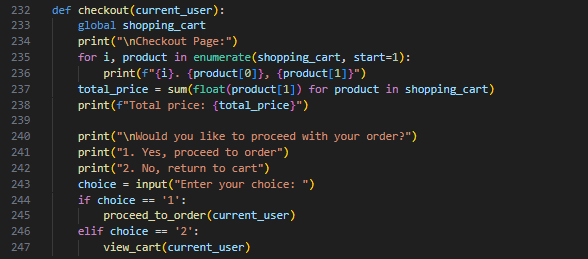


(GeeksforGeeks, 2022).

**Checkout Process**

The checkout process is fortified with a two-factor authentication system. Users receive a unique code via email, which must be entered to verify their identity. This additional security measure safeguards against unauthorized access and ensures that only legitimate transactions are processed (GitHub. (n.d.).

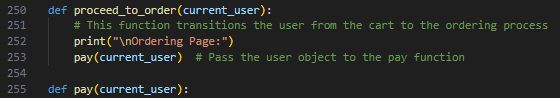




(GitHub. (n.d.)).

**Ordering Process**

After selecting their desired products, users can proceed to place an order through a streamlined ordering process. The ‘proceed\_to\_order’ function transitions the user from the shopping cart to the ordering page, where they can review their selections one final time before proceeding to payment. This function serves as a bridge to the ‘*Pay*’ function, ensuring a smooth transition from cart management to transaction processing.

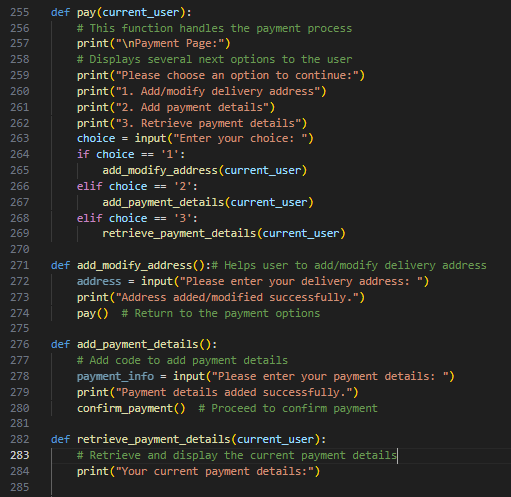


**Payment Process**

In the pay function users are presented with options to add or modify their delivery address, add new payment details, or retrieve existing payment information. This multi-step process is designed to give users full control over their transaction details, allowing them to choose the most convenient option for their needs.

* **Add/Modify Delivery Address**: Users can input or update their delivery address, ensuring that their order is shipped to the correct location.
* **Add Payment Details**: New payment information can be securely entered and saved for the current transaction and future use.
* **Retrieve Payment Details**: Users can view and confirm their saved payment details, streamlining the checkout process.

Once the payment details are confirmed, a unique order tracking number is generated, providing users with the ability to track their order’s progress post-purchase. The application confirms the successful placement of the order and offers users the option to return to the home page or sign out.



**Security Features**

My application incorporates several layers of security to protect user data and ensure safe transactions:

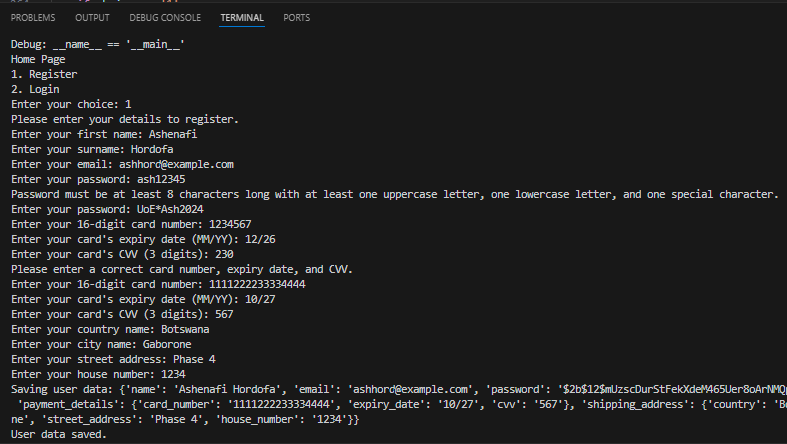
* **Secure Password Handling**: I used bcrypt to hash and salt passwords, providing a secure method of storing sensitive user credentials.
* **Two-Factor Authentication (2FA)**: An extra layer of security is added during the checkout process, requiring users to verify their identity with a unique code sent to their email.
* **Authorization**: Users are granted access only to the features and information appropriate to their role, preventing unauthorized actions within the application.
* **Secure Transmission**: All sensitive information is transmitted securely, adhering to industry-standard encryption protocols.
* **Secure Storage**: Payment details are retrieved and stored securely, following best practices for data protection and privacy.

**Testing**

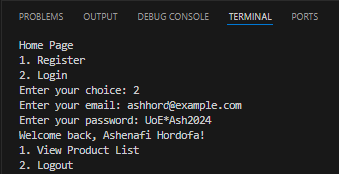
I have tried to ensure quality and security of the application by applying several testing protocols:

1. **Manual Testing**: I have tested each functionality of the application manually, ensuring that every feature works as intended. Screenshots of code executions and outputs are included to demonstrate the testing process.

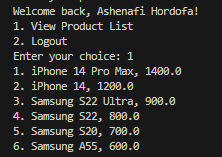
i) User Registration testing output

****

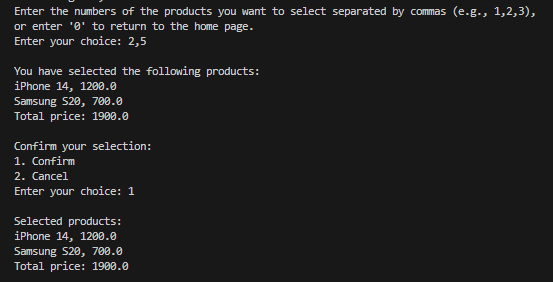
ii. User Login and Authentication testing output



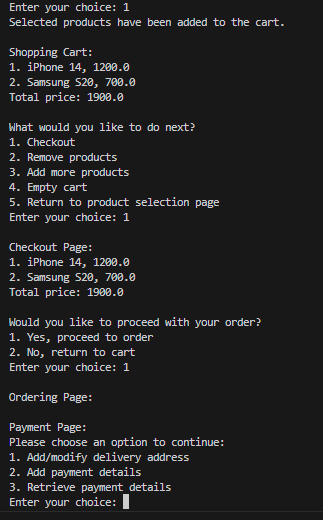
iii. View Product List testing output



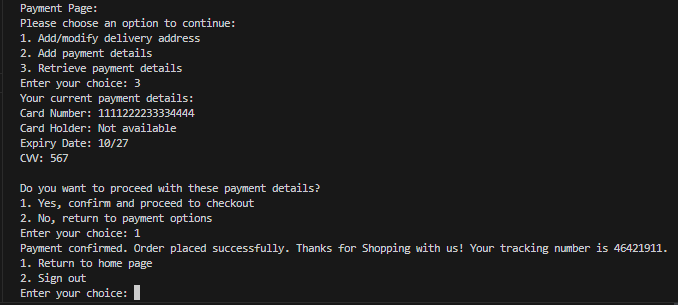
iv. Select Product testing output



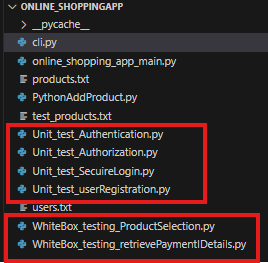
v. Ordering process testing output



vi. Payment and Order Confirmation testing output

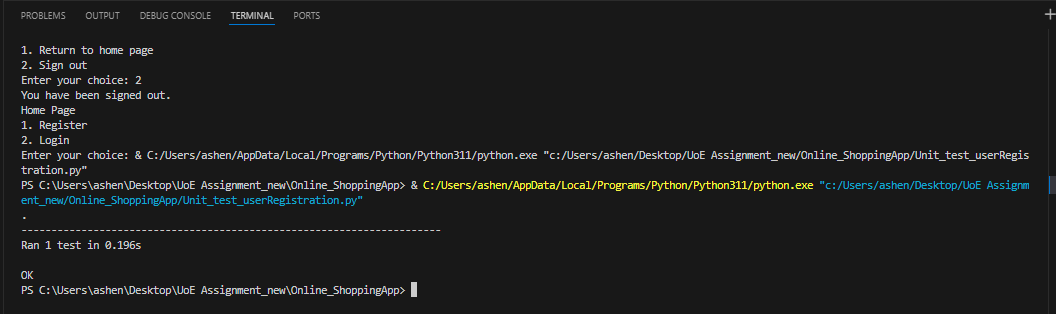


1. **Automated Testing**: several unit tests and white box tests were conducted to check the security and functionality of the application.

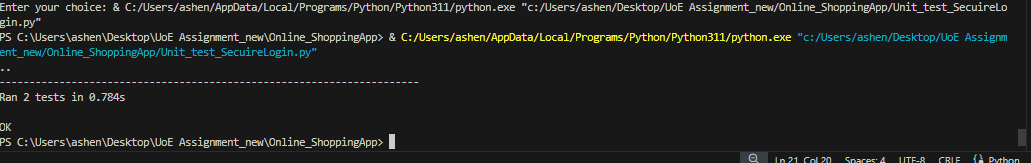


* 1. **Unit Testing**: Several components were tested in isolation to verify its functionality and robustness (BrowserStack. (n.d.).

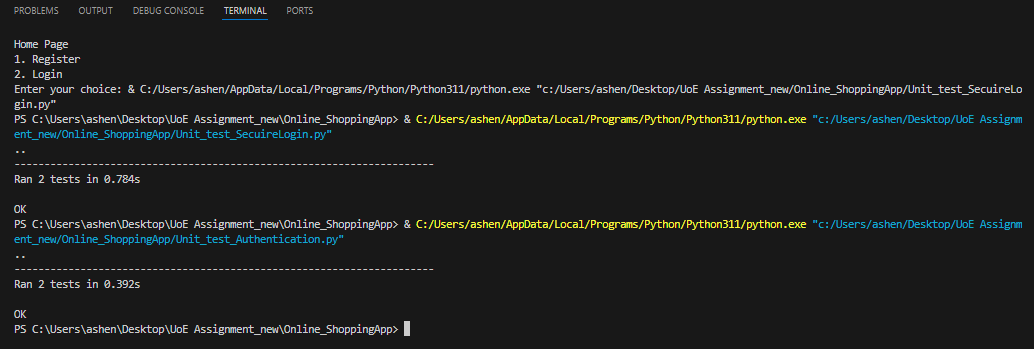
i) User registration unit test output



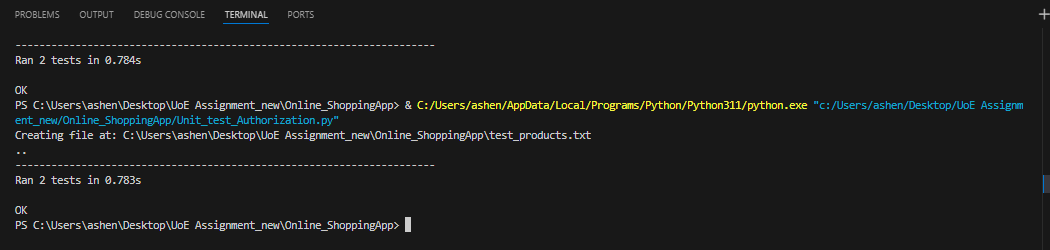
ii) Secure login unit test output



iii) Authentication unit test output



iv)Authorization unit test output



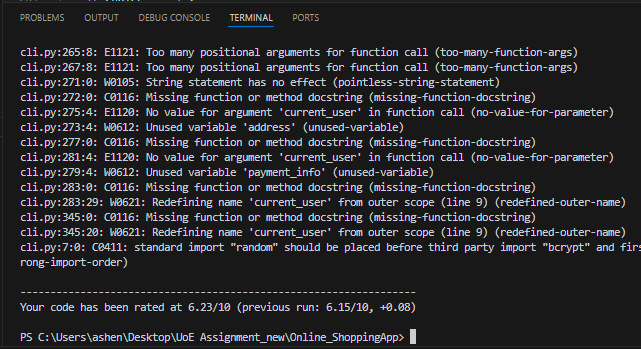
* 1. **White Box Testing**: Automated testing was run for several functionalities to examine the internal structures of the application to ensure that all logic paths are executed correctly.
* A screen shot of a computer

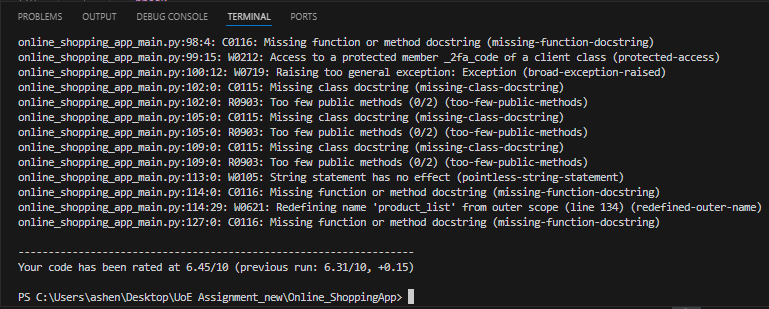
  Description automatically generated

A screen shot of a computer

Description automatically generated

* 1. **Linting with Pylint**: The code was analyzed by running pylint and the quality was improved progressively. I will continue to improve the quality based on the analysis from pylint.





**References**

BrowserStack. (n.d.). Unit Testing in Python: Detailed Tutorial. [online] Available at: <https://www.browserstack.com/guide/unit-testing-python> [Accessed 15 May 2024].

‌GeeksforGeeks. (2022). How To Hash Passwords In Python. [online] Available at: <https://www.geeksforgeeks.org/how-to-hash-passwords-in-python/> [Accessed 25 May 2024].

GitHub. (n.d.). Online-Shopping-System-using-python/Fashion.py at master · bsdharshini/Online-Shopping-System-using-python. [online] Available at: https://github.com/bsdharshini/Online-Shopping-System-using-python/blob/master/Fashion.py [Accessed 07 May 2024].

GitHub. (n.d.). Shopping-Cart-Project/main.py at main · jig-josh/Shopping-Cart-Project. [online] Available at: https://github.com/jig-josh/Shopping-Cart-Project/blob/main/main.py [Accessed 06 May 2024].

Kong, Q., Siauw, T. and Bayen, A. (2020). Python Programming and Numerical Methods A Guide for Engineers and Scientists. [online] Available at: <https://library.samdu.uz/files/8ea2cbd32e87c6d25a17c4b2510501d0_Python_Programming_and_Numerical_Methods_A_Guide_for_Engineers_and.pdf> [Accessed 21 May 2024]

Nielson, S. and Monson, C. (2019). *Practical Cryptography in Python Learning Correct Cryptography by Example*. [online] Available at: https://edu.anarcho-copy.org/Programming%20Languages/Python/Practical\_Cryptography\_in\_Python\_Learning\_Correct\_Cryptography\_by.pdf [Accessed 22 May 2024].

‌‌‌Shehmir, S.S., Nazar, M., Zaveri, A.A., Sami, N., Mashood, R., Faisal, N., Waqas, A. and Naeem, R. (2024). Enhancing the Shopping Experience in E-Commerce: A Path to Improvement – Buy the Best. Bulletin of Social Informatics Theory and Application, [online] 8(1), pp.152–164. doi:https://doi.org/10.31763/businta.v8i1.680.

Stack Overflow. (n.d.). *authentication - Salt and hash a password in Python*. [online] Available at: <https://stackoverflow.com/questions/9594125/salt-and-hash-a-password-in-python> [Accessed 15 May 2024].