

Development Individual Project: Coding Output and Evidence of Testing

I have developed an e-commerce store, with APIs built using the Python Flask framework. The store allows users to browse and purchase products while providing administrators with the ability to upload and manage products.

APIs facilitate communication between the front-end store and the back-end SQL database. Using HTML to demonstrate the shopping experience UI.

To enhance security, I implemented security mechanisms for the attack surface of the domain. These measures ensure the protection of sensitive data and maintain the integrity of the platform.

Furthermore, I conduct testing using Pylint to ensure code quality and adherence to best practices.

Overall, integrating Python Flask for the API and HTML for the front-end store with robust security implementation is a scalable and reliable e-commerce solution.

Codebase Github Repository link: <https://github.com/rocioaltairr/The-CURE-Store-Apis>

System Requirements

- *Python 3.8*
- *Flask-SQLAlchemy 3.1.1+*
- *PyJWT 2.8.0+*
- *Flask-JWT-Extended 4.6.0+*
- *flask-smorest 0.42.3+*
- *pycrypto 2.6.1+*

		Actors	
		Customer	Store Administrator
Actions	Create account	•	•
	Add product to catalogue		•
	Browse products	•	•
	Modify/delete products		•
	Create order	•	
	Add payment method	•	
	Modify/delete payment method	•	
	Modify/delete own account	•	•
	Modify/delete other account		•

Figure 1 – Functional requirements by user type

Design Patterns

Model-View-Controller (MVC) - SoC Principle

MVC architecture helps us to control the complexity of the application by dividing it into three components, it provides the idea of “Separation of concerns”.

In a typical Flask application, controllers are organized within a designated folder (e.g., resources) while models are stored in a separate folder (e.g., models). This modular structure helps maintain code organization and scalability by clearly the responsibilities of each component.

View is the part of the application responsible for displaying the data. It can be a simple return string or a fully-fledged HTML page.

Singleton

SessionStorage is an HTML 5 browser standard to enables storage and unique retrieval of simple data on the client. This is a global object that maintains a storage area that's available for the duration of the page session. SessionStorage provides a reliable way to maintain stateful data while reducing the risk of data leakage or unauthorized access across browsing sessions.

Ensuring GDPR Compliance and User Data Acceptance: Role-based Access Control for Enhanced Security

Requesting customer acceptance of data usage for GDPR

In addition to these application-specific functional requirements, relevant legislative and best practice requirements will apply, such as the General Data Protection Regulation (GDPR) and the Open World Application Security Project (OWASP) Secure Coding Practices. Where user agreement is required, such as for GDPR (European Union, 2016), the user will be presented with the relevant information and asked to continue through the user view, as shown by the activity diagram.

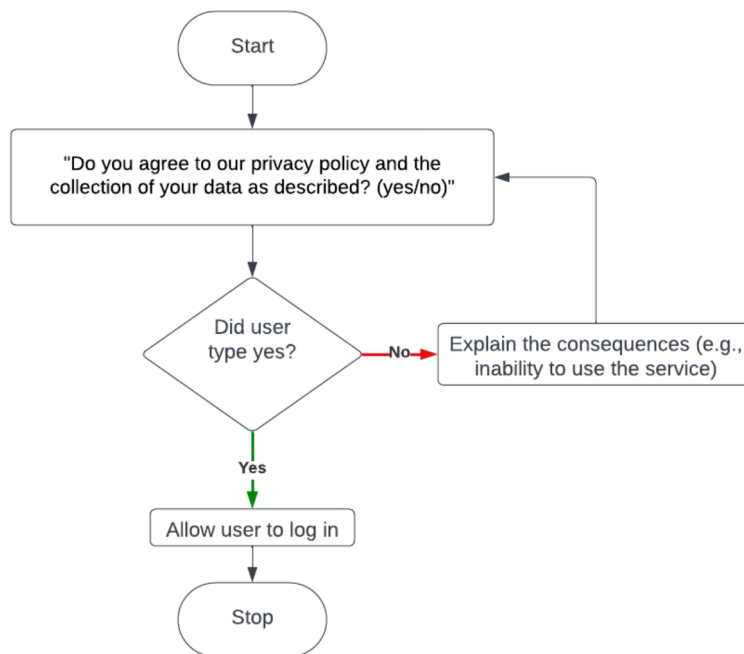


Figure 2 - Activity diagram of requesting customer acceptance of data usage for GDPR

Create Account

Actor:

Name:

Username:

Code:

By creating an account, you agree to our privacy policy and data collection terms. This Personal Data Processing Policy outlines how The CURE E-commerce Store, processes personal data about individuals with whom it interacts in various contexts (referred to as "you" hereafter). The CURE E-commerce Store ("We" or "The CURE E-commerce Store") acts as the controller of your personal data in accordance with the General Data Protection Regulation (EU) 2016/679 ("GDPR"). As the controller, we collect, store, and utilize your personal data for the purpose of conducting our business activities.

☐

Please agree to our privacy policy and data collection, otherwise you couldn't use the service

Create Account

Figure 3 - Screenshot of the application creates an account if the user doesn't confirm the policy and will explain the consequences to the user

Implementation of Role-based access control (RBAC)

Users can view store products and make purchases but cannot add items to the store.	Administrators have additional privileges allowing them to add stores and products.
---	---

This page says
Login successful!

Login

Actor:
User

Username:
Rocio

Password:

Login

Sign up

Figure 4 - Screenshot of the application login page when user login

This page says
Login successful!

Login

Actor:
Admin

Username:
Admin1

Password:

Login

Sign up

Figure 5 - Screenshot of the application login page when the administrator login

Stock

Welcome, Rocio you are User

Flowers

Sunflower - Price: \$2 - Stock: 10 0

Add to cart

Rose - Price: \$3 - Stock: 8 1

Add to cart

Tools

Adobe Ps - Price: \$200 - Stock: 4 0

Add to cart

Log Out Delete Account

Figure 6 - Screenshot of the application shopping cart page after the user successfully login

Welcome, Admin1 you are Admin

Flowers

Add Item

Sunflower - Price: \$2 10

Rose - Price: \$3 8

Tools

Add Item

Adobe Ps - Price: \$200 4

Add Store Log Out Delete Account

Figure 7 - Screenshot of the application manage store page after the user successfully login

Security Risks and Countermeasures

The online store has been designed with consideration to the potential security risks and the implementation of appropriate countermeasures to protect sensitive user information, prevent unauthorized access, and ensure the integrity of the system (Rehim, 2016). Within the system design, it is demonstrated how these risks can be mitigated through sanitizing and validating user inputs, with reference made to the OWASP Top Ten list of threats that developers should be aware of (OWASP, 2021). Details of the perceived threats and countermeasures are listed below:

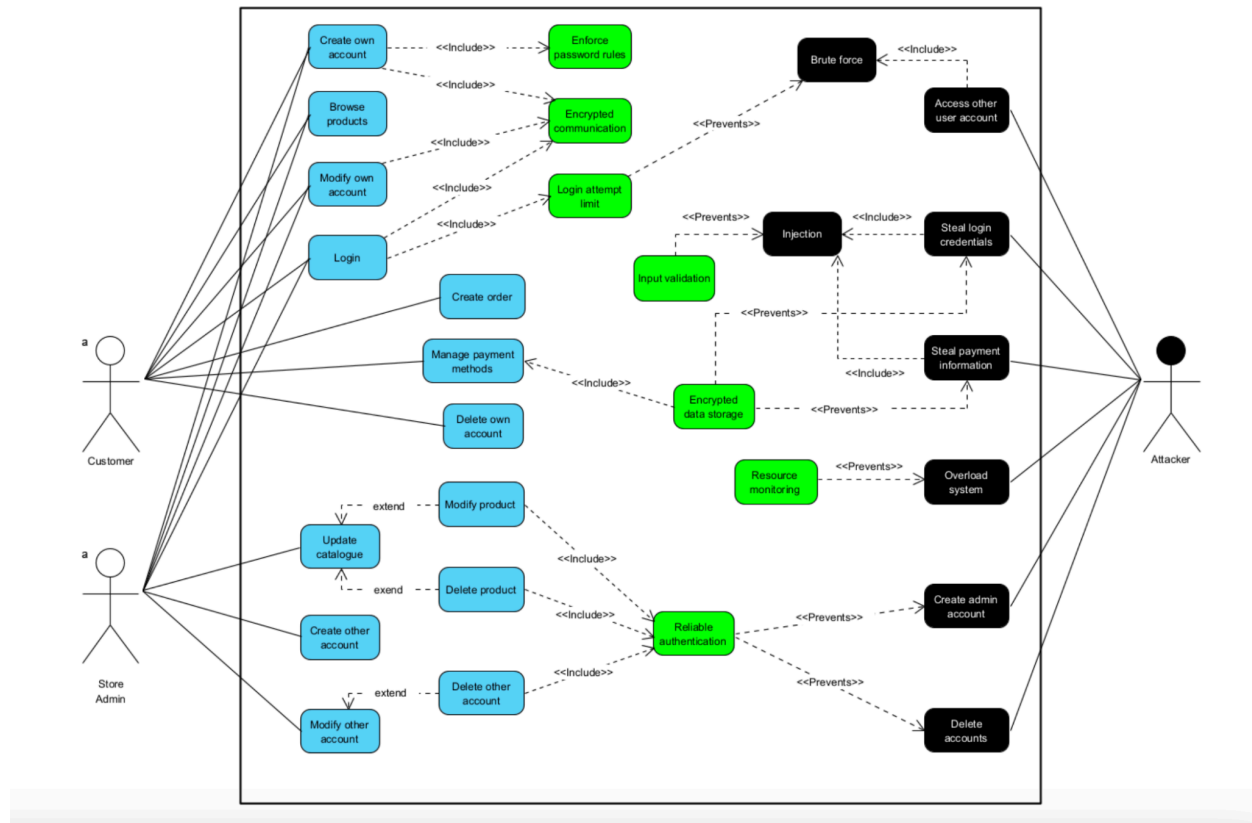


Figure 8 - System misuse case diagram

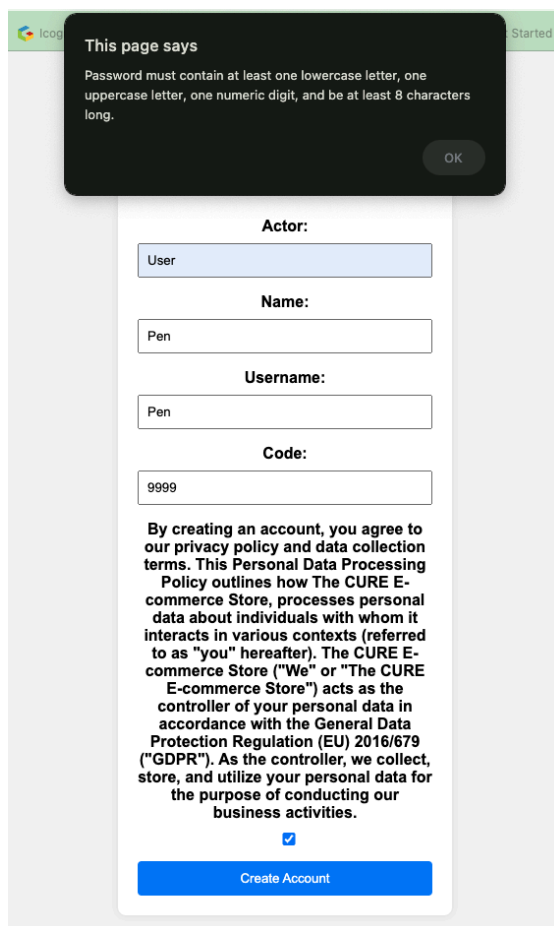
To model these threats and mitigations, the misuse case diagram (Pauli & Xu, 2005) shown in Figure 8 was produced. The identified risks (black use cases) are linked to the expected activities of users of the system (blue use cases). Mitigations are also included (green use cases) to show which expected activities will be protected and which risks will be mitigated by the relevant countermeasure.

Brute Force Attacks

A brute force attack is a hacking method that uses trial and error to crack passwords and login credentials. The hacker tries multiple usernames and passwords, often using a computer to test a wide range of combinations until they find the correct login information. The name "brute force" comes from attackers using excessively forceful attempts to gain access to user accounts.

Strong Password Policy

A strong password policy helps prevent brute-force attacks by increasing the complexity of passwords, making the password harder for attackers to crack.



The screenshot shows a 'Create Account' form with a dark notification box at the top. The notification box contains the text: 'This page says Password must contain at least one lowercase letter, one uppercase letter, one numeric digit, and be at least 8 characters long.' with an 'OK' button. The form fields are: 'Actor:' with a 'User' input, 'Name:' with a 'Pen' input, 'Username:' with a 'Pen' input, and 'Code:' with a '9999' input. Below the fields is a paragraph of text: 'By creating an account, you agree to our privacy policy and data collection terms. This Personal Data Processing Policy outlines how The CURE E-commerce Store, processes personal data about individuals with whom it interacts in various contexts (referred to as "you" hereafter). The CURE E-commerce Store ("We" or "The CURE E-commerce Store") acts as the controller of your personal data in accordance with the General Data Protection Regulation (EU) 2016/679 ("GDPR"). As the controller, we collect, store, and utilize your personal data for the purpose of conducting our business activities.' followed by a checked checkbox and a 'Create Account' button.

Figure 9 - Screenshot of Create Account page

The CURE online store password policy: Password must contain at least one lowercase letter, one uppercase letter, and one numeric digit, and be at least 8 characters long.

AES Algorithm Encryption

AES is a symmetric encryption algorithm widely regarded as secure when implemented correctly. It uses a block cipher with a variable key length (128, 192, or 256 bits), making it computationally infeasible for attackers to decrypt data without the encryption key.

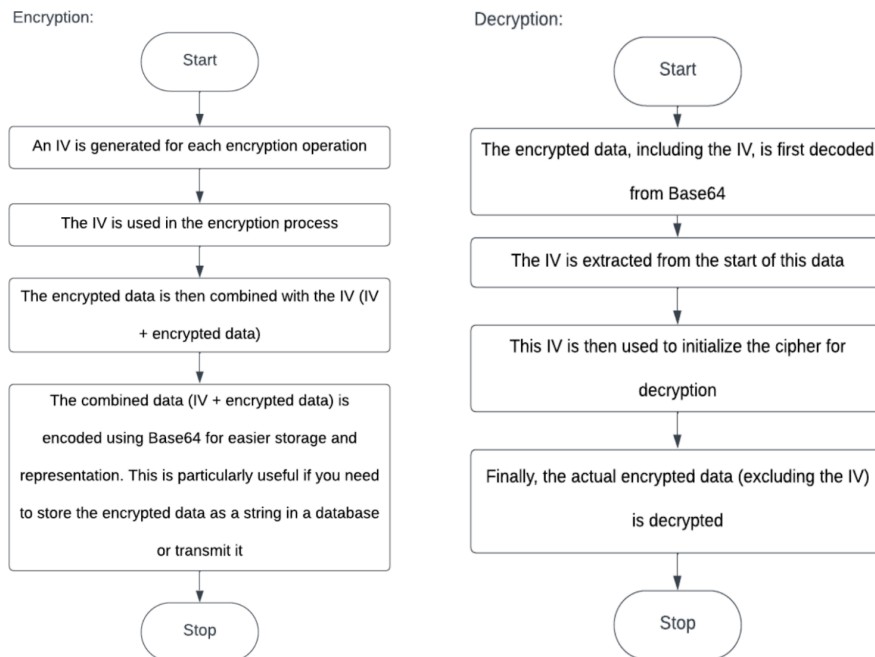


Figure 10 - AES encryption and decryption processes (note: IV = Initialisation vector)

The `encrypt.py` file implements the AES algorithm for the encryption and decryption processes on passwords. The purpose is to provide secure storage of user passwords. When a user creates an account, their password is encrypted using the AES algorithm in CBC mode with a predefined initialization vector (IV). Then base64 encodes the encrypted message before returning it. This encrypted password is then stored securely in the database.

Conversely, the `decrypt` function is used during the login process. It takes the encrypted password stored in the database, decrypts it using the same AES algorithm and IV, and then compares it with the password provided by the user during login. If the decrypted password matches the user's input, access is granted.

This mechanism helps to minimize the risk of unauthorized access to user passwords, enhancing the overall security of the application.

Denial of Service (DoS) Attack

Many DoS attacks, including those targeting login mechanisms, are automated and rely on repeated, rapid requests to overwhelm the system. Enforcing a login limit would hinder the attacker's efforts to disrupt service availability.

Set login attempts limit

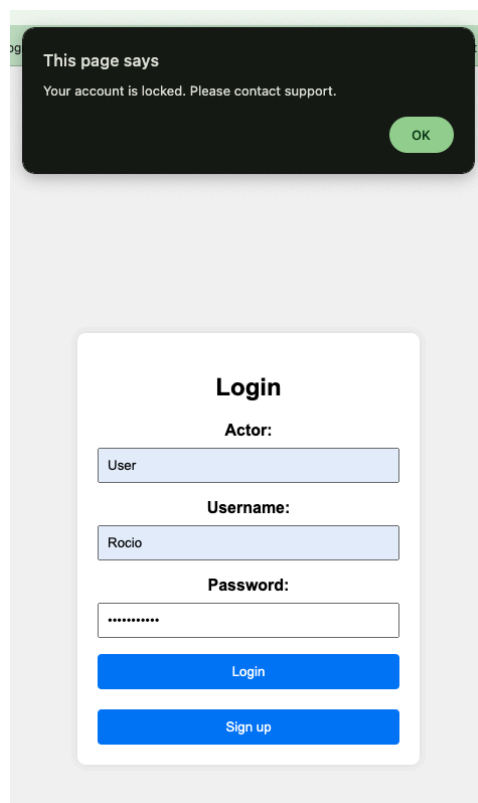
The image shows a user interface for a login system. At the top, there is a dark grey notification box with white text that reads: "This page says" followed by "Your account is locked. Please contact support." and a green "OK" button. Below this, the main login form is centered. It has a title "Login" and three input fields: "Actor:" with a placeholder "User", "Username:" with a placeholder "Rocio", and "Password:" with a masked password "*****". At the bottom of the form are two blue buttons: "Login" and "Sign up".

Figure 11 - Maximum Login Attempts 3 times for preventing Broken Access Control

Injection Attacks

If the application is vulnerable to SQL injection or other injection attacks, attackers may be able to gain unauthorized access to sensitive information, including passwords. Employing parameterized queries and input validation can help prevent injection attacks.

Figure 12 - Implementing an API endpoint to retrieve user account data, showcasing vulnerability to SQL injection.

Figure 13 - Simulate a potential attacker's SQL injection attempt by using the payload 'OR '1'='1' in a Swagger UI platform environment to demonstrate the vulnerability and potential retrieval of user data.

Figure 14 - Implementing an API endpoint to retrieve user account data implementation of parameterized queries for preventing SQL injection attack

Dictionary Attack

A dictionary attack is a method of breaking into a password-protected computer, network, or other IT resource by systematically entering every word in a dictionary, or word list, as a password. Attackers may attempt to use commonly used password keys in the dictionary to guess passwords.

Prevent usage of password key

For the user data stored in the database, I use a code instead of the password itself as the key for encryption.

	id	actor	name	user_name	code	login_attempts
1	1	User	Rocio	Rocio	Co/gFT9Eo+RERZd56g/G1w==	0

Figure 15 - User's account data stored on the database showcases the key 'code' which stores the password

Cross-Site Scripting (XSS)

JWTs are typically stored in browser cookies or local storage. By using the HttpOnly flag for cookies or implementing proper security measures for local storage, JWTs can be protected from XSS attacks, as the browser will prevent client-side scripts from accessing the JWT. When the Administrator attempts to add a store, the application verifies the JWT to ensure that it's valid and contains the necessary permissions.

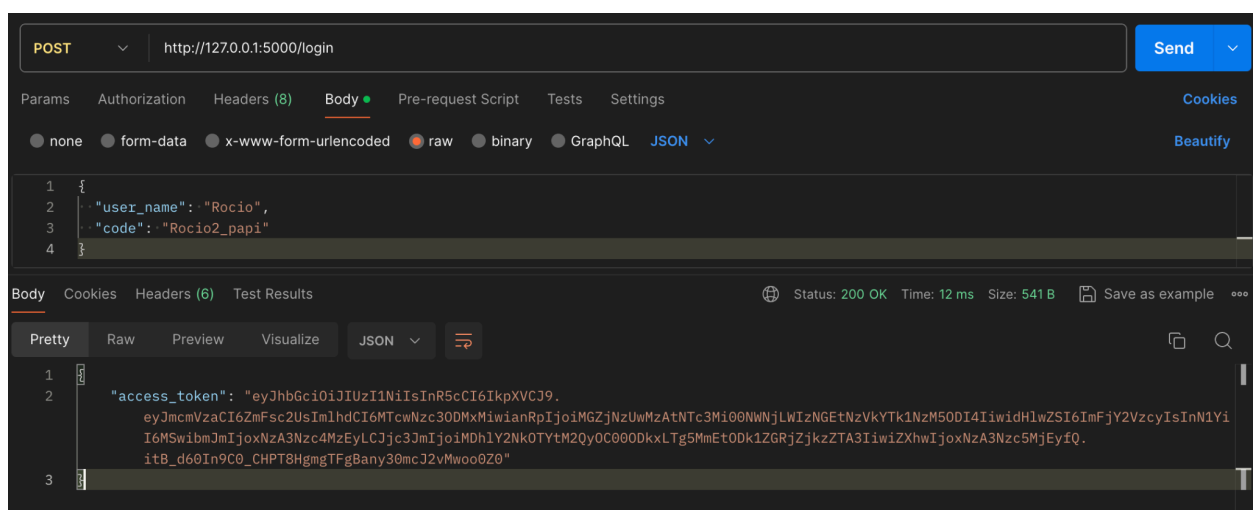


Figure 16 - Screenshot of a login API request and its result. With the JWT mechanism

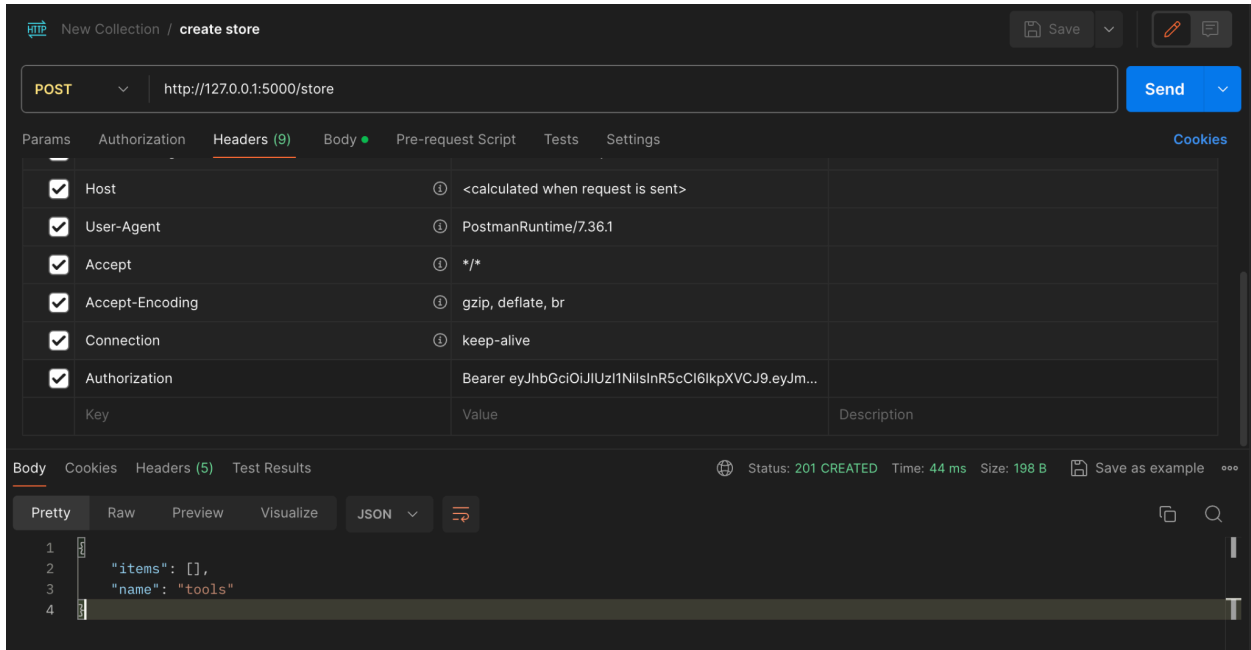


Figure 16 - Screenshot of creat store request and its result. With the JWT mechanism, you need the access token to add a store to the database

Testing with Pylint

The files under resources folder

```

rocio@V56QGT96KX resources % pylint .

-----
Your code has been rated at 10.00/10 (previous run: 9.92/10, +0.08)

```

Figure 16 - Screenshot of output captures after running Pylint of files under resources folder

In files under the models folder, the error doesn't influence the code safety

```

rocio@V56QGT96KX models % pylint .
***** Module models.store
store.py:6:0: R0903: Too few public methods (0/2) (too-few-public-methods)
***** Module models.sale_item
sale_item.py:6:0: R0903: Too few public methods (0/2) (too-few-public-methods)
***** Module models.item
item.py:6:0: R0903: Too few public methods (0/2) (too-few-public-methods)
***** Module models.account
account.py:20:0: C0303: Trailing whitespace (trailing-whitespace)
account.py:6:0: R0903: Too few public methods (0/2) (too-few-public-methods)

-----
Your code has been rated at 8.65/10 (previous run: 8.65/10, +0.00)

```

Figure 17 - Screenshot of output captures after running Pylint of files under models folder

```
● rocio@V56QGT96KX store-apis % pylint app.py

-----
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

Figure 17 - Screenshot of output captures after running Pylint of qpp.py file

Word count: 1520

References

European Union. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). Official Journal of the European Union: L119, 1-88.

Huang, L. (2022) *Full-stack E-commerce online store: A web application for purchasing and uploading products for every user*. Vaasa University of Applied Sciences.

OWASP (2021). OWASP Top Ten. Available at: <https://owasp.org/top10> (Accessed: 12 December 2023).

Auth0. (n.d.). How to Handle JWT in Python. Available at: <https://auth0.com/blog/how-to-handle-jwt-in-python/>

Pillai, A. B. (2017) Software architecture with Python: design and architect highly scalable, robust, clean, and high performance applications in Python. 1st ed. PACKT Publishing.

Tenzin, S., Lhamo, T. & Dorji, T. (2022) Design and Development of E-commerce: Web-application for Cooperative Store. *International Research Journal of Engineering and Technology (IRJET)* 9(2): 843-847.

William, G., Anthony, R. & Purnama, J. (2020) 'Development of NodeJS based Backend System with Multiple Storefronts for Batik Online Store', *International Conference on Engineering and Information Technology for Sustainable Industry (ICONETSI 2020)*. Tangerang, Indonesia, 28-29 September. New York: ACM.