DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-116
Course Title: Object Oriented Programming
Complex Engineering Problem
FE Batch 2023, Spring Semester 2024
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1	Muhammad Qambar Hussain	CS-23125
S2	Asadullah Nizami	CS-23092
S3	Ali Raza Baloch	CS-23130

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the class diagram meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) [4 marks]						
1	2	3	4			
The class diagram does not meet the desired specifications and is producing incorrect outputs.	The class diagram partially meets the desired specifications and is producing incorrect or partially correct outputs.	The class diagram meets the desired specifications but is producing incorrect or partially correct outputs.	The class diagram meets all the desired specifications and is producing correct outputs.			
Criterion 2: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) **[6 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 3: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 4: How friendly is the application interface? (CPA-1, CPA-3) **[2 marks]**						
1	2	3	4			
The application interface is difficult to understand and use.	The application interface is easy to understand and but not that comfortable to use.	The application interface is very easy to understand and use.	The application interface is very interesting/ innovative and easy to understand and use.			
Criterion 5: How does the student performed individually and as a team member? (CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	**The student worked on the assigned task, and accomplished goals partially.**	**The student worked on the assigned task, and accomplished goals satisfactorily.**	**The student worked on the assigned task, and accomplished goals beyond expectations.**			
Criterion 6: Does the report adhere to the given format and requirements? **[2 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

____
Teacher's Signature

Table of Contents

COMPLEX ENGINEERING PROBLEM:

PROBLEM DESCRIPTION:

The project aims to develop a comprehensive e-commerce website using Django. This website will provide users with an intuitive and secure shopping experience, enabling them to browse products, manage their shopping carts, and view their order history. The primary focus is on creating a seamless and efficient shopping process while incorporating robust security measures to protect user data.

DISTINGUISHING FEATURES OF OUR PROJECT

Our e-commerce platform is structured using object-oriented principles, which enhance modularity, code reuse, and maintainability. Classes and objects represent various entities such as products, users, and orders, encapsulating their attributes and behaviors effectively.

Structured Project Development: Organized Mapping and Systematic Model Management:

In our project, every aspect is meticulously mapped to three fundamental components: views for handling business logic, URLs for routing requests, and templates for rendering user interfaces. This structured approach ensures clarity and consistency across multiple apps within our system. As the project scales, this methodology becomes crucial in maintaining integrity and scalability across various modules.

Furthermore, our approach includes rigorous management of data models. Each model is first registered with the admin interface to facilitate seamless content management. Subsequently, careful migration processes are implemented to evolve database schemas in alignment with the application's evolving needs. This meticulous handling of data models enhances maintainability and ensures robustness as our project continues to grow.

Django Framework:

We leveraged the Django framework, renowned for its robustness and scalability, to develop the application. Django’s powerful ORM (Object-Relational Mapping) system facilitates seamless database interactions through Python-related queries.

SQLite Database:

Using SQLite for the database allows for efficient storage and retrieval of data. Django’s ORM handles complex SQL queries behind the scenes, providing a high-level abstraction over the database operations.

HTTP Methods and Security:

1)Request Methods: The application utilizes HTTP GET and POST methods to handle client requests. The Get method retrieves data, while the POST method submits data to the server

2)CSRF Protection: Cros-Site Request Forgery (CSRF) protection is enabled to safeguard against malicious activities by ensuring that requests come from authenticated users.

3)Validation checks:
Rigorous validation checks are in place for forms and data inputs, ensuring data Integrity and security

User Interface and Experience:

1)Navbar: A dynamic navigation bar provides easy access to different sections of the website, enhancing user navigation.

2)Templates: Django templates are used to render HTML content dynamically, supporting the display of product lists, shopping carts, and user information

3) On-Spot Quantity Stock Updates: Real-time updates for product quantities ensure accurate stock levels are maintained, preventing overselling.

Security and User Management:

1)Account Setting: Users can manage their account settings, including password changes and profile updates.

2)CAPTCHA: Integrated CAPTCHA mechanisms prevent automated bots from accessing the platform, ensuring that interactions are from genuine users.

3)Session management: Unique session keys manage user sessions, providing secure and personalized user experiences**.**

4)EMAIL confirmation: Sends a confirmation email with a time-limited link to ensure secure and timely user verification.

Order History and Navigation:

Our e-commerce platform provides a comprehensive order history feature that allows users to view their past purchases with options to filter by date or specific items. This functionality offers detailed insights into their shopping activities, enabling them to track previous orders efficiently. Additionally, users benefit from seamless navigation as they can easily return to the shopping process after reviewing their order history, thus enhancing the overall user experience by maintaining a smooth and intuitive flow throughout their interaction with the platform.

Structural Scalability:

The platform is designed with a scalable architecture that supports future enhancements and can handle higher user loads, ensuring that the system grows alongside the business needs. Its extensible design features modular code and well-defined interfaces, allowing for the straightforward integration of new functionalities such as payment gateways and multi-vendor support. This structural flexibility ensures that the platform can adapt to evolving market demands and technological advancements without significant restructuring.

Models:

In our e-commerce platform, we use several models to manage different aspects of the system. These include models for `Product`, `Abstract user`, `Customer user`, `Password`, `History`, and `cart`, each handling specific data and functions related to their respective areas. Instead of using SQL, we opted for SQLite to manage our database, offering a lightweight and efficient solution for our needs.

Views and Templates:

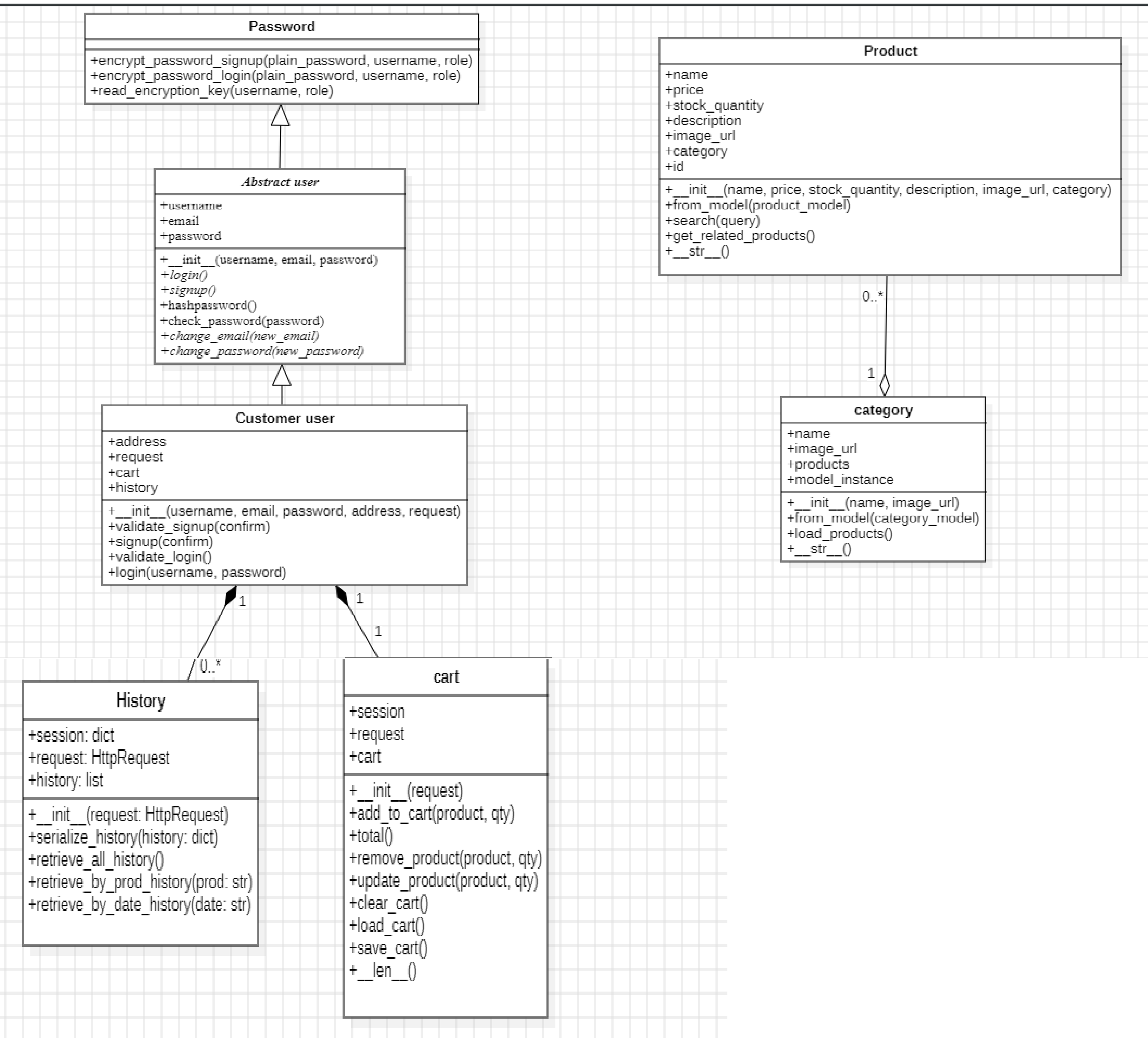
The application utilizes a range of views and templates to display and manage content dynamically. Product Views are responsible for showcasing product listings and detailed information about each item. Cart Views handle operations related to the shopping cart, including adding, updating, and removing items. Order Views manage the checkout process and display the order history, facilitating a seamless and user-friendly transaction experience. Django templates are employed to render dynamic HTML content for various pages, including product lists, the shopping cart, and order history pages.

Testing:

The development process incorporates extensive testing to ensure platform functionality and reliability. Unit tests validate individual components, like models and views, while integration tests confirm the proper interaction between these components, maintaining overall system performance and quality.

CLASS DIAGRAM:

This diagram depicts a class hierarchy and relationships for an e-commerce system. The `Abstract user` class provides basic attributes and methods for user authentication, extended by the `Customer user` class which includes additional details like address and cart. The `Product` class defines product attributes and related functions, while the `Category` class organizes products into categories. The `Password` class handles password encryption. The `History` and `Cart` classes manage the user's purchase history and shopping cart, respectively, interacting with the `Customer user` class.


Password
+encrypt_password_signup(plain_password, username, role)
+encrypt_password_login(plain_password, username, role)
+read_encryption_key(username, role)
Abstract user
+username
+email
+password
+__init__(username, email, password)
+login()
+signup()
+hashpassword()
+check_password(password)
+change_email(new_email)
+change_password(new_password)
Customer user
+address
+request
+cart
+history
+__init__(username, email, password, address, request)
+validate_signup(confirm)
+signup(confirm)
+validate_login()
+login(username, password)
1
1
1
0..*
History
+session: dict
+request: HttpRequest
+history: list
+__init__(request: HttpRequest)
+serialize_history(history: dict)
+retrieve_all_history()
+retrieve_by_prod_history(prod: str)
+retrieve_by_date_history(date: str)
cart
+session
+request
+cart
+__init__(request)
+add_to_cart(product, qty)
+total()
+remove_product(product, qty)
+update_product(product, qty)
+clear_cart()
+load_cart()
+save_cart()
+__len__()
Product
+name
+price
+stock_quantity
+description
+image_url
+category
+id
+__init__(name, price, stock_quantity, description, image_url, category)
+from_model(product_model)
+search(query)
+get_related_products()
+__str__()
0..*
1
category
+name
+image_url
+products
+model_instance
+__init__(name, image_url)
+from_model(category_model)
+load_products()
+__str__()

Any New Thing Learnt in Python While Working on the Project:

SMTP Protocol:

In developing our e-commerce platform, we concentrated on configuring settings rather than using APIs. Specifically, we set up the SMTP protocol in Django to send confirmation emails and created an environment file for managing sensitive information. Although we didn't implement APIs, we gained valuable insights into their potential benefits, such as improving communication between the front-end and back-end, supporting essential functionalities, and ensuring system interoperability. This foundational understanding of APIs, including aspects of security, versioning, and documentation, has equipped us for future advancements.

Django:

Working with Django provided a deep understanding of its architecture and best practices. We explored Django's Model-View-Template (MVT) structure, which enabled us to separate the data layer, business logic, and presentation layer effectively. This separation helped in maintaining a clean codebase and facilitated easier debugging and testing. We also delved into Django's ORM (Object-Relational Mapping) system, which simplified database interactions and allowed us to write database queries using Python code. The framework's built-in features, such as the admin interface, authentication, and form handling, greatly accelerated the development process and ensured high-quality standards for our project.

MySQL Databases:

Our project required efficient data management, leading us to work extensively with MySQL databases. We learned how to design and implement relational databases, optimize queries, and ensure data integrity through various constraints and indexing techniques. This involved setting up MySQL databases, configuring them to work seamlessly with Django, and using Django's ORM to interact with the database. We also explored advanced topics like database transactions, concurrency control, and performance tuning, which were crucial for managing large datasets and ensuring the system's reliability and efficiency. Even we couldn't use mySQL directly as Django offered mySQL lite in form of models we tried to understand it

HTML, CSS And Web dev:

During the development of our e-commerce platform, we focused on mastering the structuring and presentation of web content. We utilized HTML elements such as headings, paragraphs, lists, and links to create a well-organized website framework. CSS (Cascading Style Sheets) was used to control layout, colors, and fonts, enhancing the visual appeal. JavaScript added interactive features like form validation and real-time updates, ensuring a dynamic user experience. We also integrated Bootstrap for responsive grid systems, pre-designed components, and utilities, streamlining the design process. Additionally, Django's templating system was employed to dynamically render HTML, manage content, and create a cohesive connection between the front-end and back-end, enabling a more efficient and maintainable development process.

FUTURE EXPANSIONS:

In the next phase of development, several expansions are planned to enhance our e-commerce platform's capabilities, administration, and analytical features.

Full Admin Feature:

We will introduce a comprehensive administrative panel that provides advanced controls over various aspects of the platform. This feature will allow administrators to manage user accounts, oversee product listings, and monitor transactions. It will also offer detailed analytical insights into site performance, user behavior, and sales trends, thereby enabling informed decision-making and efficient management of the e-commerce operations.

Distinct Seller and Customer Roles:

To better serve both sellers and buyers, our platform will be upgraded to include distinct roles for sellers and customers. Sellers will have access to dedicated dashboards where they can manage their product listings, track sales, and communicate with customers. Customers, on the other hand, will benefit from enhanced features for viewing product reviews, managing orders, and interacting with sellers. This role-based approach will ensure that each user type has access to relevant tools and information, improving overall user experience and functionality.

External Server Usage:

To accommodate growing user traffic and enhance performance, we will transition to using external server infrastructure. This will involve migrating to cloud-based hosting solutions that offer scalable and reliable server management. Implementing load balancing techniques will ensure optimal traffic distribution, reducing server load and improving responsiveness. Additionally, robust data backup and recovery mechanisms will be established to protect against data loss and ensure business continuity.

Graphical Reporting and Analytics:

We plan to develop advanced graphical tools for reporting and analytics to provide deeper insights into business performance. These tools will include visual representations of sales data over time, user engagement metrics, and inventory levels. Administrators and sellers will be able to generate customized reports, facilitating better tracking of sales trends, user behavior, and product performance. This will enable data-driven decisions that can enhance sales strategies and inventory management.

Betterment of GUI:

In future expansions of our e-commerce platform, we will prioritize creating a visually appealing GUI (Graphical User Interface) from scratch, avoiding templates. Using HTML, we will structure content with elements such as headings, paragraphs, and lists. CSS will enable us to customize the GUI's layout, colors, and typography, ensuring a unique design tailored to our brand. JavaScript will be utilized to add interactive features like animations and user-friendly functionalities such as product filtering and dynamic content updates. This approach will allow us to craft a distinctive and engaging interface, enhancing user experience and usability across our platform.

Server Management and Deployment:

In our future expansions, we plan to deepen our understanding of server management and the deployment process. We aim to configure web servers, such as Apache and Nginx, to serve our Django application efficiently. This will include setting up reverse proxies, handling static and media files, and ensuring secure communication through SSL certificates. Additionally, we will explore cloud-based solutions for hosting our application, providing insights into server scaling, load balancing, and automated backups. This knowledge will enable us to deploy our platform in a live environment, ensuring it remains accessible, performant, and secure under varying loads.

Enhancing User interactivity:

Our future expansions aim to enhance user interactivity by implementing personalized features. This includes offering discounts based on users' purchase history and providing tailored advice and recommendations according to their tastes and preferences. This approach will help us build stronger connections with our customers and improve their overall shopping experience.

INDIVIDUAL PROWESS:

Muhammad Qambar Hussain:

Worked on user authentication, including history login, signup, checkout, cart functionality, and the user app integration.

AsadUllah Nizami:

Enhanced the base HTML template using Django's templating language and implemented products sorting, navbar functionality, and other frontend logic.

AliRaza Baloch:

Developed object-oriented code for categories, products, admin registration, and checkout functionalities within the app.

MOST DIFFICULT PART DURING OUR PROJECT:

Understanding Django:

One of the most challenging aspects of this project was getting to grips with Django, a high-level Python web framework that encourages rapid development and clean, pragmatic design. As beginners, we found Django's comprehensive nature both a boon and a bane. The framework's vast array of features, including an ORM (Object-Relational Mapping), a templating engine, and built-in security features, initially overwhelmed us. The learning curve was steep because we had to understand not just how to use these features, but also the underlying principles that make Django work.

Integrating Object-Oriented Programming with Django:

Integrating object-oriented programming (OOP) principles within the Django framework posed another significant challenge. Although Django itself is built on OOP principles, applying these concepts effectively in our own codebase required a deep understanding of both OOP and Django's architecture. Designing Django models that accurately represented our data structures while adhering to OOP principles such as encapsulation and inheritance was tricky. We also found it challenging to leverage class-based views effectively, as this required a solid grasp of inheritance and mix ins. Organizing our code in a clean and logical manner while balancing Django's conventions with OOP structure was a continual effort. However, these challenges provided valuable learning opportunities, and by the end of the project, we had significantly deepened our understanding of both Django and object-oriented programming.

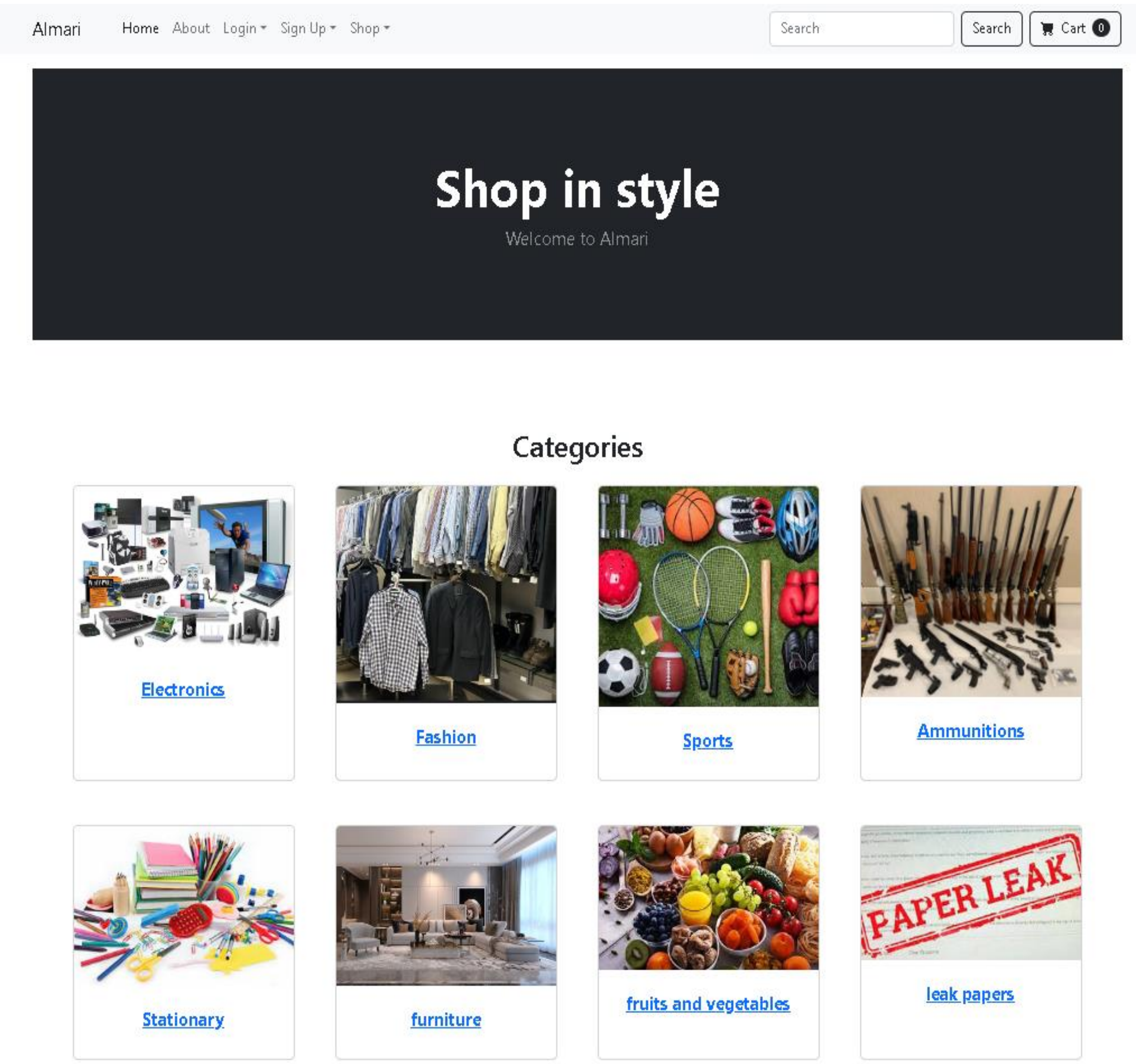
TEST CASES:

A lot of things could be added in the test cases but we decided to show some of the unique features of our website for example:

- **The Homepage**
- **Captcha**
- **Email**
- **History**
- **Breadcrumb navigations**
- **Cart**
- **CSRF Tokens and Session ID**

THE HOMEPAGE:

The homepage showcases the frontend



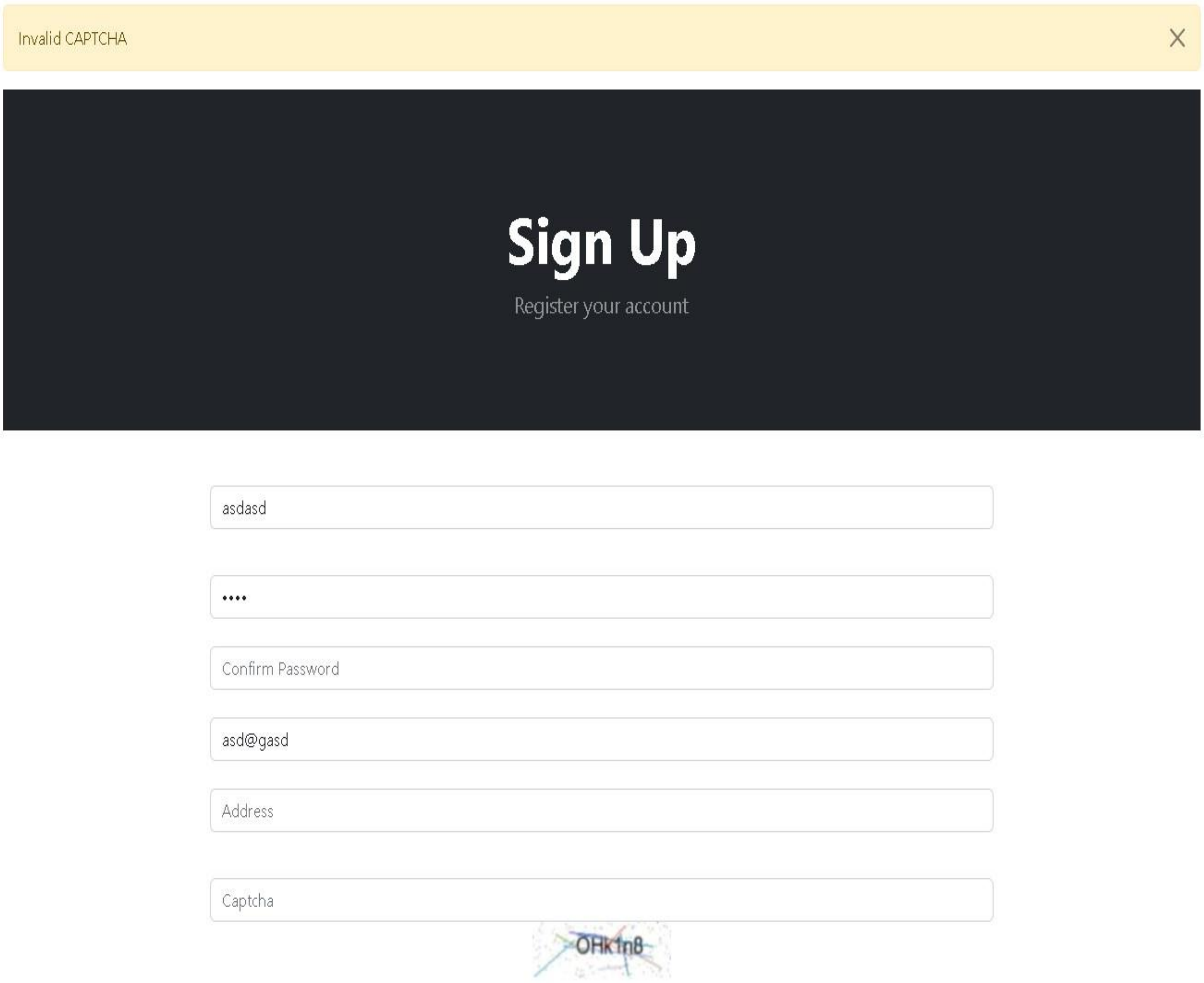
Shop in style

Welcome to Almari

Categories

CAPTCHA:

The captcha is part of the security checks installed in the code



Invalid CAPTCHA

Sign Up

Register your account

asdasd

••••

Confirm Password

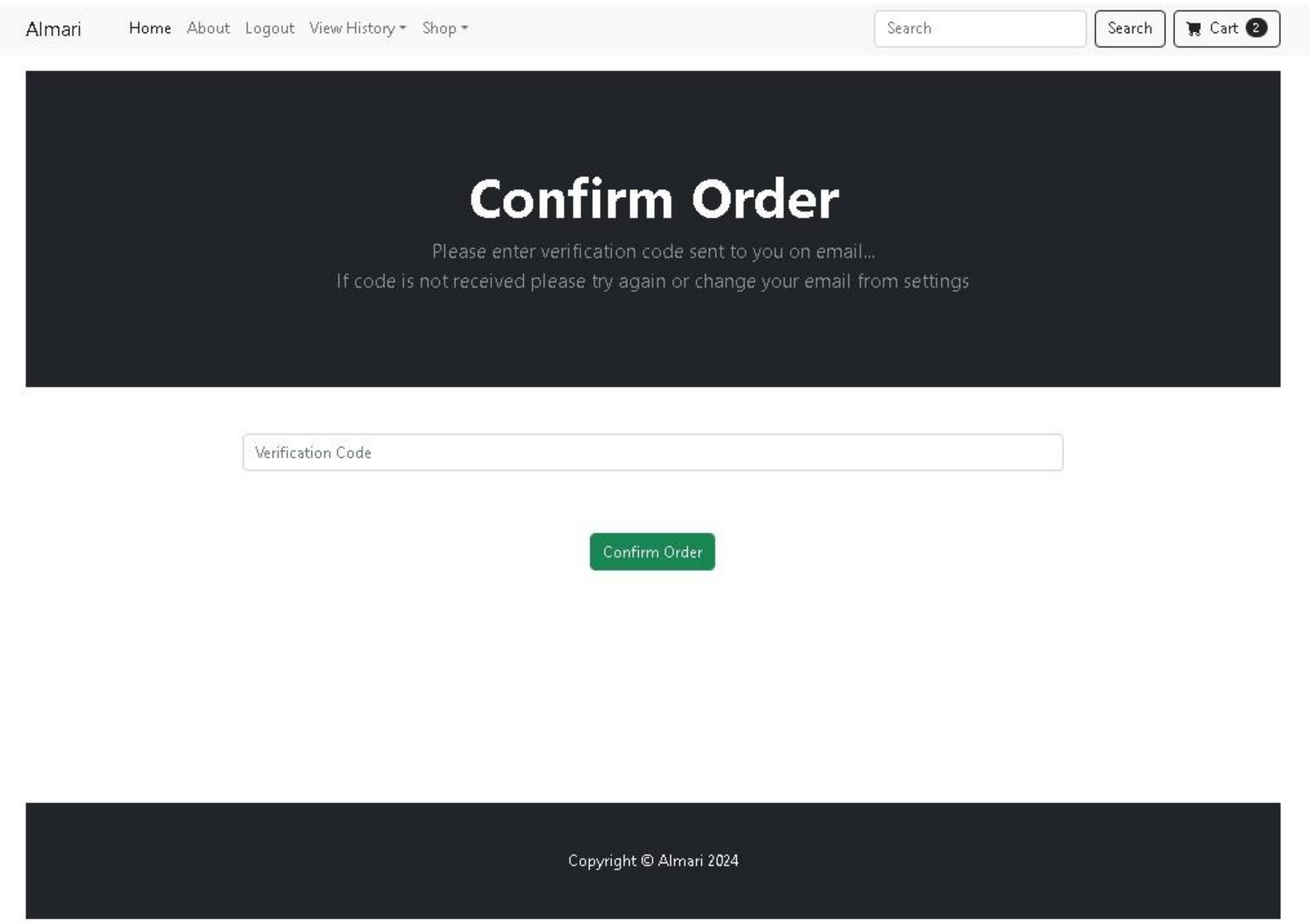
asd@gasd

Address

Captcha

EMAIL:

Another security feature is the verification code sent by an email with an expiry time of 1 minute after that it expires and another code is sent



Confirm Order

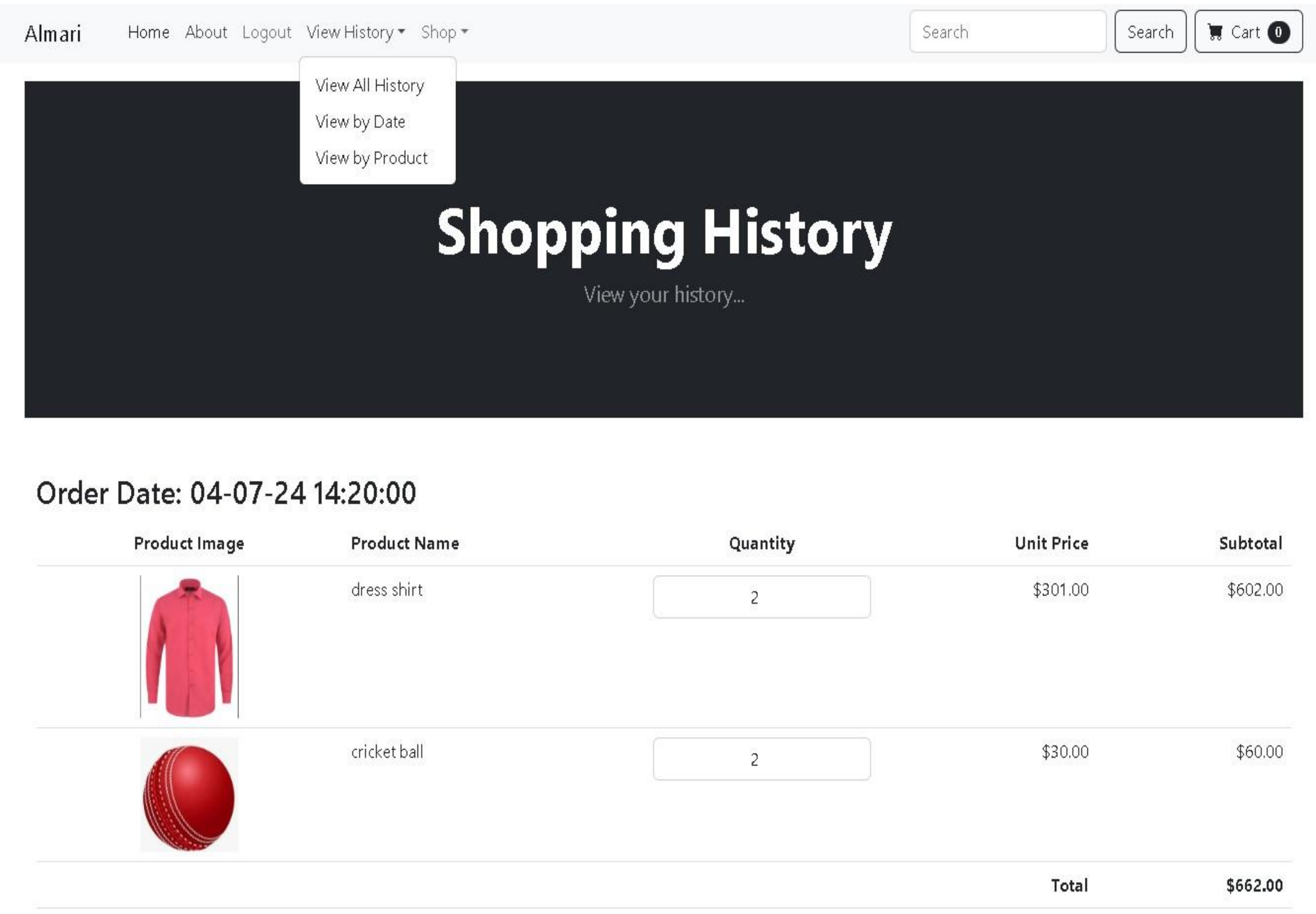
Please enter verification code sent to you on email...

If code is not received please try again or change your email from settings

Copyright © Almari 2024

HISTORY:

The history feature helps the user to view the shopping history either by date or product or every product through **All history**



Shopping History

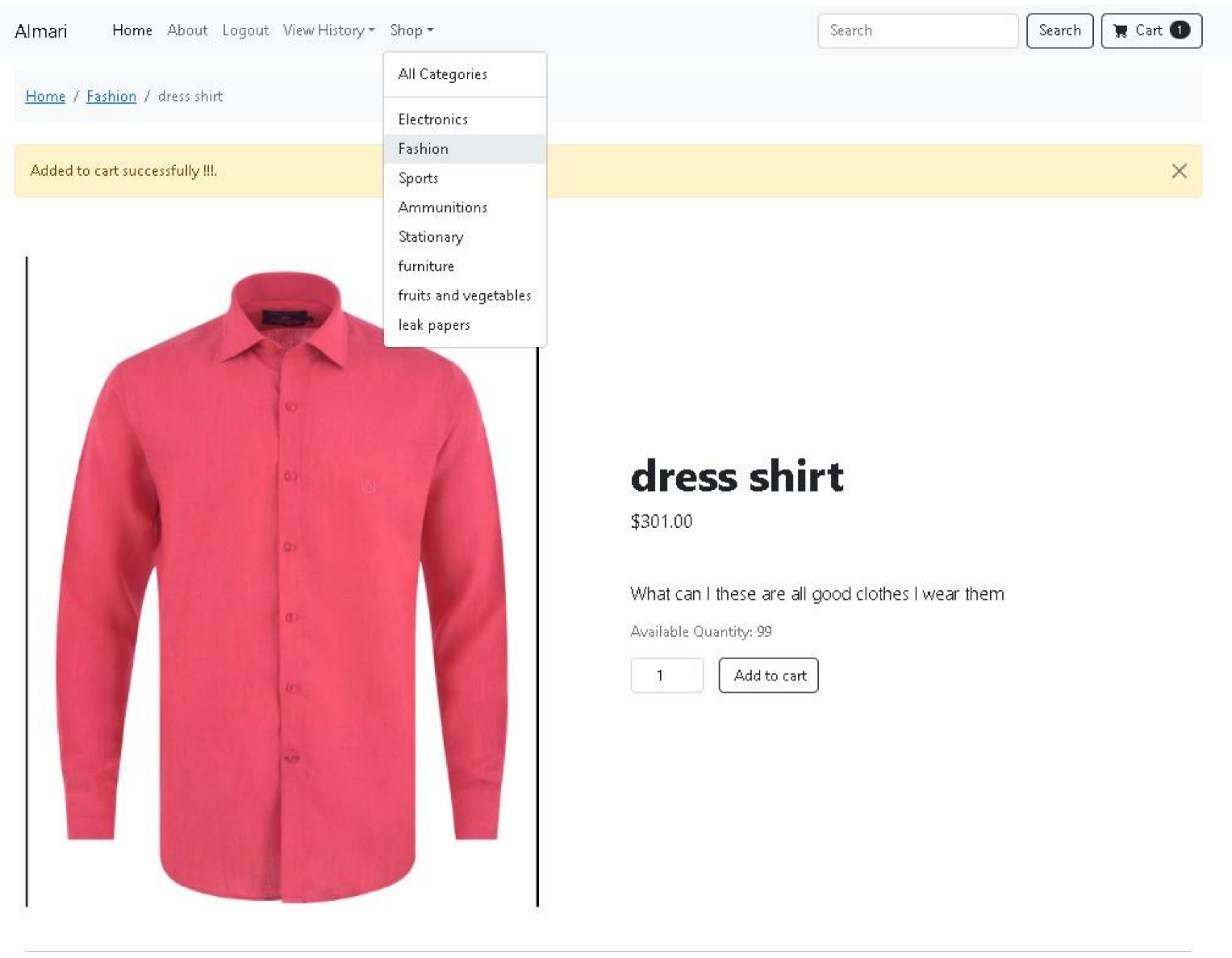
View your history...

Order Date: 04-07-24 14:20:00

Product Image	Product Name	Quantity	Unit Price	Subtotal
	dress shirt	2	$301.00	$602.00
	cricket ball	2	$30.00	$60.00
			Total	$662.00

BREADCRUMB NAVIGATION:

Another feature of our website the user friendly nature that allows the user to freely navigate using the breadcrumb navigation feature



Home / Fashion / dress shirt

Added to cart successfully !!!.

dress shirt

$301.00

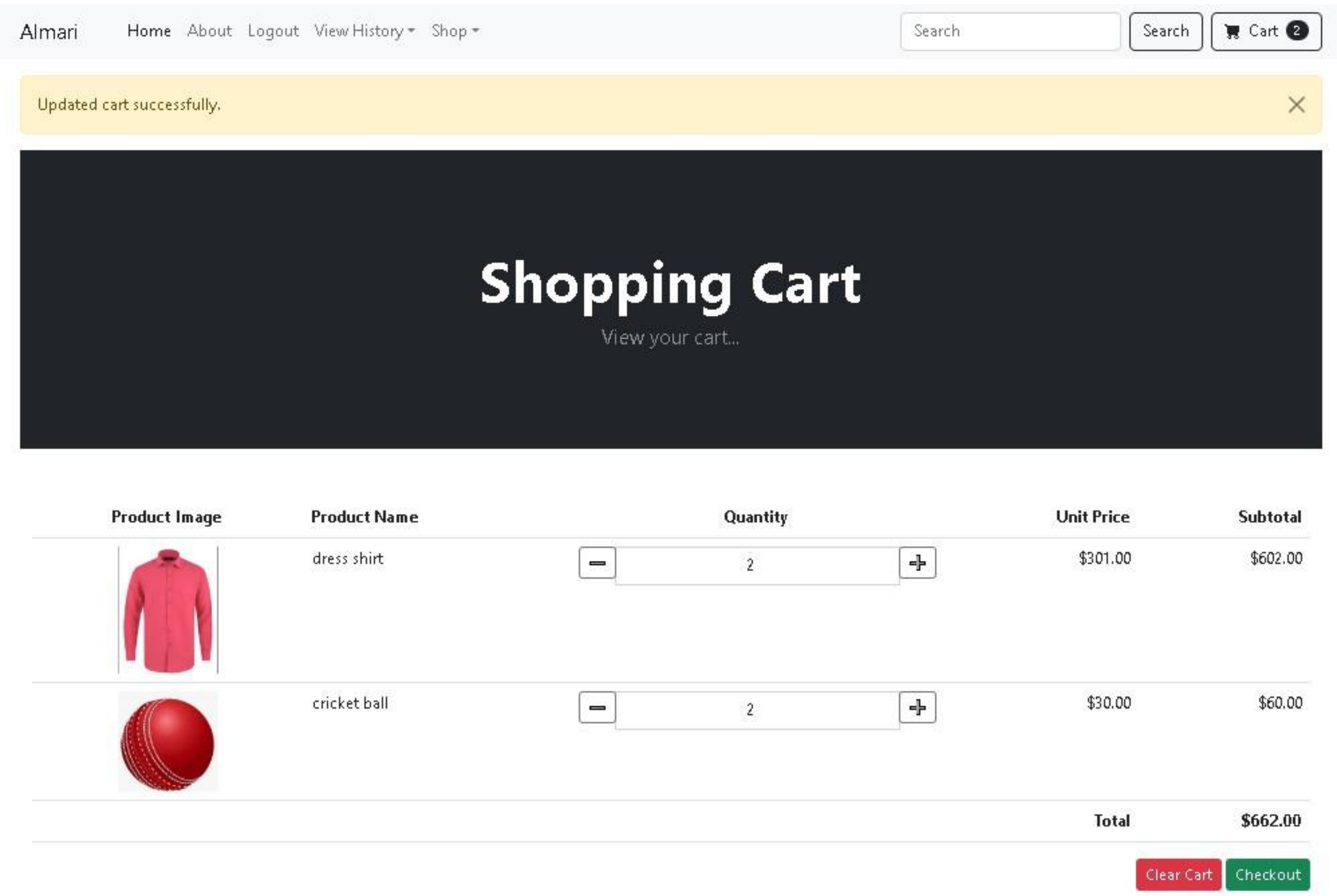
What can I these are all good clothes I wear them

Available Quantity: 99

1

CART:

A functioning cart which helps the user update the quantity on spot is another unique feature of our project



Updated cart successfully.

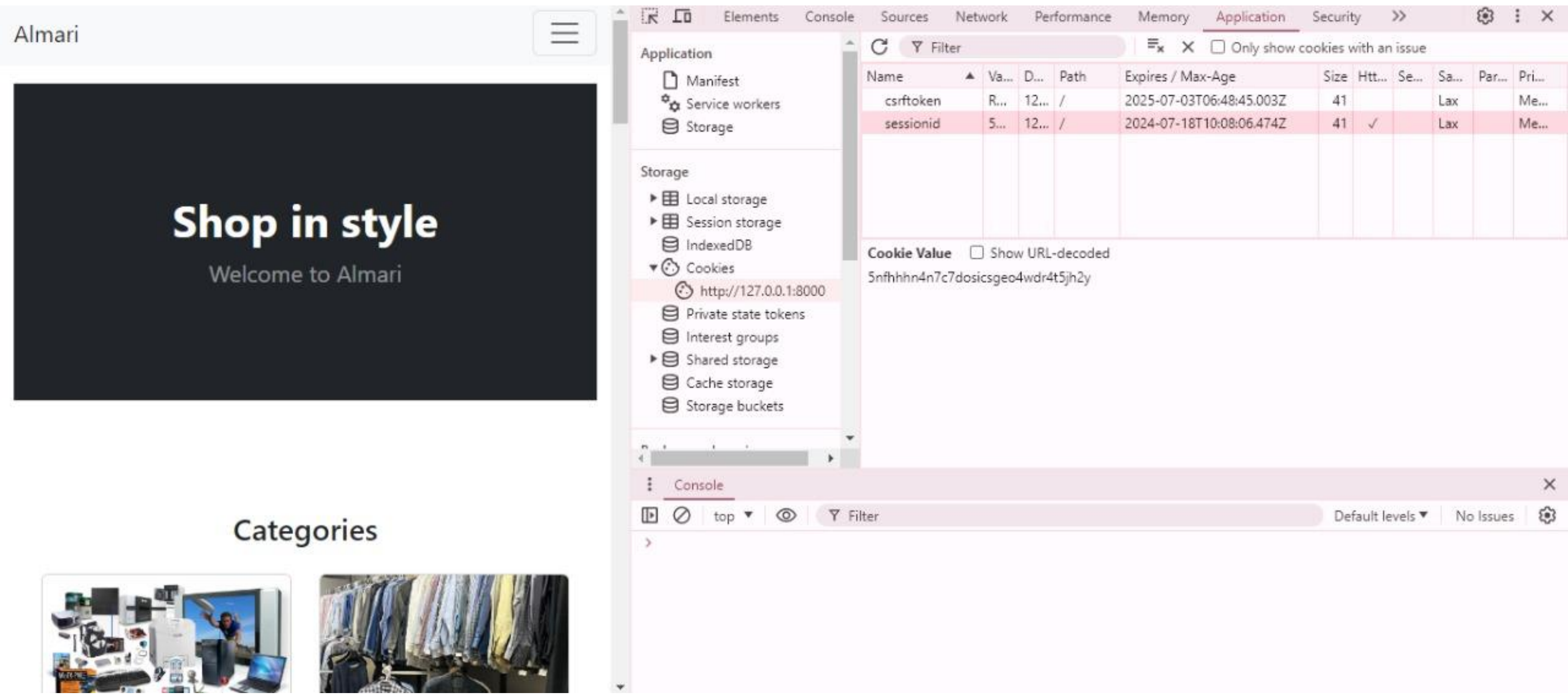
Shopping Cart

View your cart...

Product Image	Product Name	Quantity	Unit Price	Subtotal
	dress shirt	2	$301.00	$602.00
	cricket ball	2	$30.00	$60.00
			Total	**$662.00**

CRSF TOKENS:

CSRF tokens are used to ensure that requests originate from authenticated users, protecting against cross-site request forgery attacks.



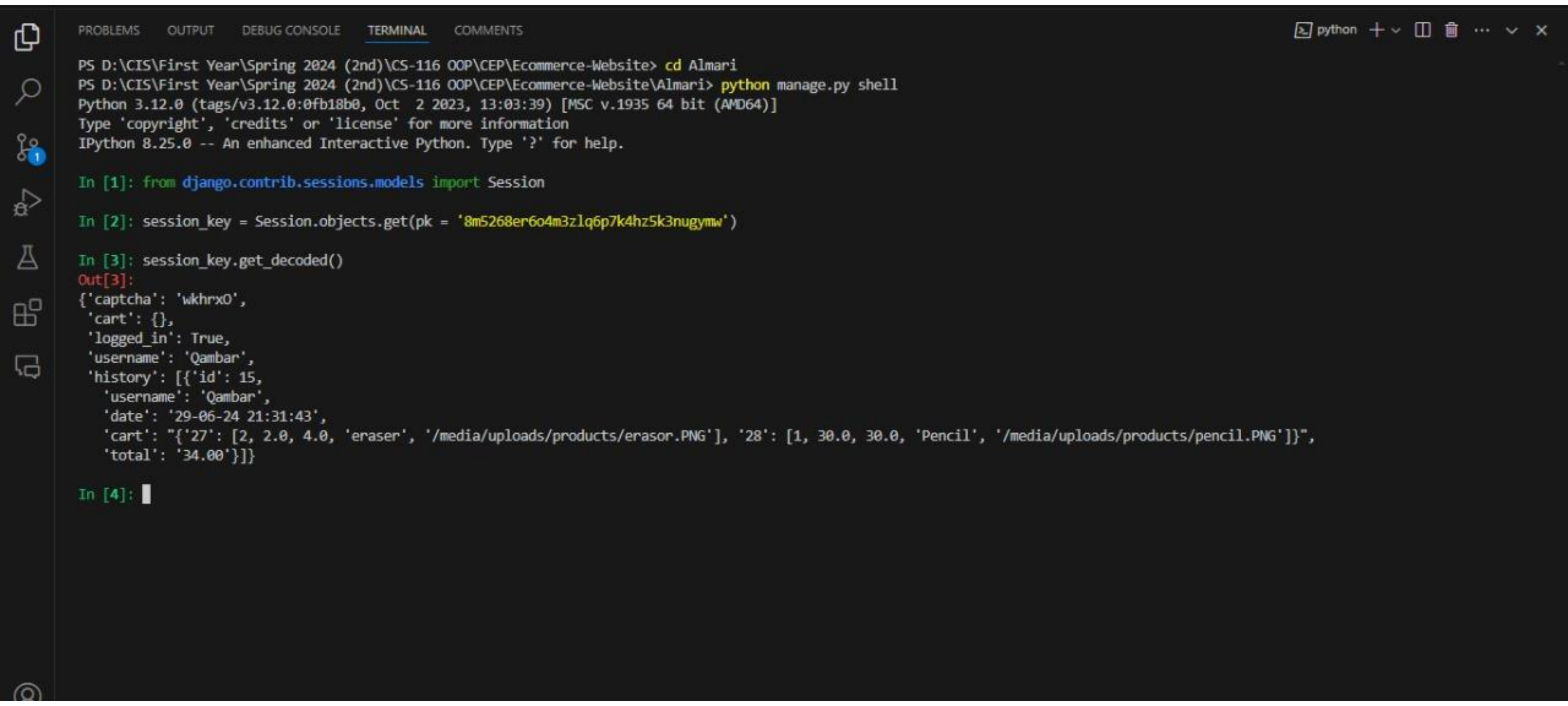
Name	Va...	D...	Path	Expires / Max-Age	Size	Htt...	Se...	Sa...	Par...	Pri...
csrftoken	R...	12...	/	2025-07-03T06:48:45.003Z	41			Lax		Me...
sessionid	5...	12...	/	2024-07-18T10:08:06.474Z	41	✓		Lax		Me...

Cookie Value ☐ Show URL-decoded

5nfhhhn4n7c7dosicsgeo4wdr4t5jh2y

SESSION ID:

Session IDs are used to bind all session data, including information like username and cart contents, to the user's session.



```
PS D:\CIS\First Year\Spring 2024 (2nd)\CS-116 OOP\CEP\Ecommerce-Website> cd Almari
PS D:\CIS\First Year\Spring 2024 (2nd)\CS-116 OOP\CEP\Ecommerce-Website\Almari> python manage.py shell
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct  2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.25.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from django.contrib.sessions.models import Session

In [2]: session_key = Session.objects.get(pk = '8m5268er6o4m3zlq6p7k4hz5k3nugymw')

In [3]: session_key.get_decoded()
Out[3]:
{'captcha': 'wkhrxO',
 'cart': {},
 'logged_in': True,
 'username': 'Qambar',
 'history': [{'id': 15,
   'username': 'Qambar',
   'date': '29-06-24 21:31:43',
   'cart': "{'27': [2, 2.0, 4.0, 'eraser', '/media/uploads/products/erasor.PNG'], '28': [1, 30.0, 30.0, 'Pencil', '/media/uploads/products/pencil.PNG']}",
   'total': '34.00'}]}

In [4]: 
```