Project 1 SIO

Vulnerabilities in software products

05/11/2023

Group Members:

Miguel Aido Miragaia	108317	miguelmiragaia@ua.pt
Cristiano Antunes Nicolau	108536	cristianonoicolau@ua.pt
Andre Lourenço Gomes	97541	alg@ua.pt
Pedro Miguel Ribeiro Rei	107463	pedrorrei@ua.pt

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Contents

Introduction

According to the project planification developing an e-commerce store about DETI was one of the goals to be achieved, however the main objective was to present both vulnerable and secure application along with all the vulnerabilities encountered and a description of how they would be explored and their impact.

The project consists in an online application that includes some vulnerabilities such as **CWE-80**: Improper Neutralization of Special Elements used in an SQL Command (‘SQL Injection’), **CWE-79**: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'), **CWE-13**: ASP.NET Misconfiguration: Password in Configuration File, **CWE-262**: Not Using Password Aging, **CWE-265**: Privilege Issues, **CWE-521**: Weak Password Requirements, **CWE-306**: Missing Authentication for Critical Function, **CWE-20**: Improper Input Validation, **CWE-80**: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS), **CWE-286**: Incorrect User Management. All these vulnerabilities will be explored and detailed in the respective section of this report, along with their severity and potential impact.

Our implementation certificates that all objectives were achieved, and every member understood the purpose of the project, all the vulnerabilities discovered, their solution and attributed a severity score based in our own perspectives.

Implementation Technologies/Techniques

The project was developed with frontend in HTML/CSS/JS, the backend in Flask, and the SQLite3 database. Initially, we used a frontend template available at https://untree.co/free-templates/furni-furniture-ecommerce-website-template-free-download/, but all customizations to the frontend, as well as the development of the backend and the database, were performed by the members of the group.

Frontend in HTML/CSS/JS

The project's frontend was built using ***HTML***, ***CSS***, and ***JavaScript*** technologies. Initially, we used the Furni Furniture E-commerce template as a starting point. However, to meet the project's specific requirements and ensure security, we made extensive modifications. This included creating login pages, registration forms, user profile pages, admin pages, and other custom features. The design was optimized for a pleasant and intuitive user experience.

Backend in Flask

The backend was developed using the ***Flask*** framework, which serves as the server for the security application. ***Flask*** provides a lightweight and flexible architecture, making it a suitable choice for our project. We implemented routes to handle frontend requests, data validation, authentication, and authorization, among other tasks. Security was a priority, and measures such as password hashing, protection against ***SQL injection***, protection against ***XSS attacks***, and token authentication were implemented to protect user data and privacy.

SQLite3 Database

The project's database was implemented using ***SQLite3***, a lightweight embedded database management system. We chose ***SQLite3*** due to its simplicity and efficiency for smaller applications. We stored user information, product information, and other data essential for the system's functionality. Security practices, such as the use of parameterized queries, were applied to prevent ***SQL injection*** attacks.

Security

Security is a critical consideration in our project due to the sensitive nature of user data and information. We implemented various security measures, including:

- **Password Hashing**: User passwords are securely stored using hashing algorithms, specifically ***Bcrypt***, to protect against the leakage of confidential information.

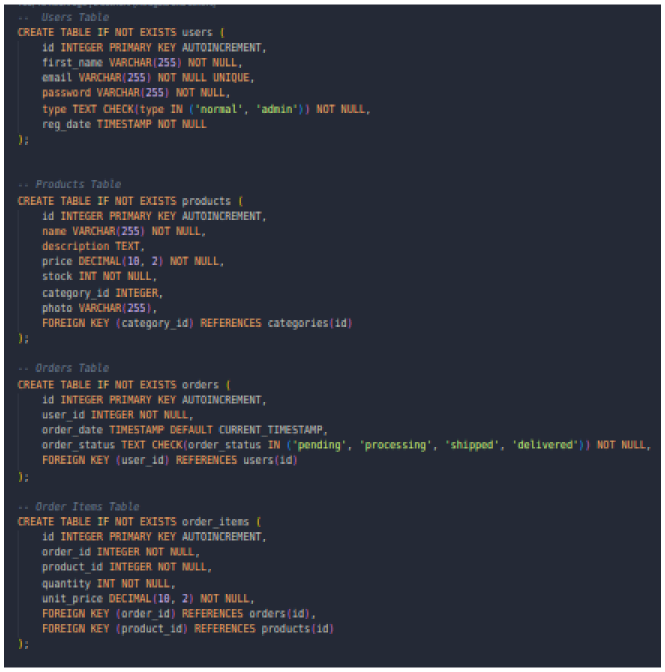
- **Protection Against SQL Injection:** We used parameterized queries and ***SQLalchemy*** to prevent SQL injection, ensuring that user data is protected against malicious attacks.

- **Token Authentication**: Token authentication was implemented to ensure that only authorized users have access to protected resources.

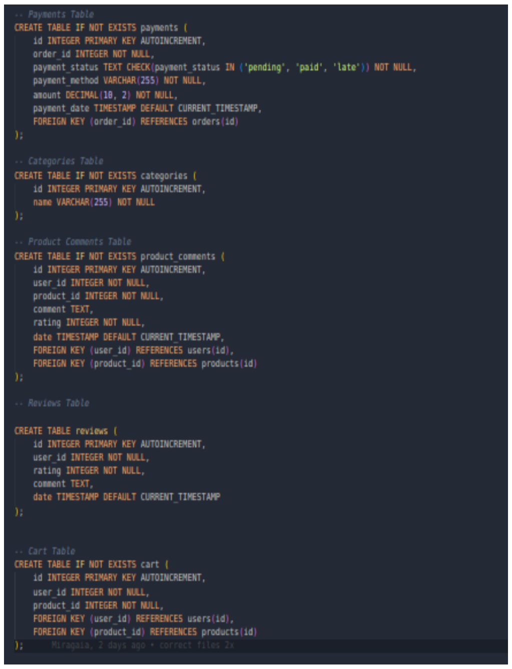
- **Data Validation**: We rigorously validate all input data to prevent vulnerabilities such as ***Cross-Site Scripting (XSS) attacks*** and ***Cross-Site Request Forgery (CSRF) attacks***.

- **Security Monitoring:** We implemented a continuous monitoring strategy to detect and respond to potential security threats, such as ***Cross-Site Scripting (XSS) attacks***. We also used ***Bleach***, which is a ***Python*** library used for cleaning and filtering text, especially in user-generated content.

The implementation of technologies and security techniques in the project was successful, providing a secure and effective system for our users. Technology choices, frontend customizations, backend development, and security measures were crucial in ensuring data integrity and user protection.



```sql
--  Users Table
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    first_name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    type TEXT CHECK(type IN ('normal', 'admin')) NOT NULL,
    reg_date TIMESTAMP NOT NULL
);


-- Products Table
CREATE TABLE IF NOT EXISTS products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL,
    category_id INTEGER,
    photo VARCHAR(255),
    FOREIGN KEY (category_id) REFERENCES categories(id)
);

-- Orders Table
CREATE TABLE IF NOT EXISTS orders (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    order_status TEXT CHECK(order_status IN ('pending', 'processing', 'shipped', 'delivered')) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- Order Items Table
CREATE TABLE IF NOT EXISTS order_items (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    order_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```



```sql
-- Payments Table
CREATE TABLE IF NOT EXISTS payments (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    order_id INTEGER NOT NULL,
    payment_status TEXT CHECK(payment_status IN ('pending', 'paid', 'late')) NOT NULL,
    payment_method VARCHAR(255) NOT NULL,
    amount DECIMAL(10, 2) NOT NULL,
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES orders(id)
);

-- Categories Table
CREATE TABLE IF NOT EXISTS categories (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(255) NOT NULL
);

-- Product Comments Table
CREATE TABLE IF NOT EXISTS product_comments (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    comment TEXT,
    rating INTEGER NOT NULL,
    date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

-- Reviews Table

CREATE TABLE reviews (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    rating INTEGER NOT NULL,
    comment TEXT,
    date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);


-- Cart Table
CREATE TABLE IF NOT EXISTS cart (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    product_id INTEGER NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);      Miragaia, 2 days ago • correct files 2x
```

API Requests

Request to know if user is logged in

Method	URL
GET	/user_info
Type	**PARAM**
HEAD	Authorization
Status	**Response**
200	User is authenticated
200	User is not authenticated

Request to login

Method	URL
POST	/reg_log
Type	**PARAM**
BODY	email
BODY	password
Status	**Response**
200	User receive token of authentication (60 minutes duration)
200	User is not authenticated

Request to regist user

Method	URL
POST	/register
Type	**PARAM**
BODY	email
BODY	name
BODY	password
BODY	confirmed_password
Status	**Response**
200	Registration successfully
200	Error in registration

Request to get all products

Method	URL
GET	/product
Status	**Response**
200	Information about all products

User information Request

Method	URL
POST	/user
Type	**PARAM**
BODY	id
Status	**Response**
200	User information
200	User does not exist

User edit Request

Method	URL
POST	/edit_user
Type	**PARAM**
BODY	id
BODY	name
BODY	email
Status	**Response**
200	Sucess
200	User does not exist

Change password Request

Method	URL
POST	/changepswd
Type	**PARAM**
BODY	id
BODY	password
Status	**Response**
200	Sucess
200	User does not exist

Request to get selected product

Method	URL
GET	/product/<int:product_id>
Status	**Response**
200	Information about selected product

Request to get all categories

Method	URL
GET	/category
Status	**Response**
200	Information about all categories

Request to review a product

Method	URL
POST	/product_review
Type	**PARAM**
BODY	Product_id
BODY	user_id
BODY	rating
BODY	comment
Status	**Response**
200	Reviewed successfully
200	Error reviewing product

Request to add a product

Method	URL
POST	/add_product
Type	**PARAM**
BODY	user_id
BODY	name
BODY	price
BODY	stock
BODY	photo
BODY	description
BODY	categorie_id
Status	**Response**
200	Product added successfully
200	Error adding product

Get order details completed

Method	URL
GET	/orders/<int:order_id>
Status	**Response**
200	Order details

Request to update a product

Method	URL
POST	/update_product
Type	**PARAM**
BODY	user_id
BODY	Product_id
BODY	name
BODY	price
BODY	stock
BODY	photo
BODY	description
BODY	categorie_id
Status	**Response**
200	Product updated successfully
200	Error updating product

Delete product Request

Method	URL
DELETE	/delet_product
Type	**PARAM**
BODY	id_user
BODY	id_product
Status	**Response**
200	Product deleted successfuly
200	Error deleting product

Add product to cart Request

Method	URL
POST	/product_details/<int:user_id>/<int:product_id>
Status	**Response**
400	Product not found
404	Product out of stock

Get user cart Request

Method	URL
GET	/cart/<int:user_id>
Status	**Response**
200	Cart list

Remove product of cart Request

Method	URL
GET	/remove_from_cart/<int:user_id>/<int:product_id>
Status	**Response**
200	Product removed

Get reviews of site Request

Method	URL
GET	/render_reviews
Status	**Response**
200	Reviews list

Request to review a site

Method	URL
POST	/reviews
Type	**PARAM**
BODY	user_id
BODY	rating
BODY	comment
Status	**Response**
200	Reviewed successfully
200	Error reviewing

Request to reorder same products

Method	URL
POST	/reorder/<int:user_id>/<int:order_id>
Status	**Response**
200	Reorder successfully

Get orders completed

Method	URL
GET	/orders/<int:user_id>
Status	**Response**
200	Orders list

Vulnerabilities

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (CWE-79)

Description

CWE-79, also known as "Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')," represents a common web application security vulnerability where untrusted data is included in the output of a web page without proper validation or escaping. This allows an attacker to inject malicious scripts into a web application, which can then be executed by unsuspecting users' browsers. The attacker can steal user data, manipulate the content of the page, or perform other malicious actions.

Scenario

The web application allows any visitor to the site, even without logging in or authenticating, to post reviews about the website or products. This means that anyone, including potential malicious attackers, can leave reviews without the need to prove their identity. However, this lack of authentication also creates a potential vulnerability for Cross-Site Scripting (XSS) attacks. Attackers can exploit this vulnerability by injecting malicious scripts into their reviews. These scripts can be executed when other users view the reviews, potentially leading to *data theft*, *content manipulation* and *browser exploitation*.

Severity Score and Evaluation

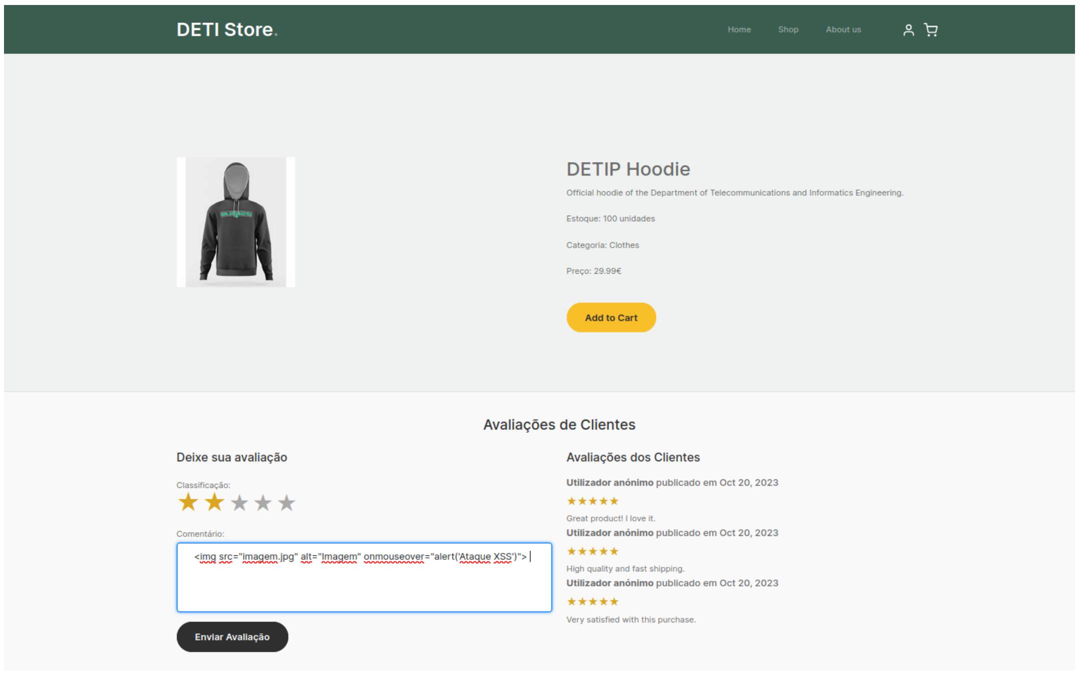
Severity Score: High: The vulnerability has a severe impact, and exploitation is relatively easy.

Points of View: Allowing attackers to inject malicious scripts into reviews, which can be executed when other users view them. Content manipulation and browser exploitation are serious problems that should never be present in a web app. The severity of the issue is evident.

Demonstration

Here is an example of how a malicious attacker could use this CWE-79 vulnerability to perform Cross-Site Scripting (XSS):

1. `<img src="imagem.jpg" alt="Imagem" onmouseover="alert('Ataque XSS')">`
2. `<script> alert('Ataque XSS')</script>`
3. `<img src=x onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041">`


DETI Store.

DETIP Hoodie

Official hoodie of the Department of Telecommunications and Informatics Engineering.

Estoque: 100 unidades

Categoria: Clothes

Preço: 29.99€

Avaliações de Clientes

Deixe sua avaliação

Classificação:

Comentário:

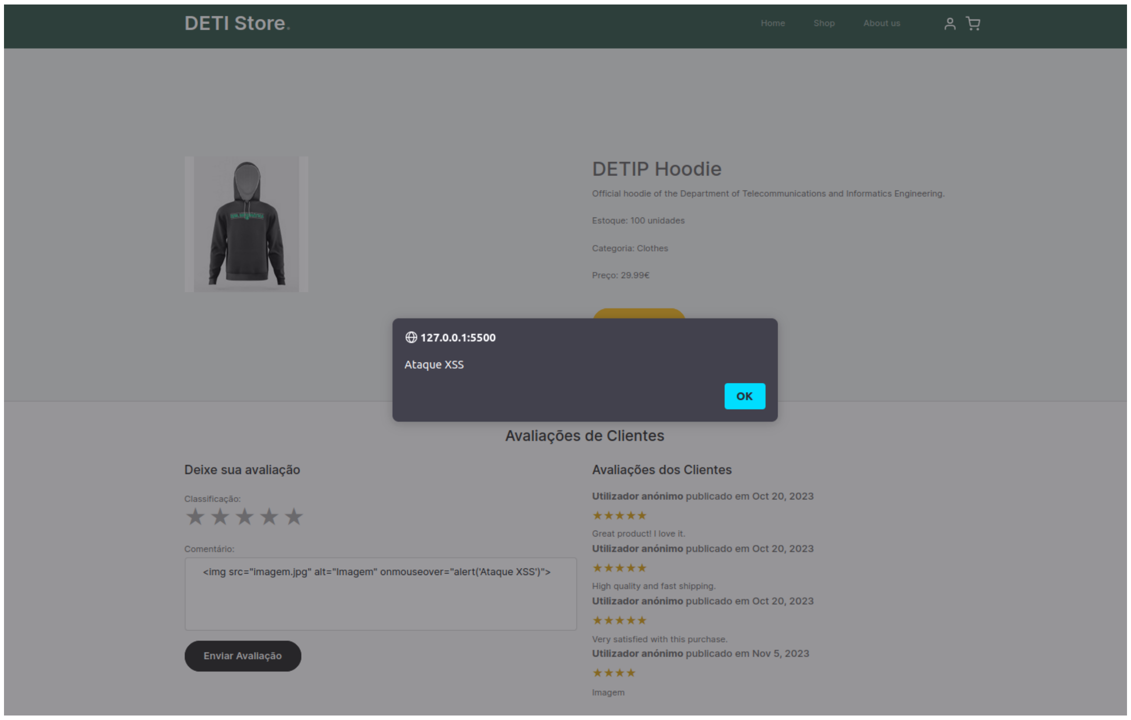
```
<img src="imagem.jpg" alt="Imagem" onmouseover="alert('Ataque XSS')">
```

Avaliações dos Clientes

Utilizador anónimo publicado em Oct 20, 2023
★★★★★
Great product! I love it.

Utilizador anónimo publicado em Oct 20, 2023
★★★★★
High quality and fast shipping.

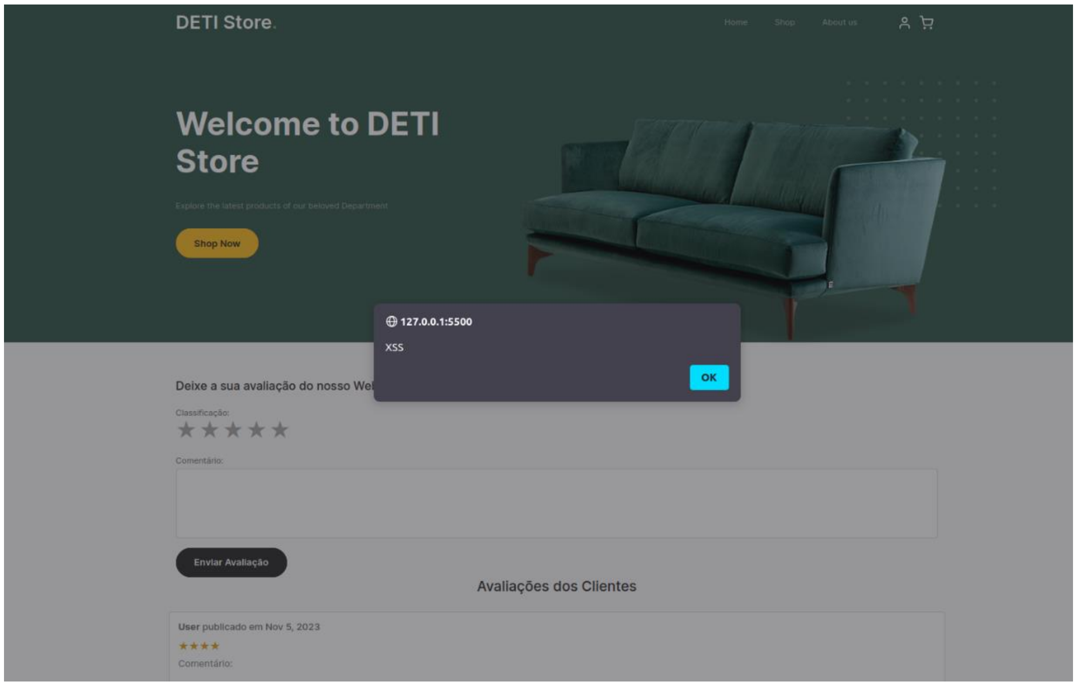
Utilizador anónimo publicado em Oct 20, 2023
★★★★★
Very satisfied with this purchase.


127.0.0.1:5500

Ataque XSS

```
<img src="imagem.jpg" alt="Imagem" onmouseover="alert('Ataque XSS')">
```

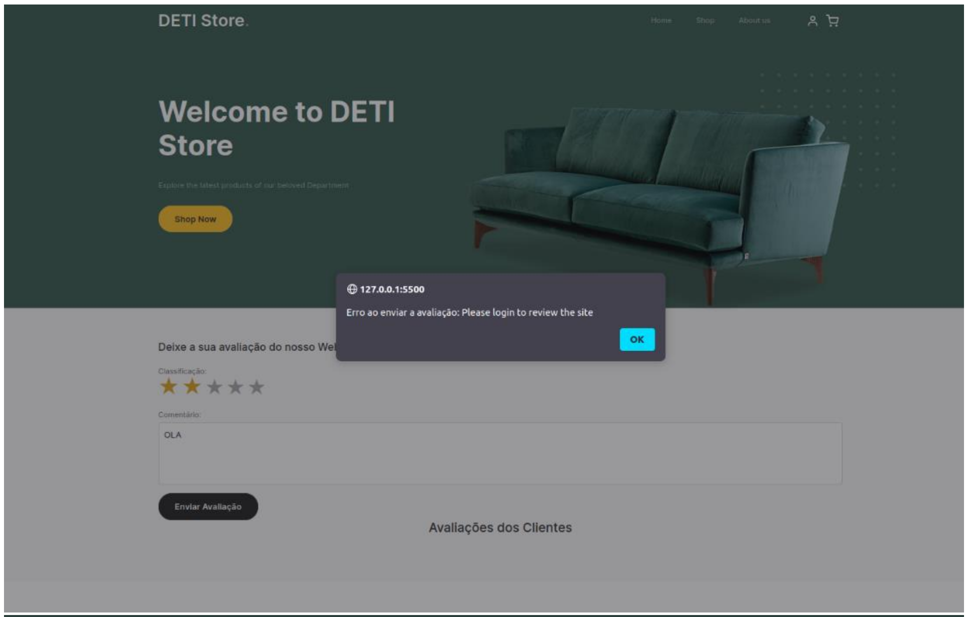
Utilizador anónimo publicado em Nov 5, 2023
★★★★
Imagem



Solution

1. **Authentication Requirement:** We have made it mandatory for users to be authenticated and logged in before they can post reviews.
2. **Mandatory Field Validation:** In addition to user authentication, we require all review fields to be filled out, including the rating.
3. **Text Filtering with Bleach:** We have integrated the Bleach library into our application to filter and sanitize the text input in reviews. Bleach helps prevent the injection of malicious scripts by removing or escaping potentially harmful elements and attributes from user-generated content.

By implementing these measures, we have enhanced the security of our web application and significantly reduced the risk of XSS attacks. Users must be authenticated, complete all review fields, and any user-generated text is carefully filtered to ensure a safer and more trustworthy experience for everyone.

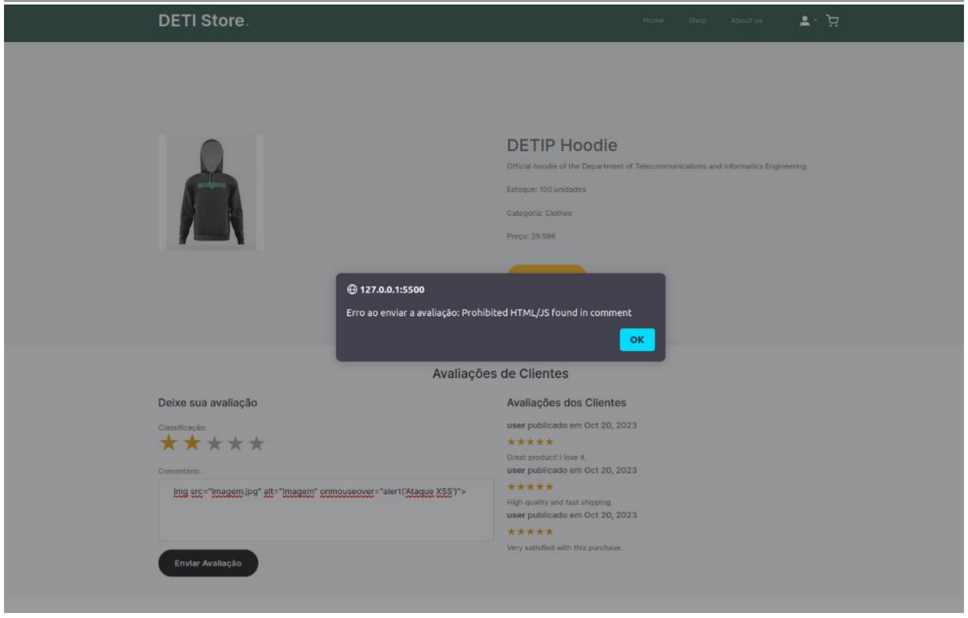


127.0.0.1:5500

Erro ao enviar a avaliação: Please login to review the site

Comentário:

OLA



DETIP Hoodie

Official hoodie of the Department of Telecommunications and Informatics Engineering.

Estoque: 100 unidades

Categoria: Clothes

Preço: 29.99€

127.0.0.1:5500

Erro ao enviar a avaliação: Prohibited HTML/JS found in comment

Comentário:

```
Img src="Imagem.jpg" alt="Imagem" onmouseover="alert('Ataque XSS')">
```

Avaliações dos Clientes

user publicado em Oct 20, 2023

★★★★★

Great product! I love it.

user publicado em Oct 20, 2023

★★★★★

High quality and fast shipping.

user publicado em Oct 20, 2023

★★★★★

Very satisfied with this purchase.

Improper Neutralization of Script-Related HTML Tags in a Web Page (CWE-80)

Description

CWE-80, also known as "Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)," is a type of Cross-Site Scripting (XSS) vulnerability. It occurs when a web application fails to properly neutralize or sanitize input, allowing attackers to inject malicious scripts into web pages.

Scenario

In our web application, user inputs are not adequately sanitized or neutralized before rendering in web pages. As a result, attackers can inject malicious scripts, such as JavaScript, into input fields, comments, or other forms of user-generated content. When other users view these pages or content, the injected scripts can execute within their browsers, potentially leading to various security risks.

Potential consequences are Content Manipulation, that occurs when malicious scripts can manipulate the content of web pages, altering the appearance or behavior of the application.

An example would be adding a button through XSS that when clicked redirects to an external page, or pops an alert with some messages, in general managing the appearances and adding some actions to the website.

Severity Score and Evaluation

- **Severity Score**: High: The vulnerability has a severe impact, and exploitation is relatively easy.
- **Points of View**: Content manipulation and the execution of malicious scripts can have a detrimental impact on the application and user experience. An attacker could exploit this vulnerability by adding a button that redirects users to external pages or displays alert messages. Such actions can seriously compromise the application's security and user trust.

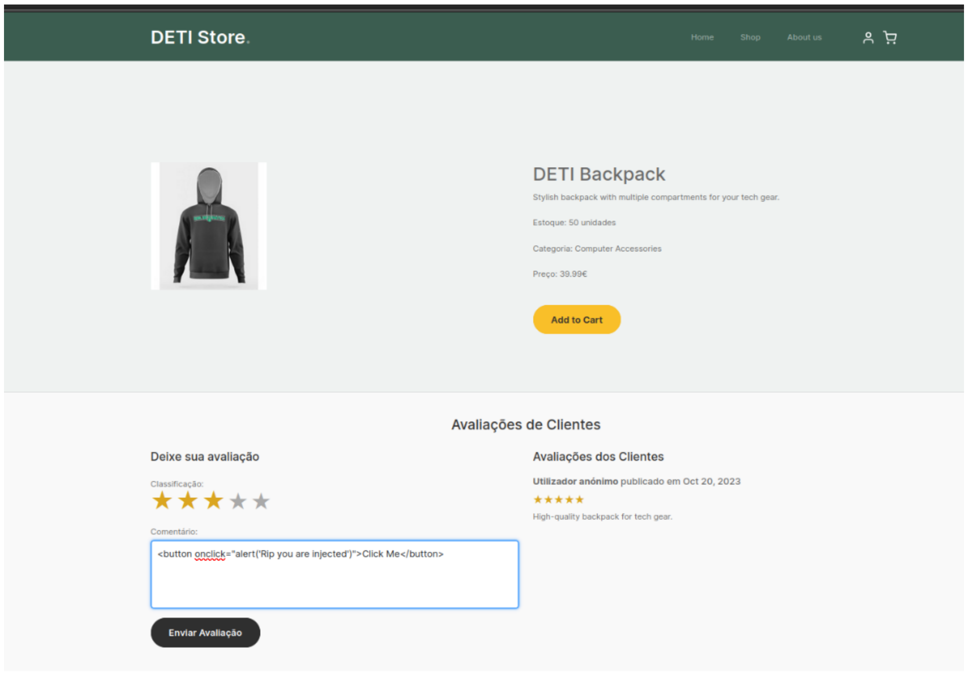
Demonstration

In this scenario, we will illustrate how an attacker could exploit the CWE-80 vulnerability to inject malicious scripts into user-generated content. For this demonstration, consider the following malicious code:

```
<button onclick=alert("Rip you are injected")>Click Me</button>
```

When an attacker injects this code into a comment or input field, and it gets displayed on a web page, any user who views this content will see a button. If they click the button, it triggers an alert with a message indicating a severe security breach, such as "Rip you are injected."

This demonstrates that the CWE-80 vulnerability can potentially lead to not only content manipulation but also alarming messages that may cause panic and harm to users.



DETI Backpack

Stylish backpack with multiple compartments for your tech gear.

Estoque: 50 unidades

Categoria: Computer Accessories

Preço: 39.99€

Avaliações de Clientes

Deixe sua avaliação

Classificação:

Comentário:

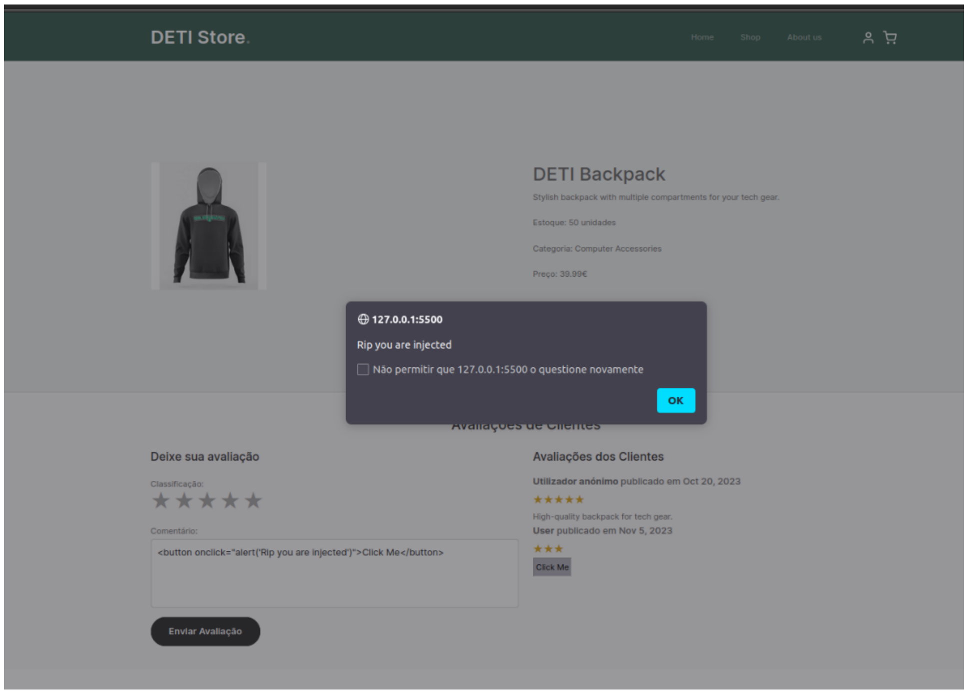
```
<button onclick="alert('Rip you are injected')">Click Me</button>
```

Avaliações dos Clientes

Utilizador anónimo publicado em Oct 20, 2023

★★★★★

High-quality backpack for tech gear.



DETI Backpack

Stylish backpack with multiple compartments for your tech gear.

Estoque: 50 unidades

Categoria: Computer Accessories

Preço: 39.99€

127.0.0.1:5500

Rip you are injected

Não permitir que 127.0.0.1:5500 o questione novamente

Avaliações de Clientes

Deixe sua avaliação

Classificação:

Comentário:

```
<button onclick="alert('Rip you are injected')">Click Me</button>
```

Avaliações dos Clientes

Utilizador anónimo publicado em Oct 20, 2023

★★★★★

High-quality backpack for tech gear.

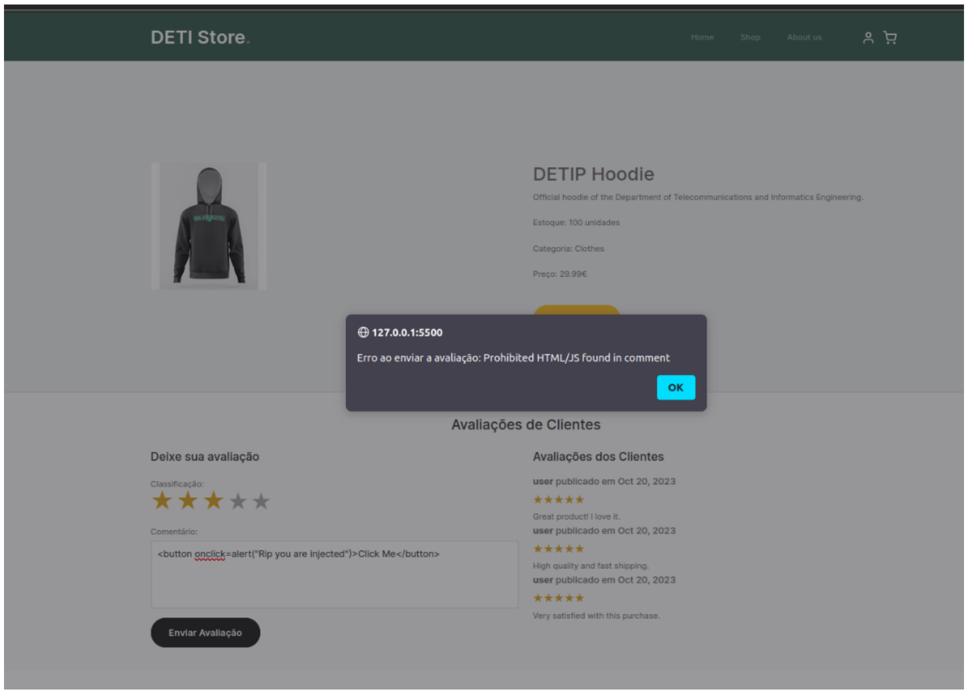
User publicado em Nov 5, 2023

★★★

Solution

To address the CWE-80 vulnerability, we have implemented input validation and sanitation using the Bleach library. All user-generated content, such as comments, input fields, and other forms of data, undergo rigorous scanning to ensure that no HTML or potentially harmful elements are executed. This sanitation process effectively mitigates the risk of malicious script injection, preserving the security, appearance, and functionality of the web application.

By implementing these security measures, we have significantly improved the application's security posture, preventing unauthorized script injection and content manipulation. The combination of input validation and sanitation effectively mitigates the CWE-80 vulnerability, ensuring a safer and more trustworthy user experience.



DETIP Hoodie

Official hoodie of the Department of Telecommunications and Informatics Engineering.

Estoque: 100 unidades

Categoria: Clothes

Preço: 29.99€

127.0.0.1:5500

Erro ao enviar a avaliação: Prohibited HTML/JS found in comment

Avaliações de Clientes

Deixe sua avaliação

Classificação:

Comentário:

```
<button onclick=alert("Rip you are injected")>Click Me</button>
```

Avaliações dos Clientes

user publicado em Oct 20, 2023

★★★★★

Great product! I love it.

user publicado em Oct 20, 2023

★★★★★

High quality and fast shipping.

user publicado em Oct 20, 2023

★★★★★

Very satisfied with this purchase.

Improper User Management (CWE-286)

Description

CWE-286, also known as "Incorrect User Management," is a security vulnerability related to how user accounts are managed within a web application. It encompasses issues like inadequate authentication, authorization, and session management.

Scenario

In our web application, there are some aspects of user management that have not been correctly implemented, this includes Authorization Issues such as not enforcing proper authorization checks, allowing users to access restricted resources or perform actions they shouldn't be allowed to.

An example would be “normal” users could access restricted areas such as the admin dashboard through altering the url. This could lead to serious damage to the database as the users could manage the stock and every other aspect of it.

Severity Score and Evaluation

- **Severity Score**: High: The vulnerability has a severe impact, and exploitation is relatively easy.
- **Points of View**: Both authorized and unauthorized users to perform actions that should only be available to authenticated and authorized users. Unauthorized and unauthenticated users can access admin pages, update the database, and perform actions they shouldn't be allowed to, highlighting the severity of this issue.

Demonstration

In our web application, there exists a critical vulnerability where sensitive functions or actions are accessible without proper authentication. This means that both authorized and unauthorized users can perform actions that should only be available to authenticated and authorized users, including accessing admin pages through URL manipulation and then it`s possible to updating the database by adding or removing products, adding products with negative price, dropping database tables, viewing the content of tables, and even altering product prices. This can have serious implications for the application's security and functionality.

Solution

To address this security issue, we have implemented the following measures:

1. **Role-Based Access Control:** We have introduced role-based access control, where only users with the "admin" attribute can access and perform critical functions in the application. Users without the "admin" attribute are redirected to the home page if they attempt to access administrative areas via URLs.
2. **User Authentication Check:** In addition to role-based access control, we have also implemented checks to ensure that actions performed by administrators require authentication. We validate the user's identity by verifying their "admin" attribute before allowing them to execute sensitive functions.

By implementing these security measures, we have significantly improved the application's security posture, preventing unauthorized access to critical functions and specifically addressing the issue of unauthorized access to admin pages via URL manipulation, mitigating the CWE-306 vulnerability.

Improper Input Validation (CWE-20)

Description

CWE-20, also known as "Improper Input Validation," is a web application security vulnerability that occurs when the application does not properly validate or sanitize user inputs. Without adequate input validation, attackers can supply malicious inputs that may lead to various security issues.

Scenario

In our web application, the user input validation process is not adequately implemented. For example, when users submit forms or input data, the application does not check and sanitize the inputs for malicious or unexpected content. This oversight leaves the application vulnerable to various attack vectors, such as SQL Injection and Cross-Site Scripting (XSS). This includes manipulating the product table, that should only be accessible to the admin, by defining negative prices, altering the stock of a product, etc...

This vulnerability comes as an add-on of the CWE-286 as the user needs to access first the admin dashboard as a "normal" user.

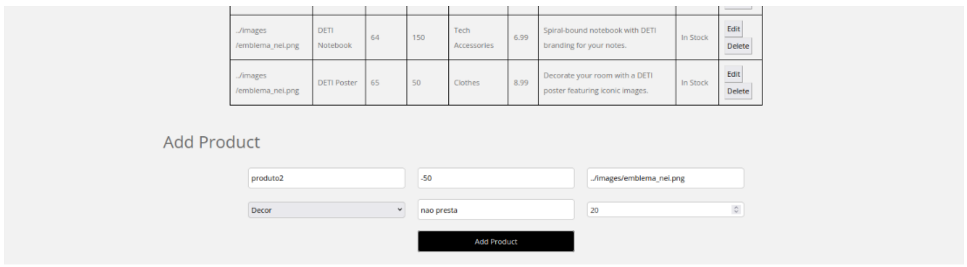
Severity Score and Evaluation

Severity Score: High: The vulnerability has a severe impact, and exploitation is relatively easy.

Points of View: The potential consequences include manipulation of the product table, which should only be accessible to administrators, and can involve actions like defining negative prices and altering product stock. These are all severe security problems that risk the security of the website and its owners.

Demonstration

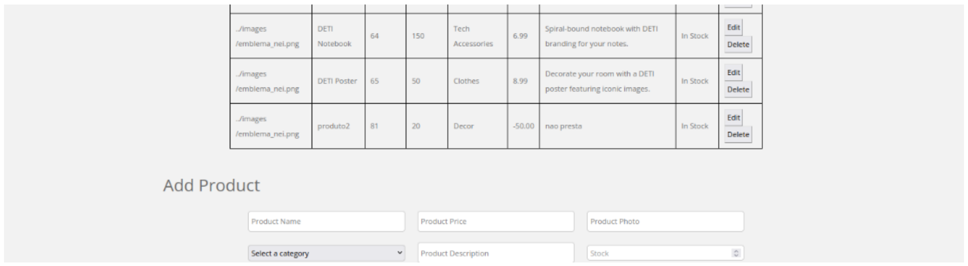
Here is an example of how a malicious attacker could use this CWE-20 vulnerability to perform a product table update of a product price.



../images /emblema_nei.png	DETI Notebook	64	150	Tech Accessories	6.99	Spiral-bound notebook with DETI branding for your notes.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Poster	65	50	Clothes	8.99	Decorate your room with a DETI poster featuring iconic images.	In Stock	Edit Delete

Add Product

produto2	-50	../images/emblema_nei.png
Decor	nao presta	20



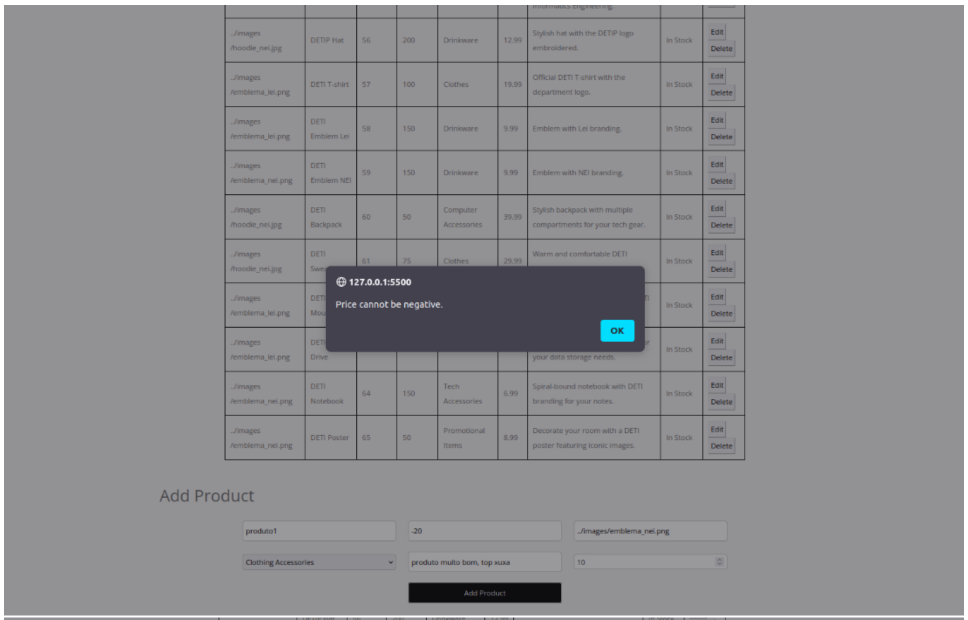
../images /emblema_nei.png	DETI Notebook	64	150	Tech Accessories	6.99	Spiral-bound notebook with DETI branding for your notes.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Poster	65	50	Clothes	8.99	Decorate your room with a DETI poster featuring iconic images.	In Stock	Edit Delete
../images /emblema_nei.png	produto2	81	20	Decor	-50.00	nao presta	In Stock	Edit Delete

Add Product

Product Name	Product Price	Product Photo
Select a category	Product Description	Stock

Solution

To address this vulnerability, we have implemented robust input validation for all user inputs in the application. For example, we ensure that prices must be greater than 0, and stock quantities must be positive or equal to 0. Furthermore, we have introduced user role-based access control to prevent non-administrator users from performing unauthorized actions within the product table.



../images /hoodie_nei.jpg	DETIP Hat	56	200	Drinkware	12.99	Stylish hat with the DETIP logo embroidered.	In Stock	Edit Delete
../images /emblema_lei.png	DETI T-shirt	57	100	Clothes	19.99	Official DETI T-shirt with the department logo.	In Stock	Edit Delete
../images /emblema_lei.png	DETI Emblem Lei	58	150	Drinkware	9.99	Emblem with Lei branding.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Emblem NEI	59	150	Drinkware	9.99	Emblem with NEI branding.	In Stock	Edit Delete
../images /hoodie_nei.jpg	DETI Backpack	60	50	Computer Accessories	39.99	Stylish backpack with multiple compartments for your tech gear.	In Stock	Edit Delete
../images /hoodie_nei.jpg	DETI Swe	61	75	Clothes	29.99	Warm and comfortable DETI	In Stock	Edit Delete
../images /emblema_lei.png	DETI Mou						In Stock	Edit Delete
../images /emblema_lei.png	DETI Drive					your data storage needs.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Notebook	64	150	Tech Accessories	6.99	Spiral-bound notebook with DETI branding for your notes.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Poster	65	50	Promotional Items	8.99	Decorate your room with a DETI poster featuring iconic images.	In Stock	Edit Delete

127.0.0.1:5500

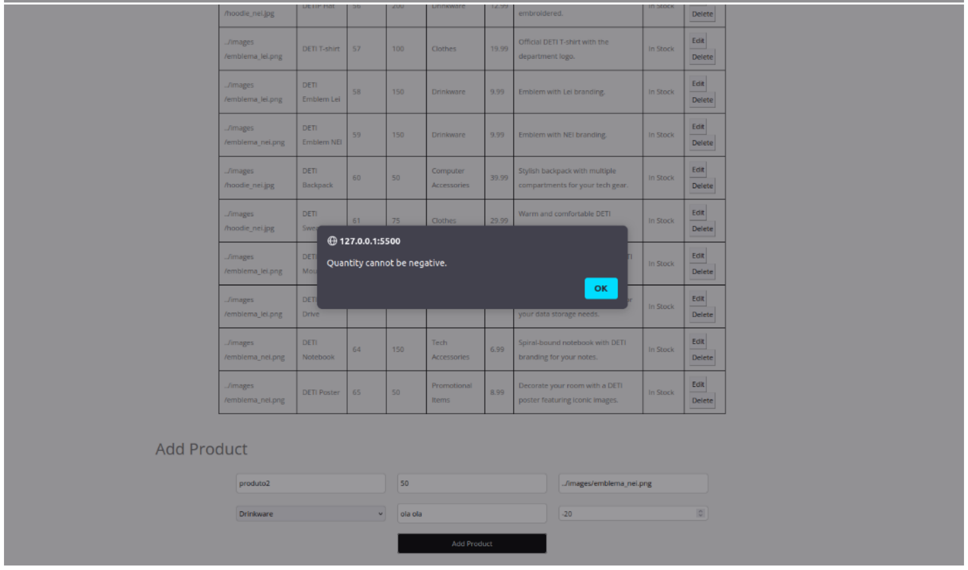
Price cannot be negative.

OK

Add Product

produto1	-20	../images/emblema_nei.png
Clothing Accessories	produto muito bom, top xuxa	10

Add Product



/hoodie_nei.jpg						embroidered.		Delete
../images /emblema_lei.png	DETI T-shirt	57	100	Clothes	19.99	Official DETI T-shirt with the department logo.	In Stock	Edit Delete
../images /emblema_lei.png	DETI Emblem Lei	58	150	Drinkware	9.99	Emblem with Lei branding.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Emblem NEI	59	150	Drinkware	9.99	Emblem with NEI branding.	In Stock	Edit Delete
../images /hoodie_nei.jpg	DETI Backpack	60	50	Computer Accessories	39.99	Stylish backpack with multiple compartments for your tech gear.	In Stock	Edit Delete
../images /hoodie_nei.jpg	DETI Swe	61	75	Clothes	29.99	Warm and comfortable DETI	In Stock	Edit Delete
../images /emblema_lei.png	DETI Mou						In Stock	Edit Delete
../images /emblema_lei.png	DETI Drive					your data storage needs.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Notebook	64	150	Tech Accessories	6.99	Spiral-bound notebook with DETI branding for your notes.	In Stock	Edit Delete
../images /emblema_nei.png	DETI Poster	65	50	Promotional Items	8.99	Decorate your room with a DETI poster featuring iconic images.	In Stock	Edit Delete

127.0.0.1:5500

Quantity cannot be negative.

OK

Add Product

produto2	50	../images/emblema_nei.png
Drinkware	ola ola	-20

Add Product

Missing Authentication for Critical Function (CWE-306)

Description

CWE-306, also known as "Missing Authentication for Critical Function," is a security vulnerability that occurs when a web application fails to adequately authenticate users for important or sensitive functions. This oversight can lead to unauthorized access and misuse of critical application features.

Scenario

In our web application, there are critical functions or actions that are accessible without proper authentication. This means that users, including unauthorized individuals, can perform actions that should only be available to authenticated and authorized users.

Potential consequences are Unauthorized Access, this vulnerability can result in unauthorized users gaining access to critical functions, Data Exposure, sensitive data may be at risk of exposure when critical functions lack proper authentication, Malicious Activities, where attackers or malicious users may exploit this vulnerability to perform unauthorized actions, potentially disrupting the application's intended functionality or causing harm

An example would be sensitive operations like updating the database by adding products, dropping tables, seeing the content of a table, etc...

Severity Score and Evaluation

Severity Score: High: The vulnerability has a severe impact, and exploitation is relatively easy.

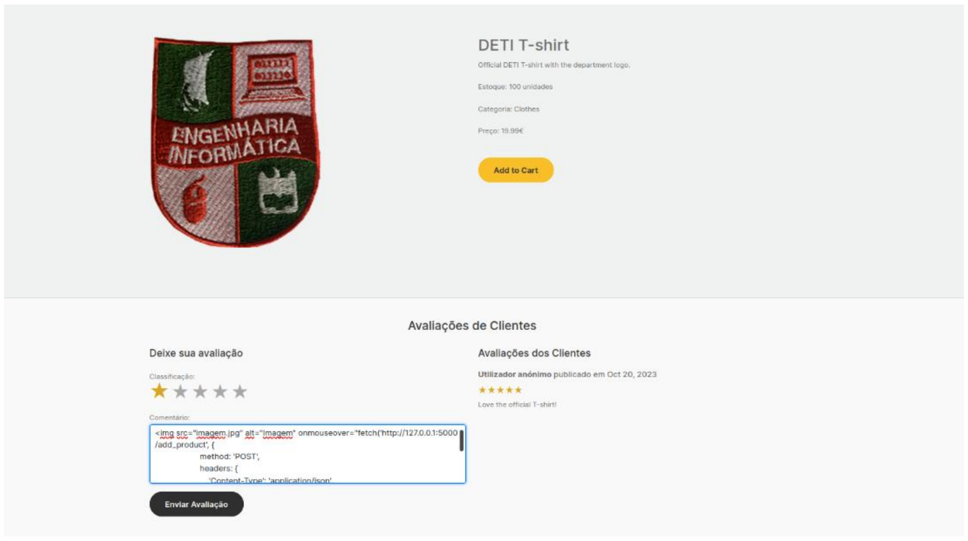
Points of View: The vulnerability involves the accessibility of critical functions without proper authentication, which can lead to unauthorized access to sensitive operations, data exposure, and the potential for malicious activities that can disrupt the application's functionality. Adding a product with a negative price to the database demonstrates the severity of the issue.

Demonstration

In this scenario, we will illustrate how a malicious attacker could exploit the CWE-306 vulnerability to gain unauthorized access to sensitive operations within our web application. The attacker can use a crafted image tag with a JavaScript payload, as shown below:

```
1. <img src="imagem.jpg" alt="Imagem"
   onmouseover="fetch('http://127.0.0.1:5000/add_product', {  method: 'POST',
   headers: { 'Content-Type': 'application/json',  },
              body: JSON.stringify({ name: 'p1', price: -50, stock: 100, photo: 'ffv',
                categories_id: 2,description: 'wrvebwe',  }),})">
```

This payload, when executed, triggers an unauthorized operation to add a product to the database with a negative price.



DETI T-shirt

Official DETI T-shirt with the department logo.

Estoque: 100 unidades

Categoria: Clothes

Preço: 19.99€

Avaliações de Clientes

Deixe sua avaliação

Classificação:

★★★★★

Comentário:

```
<img src="imagem.jpg" alt="imagem" onmouseover="fetch('http://127.0.0.1:5000
/add_product', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
```

Avaliações dos Clientes

Utilizador anónimo publicado em Oct 20, 2023

★★★★★

Love the official T-shirt!

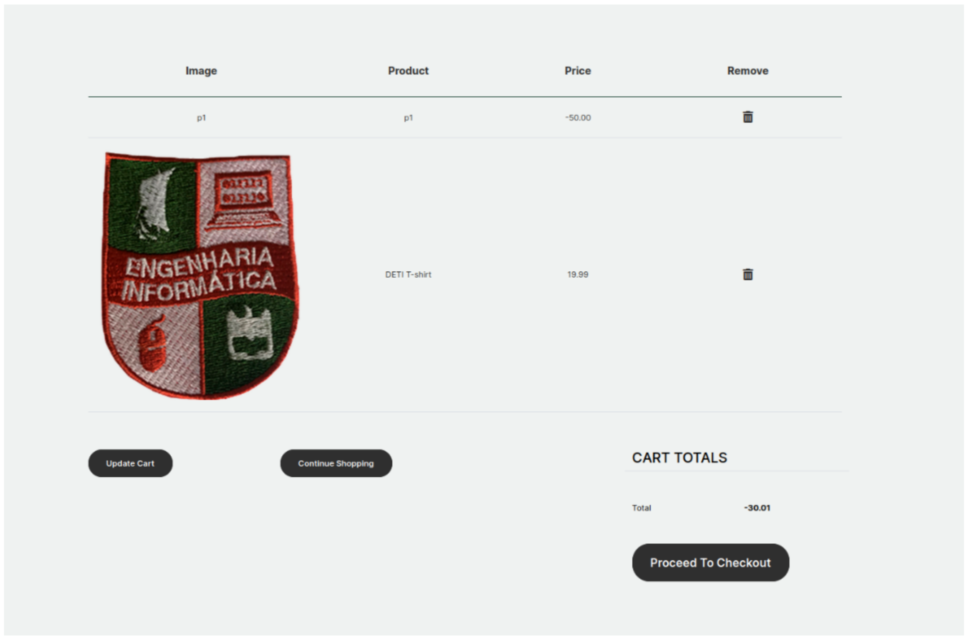


Image	Product	Price	Remove
p1	p1	-50.00	🗑
	DETI T-shirt	19.99	🗑

CART TOTALS

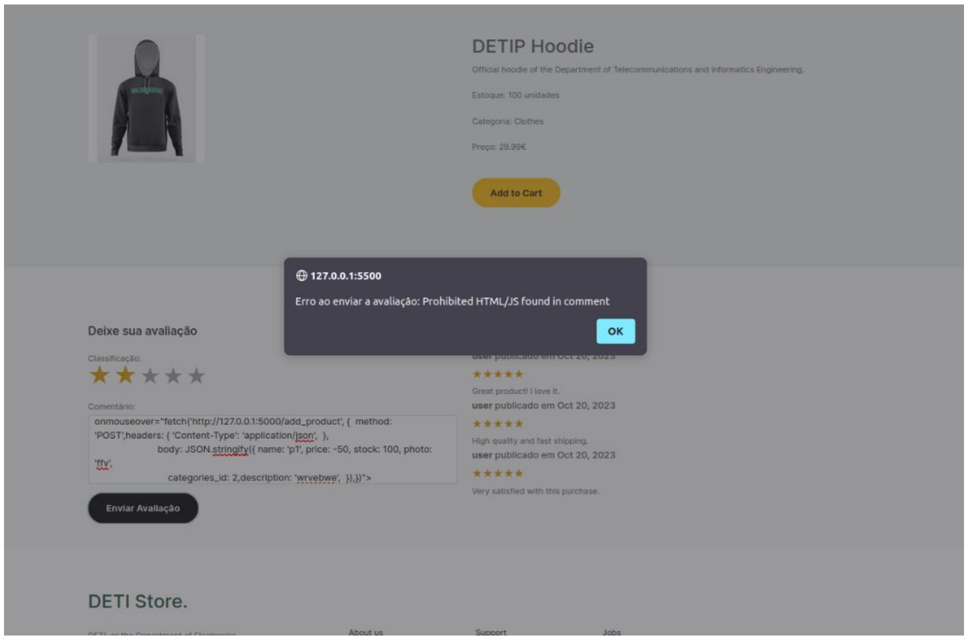
Total	-30.01

Solution

To address this security issue, we have implemented the following measures:

1. **Role-Based Access Control:** We have introduced role-based access control, where only users with the "admin" attribute can access and perform critical functions in the application. Users without the "admin" attribute are restricted from executing these functions, preventing unauthorized access.
2. **Input Sanitization:** We have implemented input validation and sanitation for user-generated content, particularly in areas such as product reviews. The Bleach library is used to scan and sanitize user inputs to ensure that no HTML or potentially harmful elements are executed. This measure prevents unauthorized operations like adding products with negative prices.

By implementing these security measures, we have significantly improved the application's security posture, preventing unauthorized access to sensitive operations and enhancing the overall security and functionality of the application. The combination of role-based access control and input sanitation effectively mitigates the CWE-306 vulnerability.



DETIP Hoodie

Official hoodie of the Department of Telecommunications and Informatics Engineering.

Estoque: 100 unidades

Categoria: Clothes

Preço: 29.99€

127.0.0.1:5500

Erro ao enviar a avaliação: Prohibited HTML/JS found in comment

Deixe sua avaliação

Classificação: ★★☆☆☆

Comentário:

```
onmouseover="fetch('http://127.0.0.1:5000/add_product', {  method:
'POST',headers: { 'Content-Type': 'application/json',  },
                body: JSON.stringify({ name: 'p1', price: -50, stock: 100, photo:
'ffy',
                  categories_id: 2,description: 'wrvebwe',  }),})">
```

★★★★★

Great product! I love it.

user publicado em Oct 20, 2023

★★★★★

High quality and fast shipping.

user publicado em Oct 20, 2023

★★★★★

Very satisfied with this purchase.

DETI Store.

Password in Configuration File (CWE-260)

Description

CWE-260, also known as "Password in Configuration File," is a security vulnerability that occurs when sensitive information, such as passwords or connection strings, is stored in plain text within the configuration files of an application. This misconfiguration can lead to unauthorized access to sensitive resources.

Scenario

In our web application, sensitive information, such as database connection strings, is stored in configuration files without proper encryption or protection. These configuration files may be accessible to unauthorized individuals or exposed through code repositories. This misconfiguration can potentially allow attackers to access sensitive resources, compromise security, and misuse these credentials.

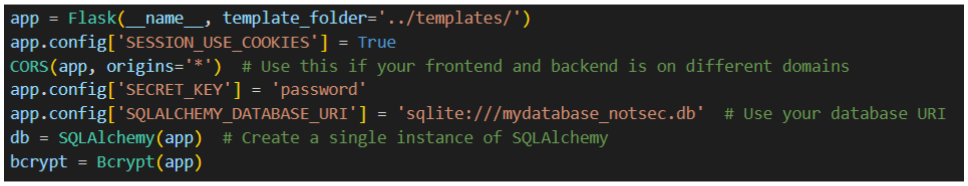
Severity Score and Evaluation

Severity Score: "Low": The vulnerability doesn´t have that big of an impact, and exploitation seems hard.

Points of View: Although having sensitive information, in plain text, on a public file, is never a good practice, in this case, the attacker can´t do much with the database credentials, since SQLAlchemy is a safe python toolkit and Object Relational Mapper.

Demonstration

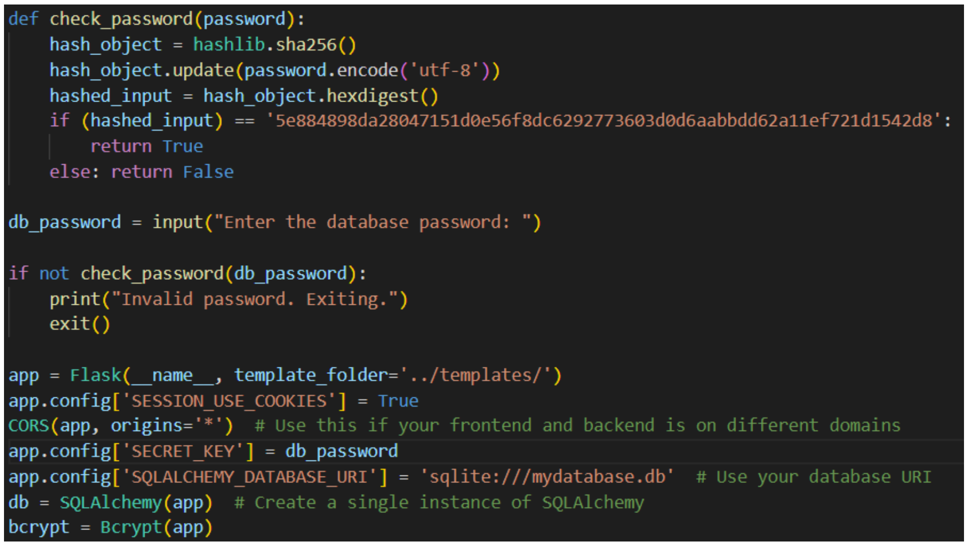
Here, on our app, as seen on the fourth line of the displayed code snippet, the 'SECRET_KEY' used to configure the app is stored in plaintext ('password').



```
app = Flask(__name__, template_folder='../templates/')
app.config['SESSION_USE_COOKIES'] = True
CORS(app, origins='*')  # Use this if your frontend and backend is on different domains
app.config['SECRET_KEY'] = 'password'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///mydatabase_notsec.db'  # Use your database URI
db = SQLAlchemy(app)  # Create a single instance of SQLAlchemy
bcrypt = Bcrypt(app)
```

Solution

To address this security issue, in our secure application, we now ask the user for the database password before running the controller.py file; The password the user inserts is then encrypted using the SHA-256 algorithm, which is an irreversible hashing algorithm, and compared to the hashed version of our actual database password; Now we are able to make sure that non-authorized users don't have access to our database credentials, by not having our 'SECRET_KEY' in plaintext and by making it only runnable with previous knowledge of the password.



```python
def check_password(password):
    hash_object = hashlib.sha256()
    hash_object.update(password.encode('utf-8'))
    hashed_input = hash_object.hexdigest()
    if (hashed_input) == '5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8':
        return True
    else: return False

db_password = input("Enter the database password: ")

if not check_password(db_password):
    print("Invalid password. Exiting.")
    exit()

app = Flask(__name__, template_folder='../templates/')
app.config['SESSION_USE_COOKIES'] = True
CORS(app, origins='*')  # Use this if your frontend and backend is on different domains
app.config['SECRET_KEY'] = db_password
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///mydatabase.db'  # Use your database URI
db = SQLAlchemy(app)  # Create a single instance of SQLAlchemy
bcrypt = Bcrypt(app)
```

Not Using Password Aging (CWE-262)

Description

CWE-262, "Not Using Password Aging," is a security vulnerability that occurs when a web application does not implement password aging policies. Password aging helps ensure that users regularly update their passwords, reducing the risk of unauthorized access.

Scenario

In our web application, there is no implementation of password aging policies. Users are not required to change their passwords periodically, and there are no mechanisms in place to enforce password expiration. As a result, users can continue using the same passwords indefinitely, increasing the risk of compromised accounts.

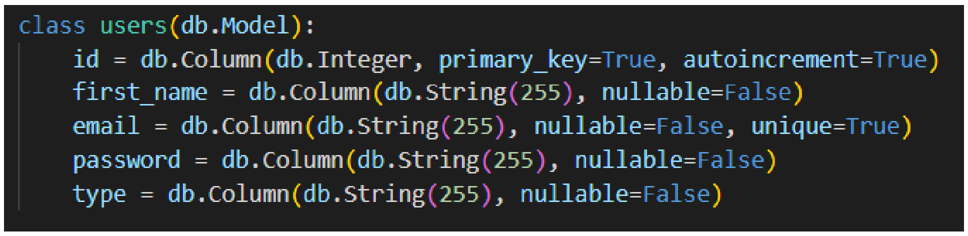
Severity Score and Evaluation

Severity Score: "Low": The vulnerability doesn´t have a direct impact

Points of View: The lack of password expiration policies can lead to weaker security, as users may not feel compelled to regularly update their passwords, and attackers who gain access to user credentials may be able to use them for an extended period without the need to change passwords. Despite this information, unless our application is already exposed to attackers, no harm can be done just by not using password aging.

Demonstration

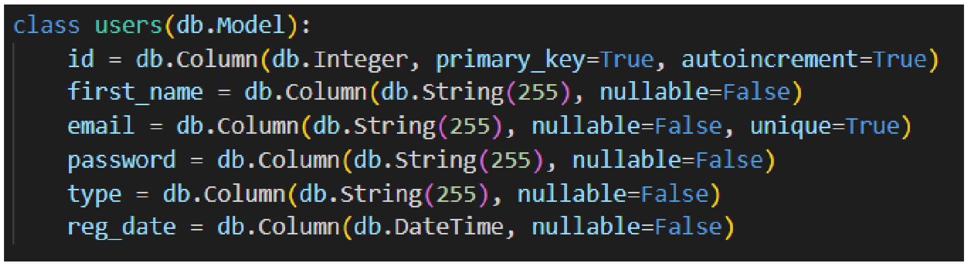
On our app, as seen, we don't have any field on the 'users' table to store the last time a user has changed their password, meaning we don't have a way to protect ourselves against password aging.



```
class users(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    first_name = db.Column(db.String(255), nullable=False)
    email = db.Column(db.String(255), nullable=False, unique=True)
    password = db.Column(db.String(255), nullable=False)
    type = db.Column(db.String(255), nullable=False)
```

Solution

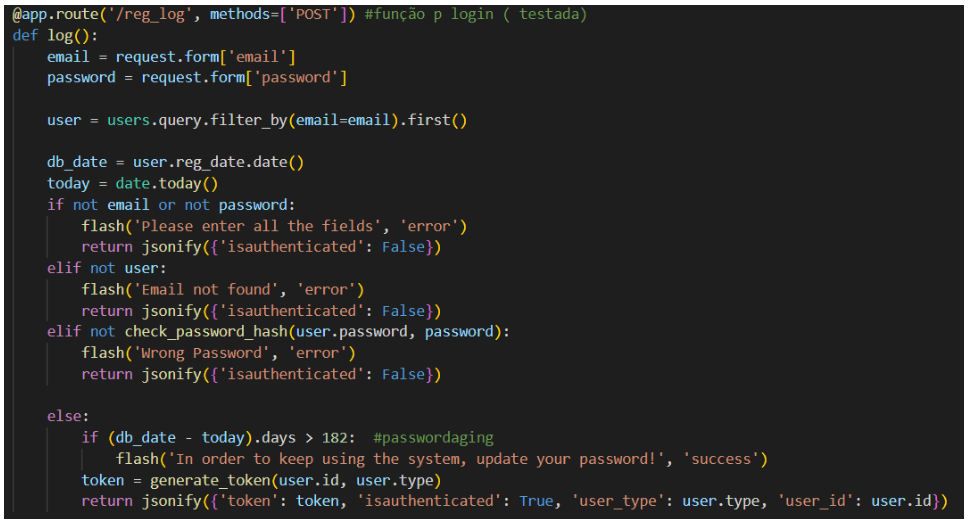
In order to protect our app against password aging, we now have a field in our database table 'users' that stores the last time our users changed their password; When a user registers onto our app, the present date is stored to 'reg_date';



```python
class users(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    first_name = db.Column(db.String(255), nullable=False)
    email = db.Column(db.String(255), nullable=False, unique=True)
    password = db.Column(db.String(255), nullable=False)
    type = db.Column(db.String(255), nullable=False)
    reg_date = db.Column(db.DateTime, nullable=False)
```

When our users log in onto our secure application, if the difference between the current date and the value stored in the users 'reg_date' is bigger than half a year (182 days), then a message will be shown to the user, encouraging them to change their password.

With this solution, users are more aware of the security issues associated with their passwords and more likely to periodically change them.



```python
@app.route('/reg_log', methods=['POST']) #função p login ( testada)
def log():
    email = request.form['email']
    password = request.form['password']

    user = users.query.filter_by(email=email).first()

    db_date = user.reg_date.date()
    today = date.today()
    if not email or not password:
        flash('Please enter all the fields', 'error')
        return jsonify({'isauthenticated': False})
    elif not user:
        flash('Email not found', 'error')
        return jsonify({'isauthenticated': False})
    elif not check_password_hash(user.password, password):
        flash('Wrong Password', 'error')
        return jsonify({'isauthenticated': False})

    else:
        if (db_date - today).days > 182:  #passwordaging
            flash('In order to keep using the system, update your password!', 'success')
        token = generate_token(user.id, user.type)
        return jsonify({'token': token, 'isauthenticated': True, 'user_type': user.type, 'user_id': user.id})
```

Plaintext Storage of a Password (CWE-256)

Description

CWE-256, "Plaintext Storage of a Password," is a security vulnerability that occurs when a web application stores user passwords in plaintext, without proper encryption or hashing. Storing passwords in plaintext poses a significant security risk.

Scenario

In our web application, user passwords are stored in plaintext within the application's database. This means that anyone with access to the database can view and potentially misuse user passwords, leading to security vulnerabilities.

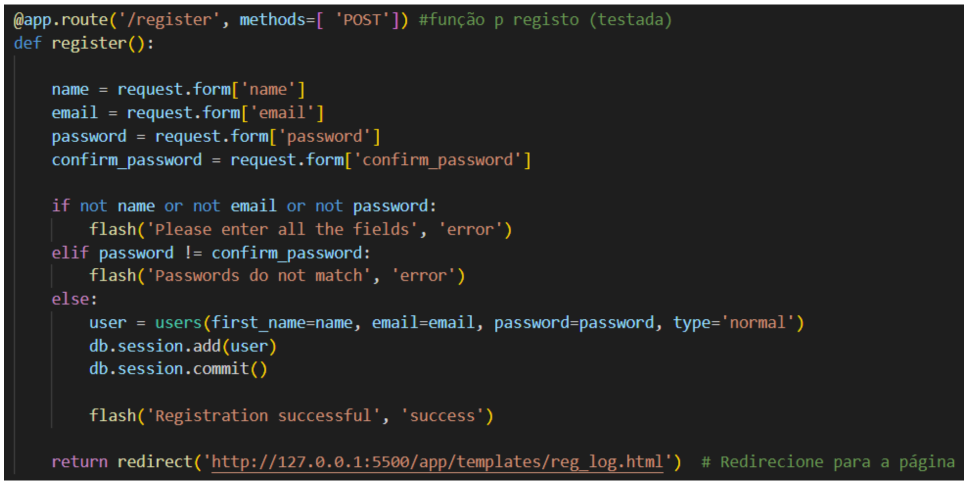
Severity Score and Evaluation

Severity Score: "Medium": The vulnerability can have some impact.

Points of View: Storing passwords in plaintext allows anyone with access to the database to view user passwords, putting user accounts at risk. However, since no data associated with payment methods is stored into the database, even if an attacker had access to the users' passwords, the most damage he could do would be either to log in to that user's account and change the password, making it so that the user could never create another account with that email address, or to delete the users account.

Demonstration

As seen in the code snippet below, when a user registers to our app, the password they chose is directly stored in the database in plaintext, making it vulnerable to attackers.

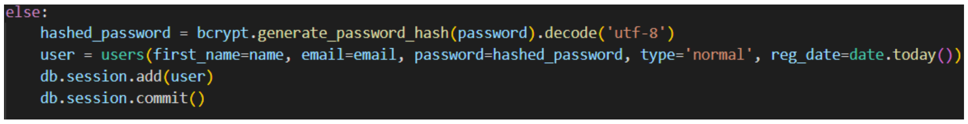

```
@app.route('/register', methods=[ 'POST']) #função p registo (testada)
def register():

    name = request.form['name']
    email = request.form['email']
    password = request.form['password']
    confirm_password = request.form['confirm_password']

    if not name or not email or not password:
        flash('Please enter all the fields', 'error')
    elif password != confirm_password:
        flash('Passwords do not match', 'error')
    else:
        user = users(first_name=name, email=email, password=password, type='normal')
        db.session.add(user)
        db.session.commit()

        flash('Registration successful', 'success')

    return redirect('http://127.0.0.1:5500/app/templates/reg_log.html')  # Redirecione para a página
```

Solution

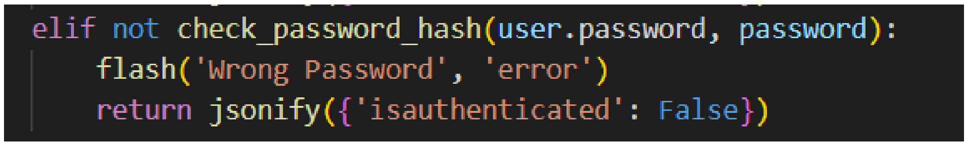
In order to protect the app against this weakness, in the secure version of our application, we store only the hashed version of the users' passwords. We do this by using 'Bcrypt', a Flask extension that provides bcrypt hashing utilities, and the 'check_password_hash' function;

```
from flask_bcrypt import Bcrypt, check_password_hash
```

Now, in the 'register' function of our secure 'controller.py', as previously said, we register the user using the hashed version of their password to the database;


```
else:
    hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')
    user = users(first_name=name, email=email, password=hashed_password, type='normal', reg_date=date.today())
    db.session.add(user)
    db.session.commit()
```

In our 'log' function, to make sure that the user has used the right credentials to access their account, we use the previously mentioned 'check_password_hash' function to compare the user's input to the data stored in the database.


```
elif not check_password_hash(user.password, password):
    flash('Wrong Password', 'error')
    return jsonify({'isauthenticated': False})
```

Weak Password Requirements (CWE-521)

Description

CWE-521, "Weak Password Requirements," is a security vulnerability that occurs when a web application does not enforce strong password requirements for user accounts. Weak password requirements can lead to easily guessable or cracked passwords.

Scenario

In our web application, the password requirements for user accounts are non-existent. This means that users can create weak and easily guessable passwords, such as short and simple combinations. Weak password requirements can make it easier for attackers to guess or crack user passwords.

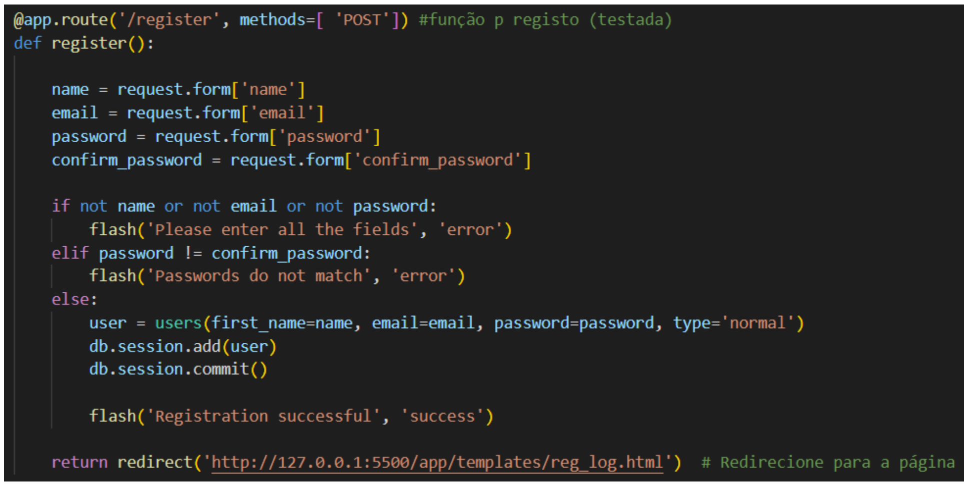
Severity Score and Evaluation

Severity Score: "Low": The vulnerability alone isn't enough for an attack other than brute force;

Points of View: Using no password requirements can make it easier for attackers to perform brute-force attacks to guess user passwords. However, just like mentioned above in the evaluation of the CWE-256 weakness, since no critical data is stored in our database, the most damage an attacker could do is impersonation and password changing.

Demonstration

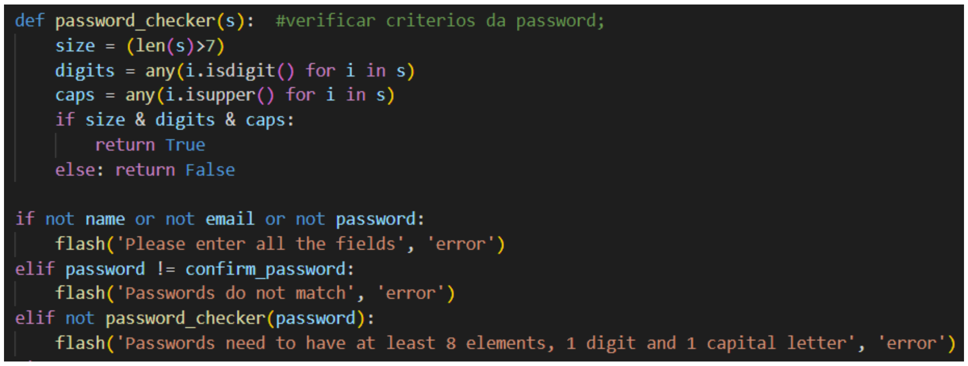
On our app, as seen in the register function below, we have no criteria for what a password should be like, making it so, that if a user wants their password to be a single character, that is allowed.



```
@app.route('/register', methods=[ 'POST']) #função p registo (testada)
def register():

    name = request.form['name']
    email = request.form['email']
    password = request.form['password']
    confirm_password = request.form['confirm_password']

    if not name or not email or not password:
        flash('Please enter all the fields', 'error')
    elif password != confirm_password:
        flash('Passwords do not match', 'error')
    else:
        user = users(first_name=name, email=email, password=password, type='normal')
        db.session.add(user)
        db.session.commit()

        flash('Registration successful', 'success')

    return redirect('http://127.0.0.1:5500/app/templates/reg_log.html')  # Redirecione para a página
```

Solution

In order to ensure that the users don't have a weak password, in the secure version of our application, more concretely, in the register flask function, we implemented the 'password_checker' function, which will verify if the password chosen by the user is, at least, 8 characters long, has a capital character and has a digit. This will ensure that our users will always have a minimally complex password.



```
def password_checker(s):  #verificar criterios da password;
    size = (len(s)>7)
    digits = any(i.isdigit() for i in s)
    caps = any(i.isupper() for i in s)
    if size & digits & caps:
        return True
    else: return False

if not name or not email or not password:
    flash('Please enter all the fields', 'error')
elif password != confirm_password:
    flash('Passwords do not match', 'error')
elif not password_checker(password):
    flash('Passwords need to have at least 8 elements, 1 digit and 1 capital letter', 'error')
```

Improper Neutralization of Special Elements used in an SQL Command (CWE-89)

Description

CWE-89, also known as "Improper Neutralization of Special Elements used in an SQL Command," is a common security vulnerability in web applications where user inputs are not properly validated or sanitized before being included in SQL queries. This can lead to SQL Injection attacks.

Scenario

In our web application, there is an instance where user inputs in a forms field are directly incorporated into SQL queries without proper validation or sanitization. This oversight can potentially allow attackers to inject malicious SQL code into this input, leading to SQL Injection vulnerabilities.

Severity Score and Evaluation

Severity Score: "High": The vulnerability has a severe impact, and exploitation is relatively easy.

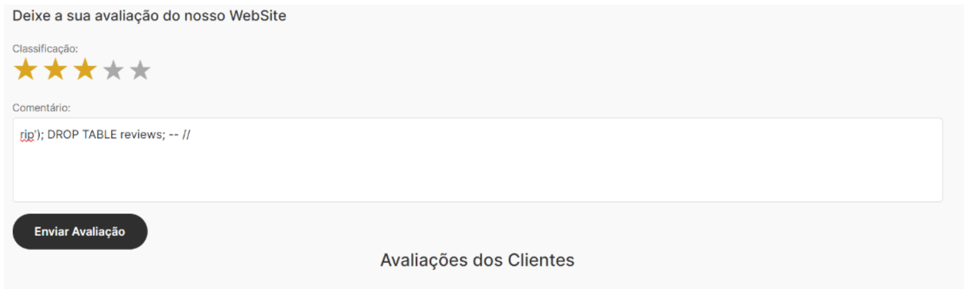
Points of View: By not having proper validation or sanitization of the input given by users onto the vulnerable field, we are allowing users to being able to use SQL Injection, which can, in the most critical scenario, be used to drop the entire database.

Demonstration

On our app, in the reviews' page, we are performing the following query to send the reviews to the database:

```
insert_query = f"INSERT INTO reviews (user_id, rating, comment) VALUES
({user_id}, {rating}, '{comment}')"

cursor.executescript(insert_query)
```

However, if the value of 'comment' were to be something like the input given in the picture below, this would result in the execution of the previous query (with the value of 'comment' being 'rip'), followed by the execution of the 'DROP TABLE reviews;' query, with '-- //' being used to ignore the remainder of query in our code.


Deixe a sua avaliação do nosso WebSite

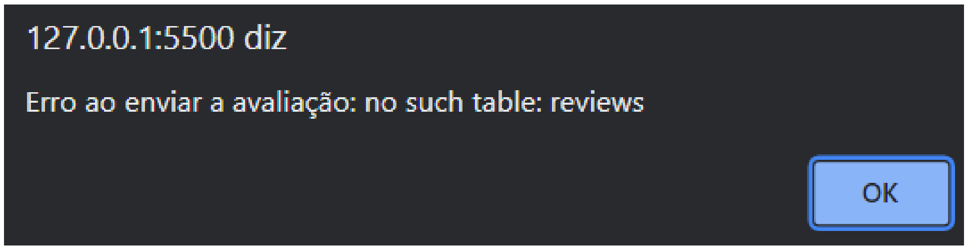
Classificação:

Comentário:

rip'); DROP TABLE reviews; -- //

Avaliações dos Clientes

If we were to try and post another review, the following message would show up, meaning that the SQL Injection try was successful.


127.0.0.1:5500 diz

Erro ao enviar a avaliação: no such table: reviews

Solution

In order to ensure that we aren't allowing SQL Injection in our secure application, we changed our query to be like this:

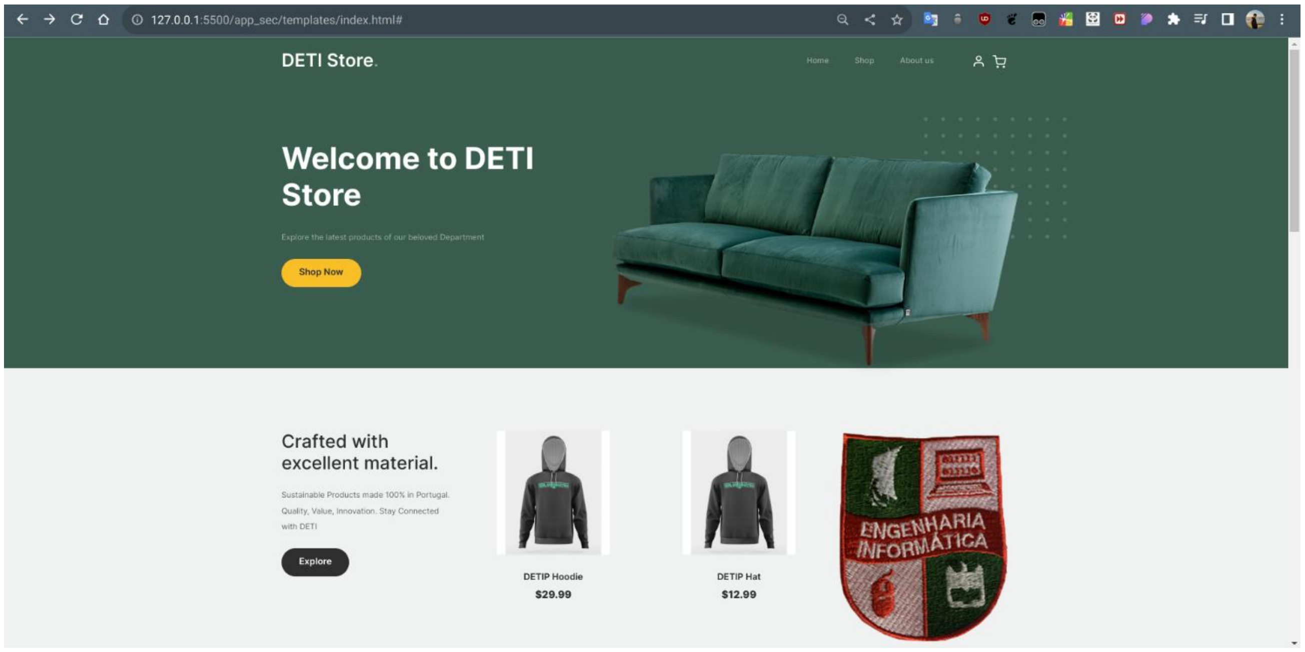
```
review = reviews(user_id=user_id, rating=rating, comment=clean_comment)

db.session.add(review)

db.session.commit()
```

This will ensure that SQL Injection won't work, since we are now using SQLAlchemy, an Object Relational Mapper that will prevent any type of injection.

Shop Overview

To start presenting our user interface we will show our e-commerce website step by step as it was built.

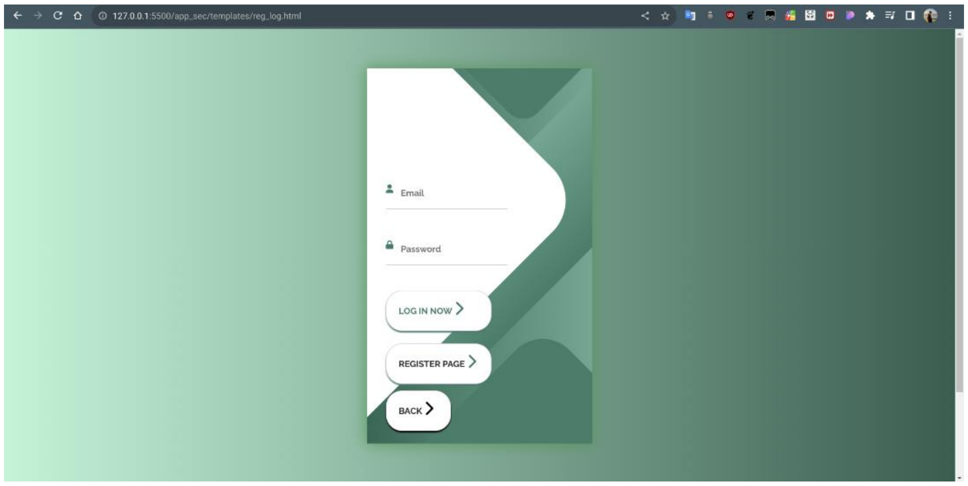
- **Index.html**


127.0.0.1:5500/app_sec/templates/index.html#
DETI Store.
Home
Shop
About us
Welcome to DETI Store
Explore the latest products of our beloved Department
Shop Now
Crafted with excellent material.
Sustainable Products made 100% in Portugal. Quality, Value, Innovation. Stay Connected with DETI
Explore
DETIP Hoodie
$29.99
DETIP Hat
$12.99
ENGENHARIA INFORMÁTICA

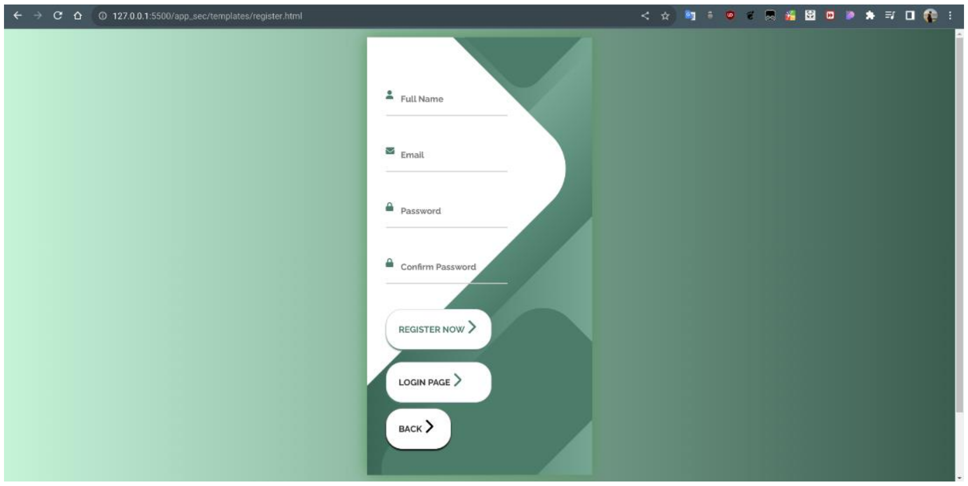
We begin by examining our main page, the index, focusing on the navbar which includes several features: "Home" (leading to index.html), "Shop" (leading to our shop page), and "About Us" (providing information about our project). Additionally, there are two icons: a "person" icon for login or registration, and a "shopping cart" icon displaying selected items for purchase.

• Login and regist new user

Right after clicking on the "person" icon in the navbar, as mentioned earlier, you will be redirected to this section. If you already have an account, you will be able to enter your email and respective password in this section and then click on "Log in Now."



If you are a new user and don´t yet have an account, you should click the "Register Page" button to create one. The user interface should appear as shown above.

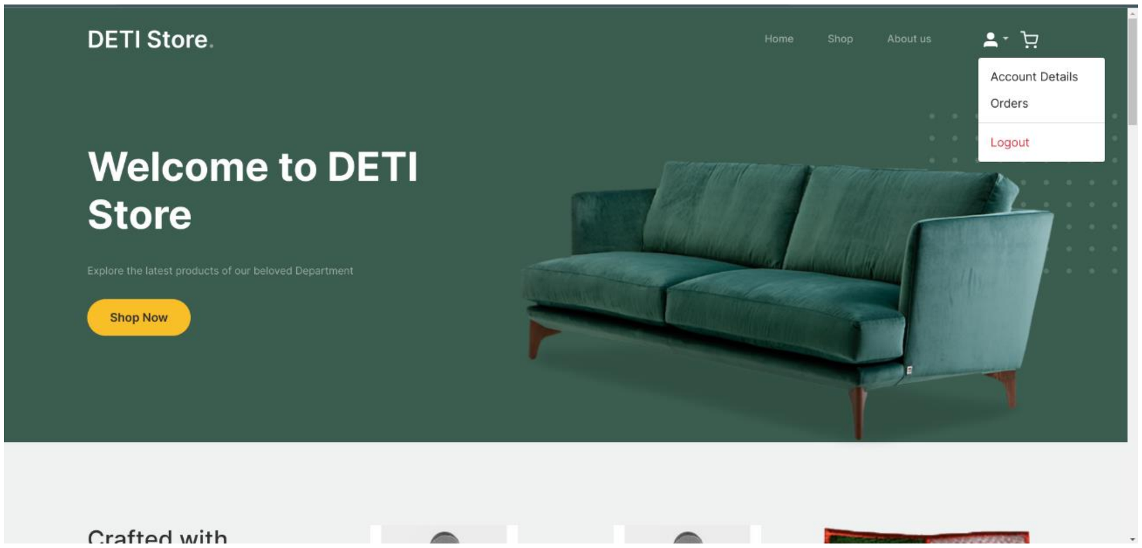


You must fill in the fields with your informations and then click on the button “Register Now” and now you should be able to log in into our website.

- **User profiles**

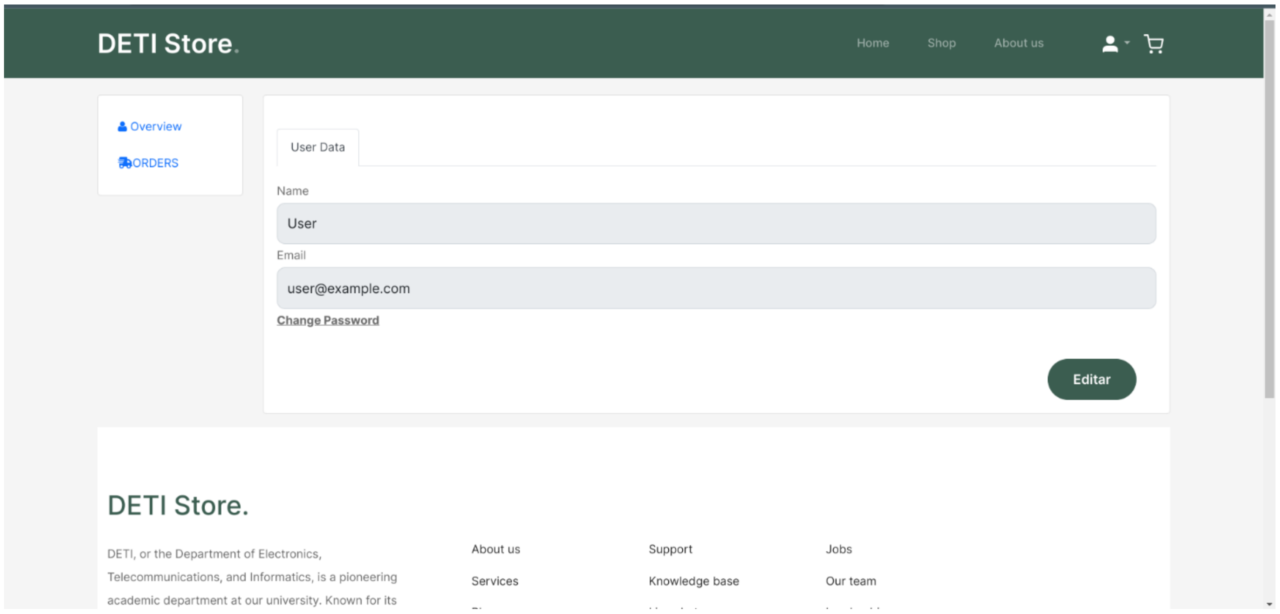
After successfully logging into our website, you should now see a dropdown menu when you click on the person's icon. This menu provides you with the following options:

- Account Details
- Orders
- Logout



- ## User’s dashboard

When you click on "Account Details," you will be redirected to the user's dashboard, which is a simple webpage where you can access your personal information, including your name, email address, and password.



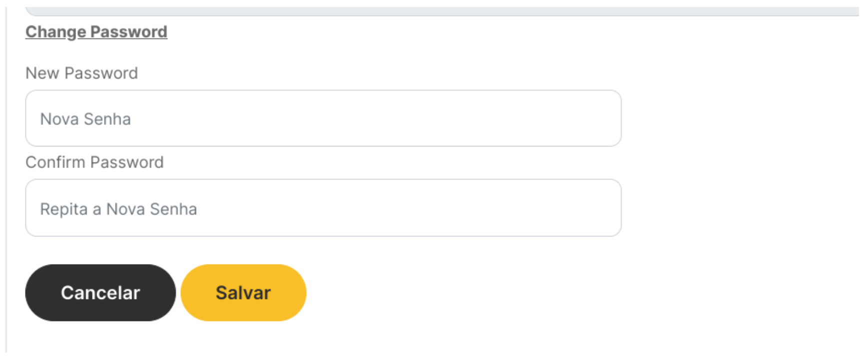
Name: User

Email: user@example.com

Change Password

DETI, or the Department of Electronics, Telecommunications, and Informatics, is a pioneering academic department at our university. Known for its

Your name and email address will be displayed as shown in the picture, and you can change your password by clicking on the "Change Password" option. Here is what you will see:

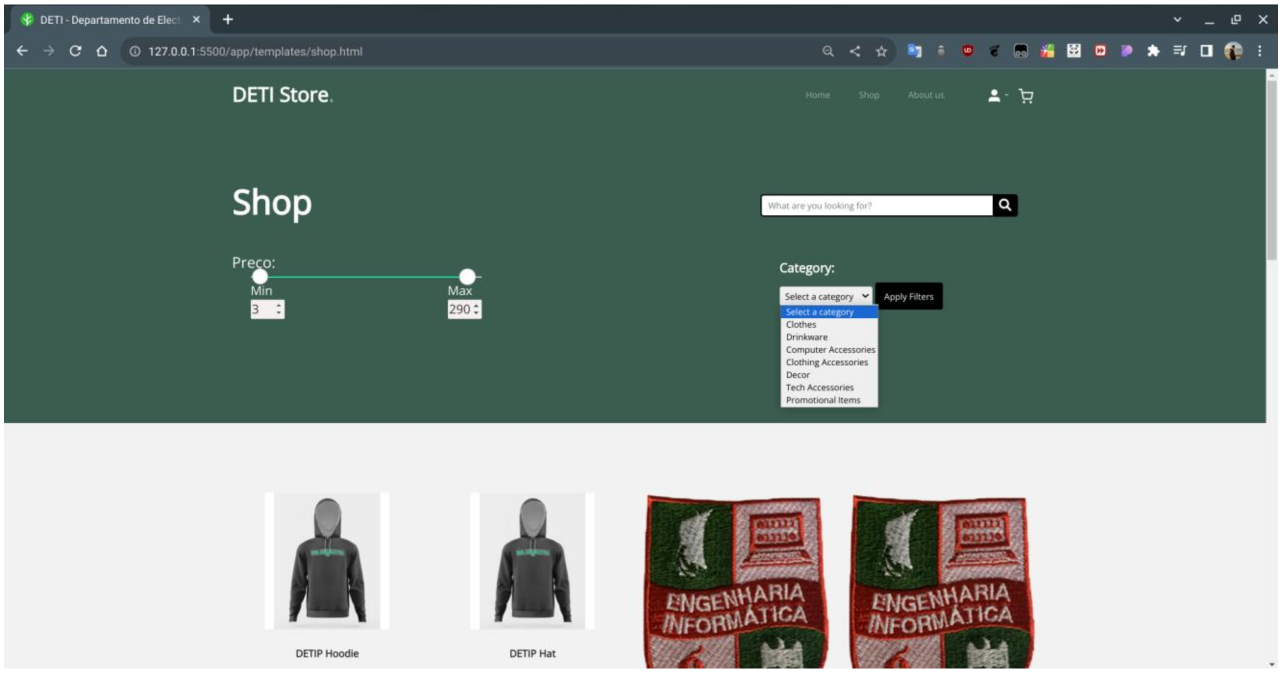


- **Product Catalog:**

  • **Product categories and filters**

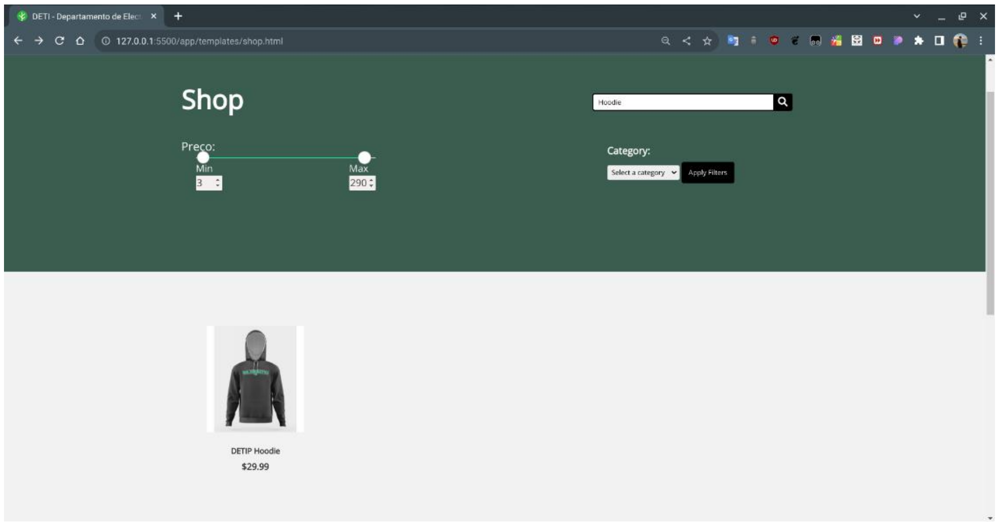
When you press the “Shop” in navbar you will be redirected to our store where you can choose and purchase our items.

When entering our shop section, you will be able to use our filters that appear at the top of our shop (filter by price, search bar for product name search and search by category).

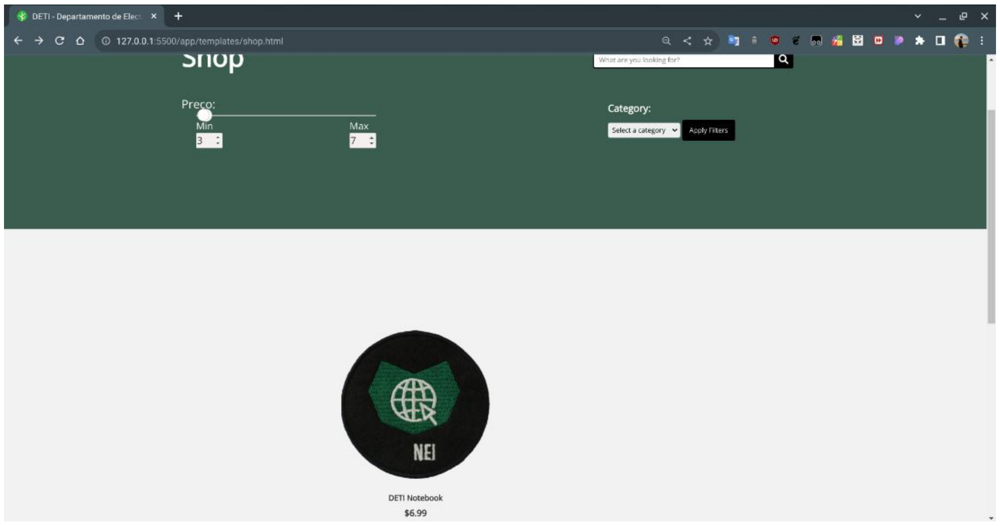


• **Product search functionality**

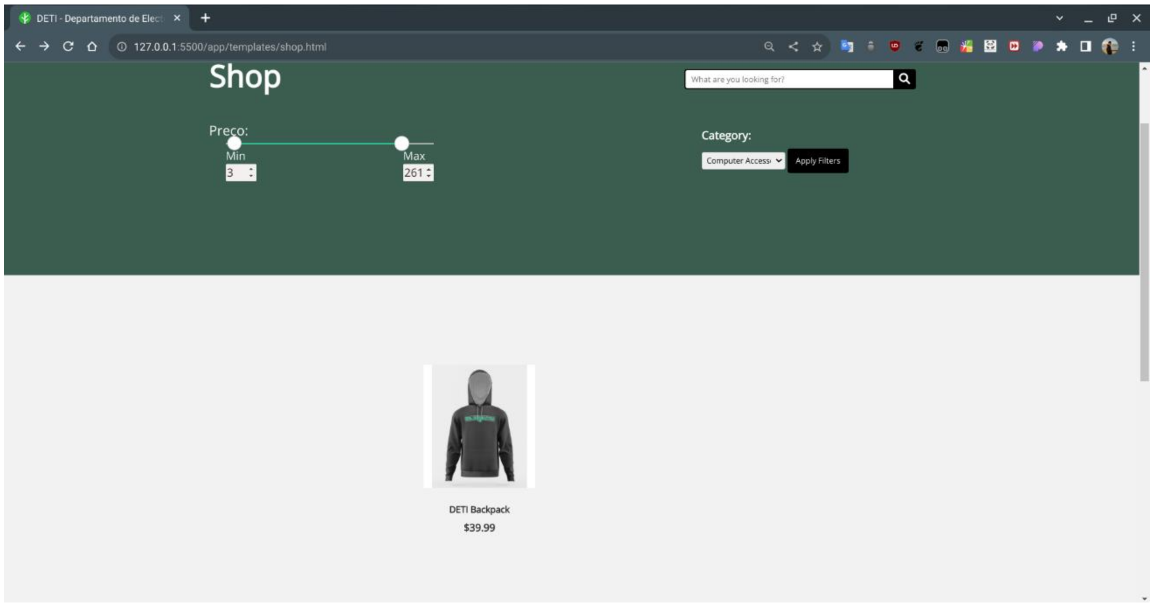
Filter by product name:



Filter by price:

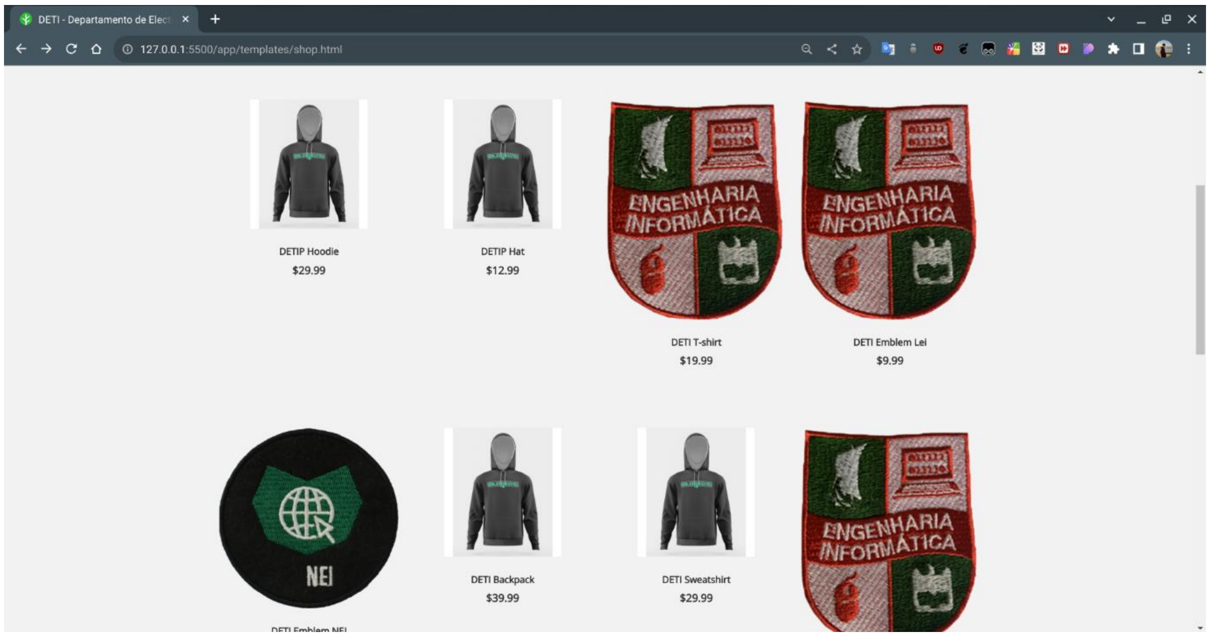

DETI - Departamento de Elect
127.0.0.1:5500/app/templates/shop.html
Shop
What are you looking for?
Preço:
Min
3
Max
7
Category:
Select a category
Apply Filters
NEI
DETI Notebook
$6.99

Filter by Category:

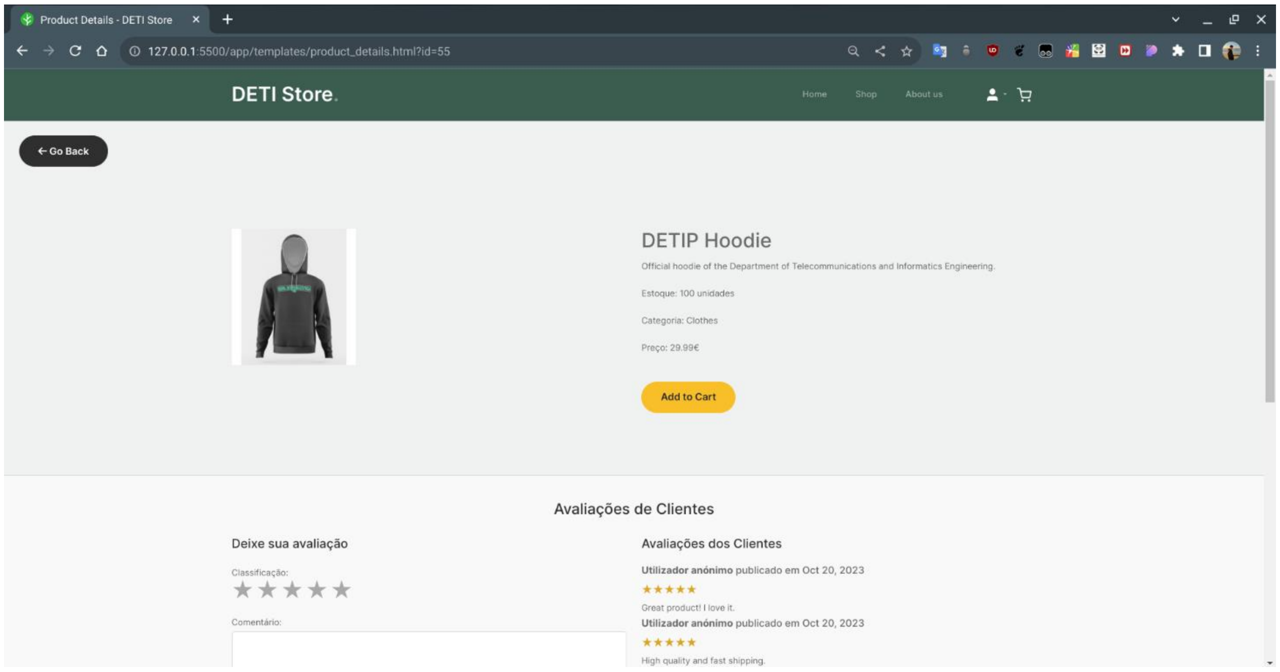

DETI - Departamento de Elect
127.0.0.1:5500/app/templates/shop.html
Shop
What are you looking for?
Preço:
Min
3
Max
261
Category:
Computer Access
Apply Filters
DETI Backpack
$39.99

• Product listings with details (name, description, price, images)

As you navigate through our available products, you will discover a wide variety of items available for sale.



After you right click on one of our products, you will be able to see more information about the item, despite the price that you already know from the product list.


Product Details - DETI Store
127.0.0.1:5500/app/templates/product_details.html?id=55
DETI Store.
Home
Shop
About us
Go Back
Add to Cart

DETIP Hoodie

Official hoodie of the Department of Telecommunications and Informatics Engineering.

Estoque: 100 unidades

Categoria: Clothes

Preço: 29.99€

Avaliações de Clientes

Deixe sua avaliação

Classificação: ★★★★★

Comentário:

Avaliações dos Clientes

Utilizador anónimo publicado em Oct 20, 2023

★★★★★

Great product! I love it.

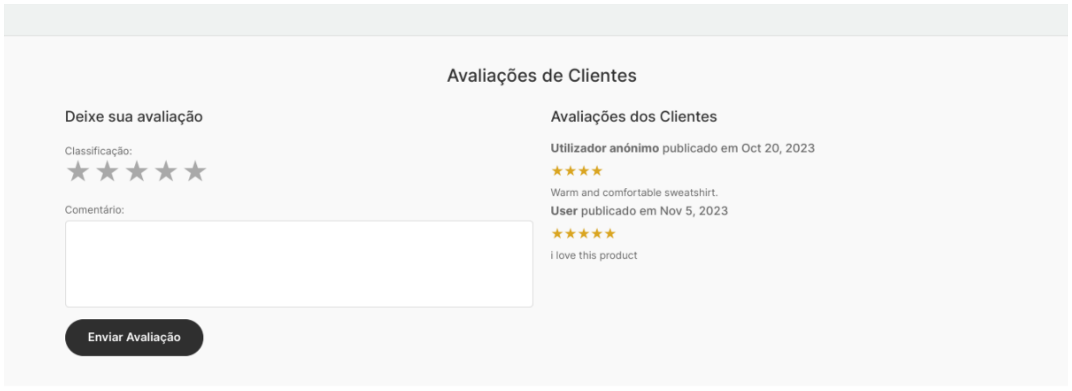
Utilizador anónimo publicado em Oct 20, 2023

★★★★★

High quality and fast shipping.

- **Allow customers to rate and review products**

The user can rate and review the product on a section showed in the product description.



Avaliações de Clientes

Deixe sua avaliação

Classificação: ★★★★★

Comentário:

Enviar Avaliação

Avaliações dos Clientes

Utilizador anónimo publicado em Oct 20, 2023

★★★★

Warm and comfortable sweatshirt.

User publicado em Nov 5, 2023

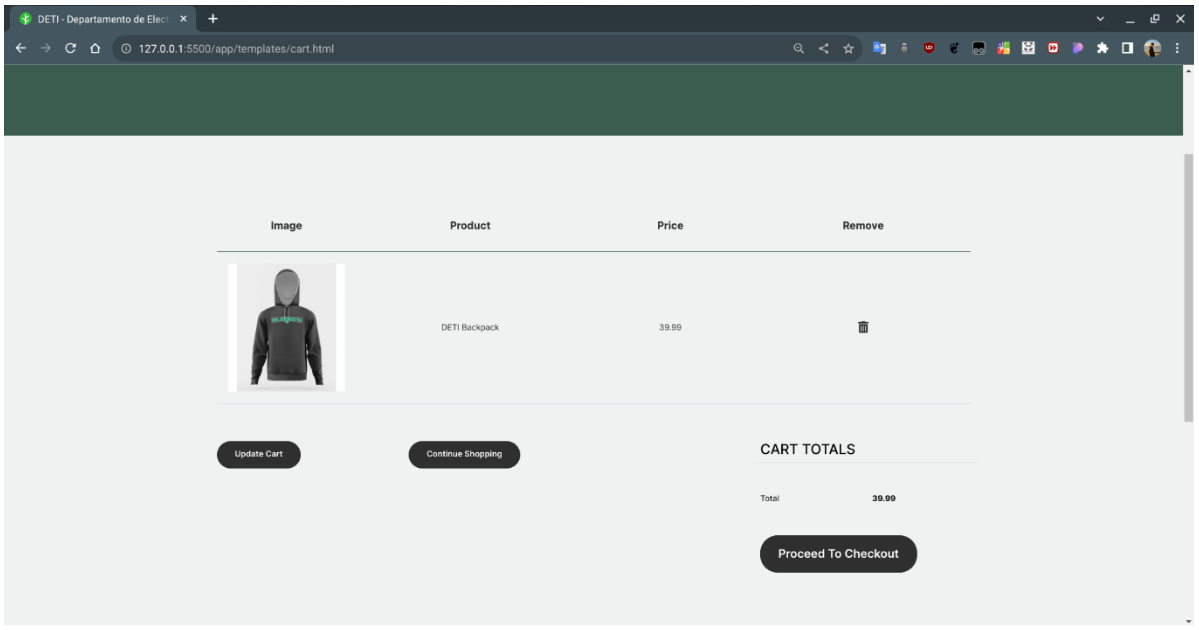
★★★★★

i love this product

- **Shopping Cart:**
  - **Cart management**

When you're ready to purchase a product, the first step is to add it to your cart. To do this, simply select a product, view more information about it, and then click the "Add to Cart" button.

After adding items to your cart, if you visit the cart page by clicking on the shopping cart icon in the navbar, you should see a page that looks something like this:



DETI - Departamento de Elect

127.0.0.1:5500/app/templates/cart.html

Image	Product	Price	Remove
	DETI Backpack	39.99	🗑

Update Cart

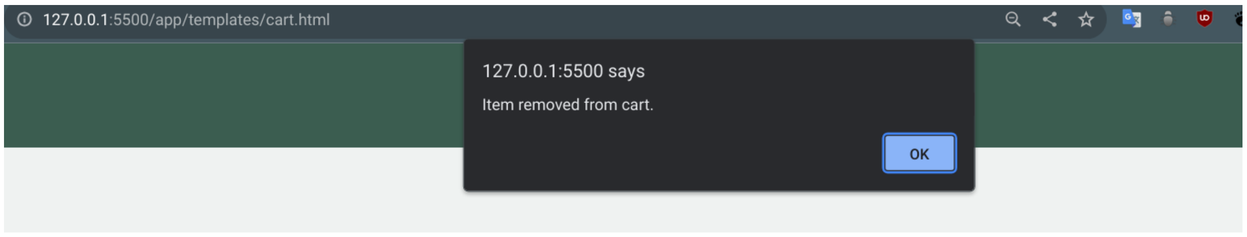
Continue Shopping

CART TOTALS

Total	39.99

Proceed To Checkout

To remove an item from your cart, simply click on the Trash icon visible on the screen. The cart will update, and you will also receive an alert in your browser confirming that the item has been successfully removed from your cart.



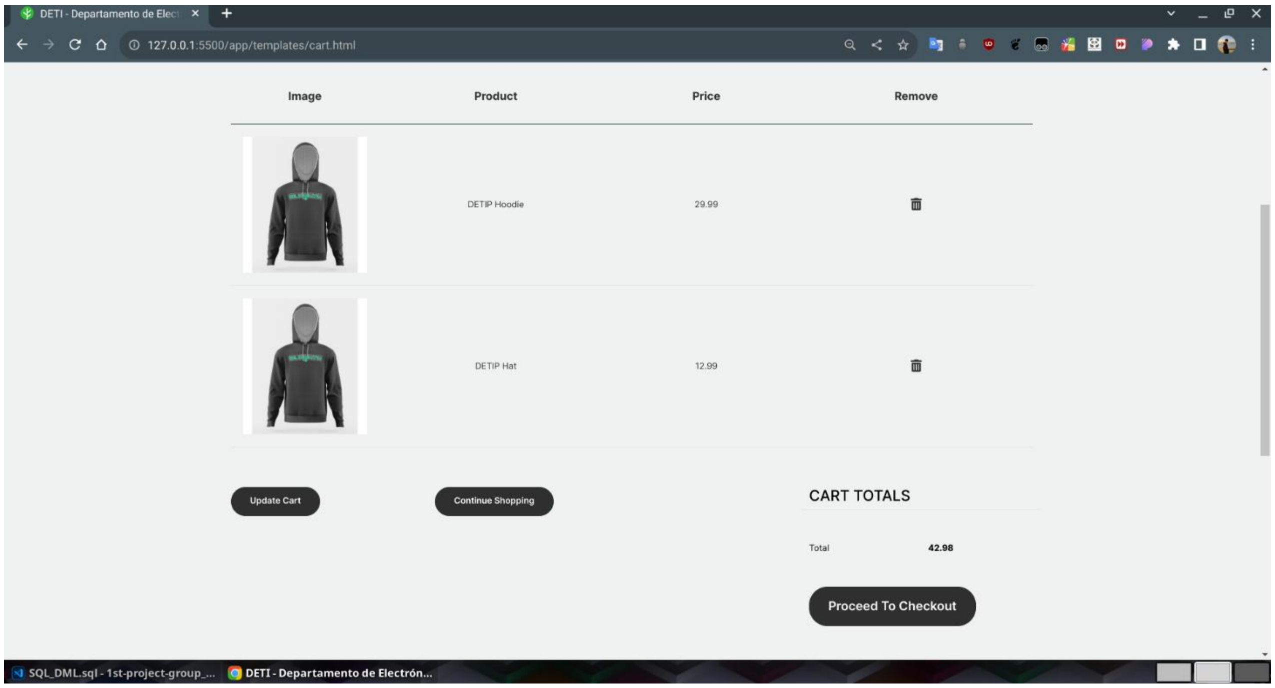
127.0.0.1:5500/app/templates/cart.html

127.0.0.1:5500 says

Item removed from cart.

- ## Cart total calculation

To view the total price of your cart when ordering more than one product, it's simple and efficient. Just add the products to your cart, and the cart will update automatically to display the total price.



127.0.0.1:5500/app/templates/cart.html

Image	Product	Price	Remove
	DETIP Hoodie	29.99	
	DETIP Hat	12.99	

CART TOTALS

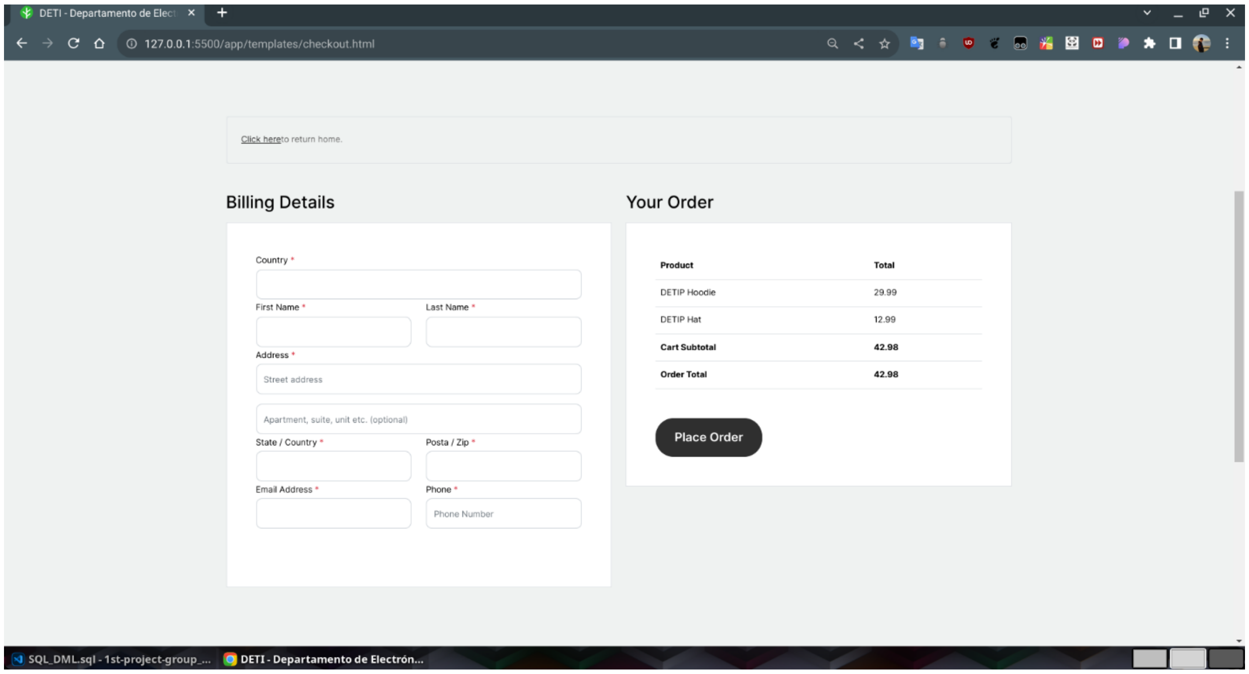
Total 42.98

- ## Save cart for Later

You can simply leave the items in your cart until you make a final decision. You can also log out and then log in again without losing the items in your cart.

- **Checkout Process:**

To check you a order is simple, when you are In the cart just hit the button “Proceed to Checkout” that will lead you to another page of checkout as this one.


DETI - Departamento de Elect

127.0.0.1:5500/app/templates/checkout.html

Click hereto return home.

Billing Details

Country *

First Name * | Last Name *

Address *

Street address

Apartment, suite, unit etc. (optional)

State / Country * | Posta / Zip *

Email Address * | Phone *

Phone Number

Your Order

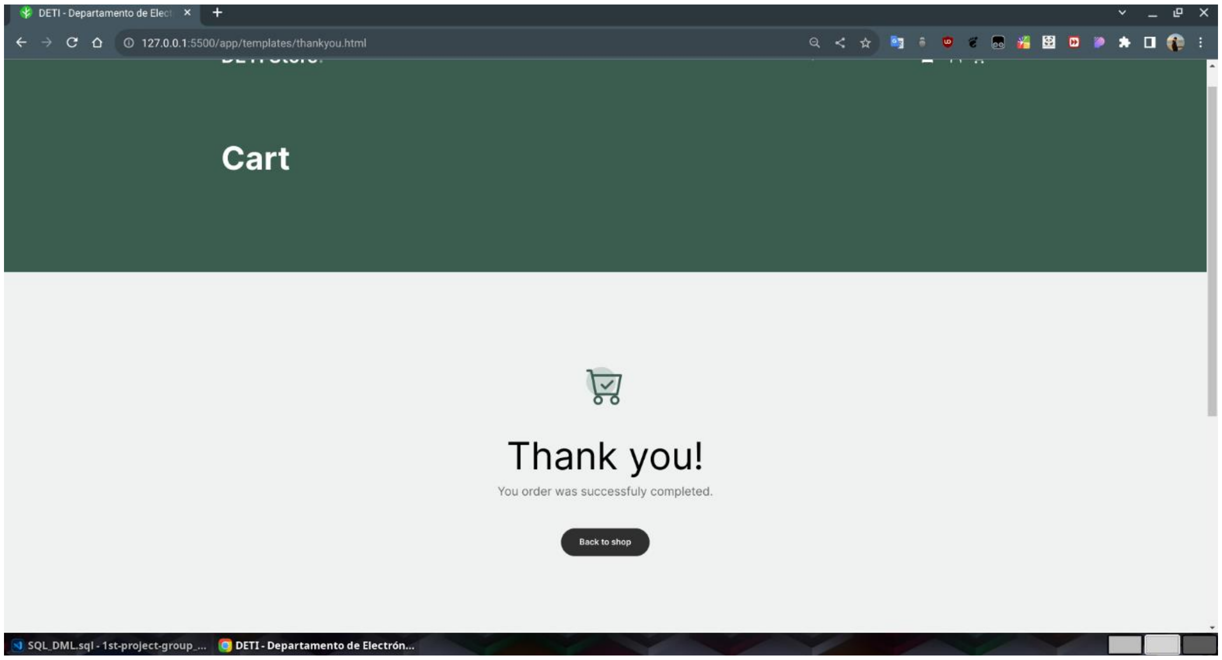
Product	Total
DETIP Hoodie	29.99
DETIP Hat	12.99
Cart Subtotal	**42.98**
Order Total	**42.98**

Place Order

SQL_DML.sql - 1st-project-group_... DETI - Departamento de Electrón...

In this section, you need to fill in the fields with your billing details, and you will have the option to review your order once more. This review will include details of the products, their individual prices, and the total price of the order.

To place your order, just click on the "Place Order" button, and you will receive the following information:

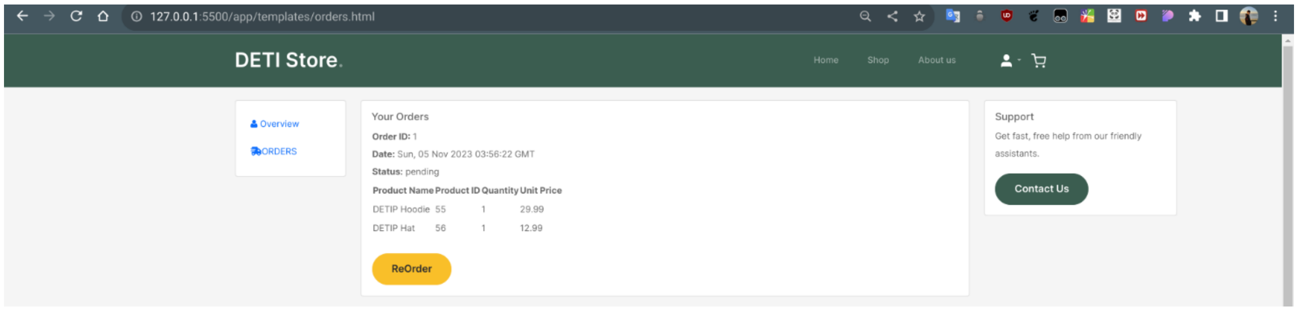


Thank you!

You order was successfuly completed.

- ## Verify past order

When you navigate through the dropdown menu by clicking on the user icon, you will find the "Orders" button. When you press it, the interface is intuitive, and you should see your order ID and the contents of your previous orders.



Your Orders

Order ID: 1

Date: Sun, 05 Nov 2023 03:56:22 GMT

Status: pending

Product Name	Product ID	Quantity	Unit Price
DETIP Hoodie	55	1	29.99
DETIP Hat	56	1	12.99

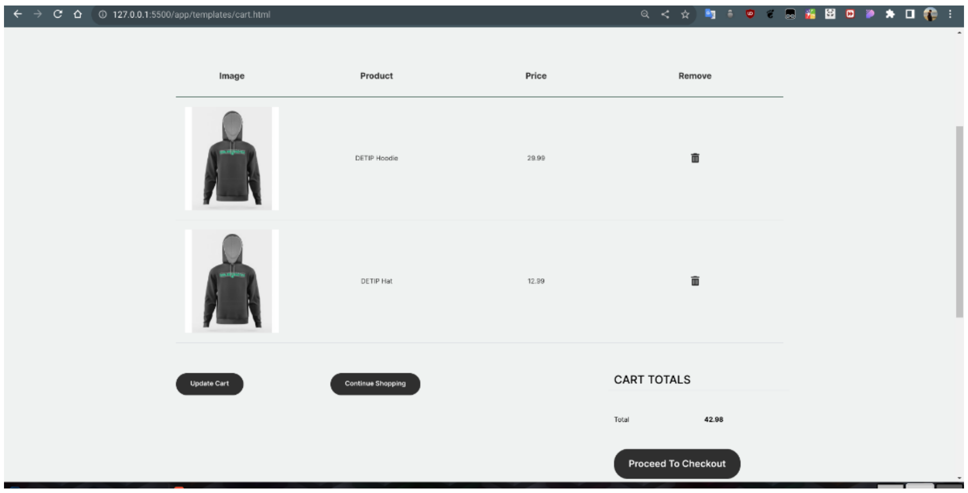
Support

Get fast, free help from our friendly assistants.

You should be able to view your order ID, the date it was placed, and detailed information about the products included in that order.

- **Reorder one past order**

To reorder the same order that you made some time ago, simply check your past orders and look for the button that says "Reorder." When you click on it, the same products will be automatically added to your cart, allowing you to complete your order once again.

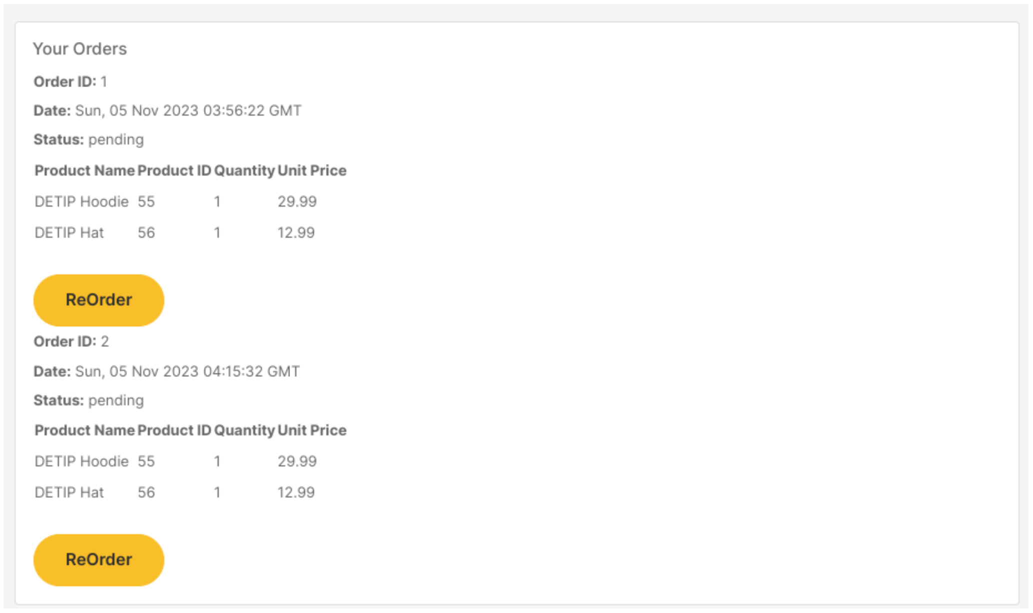


127.0.0.1:5500/app/templates/cart.html

Image	Product	Price	Remove
	DETIP Hoodie	29.99	
	DETIP Hat	12.99	

CART TOTALS

Total	42.98



Your Orders

Order ID: 1

Date: Sun, 05 Nov 2023 03:56:22 GMT

Status: pending

Product Name	Product ID	Quantity	Unit Price
DETIP Hoodie	55	1	29.99
DETIP Hat	56	1	12.99

Order ID: 2

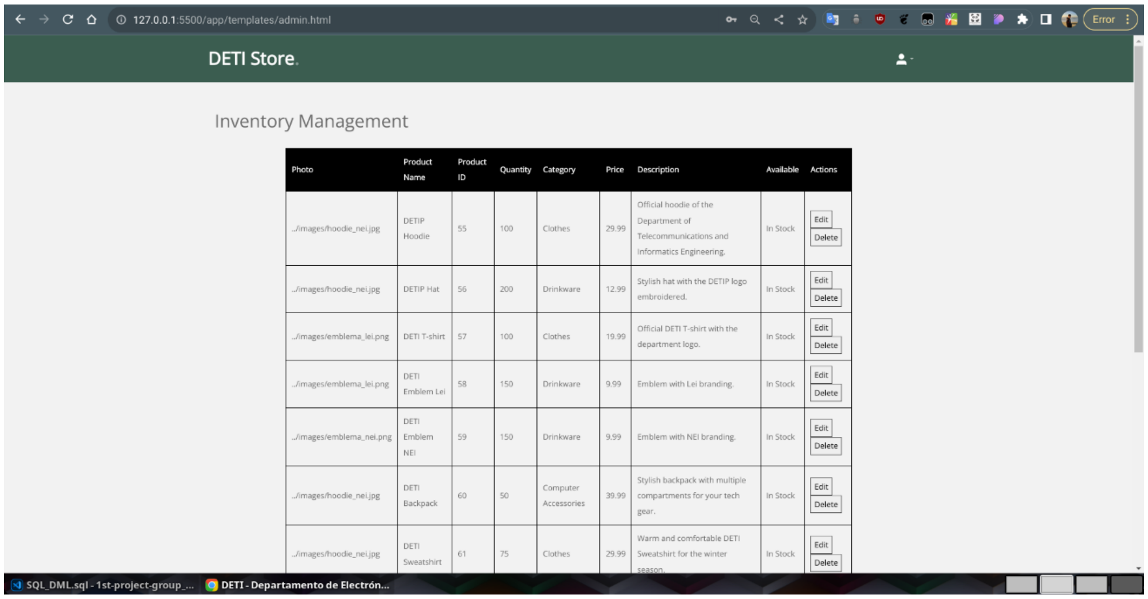
Date: Sun, 05 Nov 2023 04:15:32 GMT

Status: pending

Product Name	Product ID	Quantity	Unit Price
DETIP Hoodie	55	1	29.99
DETIP Hat	56	1	12.99

- **Tracking product availability**

  By entering in our admin page, we can observe the inventory of our company.



127.0.0.1:5500/app/templates/admin.html

DETI Store.

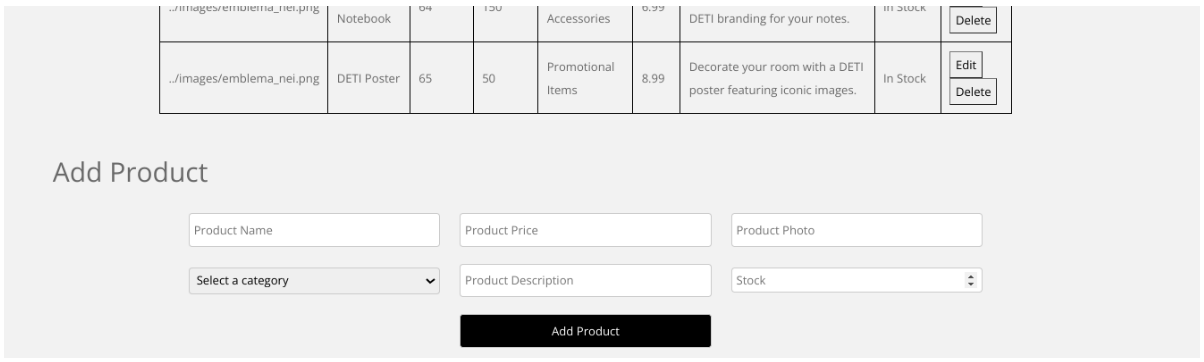
Inventory Management

Photo	Product Name	Product ID	Quantity	Category	Price	Description	Available	Actions
../images/hoodie_nei.jpg	DETIP Hoodie	55	100	Clothes	29.99	Official hoodie of the Department of Telecommunications and Informatics Engineering.	In Stock	Edit Delete
../images/hoodie_nei.jpg	DETIP Hat	56	200	Drinkware	12.99	Stylish hat with the DETIP logo embroidered.	In Stock	Edit Delete
../images/emblema_lei.png	DETI T-shirt	57	100	Clothes	19.99	Official DETI T-shirt with the department logo.	In Stock	Edit Delete
../images/emblema_lei.png	DETI Emblem Lei	58	150	Drinkware	9.99	Emblem with Lei branding.	In Stock	Edit Delete
../images/emblema_nei.png	DETI Emblem NEI	59	150	Drinkware	9.99	Emblem with NEI branding.	In Stock	Edit Delete
../images/hoodie_nei.jpg	DETI Backpack	60	50	Computer Accessories	39.99	Stylish backpack with multiple compartments for your tech gear.	In Stock	Edit Delete
../images/hoodie_nei.jpg	DETI Sweatshirt	61	75	Clothes	29.99	Warm and comfortable DETI Sweatshirt for the winter season.	In Stock	Edit Delete

SQL_DML.sql - 1st-project-group_...
DETI - Departamento de Electrón...
Error

- **Add products to our database**

  We can also add products to our shop by fill in the fields there.

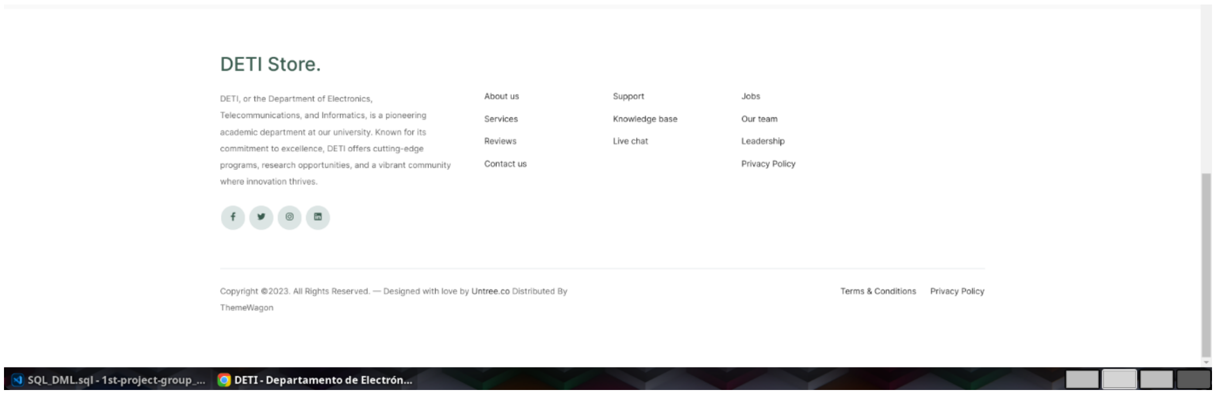


../images/emblema_nei.png	Notebook	64	150	Accessories	6.99	DETI branding for your notes.	In Stock	Delete
../images/emblema_nei.png	DETI Poster	65	50	Promotional Items	8.99	Decorate your room with a DETI poster featuring iconic images.	In Stock	Edit Delete

Add Product

- **Review website**

In our footer, you will find a section named "reviews" that you can review and see other user's reviews of the site.



DETI Store.

DETI, or the Department of Electronics, Telecommunications, and Informatics, is a pioneering academic department at our university. Known for its commitment to excellence, DETI offers cutting-edge programs, research opportunities, and a vibrant community where innovation thrives.

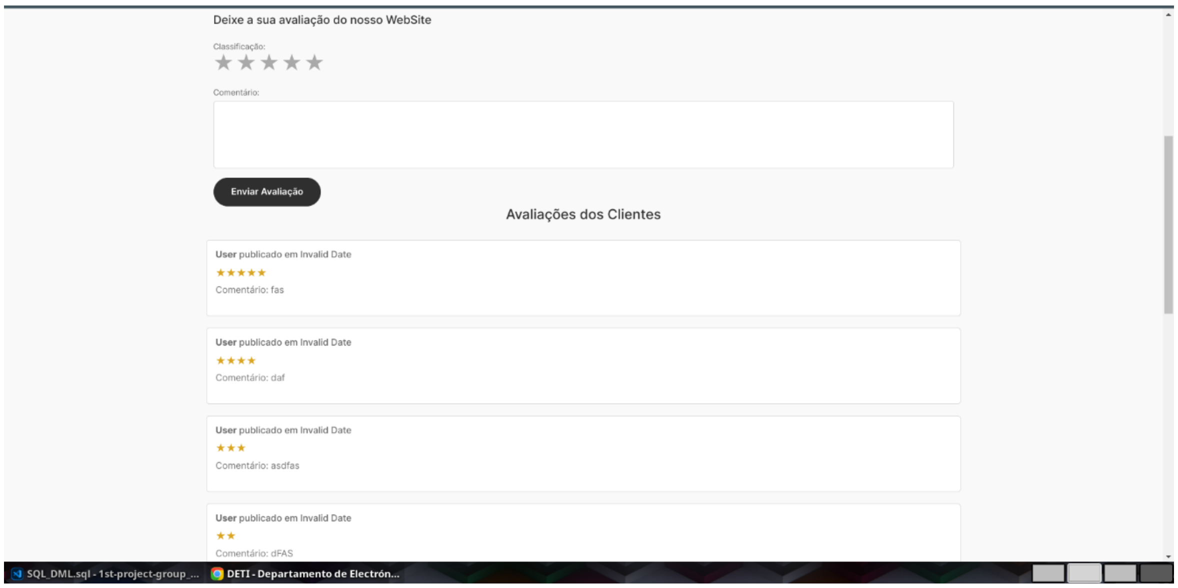
About us	Support	Jobs
Services	Knowledge base	Our team
Reviews	Live chat	Leadership
Contact us		Privacy Policy

Copyright ©2023. All Rights Reserved. — Designed with love by Untree.co Distributed By ThemeWagon

Terms & Conditions Privacy Policy

SQL_DML.sql - 1st-project-group_... DETI - Departamento de Electrón...

By going there, you will submit or see reviews.



Deixe a sua avaliação do nosso WebSite

Classificação:

Comentário:

Avaliações dos Clientes

User publicado em Invalid Date
★★★★★
Comentário: fas

User publicado em Invalid Date
★★★★
Comentário: daf

User publicado em Invalid Date
★★★
Comentário: asdfas

User publicado em Invalid Date
★★
Comentário: dFAS

SQL_DML.sql - 1st-project-group_... DETI - Departamento de Electrón...

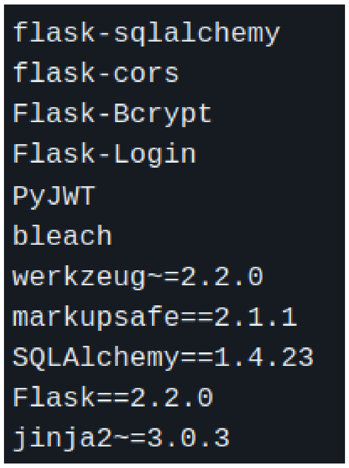
Notes

Prints

All the prints presented in this report are stored in directory analyses/prints, each one that is accoupled to a CWE is also present in their respective directory analyses/vulnerabilities/CWE-XX

Requirements

To run this project there are some requirements:



```
flask-sqlalchemy
flask-cors
Flask-Bcrypt
Flask-Login
PyJWT
bleach
werkzeug~=2.2.0
markupsafe==2.1.1
SQLAlchemy==1.4.23
Flask==2.2.0
jinja2~=3.0.3
```

Make sure you have every requirement by running this command in console at the directory "1ST-PROJECT-GROUP_13: **"pip install –r requirements"**

Test Accounts

There are different testing accounts for the vulnerable and for the secure app. The secure app requires more secure passwords, using capital letters, numbers, and special characters and a minimum number of characters

app:

User

email: user@example.com

password: pass

Admin

email: admin@example.com

password: password

app_sec:

User

email: usersec@example.com

password: Password123!

Admin

email: adminsec@example.com

password: Password123!

How to run

- **Step 1:** Install the required dependencies: **“pip install -r requirements”**

- **Step 2:** Run the backend server:
  - For **'app'**, navigate to the '/app/backend' directory and execute the following command:
    “**python3 controller.py”**
  - For **'app_sec'**, navigate to the '/app_sec/backend' directory and execute the following command:
    “**python3 controller.py**

- **Step 3:** If using '**app_sec**', provide the database password: **"password"**

- **Step 4:** Launch the HTML server using one of the following options:
  - **Option 1:** Use the Visual Studio Code (VSCode) extension 'Live Server.' Open the 'index.html' file in the 'templates' folder and click "Go Live" in the bottom right corner.

  - **Option 2:** Start a Python server in the '1ST-PROJECT-GROUP_13' folder using the command: **“python3 -m http.server 5500”**
    - Then, open the browser with the following URLs:
      - 'app': http://127.0.0.1:5500/app/templates/index.html
      - 'app_sec': http://127.0.0.1:5500/app_sec/templates/index.html

  - **Option 3:** Launch a Node server in the '1ST-PROJECT-GROUP_13' folder. First, install it using: **“npm install -g http-server"**
    - Then, run the server with the command: **“npx http-server -p 5500”**
    - Open the browser with the following URLs:
      - 'app': http://127.0.0.1:5500/app/templates/index.html
      - 'app_sec': http://127.0.0.1:5500/app_sec/templates/index.html