Reflective report

57170350 Fok Chun Hei

205CDE Project

GitHub: https://github.com/FokChunHei/205CDE

Content

Introduction

When I first started designing my website, I had a basic understanding of HTML5, CSS, and JavaScript. I knew how to create a basic webpage with HTML5 and style it with CSS. However, I wanted to take my website design to the next level, and that's when I decided to learn more advanced techniques. Like using classes and IDs to apply styles to specific HTML elements. This allows me to apply different styles to different elements and helps me easier to maintain the styles if changes need to be made. When designing CSS styles, I keep the user experience in mind. The styles should enhance the usability and accessibility of the website, rather than hindering it. I work to create effective and maintainable CSS styles that improve the overall design and usability of my web pages.

I started by learning JavaScript, which allowed me to create more dynamic and interactive web pages. I also learned how to use Python/Flask to create a server-side application that would enable me to add more functionality to my website. With Python/Flask, I could process user input, create a database, and interact with it. Flask-Login is a Flask extension that provides user session management for Flask applications. It provides a user authentication system, session management, and the ability to restrict access to certain views based on whether a user is logged in or not. The main usage of Flask-Login is to handle user authentication and session management in Flask applications. With Flask-Login, you can create user accounts, handle login and logout functionality, and restrict access to certain views based on user authentication. For example, `@app.route('/admin')`

```
def admin():
    if 'user_id' in session and session['role_id']==1:
        cur = mysql.connection.cursor()
        cur.execute('SELECT id, username FROM users where role_id=2;')
        users = cur.fetchall()
        histories={}
        for user in users:
            cur.execute('SELECT products.name,
products.price,cart_items.quantity,products.image FROM products,cart_items, users where
cart_items.hasCheckout=1 and products.id=cart_items.product_id and
users.id=cart_items.user_id and users.id=%s;',[user[0]])
            history = cur.fetchall()
            total = 0
            for his in history:
                total += his[1]*his[2]
            histories[user[1]]=(history,total)
        return render_template('admin.html',histories = histories)


   return render_template('index.html',error='You are not #authorized to access admin
page.')

@app.route('/deleteProduct/<int:product_id>')
def deleteProduct(product_id):
    cur = mysql.connection.cursor()
    cur.execute('SELECT id, name,image FROM products WHERE id = %s', [product_id])
    product = cur.fetchone()
    if product:
        try:
            cur.execute('delete FROM products WHERE id = %s', [product_id])
            mysql.connection.commit()
            try:
                os.remove(os.path.join('static',app.config['UPLOAD_FOLDER'], product[2]))
                flash('Product '+product[1]+' was deleted!')
            except:
                flash('Product '+product[1]+' image was deleted before!')

           return redirect(url_for('products'))
        except:
            cur = mysql.connection.cursor()
            cur.execute('SELECT id, name, description,price, image FROM products')
            products = cur.fetchall()
            return render_template('products.html',products=products,error='Product 
'+product[1]+' cannot be deleted since it has been bought by customers.')
    else:
        return redirect(url_for('products'), error="Already deleted.")
```

is used to access admin.html and the delete function for admin.

It is used to provide default implementations for the methods that Flask login expects user object to have. One of the benefits of using Flask-Login is that it abstracts away much of the low-level details of user session management and authentication. It provides a simple and easy-to-use interface that allows developers to add user authentication and session management to their Flask applications quickly.

Also, I have used flask blueprint to create modular and reusable components for Flask applications. With Flask Blueprint, I can define a set of routes, views, templates, and static files that can be easily integrated into my Flask application. To use Flask Blueprint, I need to first create a Blueprint object and define my routes and views. Like:

```
@app.route('/addProduct', methods=['GET','POST'])
```

```
@app.route('/about')
```

```
@app.route('/signup', methods=['GET', 'POST'])
```

Furthermore, I have used the Jinja templating language, {% endblock %}. A block is a section of a template that can be overridden by a child template. By using the {% block %} and {% endblock %} tags, you can define a block of content in a parent template that can be customized or replaced by child templates. This can improve the mobility of my coding between the conjunction of html and python.

The database was one of the most important aspects of my website design. It allowed me to store user information, such as login details, preferences, and other data. I learned how to use a database management system (DBMS) like Flask-SQLAlchemy to create, read, and update data. Flask-SQLAlchemy is a Flask extension that provides integration between Flask and SQLAlchemy. Flask-SQLAlchemy simplifies the process of my working with databases in Flask applications and allows me to work with databases in a more object-oriented way.

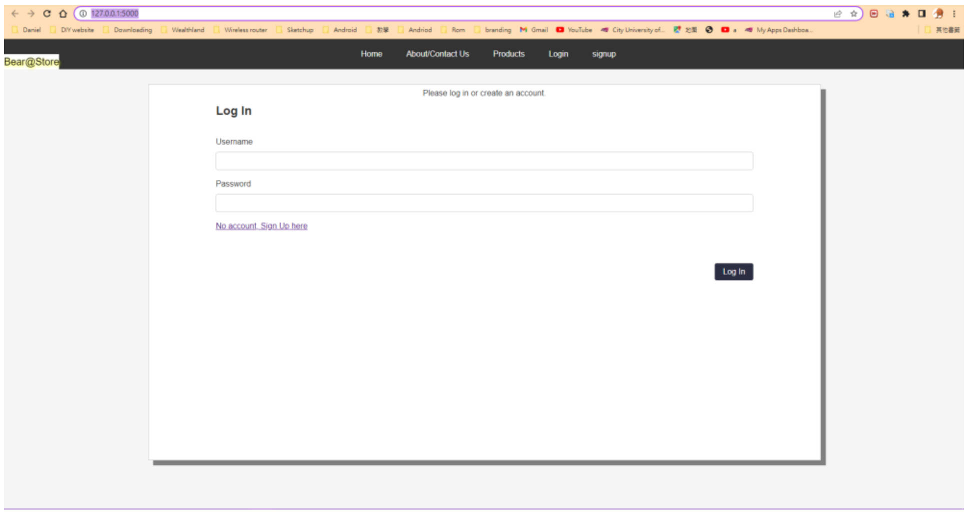
The main usage of Flask-SQLAlchemy is to define my database models, perform database operations, and interact with databases in Flask applications. With Flask-SQLAlchemy, it can define my database models as Python classes, and the extension will handle the underlying SQL queries and database operations. Also, I've used Bootstrap, a popular front-end web development framework that provides a collection of HTML, CSS, and JavaScript components for building responsive and visually appealing web applications. Its main usage is to simplify the process of designing and building responsive web applications, and it provides a set of pre-built components and styles that can be easily customized to fit the needs of a particular application. That completes my whole website style and design. Significantly reduce the amount of custom CSS and JavaScript code that needs to be written, as many common UI components and styles are already provided by the framework. By importing on the heading.

```
<link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
```

In security, I've used password hash as a function provided from the `werkzeug and Bcrypt` that is used to securely hash passwords for storage in a database. The purpose of hashing passwords is to ensure that even if an attacker gains access to a database, they cannot easily retrieve the original passwords of users.

Case Study

Login


Bear@Store

Home About/Contact Us Products Login signup

Please log in or create an account.

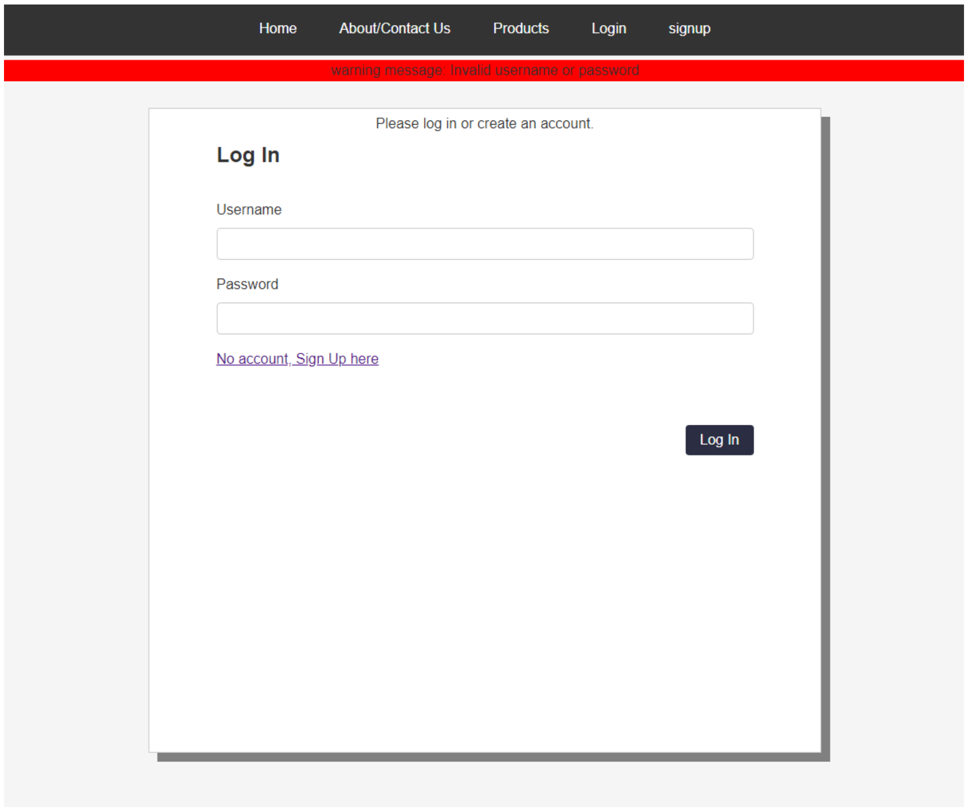
Log In

Username

Password

No account, Sign Up here

Log In


Home About/Contact Us Products Login signup

warning message: Invalid username or password

Please log in or create an account.

Log In

Username

Password

No account, Sign Up here

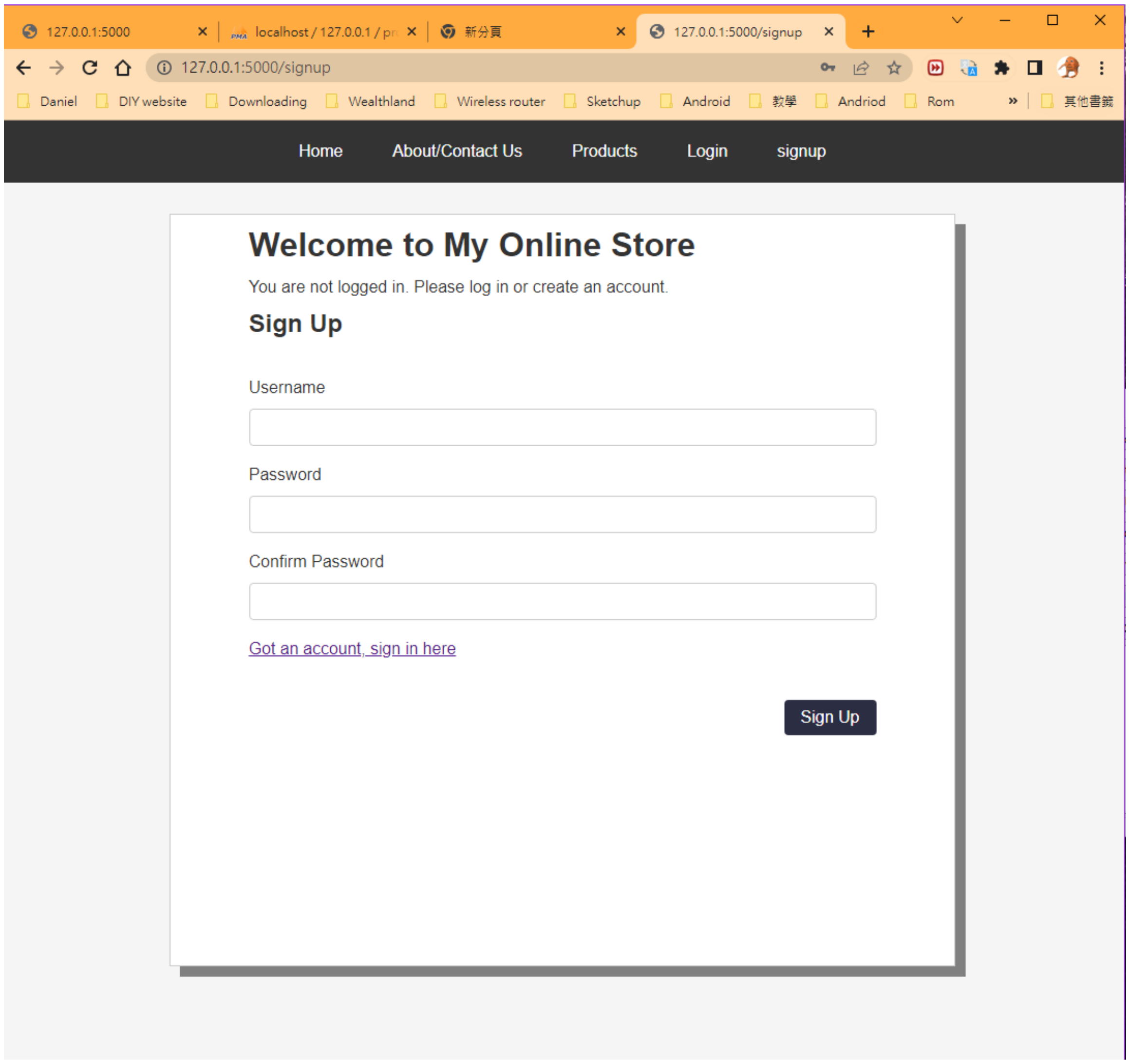
Log In

If you login with an unauthorized account, the system first searches for the account you input that didn't exist in the database. If not, it will output a bubble to show the account wasn't right.

And, if the password is incorrect. A notice will also be shown.

Sign-up

User



Welcome to My Online Store

You are not logged in. Please log in or create an account.

Sign Up

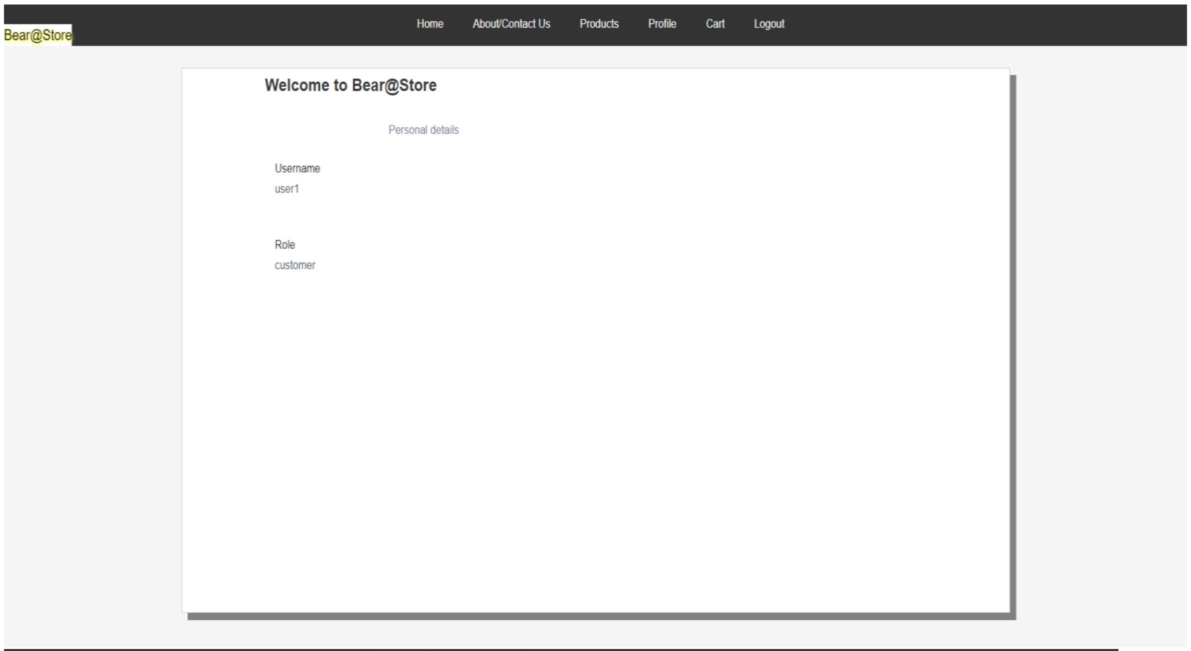
Username

Password

Confirm Password

Got an account, sign in here

User


Bear@Store

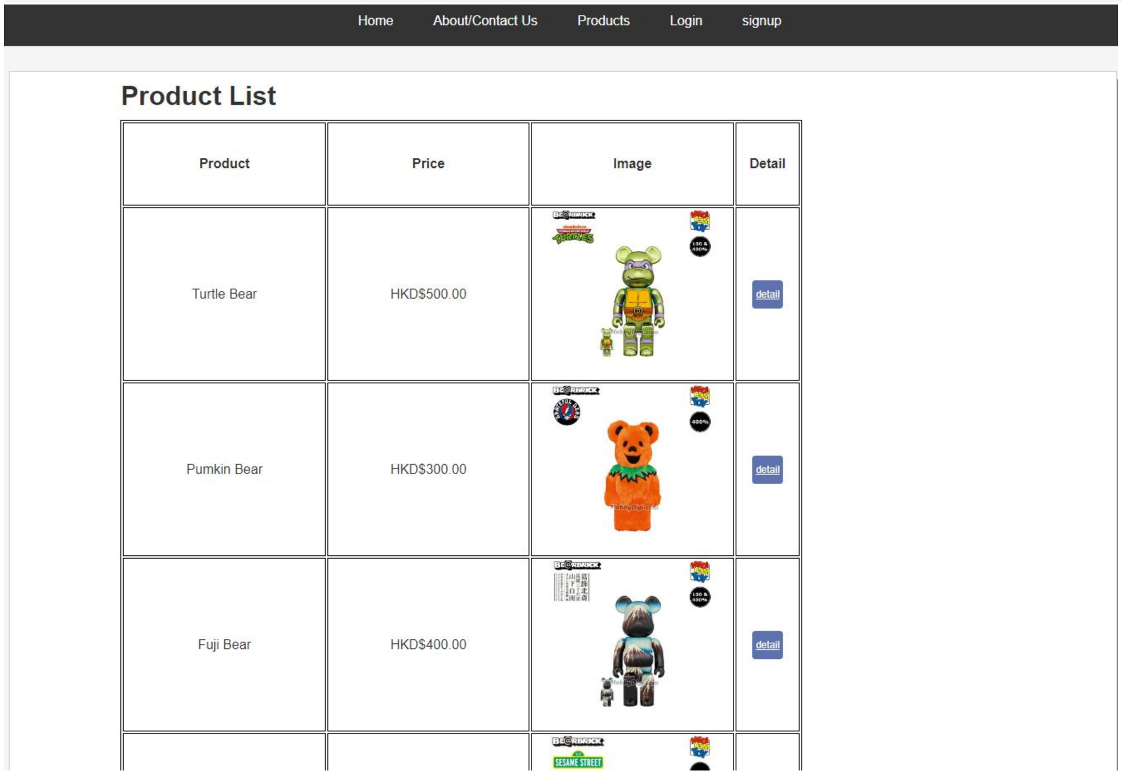
Home About/Contact Us Products Profile Cart Logout

Welcome to Bear@Store

Personal details

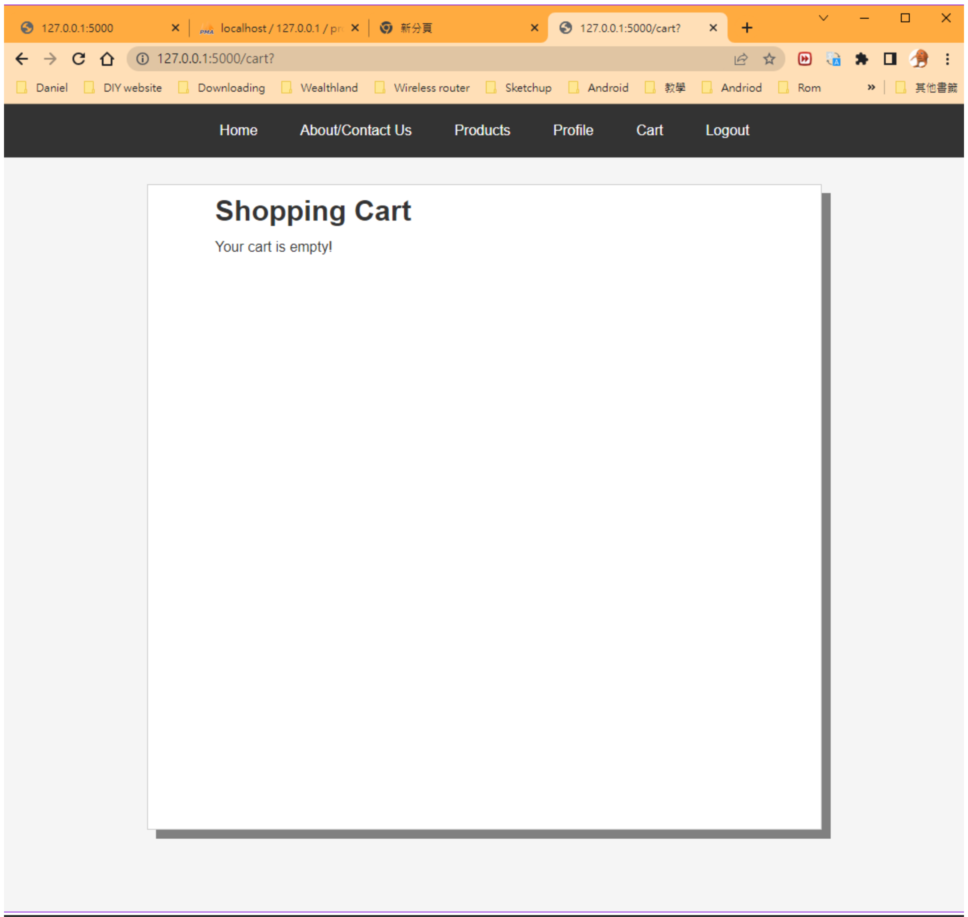
Username
user1

Role
customer


Home About/Contact Us Products Login signup

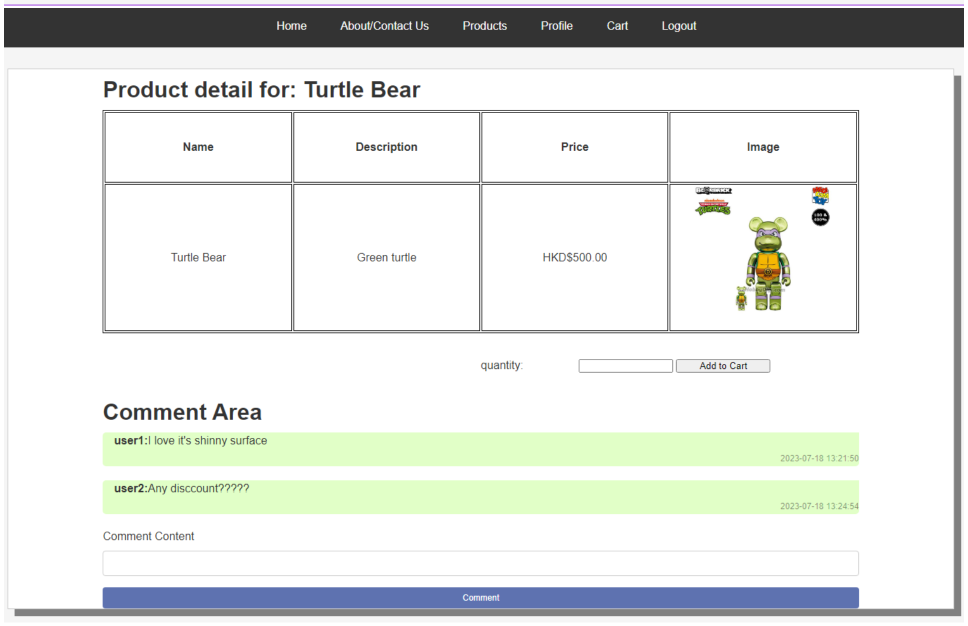
Product List

Product	Price	Image	Detail
Turtle Bear	HKD$500.00		detail
Pumkin Bear	HKD$300.00		detail
Fuji Bear	HKD$400.00		detail



Shopping Cart

Your cart is empty!



Product detail for: Turtle Bear

Name	Description	Price	Image
Turtle Bear	Green turtle	HKD$500.00	

quantity: Add to Cart

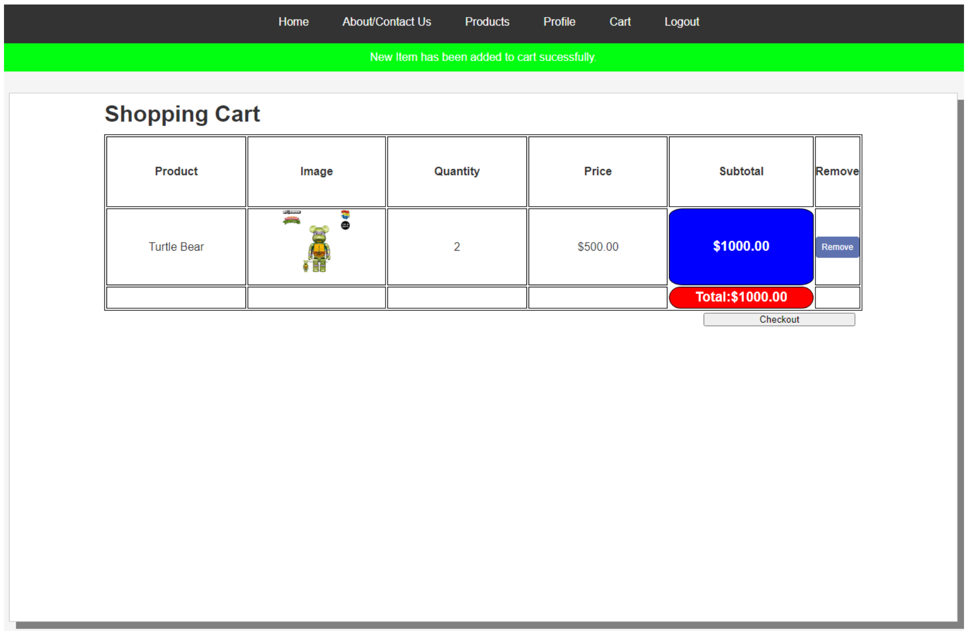
Comment Area

user1:I love it's shinny surface
2023-07-18 13:21:50

user2: Any disccount?????
2023-07-18 13:24:54

Comment Content

Comment


Home About/Contact Us Products Profile Cart Logout

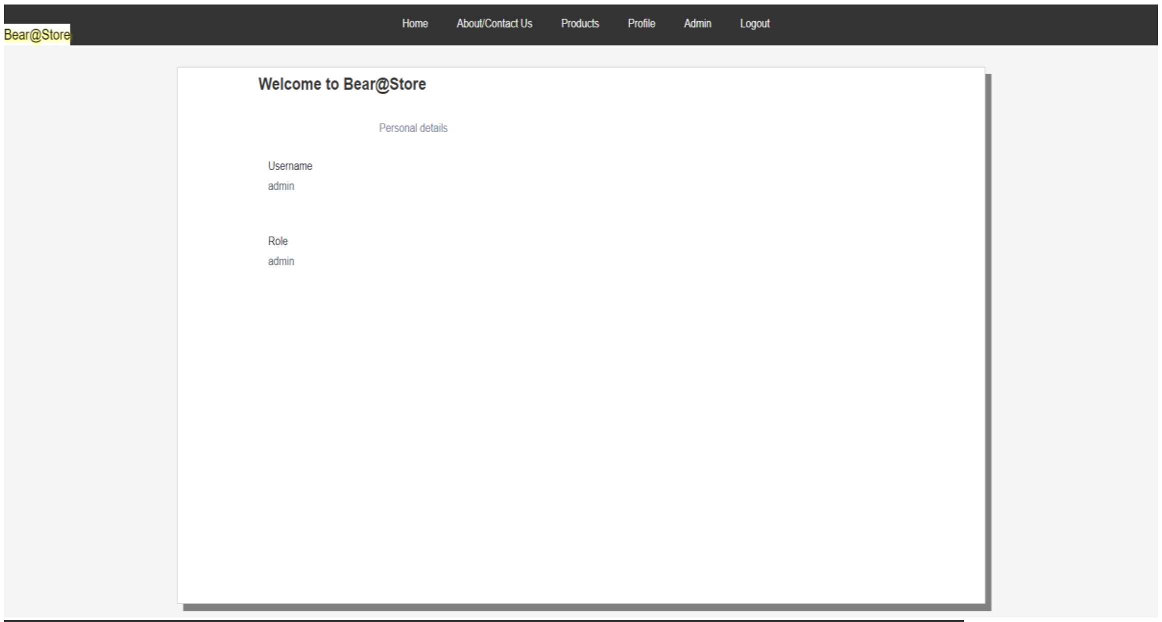
New Item has been added to cart sucessfully.

Shopping Cart

Product	Image	Quantity	Price	Subtotal	Remove
Turtle Bear		2	$500.00	$1000.00	Remove
				Total:$1000.00	

Checkout

Admin

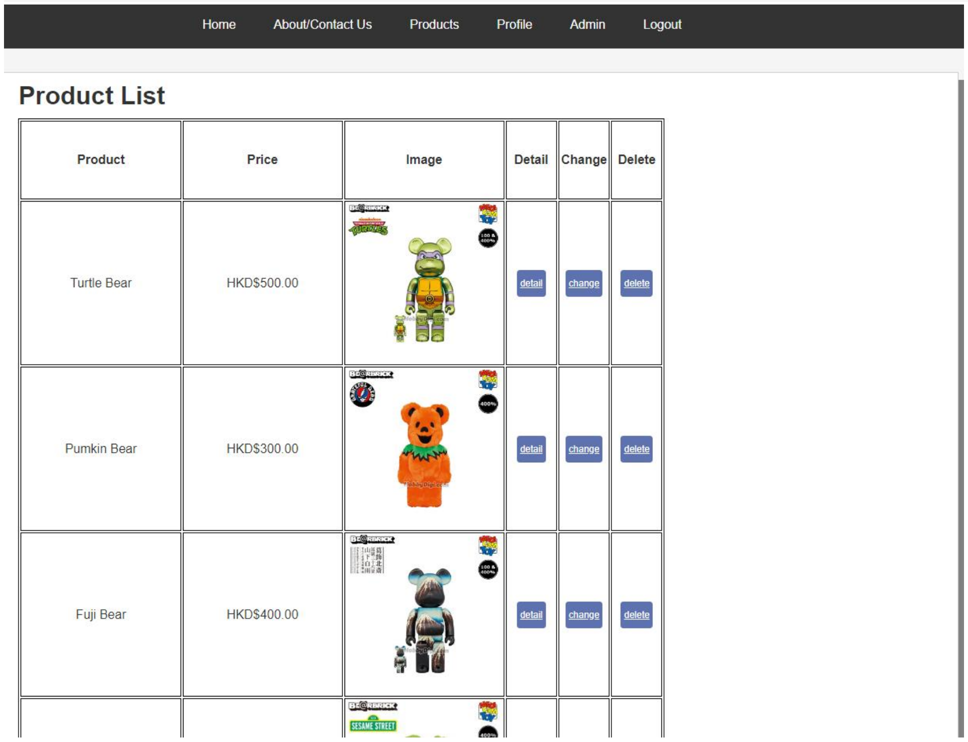


Welcome to Bear@Store

Personal details

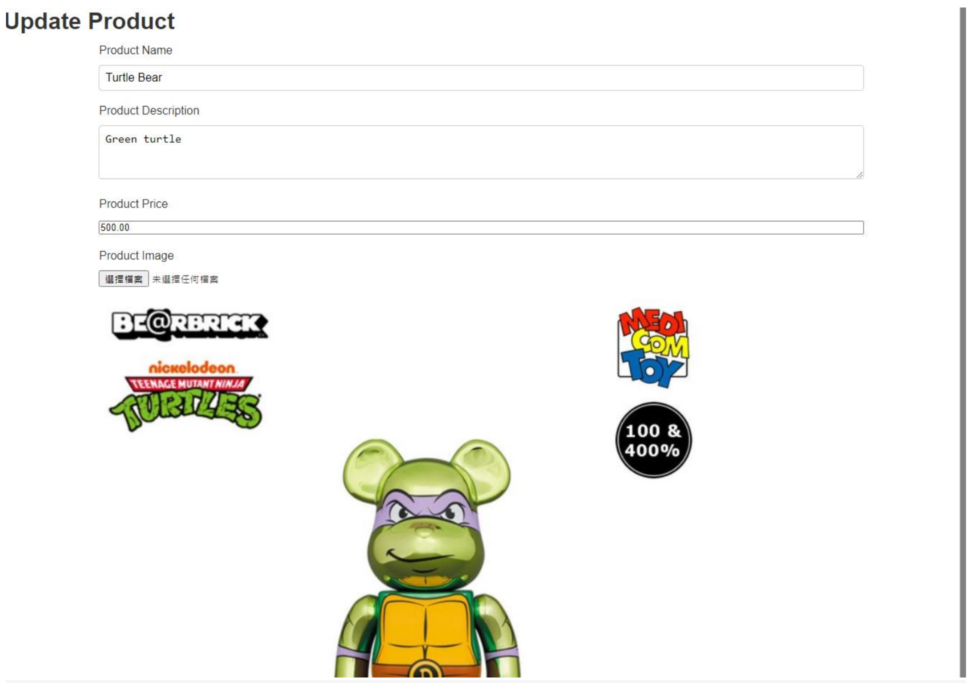
Username
admin

Role
admin



Product List

Product	Price	Image	Detail	Change	Delete
Turtle Bear	HKD$500.00		detail	change	delete
Pumkin Bear	HKD$300.00		detail	change	delete
Fuji Bear	HKD$400.00		detail	change	delete



Update Product

Product Name

Turtle Bear

Product Description

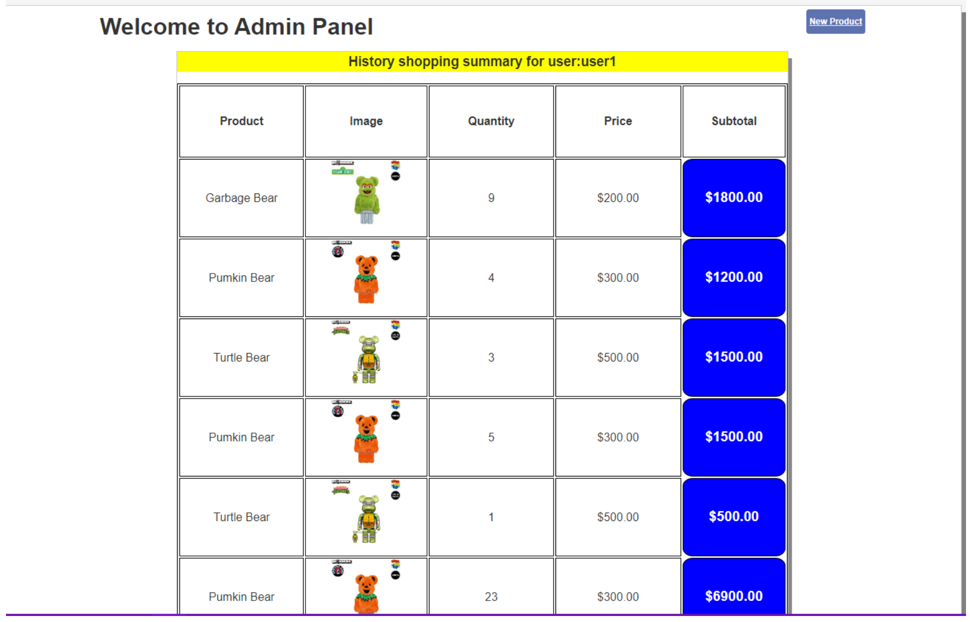
Green turtle

Product Price

500.00

Product Image

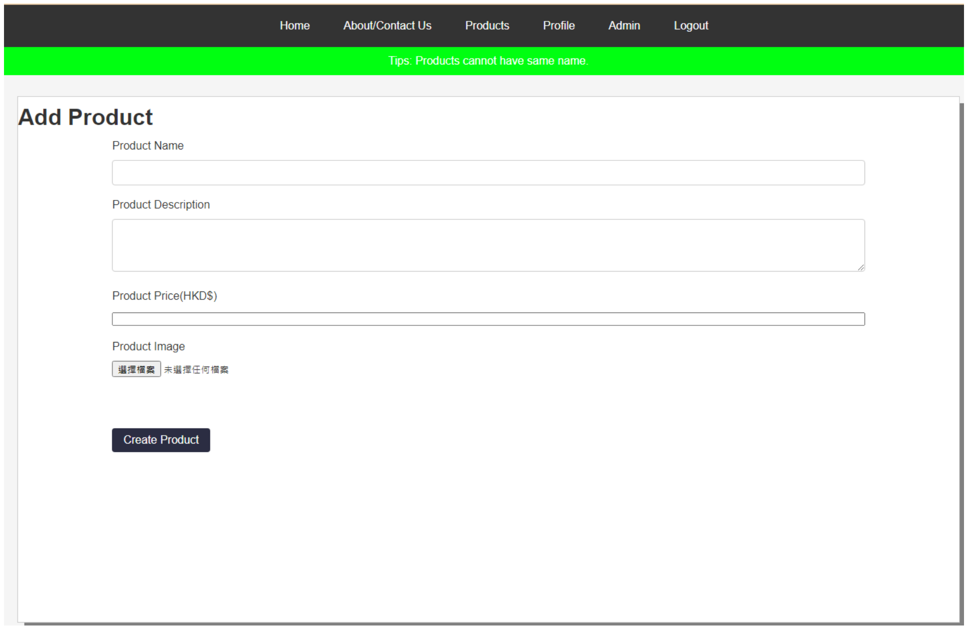
選擇檔案 未選擇任何檔案



Welcome to Admin Panel

History shopping summary for user:user1

Product	Image	Quantity	Price	Subtotal
Garbage Bear		9	$200.00	$1800.00
Pumkin Bear		4	$300.00	$1200.00
Turtle Bear		3	$500.00	$1500.00
Pumkin Bear		5	$300.00	$1500.00
Turtle Bear		1	$500.00	$500.00
Pumkin Bear		23	$300.00	$6900.00


Home About/Contact Us Products Profile Admin Logout

Tips: Products cannot have same name.

Add Product

Product Name

Product Description

Product Price(HKD$)

Product Image

選擇檔案 未選擇任何檔案

Create Product

Conclusion

As I worked on my website, I encountered several challenges. One of the most significant challenges was to ensure that my website was responsive and would work on different devices with different screen sizes. I had to learn how to use media queries in CSS to adjust the layout of my website based on the device being used to access it.

Another challenge was to ensure that my website was secure. I had to learn about web security and implement measures like password hashing, encryption, and secure communication protocols like HTTPS.

Overall, designing my website was an incredible learning experience. I learned new skills and technologies, encountered challenges, and overcame them. Through this process, I gained a deeper understanding of how websites work and how to create a user-friendly and engaging web experience.

HTML5 provided the structure for my website, CSS was used for styling, and JavaScript for dynamic and interactive functionality. Python/Flask allowed me to create a server-side application, process user input, and interact with the database. The DBMS enabled me to store, retrieve, and manipulate user data.

If I were to build a similar website in the future, there are a few things I would do differently. Firstly, I would focus more on testing and debugging. While building my website, I encountered several issues that required significant time and effort to fix. With better testing and debugging practices, I could have avoided many of these issues and saved myself a lot of time.

Secondly, I would pay more attention to user experience (UX) design. While my website was functional and had robust features, it wasn't always easy to use or navigate. In the future, I would prioritize UX design to ensure that my website is intuitive, user-friendly, and accessible to everyone.

I would explore more modern web development frameworks and libraries. While the technology stack I used was reliable and comprehensive, there are newer frameworks and libraries that offer even better functionality, performance, and ease of use. By exploring these newer technologies, I could build even better websites with less time and effort

Reference

1.https://www.youtube.com/watch?v=7S_tz1z_5bA

2.https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/

3.https://www.youtube.com/watch?v=dam0GPOAvVI

4.https://www.youtube.com/watch?v=dYulda6wEWA

5.https://ithelp.ithome.com.tw/articles/10258223