ECOMMERCE WEBSITE PROJECT

Anne Nasimiyu Makhanu

Abstract

This project is an example of an e-commerce website. The client is an electronics vendor. A user can visit the site, view available products, and add them to the cart before checkout. On checkout, they will fill a form with their details for delivery. The products will be delivered to them by a delivery company on the next day. A non-registered user can only view the products but cannot place an order. To place an order, they would have to sign up then log in. A payment section was, however, not implemented. There is also an admin section for the company staff. Here, they see all registered users and their credentials, view all orders, add and delete products, view the inventory, update the products' stock, and place reorders to manufacturers. At the moment, there is no admin login. The website manages only one store at a time and not several stores and warehouses. The about us and FAQ sections are currently inactive links.

Contents

Introduction

Online shopping has been thriving in recent years and has proved to be quite advantageous. For this reason, many companies have developed online sites to enable customers to view and purchase their products.

The objective of this project is to develop an application for an electronics vendor. This website will enable customers to buy goods online and help the staff manage products, purchases, and clients.

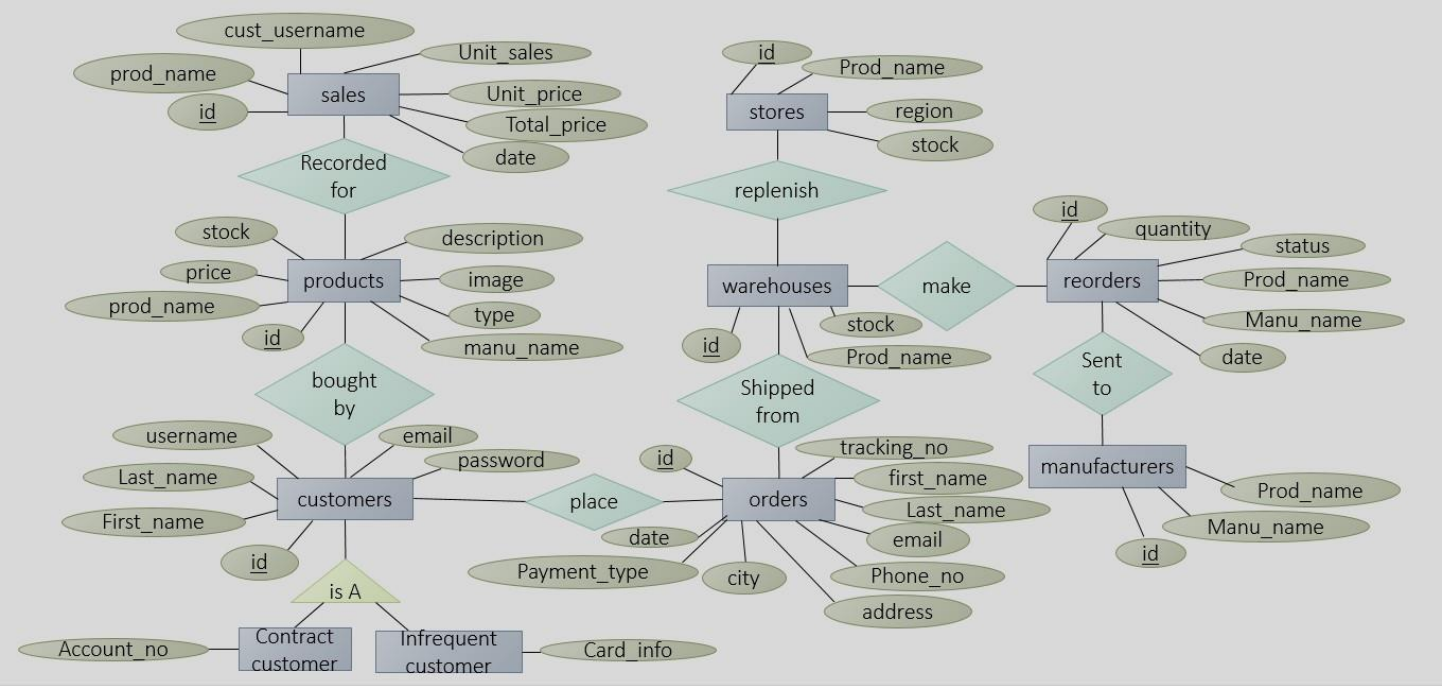
The application would be a great solution to many of the challenges the company faces in its business. The following are some of the problems the application would solve: a customer has to be physically present in their stores to view and purchase their products, keeping track of frequent customers without a means to store their information on a database is quite stressful, the stocking clerks have to manually keep track of the inventory- a very tedious process, there is an unnecessarily large amount of paperwork involved in storing customer data and inventory, keeping track of orders and deliveries is rather labor-intensive and time-consuming.

Related Work

A similar published project is the ‘Ecommerce Website’ by Jovani Samuels on GitHub. His project was a simulation of an e-commerce website for a shop that sells computers. The client also needed an application that will enable customers to view and buy products online. The goods are then shipped to them at a specified date. It was a school project under Database Management System. He used flask, python, C++, Html, CSS, and SQLite in his implementation. His approach was altogether similar to mine; in the structure of the directories and the backend section.

Methods

ER Diagram



A brief explanation:

- The sales are recorded for the products that are bought.
- Customers buy the products.
- Customers also place orders.
- The orders are shipped from Warehouses.
- Warehouses also replenish stores.
- Products that are out of stock are reordered from the manufacturers.

The primary keys have been underlined.

Relational Models

Sales

id	prod_name	Cust_username	Unit_sales	Unit_price	Total_Price	date

Products

id	prod_name	Price	Description	Image	Type	Manu_name	stock

Customers

id	First_name	Last_name	Username	Email	Password

Contract Customers

id	Username	Account_number	Billing_date

Infrequent Customers

id	Username	Card_info

Orders

id	Tracking_no	First_name	Last_name	Email	Phone_no	Address	City	Payment_type	date

Warehouses

id	Prod_name	Stock

Reorders

id	Prod_name	quantity	Manu_name	date	status

Stores

id	Store_region	Prod_name	Stock

Manufacturers

id	Manu_name	Prod_name

Models

Below are the models creating the tables used in the website's database:

```
class Customers(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    first_name = db.Column(db.String(20))
    last_name = db.Column(db.String(20))
    email = db.Column(db.String(50), unique=True)
    username = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(100))
    is_admin = db.Column(db.Boolean, default=False)

class Product(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50), unique=True)
    price = db.Column(db.Integer) #in shillings
    stock = db.Column(db.Integer)
    description = db.Column(db.String(500))
    image = db.Column(db.String(100))
    manu_name = db.Column(db.String(100))
    type = db.Column(db.String(100))
    orders = db.relationship('Order_Item', backref = 'product', lazy = True)

class Contract_Customers(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True)
    account_number = db.Column(db.String(50))
    billing_date = db.Column(db.String(500))

class Order(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    reference = db.Column(db.String(5))
    first_name = db.Column(db.String(20))
    last_name = db.Column(db.String(20))
    phone_number = db.Column(db.Integer)
    email = db.Column(db.String(50))
    address = db.Column(db.String(100))
    city = db.Column(db.String(100))
    status = db.Column(db.String(10))
    payment_type = db.Column(db.String(10))
    items = db.relationship('Order_Item', backref='order', lazy = True)
    date = db.Column(db.DateTime(timezone=True), default=func.now())
```

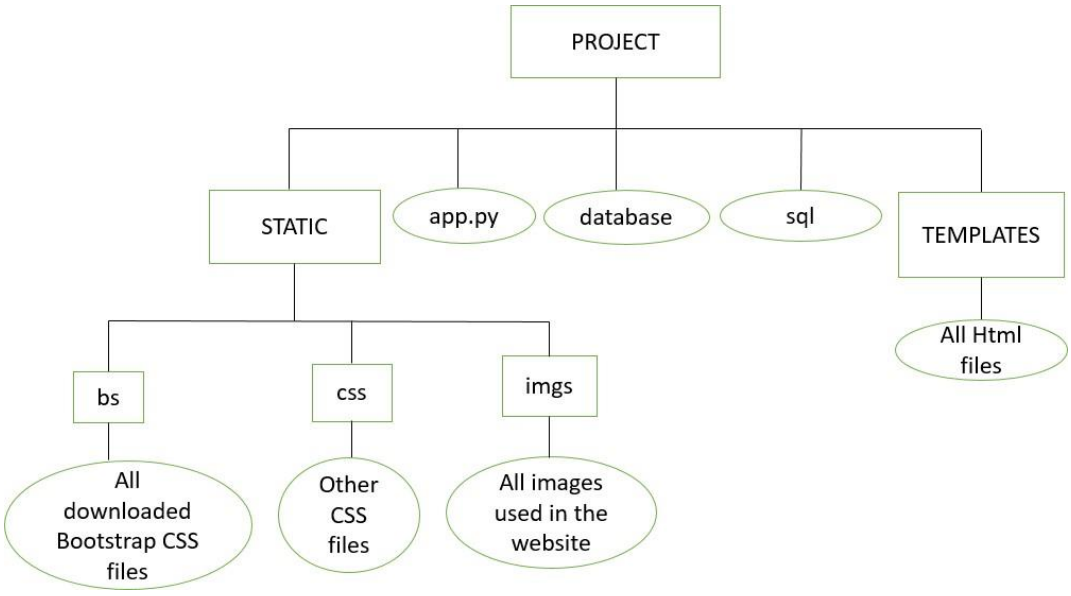
```
class Order_Item(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    order_id = db.Column(db.Integer, db.ForeignKey('order.id'))
    product_id = db.Column(db.Integer, db.ForeignKey('product.id'))
    quantity = db.Column(db.Integer)

class reorders(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    pName = db.Column(db.String(50))
    quantity = db.Column(db.Integer)
    manu_name = db.Column(db.String(50))
    date = db.Column(db.String(50))
    status = db.Column(db.String(50))
```

Tools and Platform

I used Visual Studio Code as my editor, python 3.8, flask, Html, CSS, and SQLAlchemy for this project. To host the website, I used Pythonanywhere. It is an online integrated development environment and web-hosting service. I used Werkzeug's security methods like generate_password_hash to hash passwords and check_password_hash to ensure the hashed password matches the passwords entered. Flask-Migrate is an extension that handled the database migrations using Alembic. Migrations propagate changes made in the models to the database. Models are representations of a database table in code (Mbithe, 2020).

Directory Structure



There are two folders (static and templates), a database file (projectdatabase.db), and a python file (app.py).

The static folder contains three other sub-folders: bs, CSS, and imgs. The bs contains CSS files downloaded from the Bootstrap page. The CSS folder has the styles.css file. I have referenced the tutorial from which I got some of the CSS code written by another web designer. Lastly, the imgs folder contains all the images used on the website.

The templates folder contains all the Html files.

The app.py contains the models used to create the tables. It also has the routes and functions used to run the website.

Other folders and files include the migrations folder, the sql file with the actual sql statements (these are simply for illustration since I used models to create the tables), and requirements file with the software dependencies.

Challenges Faced

A major challenge faced was limited time. Trying to make sure the project covered most of the requirements specified within a short period and learning the tools from scratch before getting started with the project was quite difficult.

To counter this problem, the lecturer invited an alumnus to teach us the basics of website development before embarking on the project. I also watched several tutorials on e-commerce projects that were using similar tools.

I created a schedule to help me complete the project in milestones. It took me one week to develop the final draft of the Entity-Relationship Diagram, Relation Models, and the database. I used another three weeks to create the frontend and get the website to work, and the last week for hosting and documentation, coming to a total of about five weeks.

This whole experience has enabled me to learn various concepts on my own within a limited time, learn to do research, and develop a sense of integrity in doing my work. I can now proceed to do similar projects even outside the school environment.

Future Work

Given more time, I would add a payment section for the customers. After adding items to the cart and checking out, they can pay via PayPal or use their debit/credit cards. A simplified way of doing this is by using Stripe, which primarily offers payment processing software and application programming interfaces for e-commerce websites and mobile applications.

Another upgrade would be to divide the admin section into various departments. The staff will only access the database according to their respective departments. I would use Flask-Admin- a Flask extension that lets you add admin interfaces to Flask applications. It will also add a login section such that not just anyone can access the admin side.

A future project I would like to be part of is, working with a team in developing an application for a client. The client is a professor coming up with an online school. The application would allow students to register and pay to access the various courses taught, a similar approach to other learning platforms like Udemy and Coursera.

Conclusion

My application partially met the project requirements. There are various interfaces for the different users who access the database. The customers can access the website and place orders. The company staff has an admin section to view customer details, customer orders, check the inventory, and make reorders to manufacturers whenever products are out of stock.

Also, two of the five specified queries could be tested on the database successfully: In the admin section, all the customer orders have a tracking number. The tracking number, in this case, is an active link which when clicked upon, leads to a page with the customer's details and content of the order. Whenever a customer makes a purchase, the inventory is updated automatically.

References

Mbithe, N. (September 21, 2020). *Build a CRUD Web App with Python and Flask.* https://www.digitalocean.com/community/tutorials/build-a-crud-web-app-with-python-and-flask-part-one

Jovani, S. (July 28, 2020). *Ecommerce-website-flask.* https://github.com/iamjovani/ecommerce-website-flask

Tea Time. (2020, Dec 6). *Store App Flask Video Course Python Framework Flask*. [Video] https://www.youtube.com/watch?v=kU74PrIjlVw&t=4550s

Flask Documentation Version 2.0.1.

Appendix

Code used to insert data into the orders table:

```
@app.route('/checkout', methods = ['GET', 'POST'])
@login_required
def checkout():
    form = Checkout()
    products,grand_total,grand_total_plus_shipping,quantity_total = handle_cart()
    if form.validate_on_submit():
        order = Order()
        form.populate_obj(order)
        order.reference = ''.join([random.choice('ABCDE') for _ in range(5)])
        order.status = 'PENDING'

        for product in products:
            order_item = Order_Item(quantity = product['quantity'],
                                    product_id= product['id'] )
            order.items.append(order_item)

        product = Product.query.filter_by(id=product['id']).update({'stock':
                Product.stock - product['quantity']})
        db.session.add(order)
        db.session.commit()

        session['cart'] =[]
        session.modified =True
        flash('Order placed successfully!', 'success')
        return redirect(url_for('cart'))
    return render_template('checkout.html',form=form,grand_total=grand_total, grand_total_plus_
shipping=grand_total_plus_shipping,quantity_total = quantity_total)
```

Brief Manual

Click on the link to visit the website. http://annemakhanu.pythonanywhere.com/

You can only checkout after logging in. After logging in, you will be redirected to the home page. If you already had a cart with products, click on the cart icon again. All the products that you added to the cart were saved. You do not need to go through the add-to-cart process again. Proceed to checkout.

To access the admin section, use the '/admin' route
http://annemakhanu.pythonanywhere.com/admin