**MATHENGE PETER MUTHUNGU**

**F17/1757/2019**

**ELECTONIKS ONLINE STORE PROJECT DOCUMENTATION**

**28/06/2021**

Table of Contents

[Abstract 3](#_Toc75986752)

[Introduction 3](#_Toc75986753)

[Methods 4](#_Toc75986754)

[ER Diagram 4](#_Toc75986755)

[Relational models 5](#_Toc75986756)

[SQL Statements 6](#_Toc75986757)

[Directory Structure 7](#_Toc75986758)

[Challenges faced 8](#_Toc75986759)

[Conclusion 8](#_Toc75986760)

[Appendix 9](#_Toc75986761)

# Table of figures

[Table1. Warehouse Inventory 6](#_Toc75987022)

[Table 2. Customers 6](#_Toc75987023)

[Table 3. Manufacturers 6](#_Toc75987024)

[Table 4. Delivery 6](#_Toc75987025)

[Table 5. Sales 6](#_Toc75987026)

# Abstract

Electroniks online store is a web based application that has a customer interface as one of the features. The customer interface has buttons displayed under each product that enable a customer to add a product of choice to cart.The cart is like a container that holds products that a user intends to purchase.

The cart has a feature that allows a user to drop a product and a button that the user can use to comfirm purchase of a product.Once the purchase is confirmed it has to go through another phase where if the customer is infrequent, they have to add an account number to proceed with purchase.After adding an account number, the application has a button that allows the customer to authorise the transaction which leads to money being deducted from their account. If a customer has a contract with the store, this phase is skipped.

If the customer authorises the transaction, the application has a feature for processing the customer’s order and preparing it for delivery where a tracking number for the delivery is automatically generated and assingned.

The application also incorporates an administrator section that has various departments involved in maintenance of the store and customer interaction.The various departments can be accessed from then admin dashboard.

# Introduction

The project is intended to eliminate the hustle customers have to go through when shopping for electronics.Electroniks online store brings the products to the customer’s fingertips and all they need is a click of a button and the electronic is brought to their doorstep.

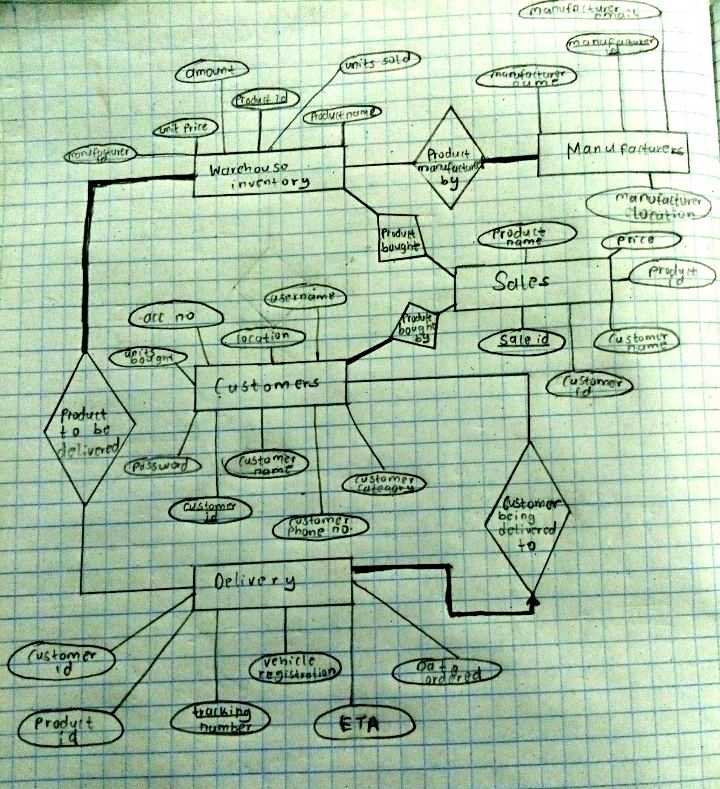
The project utilizes a database to keep a record of the customers, products, sales and deliveries made to the customer.This database is accessed by the users through interfaces where users have limited access depending on their level. The administrator have unlimited access while customers can only access the customer interface where they can place orders from. Administrator privilleges include adding new products to the inventory, updating existing products incase the products are restocked and making purchases on behalf of the customer i.e posting phone orders.

Each and every interface is interactive where buttons and forms are provided to execute commands and to connect to the database.This is achieved using a flask application which allows switching from one web page to another using view functions.

To ensure the application effectively addresses the problem stated earlier it is hosted on cloud and can be accessed at ‘http://pmathenge.pythonanywhere.com’

# Methods

## ER Diagram



Key constraints include:

* Total participation constraint

1. For one to be termed as a customer, they must have bought at least one product.
2. For a manufacturer to have a contract with the store, they must be manufacturing at least one product that the store sells
3. For a package to go to the delivery stage, it must contain atleast one product.

* Intersection of key constraint and total participation constraint

1. A particular delivery with a given tracking number will be delivered to one and only customer.

## Relational models

Table1. Warehouse Inventory

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product id | Product name | Unit price | Amount | Units sold | Manufacturer Id |
| PRIMARY KEY |  |  |  |  | FOREIGN KEY REFERENCING MANUFACTURERS |

Table 2. Customers

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Customer Id | Customer  name | Customer  category | Units  bought | Acc no | Location | Customer  Phone no. | Username | Password |
| PRIMARY  KEY |  |  |  |  |  |  |  |  |

Table 3. Manufacturers

|  |  |  |  |
| --- | --- | --- | --- |
| Manufacturer Id | Manufacturer name | Manufacturer email | Manufacturer location |
| PRIMARY KEY |  |  |  |

Table 4. Delivery

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tracking number | Vehicle registration | Date ordered | Estimated date of arrival | Customer id | Product id |
| PRIMARY KEY |  |  |  | FOREIGN KEY REFERENCING CUSTOMER | FOREIGN KEY REFERENCING  INVENTORY |

Table 5. Sales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sale id | Product id | Product name | Price | Customer id | Customer name |
| PRIMARY KEY | FOREIGN KEY REFERENCING  INVENTORY |  |  | FOREIGN KEY REFERENCING  CUSTOMER |  |

## SQL Statements

Some of the SQL statements used to modify the database include:

* INSERT INTO Customer(customer\_name,username,password)VALUES(%s,%s,%s)",(name,username,password)

Used to register new customers.

* SELECT USERNAME,PASSWORD FROM Customer WHERE username=%s",[username]

Used to select customer credentials from the database during login.

* SELECT \* FROM Inventory

Used when displaying products to the customer for purchase.

* SELECT amount FROM Inventory WHERE product\_id=%s",[*id*]

Used to determine the amount of a product remaining giving information on whether the product is out of stock or not.

* SELECT acc\_no FROM Customer  WHERE username=%s",[session['username']]

Used to determine whether a customer has registered their account which is required during a transaction.

* "UPDATE Inventory SET amount=amount-1,units\_sold=units\_sold+1 WHERE product\_id=%s",[session['id1']]

"UPDATE Customer SET units\_bought=units\_bought+1 WHERE username=%s",[session['username']]

Are the opeations that take place in the backgroud when a customer makes a purchase.

* SELECT customer\_category FROM Customer WHERE username=%s",[session['username']]

Used to determine the customer category whether they are frequent and contracted or infrequent and have to pay every time they make a purchase.

* SELECT product\_id,product\_name,product\_price from Sales WHERE customer\_id=%s",[result['customer\_id']]

Used to pick the products placed in cart by the customer.

* DELETE FROM Sales WHERE product\_id=%s",[*id*]

Used to drop a product from the cart.

* INSERT INTO Delivery (Date\_ordered,E\_T\_A,product\_id,customer\_id)VALUES(%s,%s,%s,%s)',[[date\_ordered],[e\_t\_a],session['id1'],result['customer\_id']]

Used to add data to the delivery table which is then displayed giving delivery details for a product ordered by the customer

* SELECT username,password FROM Admin WHERE username=%s",[admin\_username]

Used to fetch administrator credentials from the database during admin login.

* INSERT INTO Inventory (product\_name,unit\_price,amount,manufacturer\_id)VALUES(%s,%s,%s,%s)",(product\_name,unit\_price,amount,manufacturer\_id)

Used to add new products into the inventory.

* UPDATE Inventory SET product\_name=%s,unit\_price=%s,amount=%s WHERE product\_name=%s",(product\_name,unit\_price,amount,product\_name)

Used to update existing products attributes such as unit price and amount.

## 

## Directory Structure

Electroniks store app

├── static/

│ └── Images

│

├── templates/

│ ├── HTML files

│

├── app.py

* Static folder: Contains images that are used as background images, logos and pictures of the products in the application.
* Template folder: Contains HTML files which tell the browser what to display when a certain view function is returned.
* App.py: Is the actual application which is a flask application written in python and deployed on the cloud via pythonanywhere free web hosting platform.

# Challenges faced

Given that I had to do the project from home and that it was research intensive, I had a lot of difficulties accessing the internet which the one powerful resource that I had to use throughtout.I had to look for alternate internet providers

The other challenge I faced was I did not have any background knowledge in any of the tools used i.e python, HTML and CSS, flask and python. I had to learn them from scratch and since I was in session, I also had to learn them concurrrently with normal classes.

Finally, power outages at home occassionally left me without a power source to charge my laptop which is crucial for project implementation.In such cases, I had to look for other power sources.

Despite the challenges, I have learnt alot of skills such as designing a website, styling the website, creating a flask application, connecting the application to a database,research and working under pressure.

# Conclusion

Development of the project from designing to implementation has helped me apply my engineering knowledge in real life which for me is the first step in my engineering career.

It will a long way into boosting my confidence in trying to design other systems and applications in future.

# 

# Appendix

Some of the features of my application code i would like to highlight include:

* Adding items to cart:
* @app.route('/add\_to\_cart/<int:id>,<string:name>,<int:price>')
* @is\_logged\_in
* *def* add\_to\_cart(*id*,*name*,*price*):
* session['id']=*id*
* session['name']=*name*
* session['price']=*price*
* cur=mysql.connection.cursor()
* cur.execute("SELECT amount FROM Inventory WHERE product\_id=%s",[*id*])
* result=cur.fetchone()
* if result['amount']>0:
* cur.execute("SELECT \* FROM Inventory")
* mysql.connection.commit()
* products=cur.fetchall()
* cur.execute("SELECT customer\_id,customer\_name FROM Customer WHERE username=%s",[session['username']])
* result1=cur.fetchone()
* cur.execute("INSERT INTO Sales (customer\_id,customer\_name,product\_id,product\_name,product\_price)VALUES(%s,%s,%s,%s,%s)",[result1['customer\_id'],result1['customer\_name'],session['id'],session['name'],session['price']])
* mysql.connection.commit()
* cur.close()
* msg='Product added to cart successfully'
* return render\_template('products.html',*msg*=msg,*products*=products)
* else:
* error='This product is out of stock'
* return render\_template('product\_outofstock.html',*error*=error)
* Updating an existing product:
* @app.route('/update\_inventory',*methods*=['GET','POST'])
* @is\_admin\_logged\_in
* *def* update\_inventory():
* form=inventory\_update\_form(request.form)
* if request.method=='POST':
* product\_name=form.product\_name.data
* unit\_price=form.unit\_price.data
* amount=form.amount.data
* cur=mysql.connection.cursor()
* result=cur.execute("SELECT product\_name FROM Inventory WHERE product\_name=%s ",(product\_name,))
* if result>0:
* cur.execute("UPDATE Inventory SET product\_name=%s,unit\_price=%s,amount=%s WHERE product\_name=%s",(product\_name,unit\_price,amount,product\_name))
* mysql.connection.commit()
* msg="Product update successfull!!"
* return render\_template('update\_inventory.html',*form*=form,*msg*=msg)
* else:
* error="The product does not exist, please enter the correct product name"
* return render\_template('update\_inventory.html',*error*=error,*form*=form)
* return render\_template('update\_inventory.html',*form*=form)