ARHA-CONNECTING ARTISANS
AN E-COMMERCE PLATFORM

PHASE- 4: DEVELOPMENT PART 2

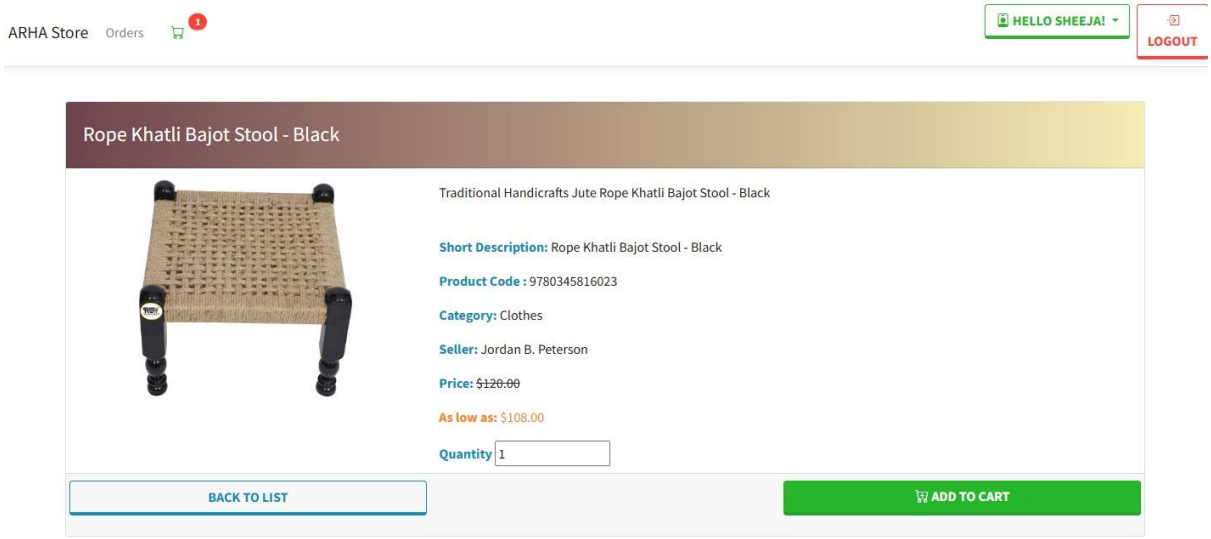
In this phase we are continuing to build our E-Commerce platform ARHA. In the previous development phase we have done few of the frontend features logo of the app, login page, register page, Home page, product page.

Now let us see about individual product page, shopping cart, orders list ,order summary, managing your account. We have also shown the flask program and the database which are the backend of our app.

INDIVIDUAL PRODUCT PAGE:

The individual product page of our app has a short description about the product, product code, the seller details, price, and the quantity the user needs. By default it is 1.The user can add the product to the cart by clicking the add to cart button in the individual products page itself.

The individual page looks something like this:

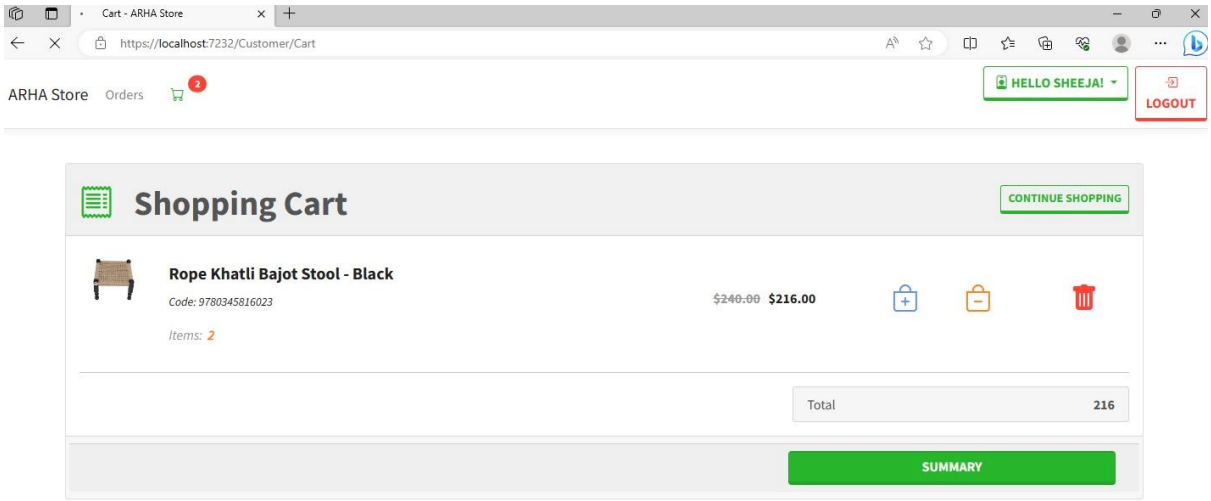


The html CSS coding for the individual product page is:

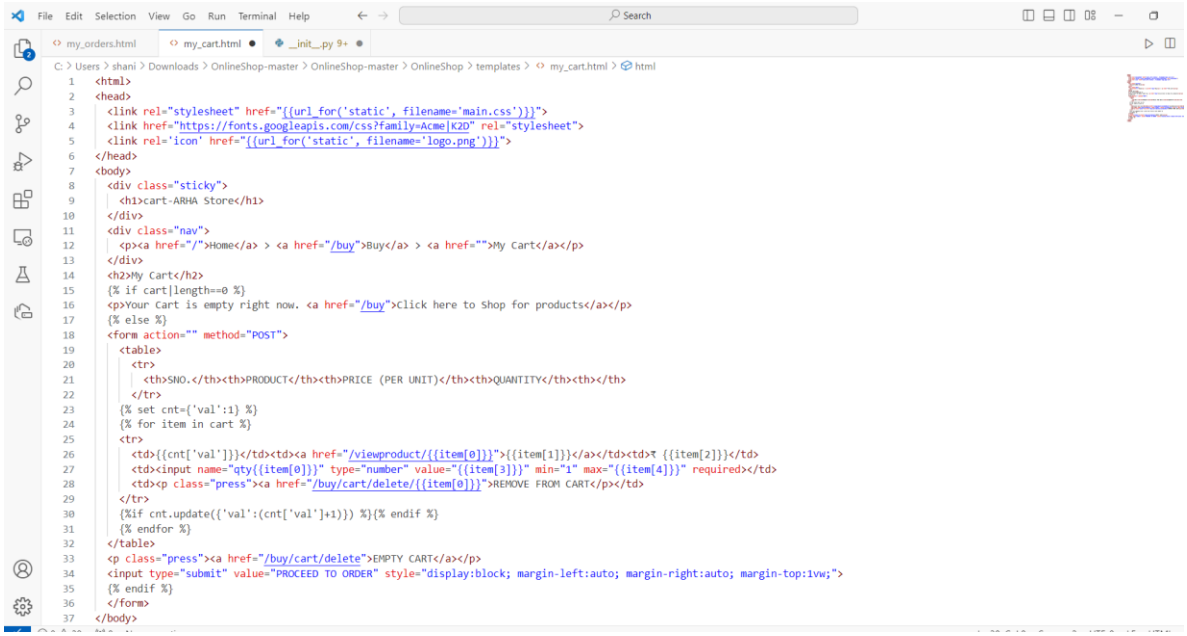
```
@{
    ViewData["Title"] = "Index";
#pragma warning disable
}
<div class="container p-3">
    <div class="row pt-4">
        <div class="col-6">
            <h2 class="text-primary">Products</h2>
        </div>
        <div class="col-6 text-end">
            <a class="btn btn-primary" asp-controller="Product" asp-action="Upsert">
                <i class="bi bi-plus-circle"></i>   New
            </a>
        </div>
    </div>
    <br>
    <table id="products" class="table table-striped table-hover stripe w-100">
        <thead>
            <tr>
                <th scope="col">
                    @Html.DisplayName("Title")
                </th>
                <th scope="col">
                    @Html.DisplayName("Code")
                </th>
                <th scope="col">
                    @Html.DisplayName("Seller")
                </th>
                <th scope="col">
                    @Html.DisplayName("Category")
                </th>
                <th scope="col">
                    @Html.DisplayName("Price")
                </th>
                <th scope="col"></th>
            </tr>
        </thead>
    </table>
</div>

@section scripts
```

SHOPPING CART :

The shopping cart enables the user to see all the product that he/she has added to the cart. The shopping cart has the name, price, quantity, total bill amount. The user also has an option to remove the item from the cart .



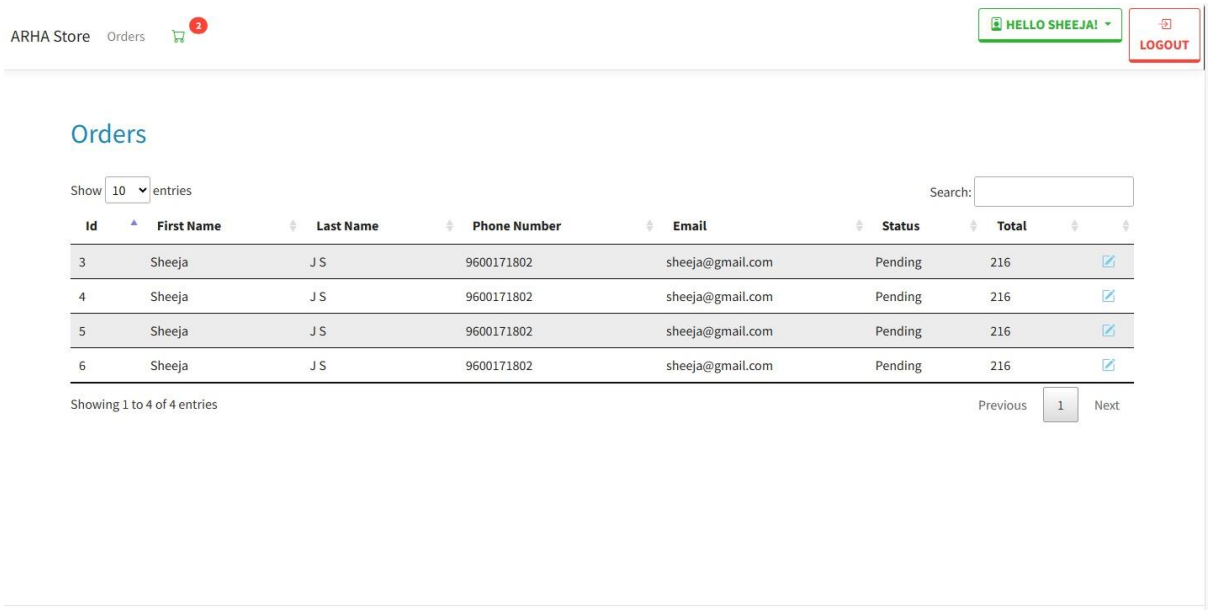
The html CSS coding that we used for this is :



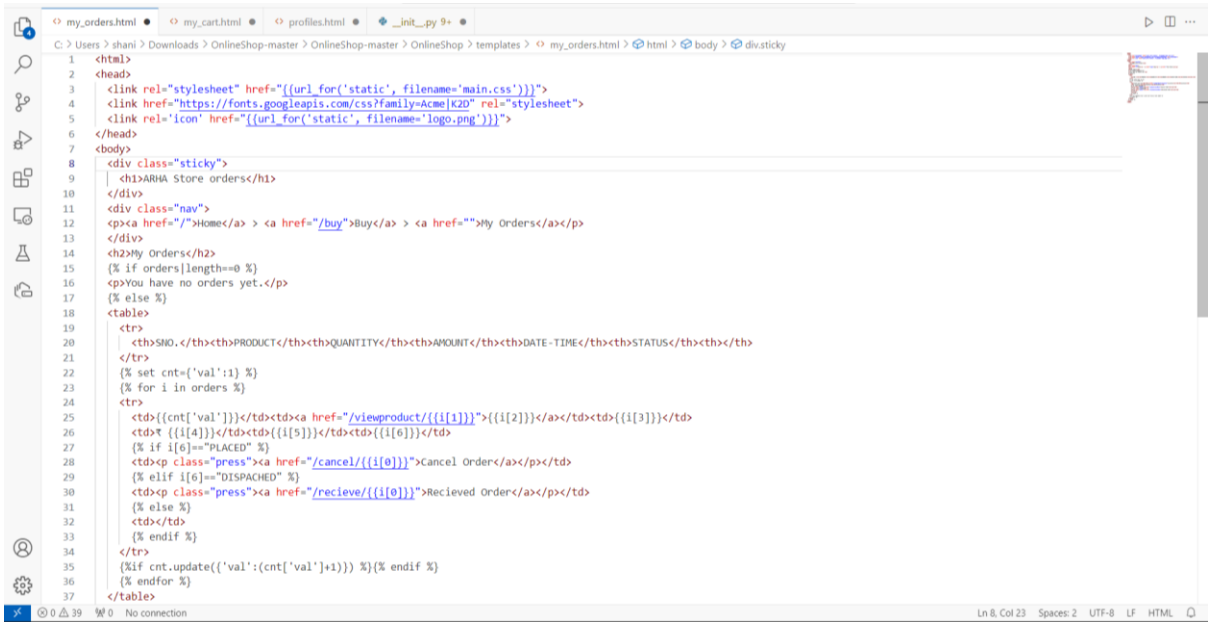
```
<html>
<head>
  <link rel="stylesheet" href="{{url_for('static', filename='main.css')}}">
  <link href="https://fonts.googleapis.com/css?family=Acme|K2D" rel="stylesheet">
  <link rel='icon' href="{{url_for('static', filename='logo.png')}}">
</head>
<body>
  <div class="sticky">
    <h1>cart-ARHA Store</h1>
  </div>
  <div class="nav">
    <p><a href="/">Home</a> > <a href="/buy">Buy</a> > <a href="">My Cart</a></p>
  </div>
  <h2>My Cart</h2>
  {% if cart|length==0 %}
  <p>Your Cart is empty right now. <a href="/buy">Click here to Shop for products</a></p>
  {% else %}
  <form action="" method="POST">
    <table>
      <tr>
        <th>SNO.</th><th>PRODUCT</th><th>PRICE (PER UNIT)</th><th>QUANTITY</th><th></th>
      </tr>
    {% set cnt={'val':1} %}
    {% for item in cart %}
    <tr>
      <td>{{cnt['val']}}</td><td><a href="/viewproduct/{{item[0]}}">{{item[1]}}</a></td><td>₹ {{item[2]}}</td>
      <td><input name="qty{{item[0]}}" type="number" value="{{item[3]}}" min="1" max="{{item[4]}}" required></td>
      <td><p class="press"><a href="/buy/cart/delete/{{item[0]}}">REMOVE FROM CART</p></td>
    </tr>
    {%if cnt.update({'val':(cnt['val']+1)}) %}{% endif %}
    {% endfor %}
  </table>
  <p class="press"><a href="/buy/cart/delete">EMPTY CART</a></p>
  <input type="submit" value="PROCEED TO ORDER" style="display:block; margin-left:auto; margin-right:auto; margin-top:1vw;">
  {% endif %}
  </form>
</body>
```

ORDERS LIST:

The user can also view all the previous or present orders he/she has made. The page looks something like this.



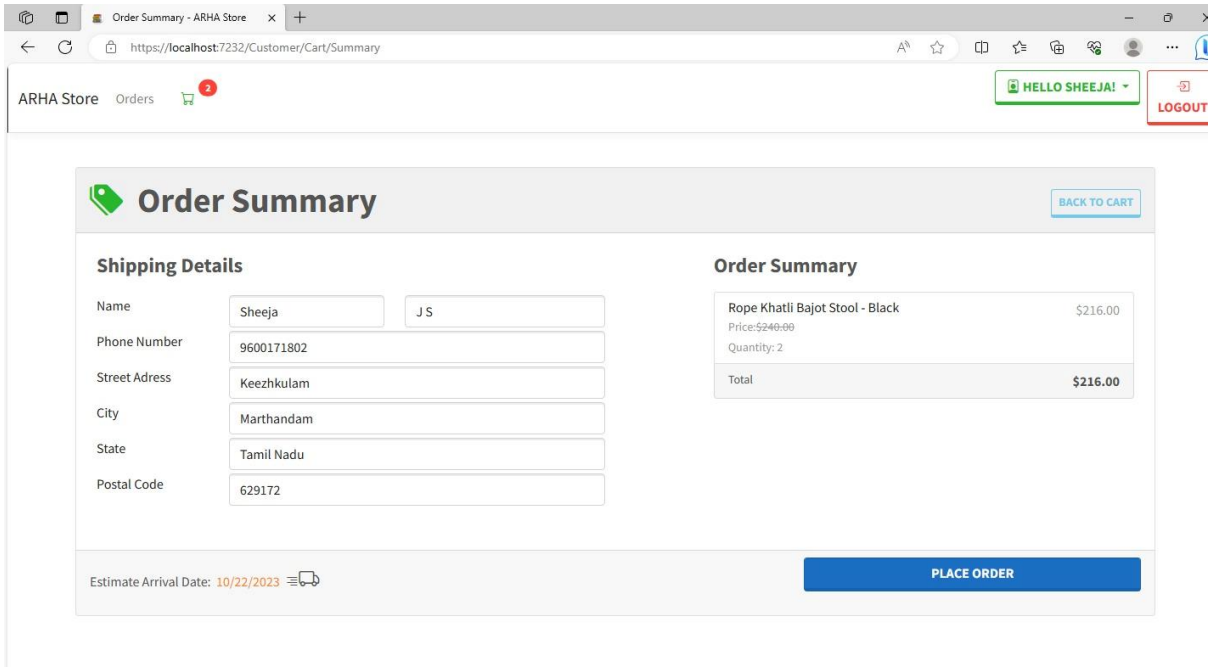
The html CSS code has been shown:



ORDER SUMMARY:

Before the user wishes to place an order, he/she will be able to view the order summary. The order summary has the details about the product, shipping details, estimated arrival date and place order button. Once the user fills up all the shipping details, he/she will be able to place the order.

The order summary page looks something like this:



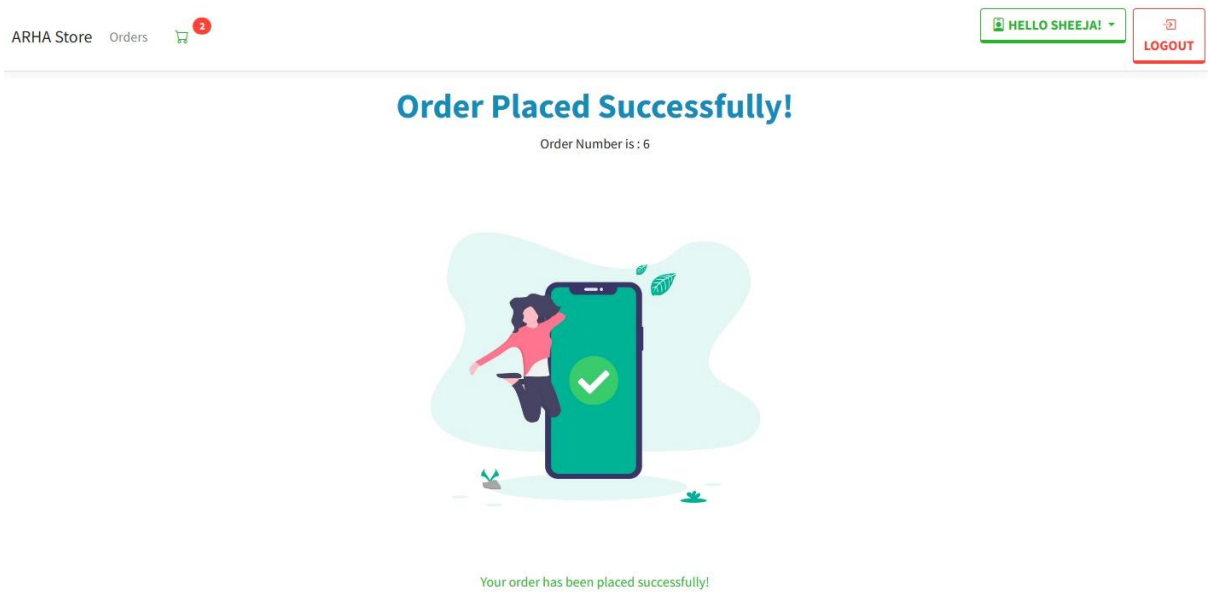
The html CSS code for this is:

```
@using Books.Domain.Utilities
@model Books.Domain.ViewModels.OrderViewModel

@{
    ViewData["Title"] = "Order";
#pragma warning disable
}

<br />
<div class="container-fluid">
    <form method="post" enctype="multipart/form-data">
        <div asp-validation-summary="All" class="text-danger"></div>
        <input asp-for="OrderHeader.Id" hidden />
        <div class="card">
            <div class="card-header bg-secondary text-dark fw-bold fs-1 ml-0">
                <div class="row">
                    <div class="col-12 d-none d-md-block col-md-6 pb-1">
                        <i class="fa fa-shopping-cart"></i><i class="bi bi-tags-fill text-success"></i>   Order Summary
                    </div>
                    <div class="col-6 text-end">
                        <a asp-action="Index" class="btn btn-outline-info btn-sm">Back to Orders</a>
                    </div>
                </div>
            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-sm-12 col-lg-6 pb-4">
                        <div class="row">
                            <h4 class="d-flex justify-content-between align-items-center mb-3">
                                <span class="text-dark fw-bold">Pick Up Details</span>
                            </h4>
                        </div>
                        <div class="row my-1">
                            <div class="col-3">
                                <label>@Html.DisplayName("Name")</label>
                            </div>
                            @if (User.IsInRole(Roles.RoleType.Admin.ToString()) || User.IsInRole(Roles.RoleType.SuperAdmin.ToStrin
                            {
                                <div class="col-4">
                                    <input asp-for="OrderHeader.FirstName" readonly class="form-control" />
```

ORDER PLACED SUCCESFULLY:

Once the user has placed the order by fulfilling the requirements in the order summary, he/she will be prompted with an order placed successfully page.

The page looks like this:

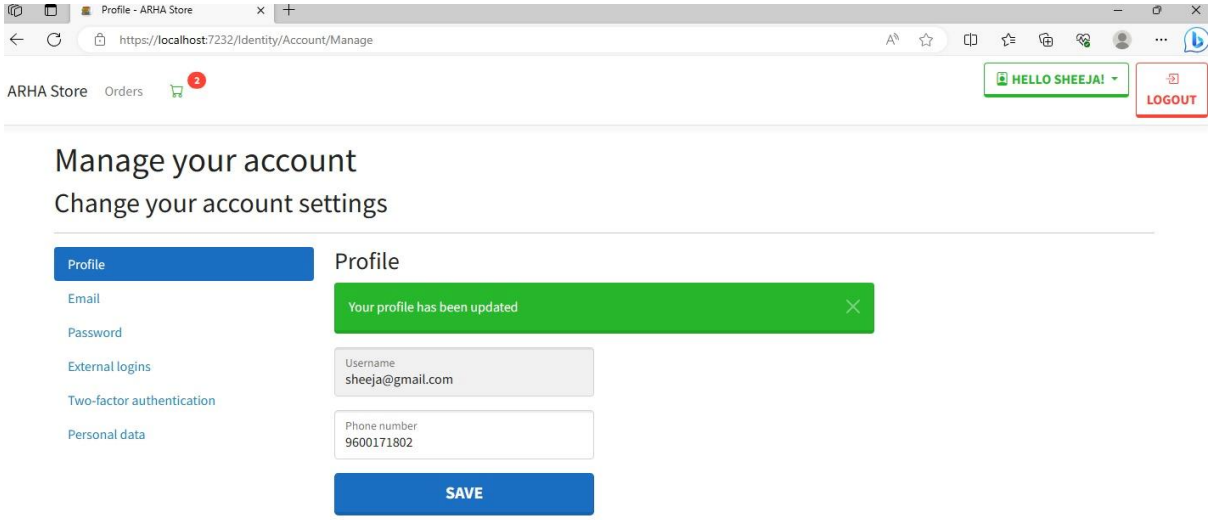


MANAGE YOUR ACCOUNTS:

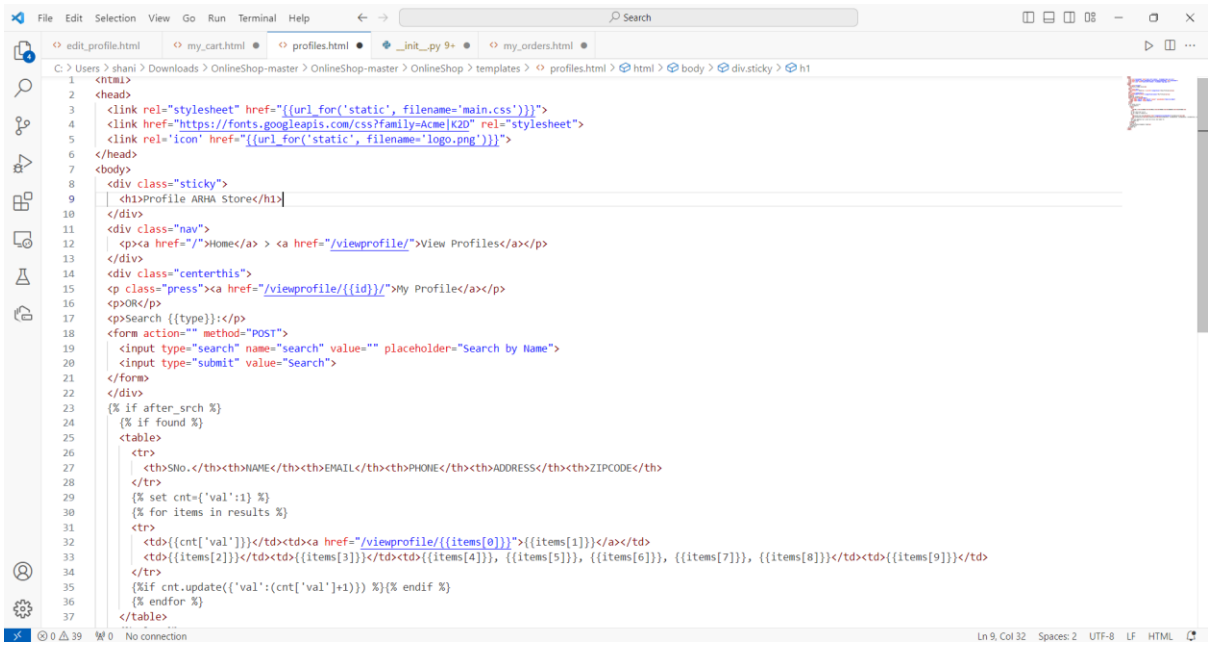
The user has an option of changing his/her account details after registration. This includes changing username, mobile number,

password in case of forgetting and other personal data.

The page looks something like this:



The HTML CSS code for this page is:



These are the front end features available in our app at present. Soon many more features will be added.

We hope that the pages will be visually pleasing and enhance user experience in our app.

Next, we are going to give an outline about the backend of our app.

BACKEND:

To develop any app, we need backend. In our app ARHA we use Flask as our backend. The entire app runs in Flask Framework.

Flask is an API of Python that allows us to build up web-applications. **Flask** is often referred to as a microframework. It is designed to keep the core of the application simple and scalable.
Flask is a web framework that allows developers

to build lightweight web applications quickly and easily with **Flask** Libraries.

HTML CSS templates are used in this as a frontend.

SQLite is used for databases.

User authentication :

User authentication in our app is performed by using bcrypt library in flask.

This can be done by using the following coding which will be incorporated in our app soon.

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_migrate import Migrate

from flask_login import (
    UserMixin,
    login_user,
```

```
    LoginManager,
    current_user,
    logout_user,
    login_required,
)
```

The flask program used to run our app is:

```
from flask import Flask, render template, request, url_for, redirect, abort, session
from flask session import Session
from ARHA.dbaccess import *
import os

app = Flask(__name__)
sess = Session()

@app.route("/")
def home():
    if "userid" in session:
        return render_template("home.html", signedin=True, id=session['userid'], name=session['name'], type=session['type'])
```

```
    return render_template("home.html", signedin=False)

@app.route("/signup/", methods = ["POST", "GET"])
def signup():
    if request.method == "POST":
        data = request.form
        ok = add_user(data)
        if ok:
            return render_template("success_signup.html")
        return render_template("signup.html", ok=ok)
    return render_template("signup.html", ok=True)

@app.route("/login/", methods=["POST", "GET"])
def login():
    if request.method == "POST":
        data = request.form
        userdat = auth_user(data)
        if userdat:
            session["userid"] = userdat[0]
            session["name"] = userdat[1]
            session["type"] = data["type"]
            return redirect(url_for('home'))
        return render_template("login.html", err=True)
```

```
    return render_template("login.html", err=False)


@app.route("/logout/")
def logout():
    session.pop('userid')
    session.pop('name')
    session.pop('type')
    return redirect(url_for('home'))


@app.route("/viewprofile/<id>/")
def view_profile(id):
    if 'userid' not in session:
        return redirect(url_for('home'))
    userid = session["userid"]
    type = session["type"]
    my = True if userid==id else False
    if not my: profile_type = "Customer" if type=="Seller" else
"Seller"
    else: profile_type = type


    det, categories = fetch_details(id, profile_type)   #details
    if len(det)==0:
        abort(404)
```

```
    det = det[0]
    return render_template("view_profile.html",
                           type=profile_type,
                           name=det[1],
                           email=det[2],
                           phone=det[3],
                           area=det[4],
                           locality=det[5],
                           city=det[6],
                           state=det[7],
                           country=det[8],
                           zip=det[9],
                           category=(None if
profile_type=="Customer" else categories),
                           my=my)


@app.route("/viewprofile/", methods=["POST", "GET"])
def profile():
    if 'userid' not in session:
        return redirect(url_for('home'))
    type = "Seller" if session['type']=="Customer" else
"Customer"
    if request.method=="POST":
```

```
        search = request.form['search']
        results = search_users(search, type)
        found = len(results)
        return render_template('profiles.html',
id=session['userid'], type=type, after_srch=True,
found=found, results=results)


    return render_template('profiles.html', id=session['userid'],
type=type, after_srch=False)


@app.route("/viewprofile/<id>/sellerproducts/")
def seller_products(id):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session["type"]=="Seller":
        abort(403)
    det, categories = fetch_details(id, "Seller")   #details
    if len(det)==0:
        abort(404)
    det = det[0]
    name=det[1]
    res = get_seller_products(id)
    return render_template('seller_products.html', name=name,
id=id, results=res)
```

```
@app.route("/editprofile/", methods=["POST", "GET"])
def edit_profile():
    if 'userid' not in session:
        return redirect(url_for('home'))

    if request.method=="POST":
        data = request.form
        update_details(data, session['userid'], session['type'])
        return redirect(url_for('view_profile',
id=session['userid']))

    if request.method=="GET":
        userid = session["userid"]
        type = session["type"]
        det, _ = fetch_details(userid, type)
        det = det[0]
        return render_template("edit_profile.html",
                               type=type,
                               name=det[1],
                               email=det[2],
                               phone=det[3],
                               area=det[4],
```

```
                    locality=det[5],
                    city=det[6],
                    state=det[7],
                    country=det[8],
                    zip=det[9])

@app.route("/changepassword/", methods=["POST", "GET"])
def change_password():
    if 'userid' not in session:
        return redirect(url_for('home'))
    check = True
    equal = True
    if request.method=="POST":
        userid = session["userid"]
        type = session["type"]
        old_psswd = request.form["old_psswd"]
        new_psswd = request.form["new_psswd"]
        cnfrm_psswd = request.form["cnfrm_psswd"]
        check = check_psswd(old_psswd, userid, type)
        if check:
            equal = (new_psswd == cnfrm_psswd)
            if equal:
                set_psswd(new_psswd, userid, type)
```

```
            return redirect(url_for('home'))
    return render_template("change_password.html",
check=check, equal=equal)


@app.route("/sell/", methods=["POST", "GET"])
def my_products():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session["type"]=="Customer":
        abort(403)
    categories = get_categories(session["userid"])
    if request.method=="POST":
        data = request.form
        srchBy = data["search method"]
        category = None if srchBy=='by keyword' else
data["category"]
        keyword = data["keyword"]
        results = search_myproduct(session['userid'], srchBy,
category, keyword)
        return render_template('my_products.html',
categories=categories, after_srch=True, results=results)
    return render_template("my_products.html",
categories=categories, after_srch=False)
```

```
@app.route("/sell/addproducts/", methods=["POST", "GET"])
def add_products():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session["type"]=="Customer":
        abort(403)
    if request.method=="POST":
        data = request.form
        add_prod(session['userid'],data)
        return redirect(url_for('my_products'))
    return render_template("add_products.html")

@app.route("/viewproduct/")
def view_prod():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        return redirect(url_for('my_products'))
    if session['type']=="Customer":
        return redirect(url_for('buy'))

@app.route("/viewproduct/<id>/")
def view_product(id):
```

```
    if 'userid' not in session:
        return redirect(url_for('home'))
    type = session["type"]
    ispresent, tup = get_product_info(id)
    if not ispresent:
        abort(404)
    (name, quantity, category, cost_price, sell_price, sellID,
desp, sell_name) = tup
    if type=="Seller" and sellID!=session['userid']:
        abort(403)
    return render_template('view_product.html', type=type,
name=name, quantity=quantity, category=category,
cost_price=cost_price, sell_price=sell_price, sell_id=sellID,
sell_name=sell_name, desp=desp, prod_id=id)


@app.route("/viewproduct/<id>/edit/", methods=["POST",
"GET"])
def edit_product(id):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Customer":
        abort(403)
    ispresent, tup = get_product_info(id)
    if not ispresent:
```

```
        abort(404)
    (name, quantity, category, cost_price, sell_price, sellID,
desp, sell_name) = tup
    if sellID!=session['userid']:
        abort(403)
    if request.method=="POST":
        data = request.form
        update_product(data, id)
        return redirect(url_for('view_product', id=id))
    return render_template('edit_product.html', prodID=id,
name=name, qty=quantity, category=category,
price=cost_price, desp=desp)


@app.route("/buy/", methods=["POST", "GET"])
def buy():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    if request.method=="POST":
        data = request.form
        srchBy = data["search method"]
        category = None if srchBy=='by keyword' else
data["category"]
```

```
        keyword = data["keyword"]
        results = search_products(srchBy, category, keyword)
        return render_template('search_products.html',
after_srch=True, results=results)
    return render_template('search_products.html',
after_srch=False)


@app.route("/buy/<id>/", methods=['POST', 'GET'])
def buy_product(id):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    ispresent, tup = get_product_info(id)
    if not ispresent:
        abort(404)
    (name, quantity, category, cost_price, sell_price, sellID,
desp, sell_name) = tup
    if request.method=="POST":
        data = request.form
        total = int(data['qty'])*float(sell_price)
        return redirect(url_for('buy_confirm', total=total,
quantity=data['qty'], id=id))
```

```
    return render_template('buy_product.html', name=name,
category=category, desp=desp, quantity=quantity,
price=sell_price)


@app.route("/buy/<id>/confirm/", methods=["POST",
"GET"])
def buy_confirm(id):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    ispresent, tup = get_product_info(id)
    if not ispresent:
        abort(404)
    (name, quantity, category, cost_price, sell_price, sellID,
desp, sell_name) = tup
    if 'total' not in request.args or 'quantity' not in request.args:
        abort(404)
    total = request.args['total']
    qty = request.args['quantity']
    if request.method=="POST":
        choice = request.form['choice']
        if choice=="PLACE ORDER":
            place_order(id, session['userid'], qty)
```

```
            return redirect(url_for('my_orders'))
        elif choice=="CANCEL":
            return redirect(url_for('buy_product', id=id))
    items = ((name, qty, total),)
    return render_template('buy_confirm.html', items=items,
total=total)


@app.route("/buy/myorders/")
def my_orders():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    res = cust_orders(session['userid'])
    return render_template('my_orders.html', orders=res)


@app.route("/cancel/<orderID>/")
def cancel_order(orderID):
    if 'userid' not in session:
        return redirect(url_for('home'))
    res = get_order_details(orderID)
    if len(res)==0:
        abort(404)
```

```
    custID = res[0][0]
    sellID = res[0][1]
    status = res[0][2]
    if session['type']=="Seller" and sellID!=session['userid']:
        abort(403)
    if session['type']=="Customer" and
custID!=session['userid']:
        abort(403)
    if status!="PLACED":
        abort(404)
    change_order_status(orderID, "CANCELLED")
    return redirect(url_for('my_orders')) if
session['type']=="Customer" else
redirect(url_for('new_orders'))


@app.route("/dispatch/<orderID>/")
def dispatch_order(orderID):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Customer":
        abort(403)
    res = get_order_details(orderID)
    if len(res)==0:
        abort(404)
```

```
    custID = res[0][0]
    sellID = res[0][1]
    status = res[0][2]
    if session['userid']!=sellID:
        abort(403)
    if status!="PLACED":
        abort(404)
    change_order_status(orderID, "DISPACHED")
    return redirect(url_for('new_orders'))

@app.route("/recieve/<orderID>/")
def recieve_order(orderID):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    res = get_order_details(orderID)
    if len(res)==0:
        abort(404)
    custID = res[0][0]
    sellID = res[0][1]
    status = res[0][2]
    if session['userid']!=custID:
```

```
        abort(403)
    if status!="DISPACHED":
        abort(404)
    change_order_status(orderID, "RECIEVED")
    return redirect(url_for('my_purchases'))

@app.route("/buy/purchases/")
def my_purchases():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    res = cust_purchases(session['userid'])
    return render_template('my_purchases.html',
purchases=res)

@app.route("/sell/neworders/")
def new_orders():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Customer":
        abort(403)
    res = sell_orders(session['userid'])
```

```
    return render_template('new_orders.html', orders=res)

@app.route("/sell/sales/")
def my_sales():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Customer":
        abort(403)
    res = sell_sales(session['userid'])
    return render_template('my_sales.html', sales=res)

@app.route("/buy/cart/", methods=["POST", "GET"])
def my_cart():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    cart = get_cart(session['userid'])
    if request.method=="POST":
        data = request.form
        qty = {}
        for i in data:
            if i.startswith("qty"):
```

```
                qty[i[3:]]=data[i]      #qty[prodID]=quantity
        update_cart(session['userid'], qty)
        return redirect("/buy/cart/confirm/")
    return render_template('my_cart.html', cart=cart)

@app.route("/buy/cart/confirm/", methods=["POST", "GET"])
def cart_purchase_confirm():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    if request.method=="POST":
        choice = request.form['choice']
        if choice=="PLACE ORDER":
            cart_purchase(session['userid'])
            return redirect(url_for('my_orders'))
        elif choice=="CANCEL":
            return redirect(url_for('my_cart'))
    cart = get_cart(session['userid'])
    items = [(i[1], i[3], float(i[2])*float(i[3])) for i in cart]
    total = 0
    for i in cart:
        total += float(i[2])*int(i[3])
```

```
    return render_template('buy_confirm.html', items=items,
total=total)


@app.route("/buy/cart/<prodID>/")
def add_to_cart(prodID):
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['type']=="Seller":
        abort(403)
    add_product_to_cart(prodID, session['userid'])
    return redirect(url_for('view_product', id=prodID))


@app.route("/buy/cart/delete/")
def delete_cart():
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['userid']=="Seller":
        abort(403)
    empty_cart(session['userid'])
    return redirect(url_for('my_cart'))


@app.route("/buy/cart/delete/<prodID>/")
def delete_prod_cart(prodID):
```

```
    if 'userid' not in session:
        return redirect(url_for('home'))
    if session['userid']=="Seller":
        abort(403)
    remove_from_cart(session['userid'], prodID)
    return redirect(url_for('my_cart'))

app.config['SECRET_KEY'] = os.urandom(17)
app.config['SESSION_TYPE'] = 'filesystem'
app.config['TEMPLATES_AUTO_RELOAD'] = True
sess.init_app(app)

if __name__=="__main__":
    app.run()
```

DATABASE:

Now we have used SQLite for our convenience as beginners.

The coding we have done for this is:

```python
import sqlite3

def gen_custID():
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    cur.execute("UPDATE metadata SET custnum = custnum + 1")
    conn.commit()
    custnum = str([i for i in cur.execute("SELECT custnum FROM metadata")][0][0])
    conn.close()
    id = "CID"+"0"*(7-len(custnum))+custnum
    return id

def gen_sellID():
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    cur.execute("UPDATE metadata SET sellnum = sellnum + 1")
    conn.commit()
    sellnum = str([i for i in cur.execute("SELECT sellnum FROM metadata")][0][0])
    conn.close()
    id = "SID"+"0"*(7-len(sellnum))+sellnum
    return id

def gen_prodID():
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    cur.execute("UPDATE metadata SET prodnum = prodnum + 1")
    conn.commit()
    prodnum = str([i for i in cur.execute("SELECT prodnum FROM metadata")][0][0])
    conn.close()
    id = "PID"+"0"*(7-len(prodnum))+prodnum
    return id

def gen_orderID():
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    cur.execute("UPDATE metadata SET ordernum = ordernum + 1")
    conn.commit()
```



```python
    ordernum = str([i for i in cur.execute("SELECT ordernum FROM metadata")][0][0])
    conn.close()
    id = "OID"+"0"*(7-len(ordernum))+ ordernum
    return id

def add_user(data):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    email = data["email"]
    if data['type']=="Customer":
        a = cur.execute("SELECT * FROM customer WHERE email=?", (email,))
    elif data['type']=="Seller":
        a = cur.execute("SELECT * FROM seller WHERE email=?", (email,))
    if len(list(a))!=0:
        return False
    tup = ( data["name"],
            data["email"],
            data["phone"],
            data["area"],
            data["locality"],
            data["city"],
            data["state"],
            data["country"],
            data["zip"],
            data["password"])
    if data['type']=="customer":
        cur.execute("INSERT INTO customer VALUES (?,?,?,?,?,?,?,?,?,?,?)",(gen_custID(), *tup))
    elif data['type']=="Seller":
        cur.execute("INSERT INTO seller VALUES (?,?,?,?,?,?,?,?,?,?,?)", (gen_sellID(), *tup))
    conn.commit()
    conn.close()
    return True

def auth_user(data):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    type = data["type"]
```

```
    type = data["type"]
    email = data["email"]
    password = data["password"]
    if type=="Customer":
        a = cur.execute("SELECT custID, name FROM customer WHERE email=? AND password=?", (email, password))
    elif type=="Seller":
        a = cur.execute("SELECT sellID, name FROM seller WHERE email=? AND password=?", (email, password))
    a = list(a)
    conn.close()
    if len(a)==0:
        return False
    return a[0]

def fetch_details(userid, type):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    if type=="Customer":
        a = cur.execute("SELECT * FROM customer WHERE custID=?", (userid,))
        a = list(a)
        b = []
    elif type=="Seller":
        a = cur.execute("SELECT * FROM seller WHERE sellID=?", (userid,))
        a = list(a)
        b = cur.execute("SELECT DISTINCT(category) from product WHERE sellID=?", (userid,))
        b = [i[0] for i in b ]
    conn.close()
    return a, b

def search_users(search, srch_type):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    search = "%"+search+"%"
    if srch_type=="Customer":
        res = cur.execute("SELECT custID, name, email, phone, area, locality, city, state, country, zipcode FROM customer WHERE LOWER(name) like ?", (search,))
    elif srch_type=="Seller":
        res = cur.execute("SELECT sellID, name, email, phone, area, locality, city, state, country, zipcode FROM seller WHERE LOWER(name) like ?", (search,))
    res = [i for i in res ]
```

```
    elif srch_type=="Seller":
        res = cur.execute("SELECT sellID, name, email, phone, area, locality, city, state, country, zipcode FROM seller WHERE LOWER(name) like ?", (search,))
    res = [i for i in res ]
    conn.close()
    return res

def update_details(data, userid, type):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    if type=="Customer":
        cur.execute("UPDATE customer SET phone=?, area=?, locality=?, city=?, state=?, country=?, zipcode=? where custID=?", (data["phone"],
                    data["area"],
                    data["locality"],
                    data["city"],
                    data["state"],
                    data["country"],
                    data["zip"],
                    userid))
    elif type=="Seller":
        cur.execute("UPDATE seller SET phone=?, area=?, locality=?, city=?, state=?, country=?, zipcode=? where sellID=?", (data["phone"],
                    data["area"],
                    data["locality"],
                    data["city"],
                    data["state"],
                    data["country"],
                    data["zip"],
                    userid))
    conn.commit()
    conn.close()

def check_psswd(psswd, userid, type):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    if type=="Customer":
        a = cur.execute("SELECT password FROM customer WHERE custID=?", (userid,))
    elif type=="Seller":
```

```
    elif type=="Seller":
        a = cur.execute("SELECT password FROM seller WHERE sellID=?", (userid,))
    real_psswd = list(a)[0][0]
    conn.close()
    return psswd==real_psswd

def set_psswd(psswd, userid, type):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    if type=="Customer":
        a = cur.execute("UPDATE customer SET password=? WHERE custID=?", (psswd, userid))
    elif type=="Seller":
        a = cur.execute("UPDATE seller SET password=? WHERE sellID=?", (psswd, userid))
    conn.commit()
    conn.close()

def add_prod(sellID, data):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    prodID = gen_prodID()
    tup = (prodID,
           data["name"],
           data["qty"],
           data["category"],
           data["price"],
           data["price"],
           data["desp"],
           sellID)
    cur.execute("INSERT INTO product VALUES (?,?,?,?,?,(SELECT profit_rate from metadata)*?,?,?)", tup)
    conn.commit()
    conn.close()

def get_categories(sellID):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    a = cur.execute("SELECT DISTINCT(category) from product where sellID=?", (sellID,))
    categories = [i[0] for i in a]
```

```
    cur = conn.cursor()
    a = cur.execute("SELECT DISTINCT(category) from product where sellID=?", (sellID,))
    categories = [i[0] for i in a]
    conn.close()
    return categories

def search_myproduct(sellID, srchBy, category, keyword):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    keyword = ['%'+i+'%' for i in keyword.split()]
    if len(keyword)==0: keyword.append('%%')
    if srchBy=="by category":
        a = cur.execute("""SELECT prodID, name, quantity, category, cost_price
                           FROM product WHERE category=? AND sellID=? """,(category, sellID))
        res = [i for i in a]
    elif srchBy=="by keyword":
        res = []
        for word in keyword:
            a = cur.execute("""SELECT prodID, name, quantity, category, cost_price
                               FROM product
                               WHERE (name LIKE ? OR description LIKE ? OR category LIKE ?) AND sellID=? """,
                               (word, word, word, sellID))
            res += list(a)
        res = list(set(res))
    elif srchBy=="both":
        res = []
        for word in keyword:
            a = cur.execute("""SELECT prodID, name, quantity, category, cost_price
                               FROM product
                               WHERE (name LIKE ? OR description LIKE ?) AND sellID=? AND category=? """,
                               (word, word, sellID, category))
            res += list(a)
        res = list(set(res))
    conn.close()
    return res

def get_product_info(id):
```

```python
def get_product_info(id):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute("""SELECT p.name, p.quantity, p.category, p.cost_price, p.sell_price,
                       p.sellID, p.description, s.name FROM product p JOIN seller s
                       WHERE p.sellID=s.sellID AND p.prodID=? """, (id,))
    res = [i for i in a]
    conn.close()
    if len(res)==0:
        return False, res
    return True, res[0]

def update_product(data, id):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    cur.execute("""UPDATE product
    SET name=?, quantity=?, category=?, cost_price=?,
    sell_price=(SELECT profit_rate from metadata)*?, description=?
    where prodID=?""",( data['name'],
                        data['qty'],
                        data['category'],
                        data['price'],
                        data['price'],
                        data['desp'],
                        id))
    conn.commit()
    conn.close()

def search_products(srchBy, category, keyword):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    keyword = ['%'+i+'%' for i in keyword.split()]
    if len(keyword)==0: keyword.append('%%')
    if srchBy=="by category":
        a = cur.execute("""SELECT prodID, name, category, sell_price
                           FROM product WHERE category=? AND quantity!=0 """,(category,))
```

```python
def search_products(srchBy, category, keyword):
    conn = sqlite3.connect("ARHA/ARHA.db")
    cur = conn.cursor()
    keyword = ['%'+i+'%' for i in keyword.split()]
    if len(keyword)==0: keyword.append('%%')
    if srchBy=="by category":
        a = cur.execute("""SELECT prodID, name, category, sell_price
                           FROM product WHERE category=? AND quantity!=0 """,(category,))
        res = [i for i in a]
    elif srchBy=="by keyword":
        res = []
        for word in keyword:
            a = cur.execute("""SELECT prodID, name, category, sell_price
                               FROM product
                               WHERE (name LIKE ? OR description LIKE ? OR category LIKE ?) AND quantity!=0 """,
                               (word, word, word))
            res += list(a)
        res = list(set(res))
    elif srchBy=="both":
        res = []
        for word in keyword:
            a = cur.execute("""SELECT prodID, name, category, sell_price
                               FROM product
                               WHERE (name LIKE ? OR description LIKE ?) AND quantity!=0 AND category=? """,
                               (word, word, category))
            res += list(a)
        res = list(set(res))
    conn.close()
    return res

def get_seller_products(sellID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute("SELECT prodID, name, category, sell_price FROM product WHERE sellID=? AND quantity!=0", (sellID,))
    res = [i for i in a]
    conn.close()
    return res
```

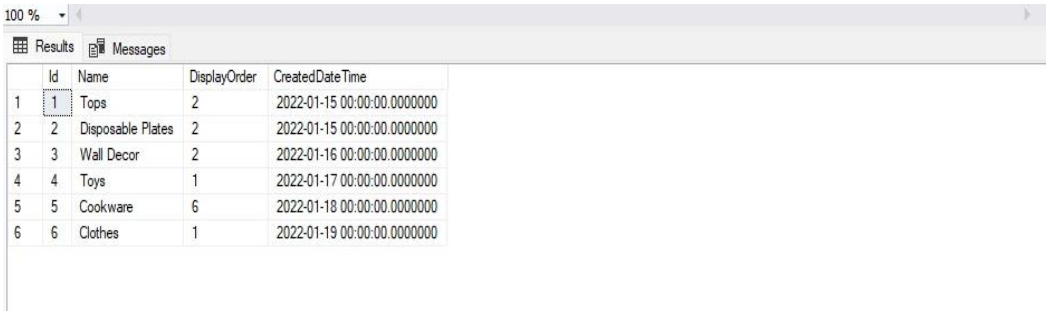
```
def get_seller_products(sellID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute("SELECT prodID, name, category, sell_price FROM product WHERE sellID=? AND quantity!=0", (sellID,))
    res = [i for i in a]
    conn.close()
    return res

def place_order(prodID, custID, qty):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    orderID = gen_orderID()
    cur.execute("""INSERT INTO orders
                SELECT ?,?,?,?,datetime('now'), cost_price*?, sell_price*?, 'PLACED'
                FROM product WHERE prodID=? """, (orderID, custID, prodID, qty, qty, qty, prodID))
    conn.commit()
    conn.close()

def cust_orders(custID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute("""SELECT o.orderID, o.prodID, p.name, o.quantity, o.sell_price, o.date, o.status
                FROM orders o JOIN product p
                WHERE o.prodID=p.prodID AND o.custID=? AND o.status!='RECIEVED'
                ORDER BY o.date DESC """, (custID,))
    res = [i for i in a]
    conn.close()
    return res

def sell_orders(sellID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute(""" SELECT o.orderID, o.prodID, p.name, o.quantity, p.quantity, o.cost_price, o.date, o.status
                FROM orders o JOIN product p
                WHERE o.prodID=p.prodID AND p.sellID=? AND o.status!='RECIEVED'
                ORDER BY o.date DESC """, (sellID,))
```

```
    res = [i for i in a]
    conn.close()
    return res

def get_order_details(orderID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute(""" SELECT o.custID, p.sellID, o.status FROM orders o JOIN product p
                WHERE o.orderID=? AND o.prodID=p.prodID """, (orderID,))
    res = [i for i in a]
    conn.close()
    return res

def change_order_status(orderID, new_status):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    cur.execute("UPDATE orders SET status=? WHERE orderID=? ", (new_status, orderID))
    if new_status=='DISPACHED':
        cur.execute("""UPDATE product SET
                quantity=quantity-(SELECT quantity FROM orders WHERE orderID=? )
                WHERE prodID=(SELECT prodID FROM orders WHERE orderID=? )""", (orderID, orderID))
    conn.commit()
    conn.close()

def cust_purchases(custID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute("""SELECT o.prodID, p.name, o.quantity, o.sell_price, o.date
                FROM orders o JOIN product p
                WHERE o.prodID=p.prodID AND o.custID=? AND o.status='RECIEVED'
                ORDER BY o.date DESC """, (custID,))
    res = [i for i in a]
    conn.close()
    return res

def sell_sales(sellID):
```

```
    cur = conn.cursor()
    a = cur.execute("""SELECT o.prodID, p.name, o.quantity, o.sell_price, o.date, o.custID, c.name
                       FROM orders o JOIN product p JOIN customer c
                       WHERE o.prodID=p.prodID AND o.custID=c.custID AND p.sellID=? AND o.status='RECIEVED'
                       ORDER BY o.date DESC """, (sellID,))
    res = [i for i in a]
    conn.close()
    return res

def add_product_to_cart(prodID, custID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    cur.execute("""INSERT INTO cart VALUES (?,?,1) """, (custID, prodID))
    conn.commit()
    conn.close()

def get_cart(custID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    a = cur.execute("""SELECT p.prodID, p.name, p.sell_price, c.sum_qty, p.quantity
                       FROM (SELECT custID, prodID, SUM(quantity) AS sum_qty FROM cart
                       GROUP BY custID, prodID) c JOIN product p
                       WHERE p.prodID=c.prodID AND c.custID=?""", (custID,))
    res = [i for i in a]
    conn.close()
    return res

def update_cart(custID, qty):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    for prodID in qty:
        cur.execute("DELETE FROM cart WHERE prodID=? AND custID=?", (prodID, custID))
        cur.execute("INSERT INTO cart VALUES (?,?,?)", (custID, prodID, qty[prodID]))
    conn.commit()
    conn.close()

def cart_purchase(custID):
```

```
    return res

def update_cart(custID, qty):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    for prodID in qty:
        cur.execute("DELETE FROM cart WHERE prodID=? AND custID=?", (prodID, custID))
        cur.execute("INSERT INTO cart VALUES (?,?,?)", (custID, prodID, qty[prodID]))
    conn.commit()
    conn.close()

def cart_purchase(custID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    cart = get_cart(custID)
    for item in cart:
        orderID = gen_orderID()
        prodID = item[0]
        qty = item[3]
        cur.execute("""INSERT INTO orders
                       SELECT ?,?,?,?,datetime('now'), cost_price*?, sell_price*?, 'PLACED'
                       FROM product WHERE prodID=? """, (orderID, custID, prodID, qty, qty, qty, prodID))
        cur.execute("DELETE FROM cart WHERE custID=? AND prodID=?", (custID, prodID))
        conn.commit()
    conn.close()

def empty_cart(custID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    cur.execute("DELETE FROM cart WHERE custID=?", (custID,))
    conn.commit()

def remove_from_cart(custID, prodID):
    conn = sqlite3.connect('ARHA/ARHA.db')
    cur = conn.cursor()
    cur.execute("DELETE FROM cart WHERE custID=? AND prodID=?", (custID, prodID))
    conn.commit()
```

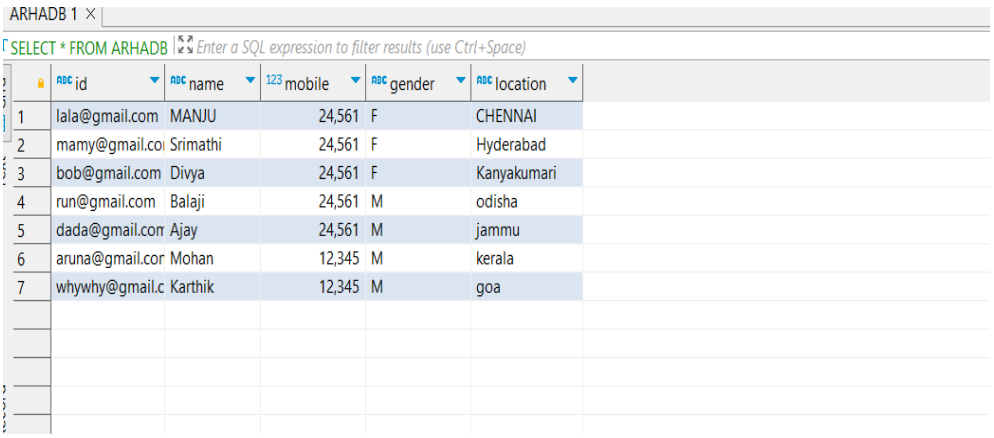
The database to store customer information looks something like this:



100 %

Results | Messages

	Id	Name	DisplayOrder	CreatedDateTime
1	1	Tops	2	2022-01-15 00:00:00.0000000
2	2	Disposable Plates	2	2022-01-15 00:00:00.0000000
3	3	Wall Decor	2	2022-01-16 00:00:00.0000000
4	4	Toys	1	2022-01-17 00:00:00.0000000
5	5	Cookware	6	2022-01-18 00:00:00.0000000
6	6	Clothes	1	2022-01-19 00:00:00.0000000



ARHADB 1

SELECT * FROM ARHADB — Enter a SQL expression to filter results (use Ctrl+Space)

	id	name	mobile	gender	location
1	lala@gmail.com	MANJU	24,561	F	CHENNAI
2	mamy@gmail.co	Srimathi	24,561	F	Hyderabad
3	bob@gmail.com	Divya	24,561	F	Kanyakumari
4	run@gmail.com	Balaji	24,561	M	odisha
5	dada@gmail.com	Ajay	24,561	M	jammu
6	aruna@gmail.cor	Mohan	12,345	M	kerala
7	whywhy@gmail.c	Karthik	12,345	M	goa