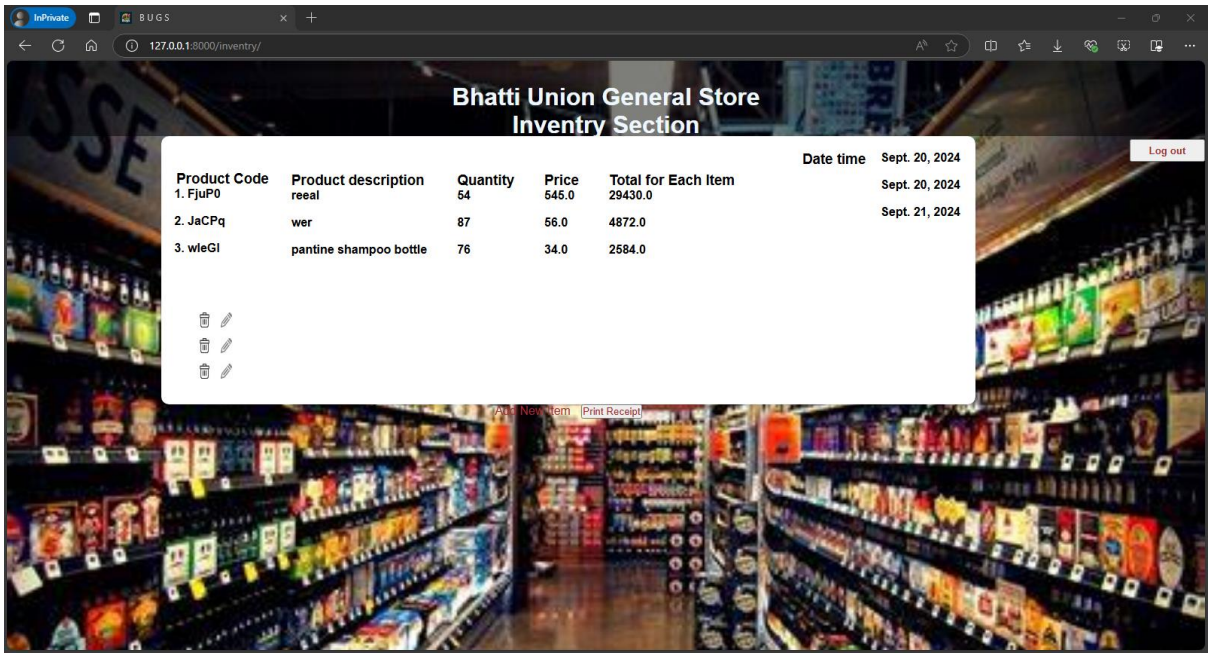
index.html:




```
{% extends "inventry/layout.html" %}

{% block body %}

<div>

   <form action="{% if is_editing %} {% url 'inventry:edit_product' prod_index prod_code %} {% else %}{% url 'inventry:add' %}{% endif %}" method="post">

      {% csrf_token %}

      <!-- Dynamic heading based on whether you are adding or editing an item -->

      <h1>

         {% if is_editing %}

            Please edit details

         {% else %}

            Please add details

         {% endif %}

      </h1>

      {% if form %}

         {{ form.as_p }}

      {% endif %}
```

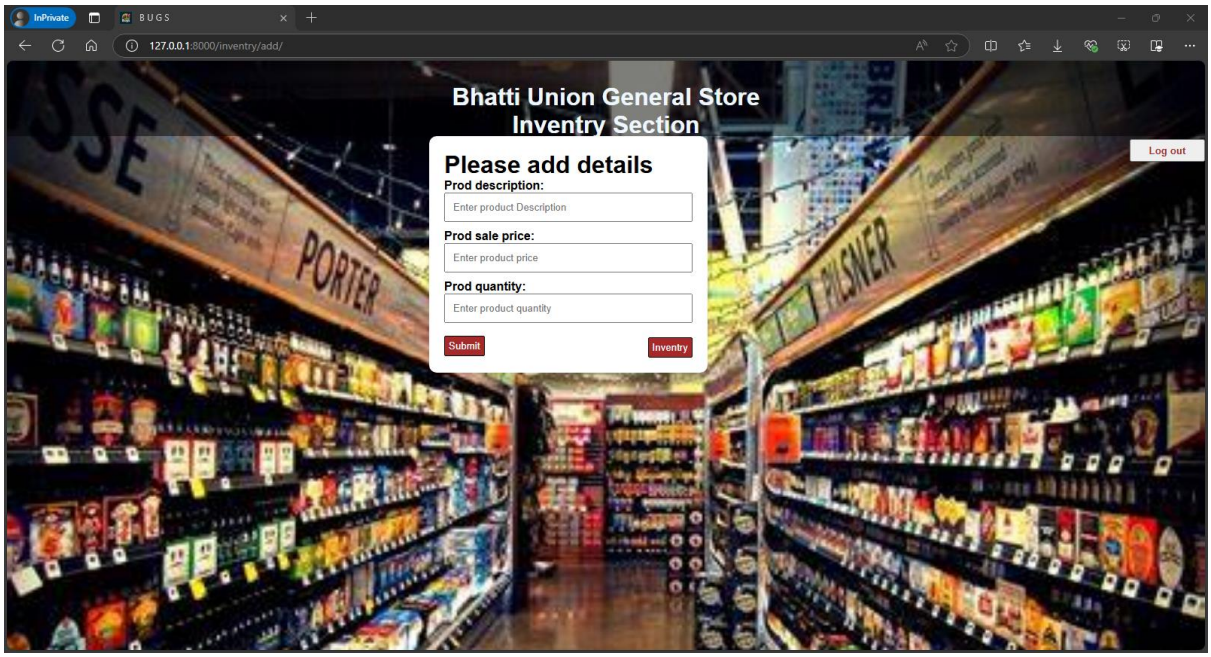
```
    <input type="submit" style="width: fit-content; height: fit-content; background-color: brown; color: white; padding: 5px; margin-top: 7px; border-radius: 2px; border: 1px solid rgb(0, 0, 0);">
  </form>

  <a href="{% url 'inventry:index' %}">
    <button style="width: fit-content; height: fit-content; background-color: brown; color: white; padding: 5px; margin-top: -25px; border-radius: 2px; border: 1px solid rgb(0, 0, 0); float: right;">
      Inventry
    </button>
  </a>
</div>

<script>
  document.addEventListener("DOMContentLoaded", function() {
    // Focus on the product name field if the customer name field is hidden
    const productNameField = document.getElementById("id_name");
    const customerNameField = document.getElementById("id_customer_name");

    if (customerNameField && !customerNameField.closest("form").querySelector(".hidden")) {
      // Focus on the customer name field if it's visible
      customerNameField.focus();
    } else if (productNameField) {
      // Focus on the product name field otherwise
      productNameField.focus();
    }
  });
</script>
{% endblock %}
```

Add.html:



layout.html: <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

{% load static %}

<link rel="stylesheet" type="text/css" href="{% static 'inventry_css/style.css' %}">

<link rel="icon" type="image/x-icon"

href="https://th.bing.com/th/id/OIG1.y_2amRAytZhrAjPsZA6Y?w=173&h=173&c=6&r=0&o=5&pid=ImgGn" alt="BUGS">

<title>B U G S</title>

</head>

<body>

<header>

<h1>Bhatti Union General Store </h1>

<h1>Inventry Section</h1>

<a href="{% url 'inventry:logout' %}">

<button type="button" class="logout-button"

```
            style="float: right;width: 100px;height: 30px; color: brown; "><strong>Log out</strong></button>
        </a>
    </header>
    <div id="outerContainer">
        <div id="container">
            {% block body %}
            {% endblock %}
        </div>
    </div>
</body>
</html>    style.css:/* Reset default margin and padding */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
/* Center the entire page content */
body {
  min-height: 100vh;
  background-image:                    url("https://th.bing.com/th/id/OIP.oJnD-qE_AcmVVeoB9YnPrAHaFj?rs=1&pid=ImgDetMain");
  background-size: cover;
  background-repeat: no-repeat;
  font-family: Arial, sans-serif;
}
/* Header styling */
header {
  width: 100%;
  height: 100px;
  text-align: center;
```

```
  background-color: rgba(0, 0, 0, 0.5);
  padding-top: 30px;
  padding-bottom: 30px;
  color: aliceblue;
}
/* Centered container */
#outerContainer {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}
#container {
  width: fit-content;
  padding: 20px;
  background-color: #ffffff;
  border-radius: 10px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  font-size: medium;
  font-weight: bold;
}
/* Receipt details section */
.receipt-details {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}
.receipt-section {
  margin: 10px;
}
.receipt-date {
```

```
  float: right;
}
.receipt-actions {
  margin-top: 10px;
}
.delete-button, .edit-button {
  display: inline-block;
  width: 15px;  /* Match the width of the image */
  height: 20px; /* Match the height of the image */
  margin-bottom: 10px;
  margin-right: 10px;
}

.action-button {
  color: brown;
  text-decoration: none;
  margin-right: 10px;
}

.print-button {
  margin-right: 0;
}

/* Print styles */
@media print {
  body * {
    visibility: hidden;
  }
  #receipt-content, #receipt-content * {
    visibility: visible;
  }
```

```
#receipt-content {
  position: absolute;
  left: 0;
  top: 0;
}
```

Views.py:

```
from django.contrib.auth import authenticate, login, logout
from django import forms
from django.shortcuts import render, redirect
from django.http import HttpResponseRedirect, HttpResponse
from django.urls import reverse
from datetime import datetime
import random
import string
from . import models

# Helper function to generate a random receipt number
def generate_random_key(length=5):
    characters = string.ascii_letters + string.digits
    random_key = ''.join(random.choices(characters, k=length))
    return random_key

# Form for adding/editing products
class NewDataForm(forms.Form):
    def for_edit_product(self,prod_desc ,pric, quant, *args,
**kwargs):
        super().__init__(*args, **kwargs)
        self.fields['prod_description'].initial = prod_desc
        self.fields['prod_sale_price'].initial = pric
        self.fields['prod_quantity'].initial = quant

    prod_description = forms.CharField(
        widget=forms.TextInput(attrs={
            'id': 'id_name',
            'placeholder': 'Enter product Description',
            'class': 'form-control',
            'style': 'width: 100%; padding: 10px; margin-bottom:
10px;'
        })
    )
```

```
    prod_sale_price = forms.IntegerField(
        widget=forms.NumberInput(attrs={
            'placeholder': 'Enter product price',
            'class': 'form-control',
            'style': 'width: 100%; padding: 10px; margin-bottom:
10px;'
        })
    )

    prod_quantity = forms.IntegerField(
        widget=forms.NumberInput(attrs={
            'placeholder': 'Enter product quantity',
            'class': 'form-control',
            'style': 'width: 100%; padding: 10px; margin-bottom:
10px;'
        })
    )

def index(request):
    if not request.user.is_authenticated:
        return HttpResponseRedirect(reverse("inventry:login"))

    # Initialize session variables if they do not exist
    # if "products" not in request.session:
    #     request.session["products"] = []
    # 2D list to store
[                name,           quantity,        price,      quanti
ty_price]
    # 2D list to store [prod_code, prod_description, prod_quantity,
prod_sale_price,prod_quantity * prod_sale_price , updated_datetime]

    # return render(request, 'inventry/index.html', {
    #     "products": request.session["products"],
    #     # 'now': datetime.now(),
    #     'length_products':range(len(request.session["products"]))
    # })
    return render(request, 'inventry/index.html', {
        "products": models.view_inventory(request),
        # 'now': datetime.now(),
        'length_products':range(len(models.view_inventory(request)))
    })
def add(request):
```

```
    if not request.user.is_authenticated:
        return HttpResponseRedirect(reverse("inventry:login"))

    # Ensure the session key 'products' is initialized
    if "products" not in request.session:
        request.session["products"] = []

    if request.method == 'POST':
        form = NewDataForm(request.POST)
        if form.is_valid():
            prod_code = generate_random_key()
            prod_description = form.cleaned_data['prod_description']
            prod_sale_price = form.cleaned_data['prod_sale_price']
            prod_quantity = form.cleaned_data['prod_quantity']
            quantity_price_sale = prod_sale_price * prod_quantity
            updated_datetime = datetime.now()

            # Append the new product to the session 'products' list
            # request.session["products"].append([
            #     prod_code,
            #     prod_description,
            #     prod_quantity,
            #     prod_sale_price,
            #     quantity_price_sale,
            #     updated_datetime.strftime("%Y-%m-%d %H:%M:%S")  #
Convert datetime to string
            # ])
            # request.session.modified = True  # Mark the session as
modified to ensure changes are saved
            models.add_each_item(prod_code, prod_description,
prod_quantity, prod_sale_price, quantity_price_sale,
updated_datetime,request.user.username)
        else:
            return render(request, 'inventry/add.html', {'form':
form})

    return render(request, 'inventry/add.html', {"form":
NewDataForm()})

def delet(request, prod_index,prod_code):
    if not request.user.is_authenticated:
        return HttpResponseRedirect(reverse("inventry:login"))
```

```
    # try:
    models.delete_item(prod_code)
    print("deleted the product")
        # delete product from database where prod_code
    # except IndexError:
    #     pass  # Handle index errors if necessary

    return redirect('inventry:index')

def edit_product(request, prod_index,prod_code):
    if not request.user.is_authenticated:
        return HttpResponseRedirect(reverse("inventry:login"))

    try:
        # Fetch the product details from the session
        product = models.get_product(prod_code)
        product=product[0]
        print(f"edit product is :: {product}")
        prod_code, prod_description, prod_quantity, prod_sale_price,
quantity_price_sale, updated_datetime,username = product

    except IndexError:
        return redirect('inventry:index')  # Redirect if invalid
index

    if request.method == 'POST':
        form = NewDataForm(request.POST)
        if form.is_valid():
            # Retrieve updated data from the form
            new_prod_description =
form.cleaned_data['prod_description']
            new_prod_sale_price =
form.cleaned_data['prod_sale_price']
            new_prod_quantity = form.cleaned_data['prod_quantity']
            new_quantity_price_sale = new_prod_sale_price *
new_prod_quantity
            new_updated_datetime = datetime.now()

            # Update the product details in the session
            # request.session["products"][prod_index] = [
            #     prod_code,
```

```
            #     new_prod_description,
            #     new_prod_quantity,
            #     new_prod_sale_price,
            #     new_quantity_price_sale,
            #     new_updated_datetime.strftime("%Y-%m-%d
%H:%M:%S")  # Convert datetime to string
            # ]
            # request.session.modified = True  # Ensure session is
saved
            models.add_each_item(prod_code, new_prod_description,
new_prod_quantity, new_prod_sale_price, new_quantity_price_sale,
new_updated_datetime,request.user.username)
            return redirect('inventry:index')
        else:
            print(f"Form is invalid: {form.errors}")

    else:
        # If GET request, prepopulate the form with existing product
details
        form = NewDataForm(initial={
            'prod_description': prod_description,
            'prod_quantity': prod_quantity,
            'prod_sale_price': prod_sale_price
        })

    return render(request, 'inventry/add.html', {
        "form": form,
        'is_editing': True,
        'prod_index': prod_index,
        'prod_code': prod_code  # Add this line
    })

def login_view(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = authenticate(request, username=username,
password=password)
        if user is not None:
            login(request, user)
            return HttpResponseRedirect(reverse("inventry:add"))
        else:
```

```
            return render(request, "inventry/login.html", {
                "message": "Invalid credentials.",
                "username": username  # Retain the entered username
            })
    return render(request, "inventry/login.html")

def logout_view(request):
    logout(request)
    return HttpResponseRedirect(reverse("inventry:login"))
```

make its css better