**Summer Project**

**On**

**E-commerce Website**

**Website Name (LORD S HOP)**

**By**

**Lordhang Rai(8641/18)**



**Submitted To:**

**Mr.Dinesh Bajracharya**

**Kantipur College of Management and Information Technology**

In Partial Fulfillment of the Requirements for the Course

Bachelor of Information Management

Mid Baneshwor, Kathmandu

sep, 2022

**ACKNOWLEDGEMENT**

I would like to express my sincere thanks and gratitude to our Teachers Mr.Dinesh Bajracharya, Mr.Indra Pc, Mr.Rakesh Shrestha for his valuable guidance and suggestions for my project. I am highly indebted to them for providing us an excellent opportunity to learn and present our studies in the form of this project.

I am grateful to all our teachers for their suggestions and inspirational lectures that paved towards the completion of this project.

**TABLE OF CONTENTS**

[CHAPTER I 1](#_Toc111563058)

[INTRODUCTION 1](#_Toc111563059)

[1.1 Scope 1](#_Toc111563060)

1.2 Technologies used in the Project…………………………………………………. 1

[1.3 **Required Skillset to Build the Project:** ……………………………1](#_Toc111563062)

1.4 ER and Use-Case Diagrams……………..................................1and 2

1.5 [**Admin Interface**](https://www.geeksforgeeks.org/python-django-admin-interface/)**:…………………………………………………….3**

CHAPTER 2

1.2 Open the project folder using a text editor. The directory structure should look like this: ……………………………………………..4

1.3 Now add store app in E-commerce website in **settings.py………5**

#### 1.4 urls.py………………………………………………………………….6

1.5 Models…………………………………………………………………………………………………6

1.6 Model and the fields required by each model……………………7

1.6.1 category.py

#### 1.6.2 customer.py

#### **1.6.3** Products.py

#### **1.6.4** Orders.py

**CHAPTER 3**

1.1 Views………………………………………………………………………………………………………8

1.2 Urls.py…………………………………………………………………………………………………….9

1,3 Home.py.....................................................................................................10

1.4 Login.py……………………………………………………………………………………………11&12

1.5 Signup.py…………………………………………………………………………………………13&14

**CONCLUSION**

1.1 Output……………………………………………………………………………………………………15

1.2 Create Accounts…………………………………………………………………………………….16

1.3 Login………………………………………………………………………………………………………16

1.4 Your Order…………………………………………………………………………………………….17

# CHAPTER I

# INTRODUCTION

This project deals with developing a Virtual website **‘E-commerce Website (LORD S HOP)’’**. It provides the user with a list of the various products available for purchase in the store. For the convenience of online shopping, a shopping cart is provided to the user. After the selection of the goods, it is sent for the order confirmation process. The system is implemented using Python’s web framework Django. To develop an e-commerce website, it is necessary to study and understand many technologies.

* 1. **Scope**:

The scope of the project will be limited to some functions of the e-commerce website. It will display products, customers can select catalogs and select products, and can remove products from their cart specifying the quantity of each item. Selected items will be collected in a cart. At checkout, the item on the card will be presented as an order. Customers can pay for the items in the cart to complete an order. This project has great future scope. The project also provides security with the use of login ID and passwords, so that no unauthorized users can access your account. The only authorized person who has the appropriate access authority can access the software.

### **1.2 Technologies used in the project:**

Django framework and SQLite database which comes by default with Django.

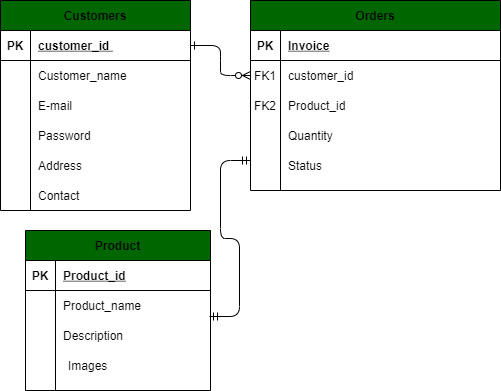
### **1.3** **Required Skillset to Build the Project:**

Knowledge of Python and basics of Django Framework.

**1.4 ER and Use-Case Diagrams**

**Customer Interface:**

1. Customer shops for a product
2. Customer changes quantity
3. The customer adds an item to the cart
4. Customer views cart
5. Customer checks out
6. Customer sends order



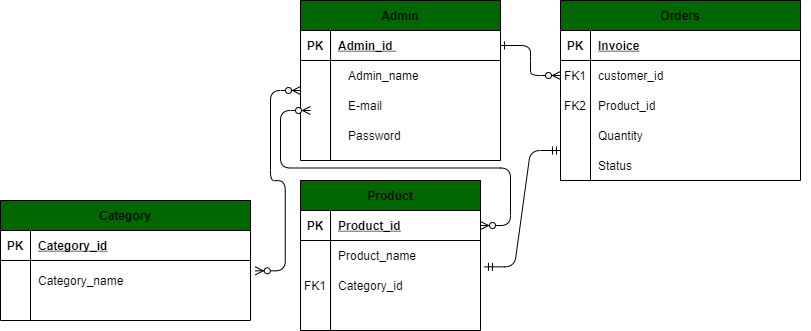
*ER-diagram for Customer*



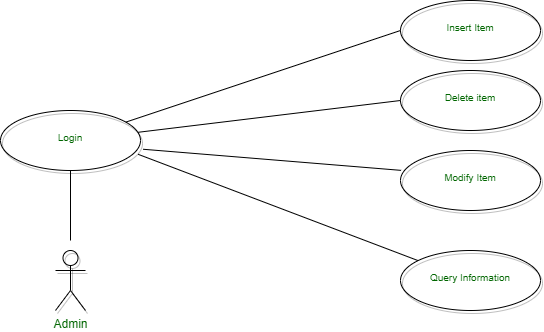
*Use-Case diagram for Customer*

**1.5**  [**Admin Interface**](https://www.geeksforgeeks.org/python-django-admin-interface/)**:**

1. Admin logs in
2. Admin inserts item
3. Admin removes item
4. Admin modifies item



*ER-diagram for Admin*

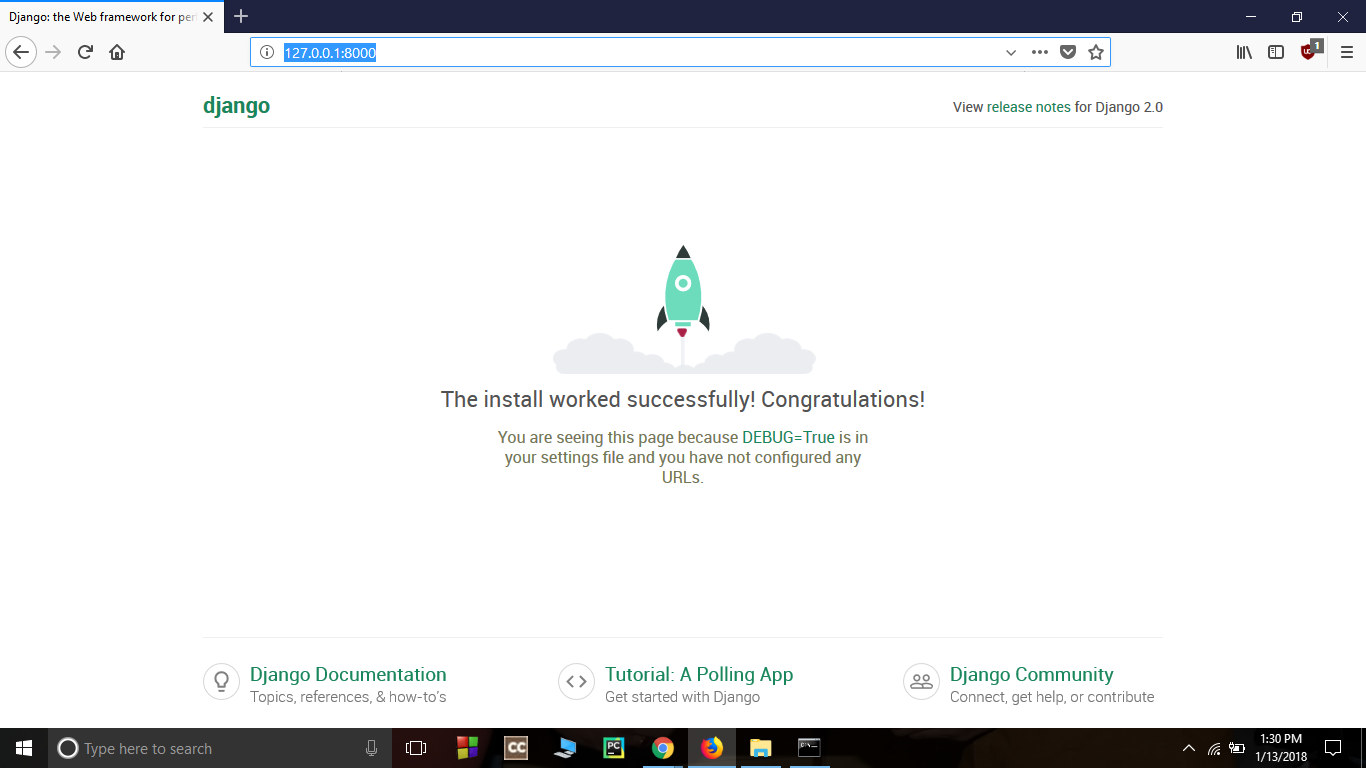


**.** *Use-Case diagram for Admin*

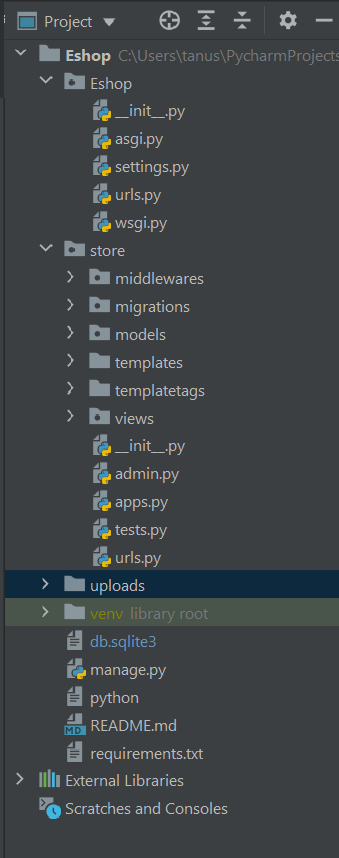
### **CHAPTER 2**

### **1.1** **Step by Step Implementation:**

* Install Django: Next, we will install the Django module from the terminal. We will use PyCharm integrated terminal to do this task. One can also use cmd on windows to install the module by running ***python -m pip install django*** command
* Check Installed Django version: To check the installed Django version, you can run the ***python -m django -version*** command as shown below.
* Create Django Project: When we execute ***django-admin startproject*** command, then it will create a Django project inside the normal project which we already have created here. ***django-admin startproject ProjectName.***
* Check Python3 version:***python3 –version***
* Run Default Django webserver:- Django internally provides a default webserver where we can launch our applications. ***python manage.py runserver***command in terminal***.*** By default, the server runs on port 8000. Access the webserver at the highlighted URL.

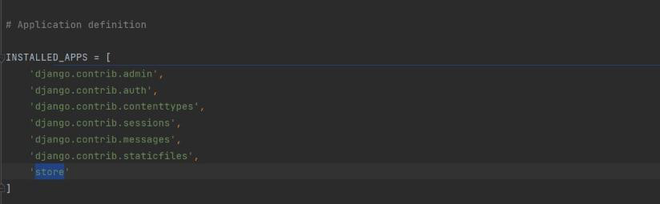


**1.2 Open the project folder using a text editor. The directory structure should look like this:**

****

*Project Structure*

**1.3 Now add store app in E-commerce website in settings.py**



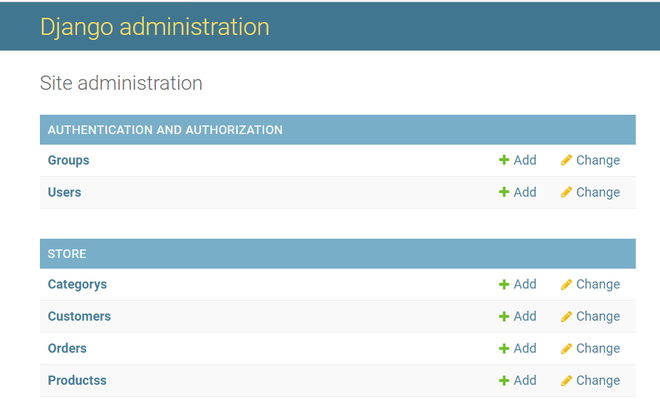
#### **1.4 urls.py**

This file contains all the URL patterns used by the website

|  |
| --- |
| from django.contrib import admin  from django.urls import path, include  from django.conf.urls.static import static  from . import setting  urlpatterns = [      path('admin/', admin.site.urls),      path('', include('store.urls'))  ] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT) |

**1.5 Models**

The below screenshot shows the required models that we will need to create. These models are tables that will be stored in the SQLite database.

****

**1.6 Model and the fields required by each model.**

#### 1.1category.py

from django.db import models

class Category(models.Model):

name = models.CharField(max\_length=50)

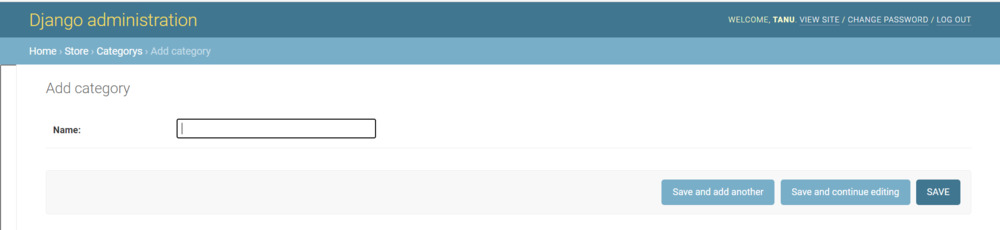
@staticmethod

def get\_all\_categories():

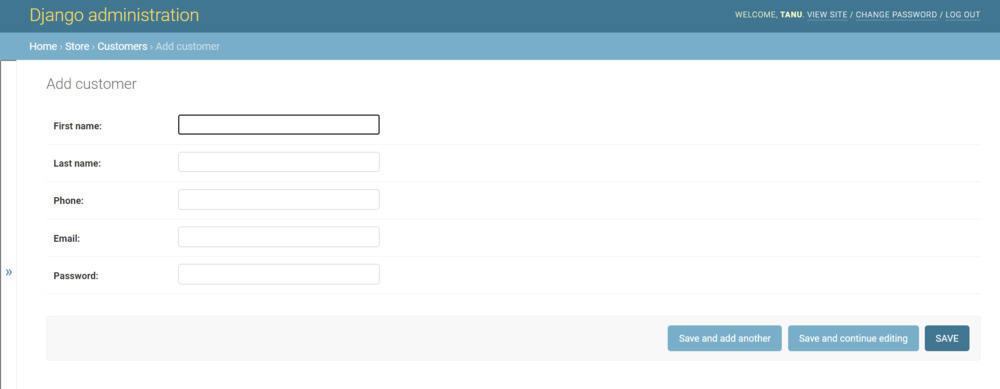
return Category.objects.all()

def \_\_str\_\_(self):

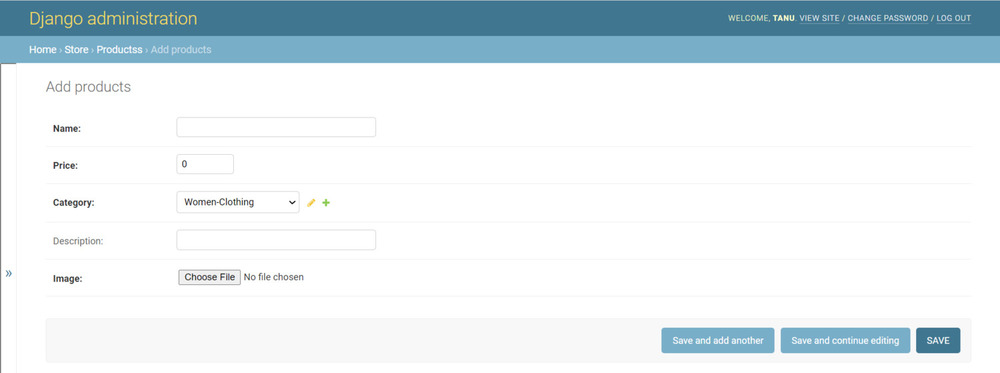
return self.name



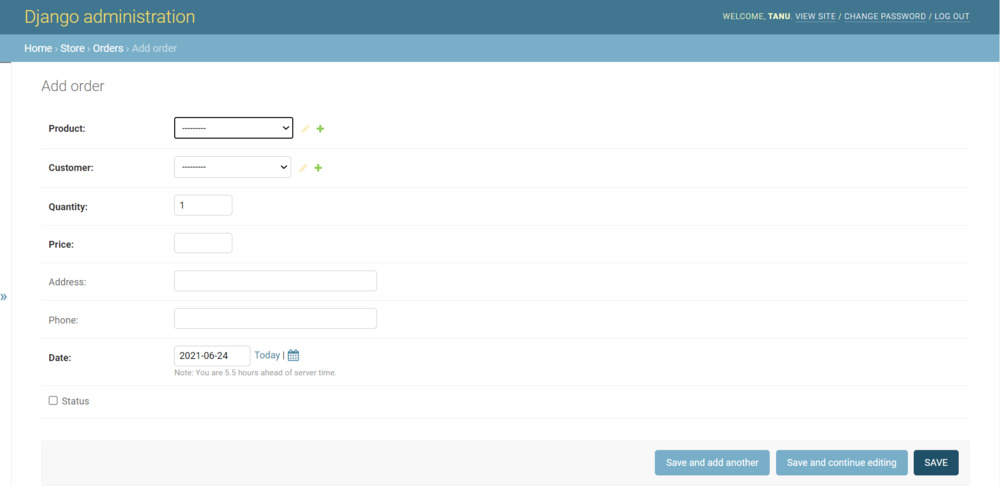
#### 1.2 customer.py



#### 1.3 products.py



#### 1.4 Orders.py



**CHAPTER 3**

* 1. **Views**

In views, we create a view named*home.py, login.py, signup.py, cart.py, checkout.py, orders.py*  which takes a request and renders an HTML as a response. Create an*home.html, login.html, signup.html, cart.html, checkout.html, orders.html* in the templates. And map the views to the store\urls.py folder.

**1.2 Urls.py**

from django.contrib import admin

from django.urls import path

from .views.home import Index, store

from .views.signup import Signup

from .views.login import Login, logout

from .views.cart import Cart

from .views.checkout import CheckOut

from .views.orders import OrderView

from .middlewares.auth import auth\_middleware

urlpatterns = [

path('', Index.as\_view(), name='homepage'),

path('store', store, name='store'),

path('signup', Signup.as\_view(), name='signup'),

path('login', Login.as\_view(), name='login'),

path('logout', logout, name='logout'),

path('cart', auth\_middleware(Cart.as\_view()), name='cart'),

path('check-out', CheckOut.as\_view(), name='checkout'),

path('orders', auth\_middleware(OrderView.as\_view()), name='orders'),

]

The below files show the views for each functionality of the site.

#### **1.3 Home.py**

**from django.shortcuts import render, redirect, HttpResponseRedirect**

**from store.models.product import Products**

**from store.models.category import Category**

**from django.views import View**

**# Create your views here.**

**class Index(View):**

**def post(self, request):**

**product = request.POST.get('product')**

**remove = request.POST.get('remove')**

**cart = request.session.get('cart')**

**if cart:**

**quantity = cart.get(product)**

**if quantity:**

**if remove:**

**if quantity <= 1:**

**cart.pop(product)**

**else:**

**cart[product] = quantity-1**

**else:**

**cart[product] = quantity+1**

**else:**

**cart[product] = 1**

**else:**

**cart = {}**

**cart[product] = 1**

**request.session['cart'] = cart**

**print('cart', request.session['cart'])**

**return redirect('homepage')**

**def get(self, request):**

**# print()**

**return HttpResponseRedirect(f'/store{request.get\_full\_path()[1:]}')**

**def store(request):**

**cart = request.session.get('cart')**

**if not cart:**

**request.session['cart'] = {}**

**products = None**

**categories = Category.get\_all\_categories()**

**categoryID = request.GET.get('category')**

**if categoryID:**

**products = Products.get\_all\_products\_by\_categoryid(categoryID)**

**else:**

**products = Products.get\_all\_products()**

**data = {}**

**data['products'] = products**

**data['categories'] = categories**

**print('you are : ', request.session.get('email'))**

**return render(request, 'index.html', data)**

#### **1.4 login.py**

from django.shortcuts import render, redirect, HttpResponseRedirect

from django.contrib.auth.hashers import check\_password

from store.models.customer import Customer

from django.views import View

class Login(View):

return\_url = None

def get(self, request):

Login.return\_url = request.GET.get('return\_url')

return render(request, 'login.html')

def post(self, request):

email = request.POST.get('email')

password = request.POST.get('password')

customer = Customer.get\_customer\_by\_email(email)

error\_message = None

if customer:

flag = check\_password(password, customer.password)

if flag:

request.session['customer'] = customer.id

if Login.return\_url:

return HttpResponseRedirect(Login.return\_url)

else:

Login.return\_url = None

return redirect('homepage')

else:

error\_message = 'Invalid !!'

else:

error\_message = 'Invalid !!'

print(email, password)

return render(request, 'login.html', {'error': error\_message})

def logout(request):

request.session.clear()

return redirect('login')

#### **1.5 signup.py**

from django.shortcuts import render, redirect

from django.contrib.auth.hashers import make\_password

from store.models.customer import Customer

from django.views import View

class Signup (View):

def get(self, request):

return render(request, 'signup.html')

def post(self, request):

postData = request.POST

first\_name = postData.get('firstname')

last\_name = postData.get('lastname')

phone = postData.get('phone')

email = postData.get('email')

password = postData.get('password')

# validation

value = {

'first\_name': first\_name,

'last\_name': last\_name,

'phone': phone,

'email': email

}

error\_message = None

customer = Customer(first\_name=first\_name,

last\_name=last\_name,

phone=phone,

email=email,

password=password)

error\_message = self.validateCustomer(customer)

if not error\_message:

print(first\_name, last\_name, phone, email, password)

customer.password = make\_password(customer.password)

customer.register()

return redirect('homepage')

else:

data = {

'error': error\_message,

'values': value

}

return render(request, 'signup.html', data)

def validateCustomer(self, customer):

error\_message = None

if (not customer.first\_name):

error\_message = "Please Enter your First Name !!"

elif len(customer.first\_name) < 3:

error\_message = 'First Name must be 3 char long or more'

elif not customer.last\_name:

error\_message = 'Please Enter your Last Name'

elif len(customer.last\_name) < 3:

error\_message = 'Last Name must be 3 char long or more'

elif not customer.phone:

error\_message = 'Enter your Phone Number'

elif len(customer.phone) < 10:

error\_message = 'Phone Number must be 10 char Long'

elif len(customer.password) < 5:

error\_message = 'Password must be 5 char long'

elif len(customer.email) < 5:

error\_message = 'Email must be 5 char long'

elif customer.isExists():

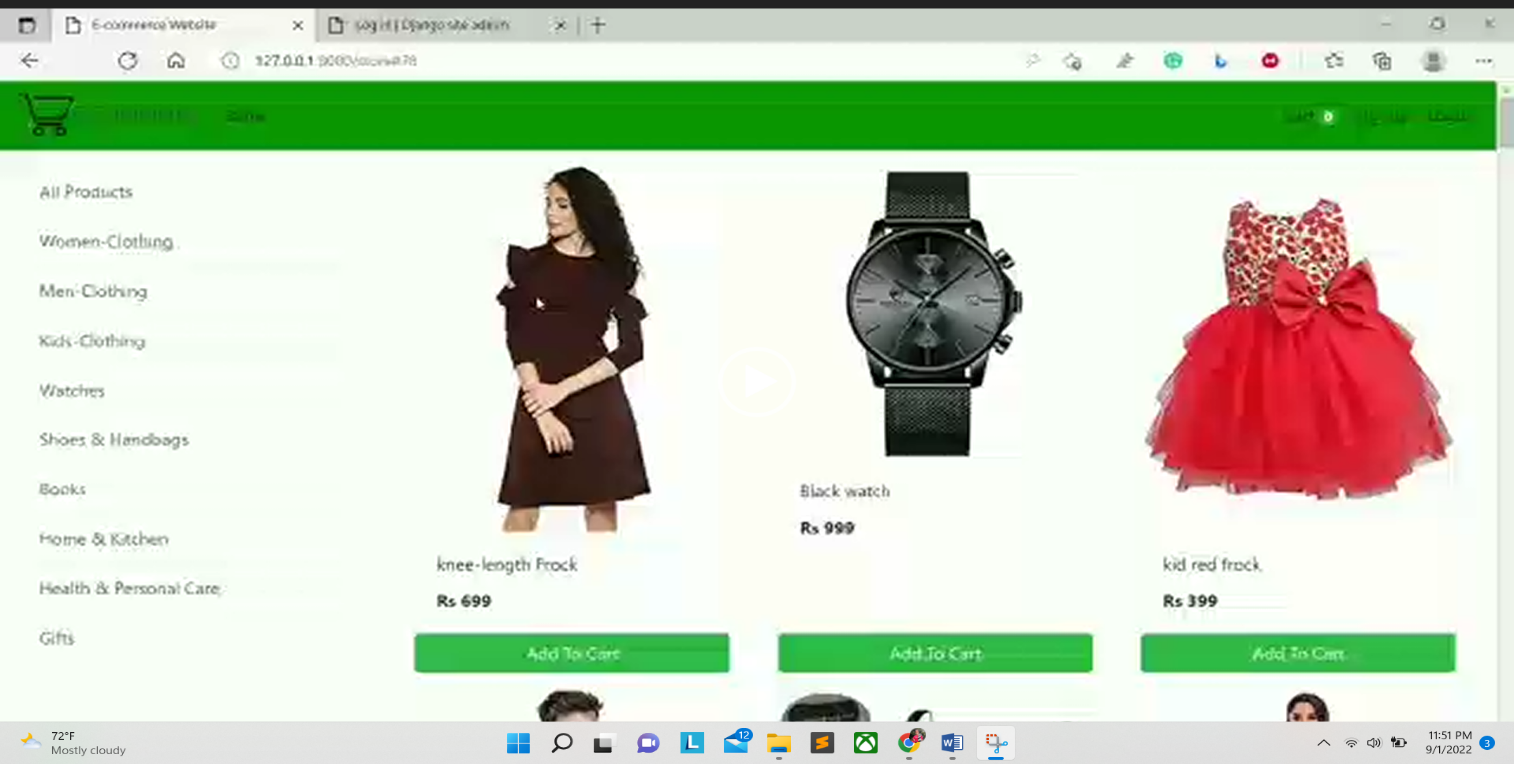
error\_message = 'Email Address Already Registered..'

# saving

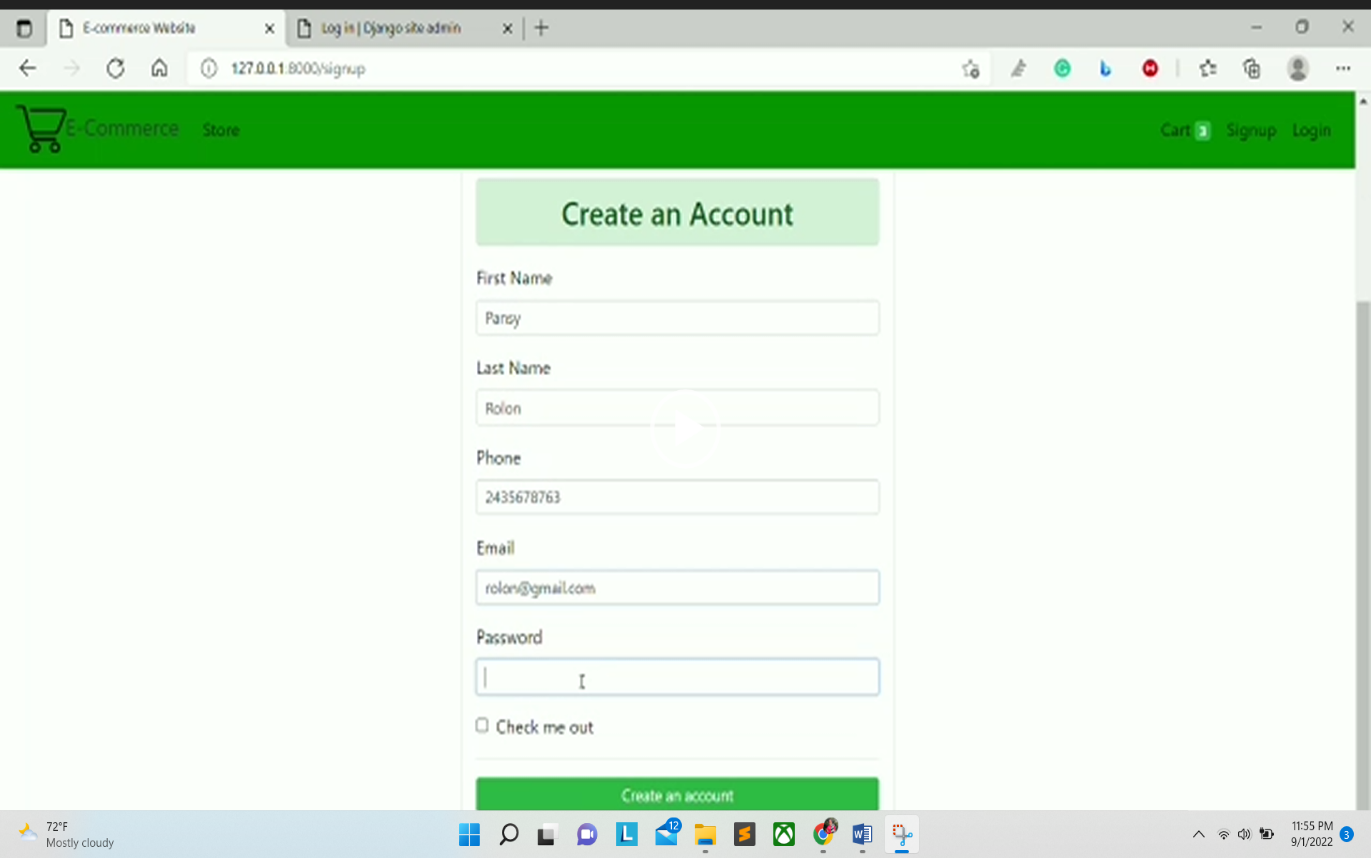
return error\_message

**CONCLUSION**

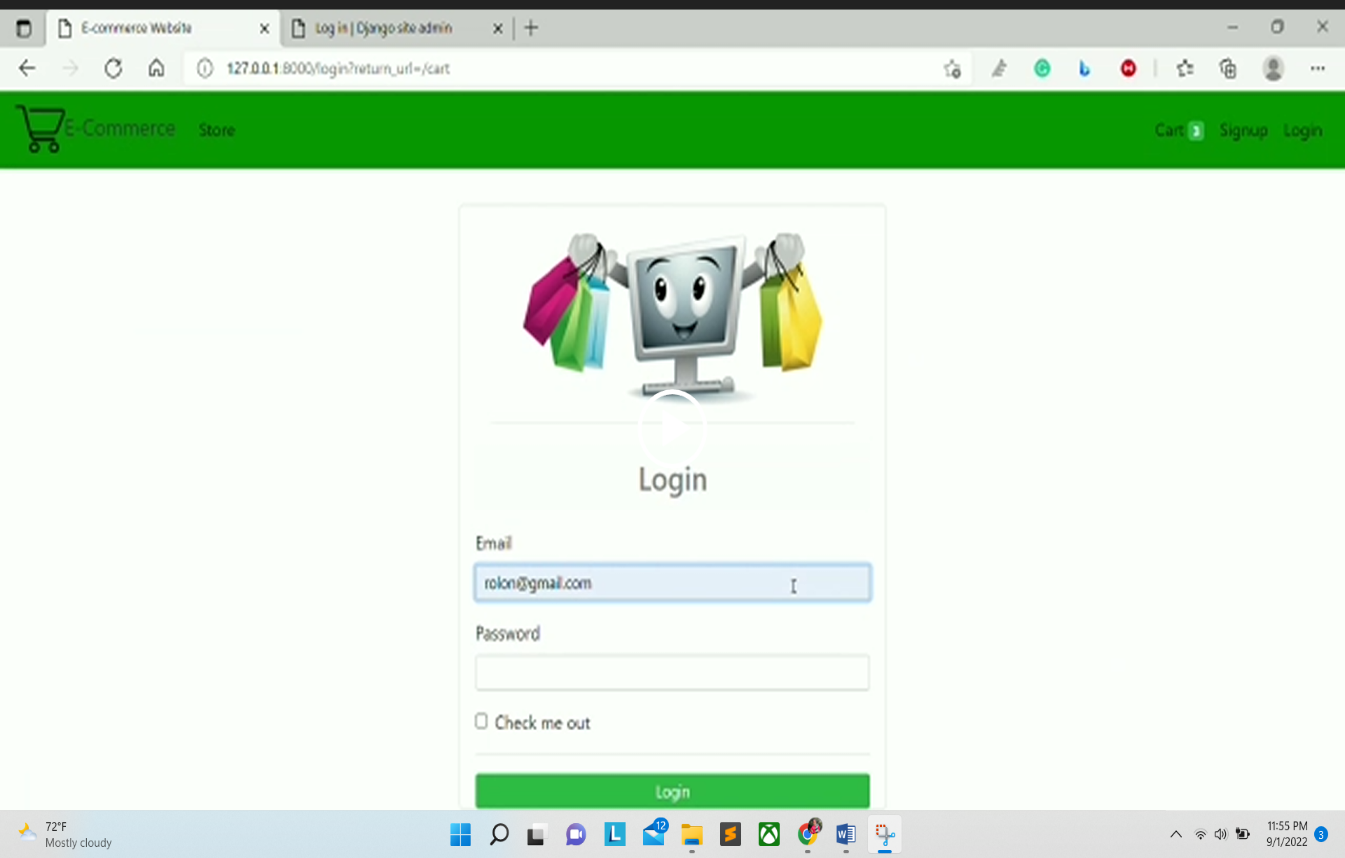
### 1.1 Output

****

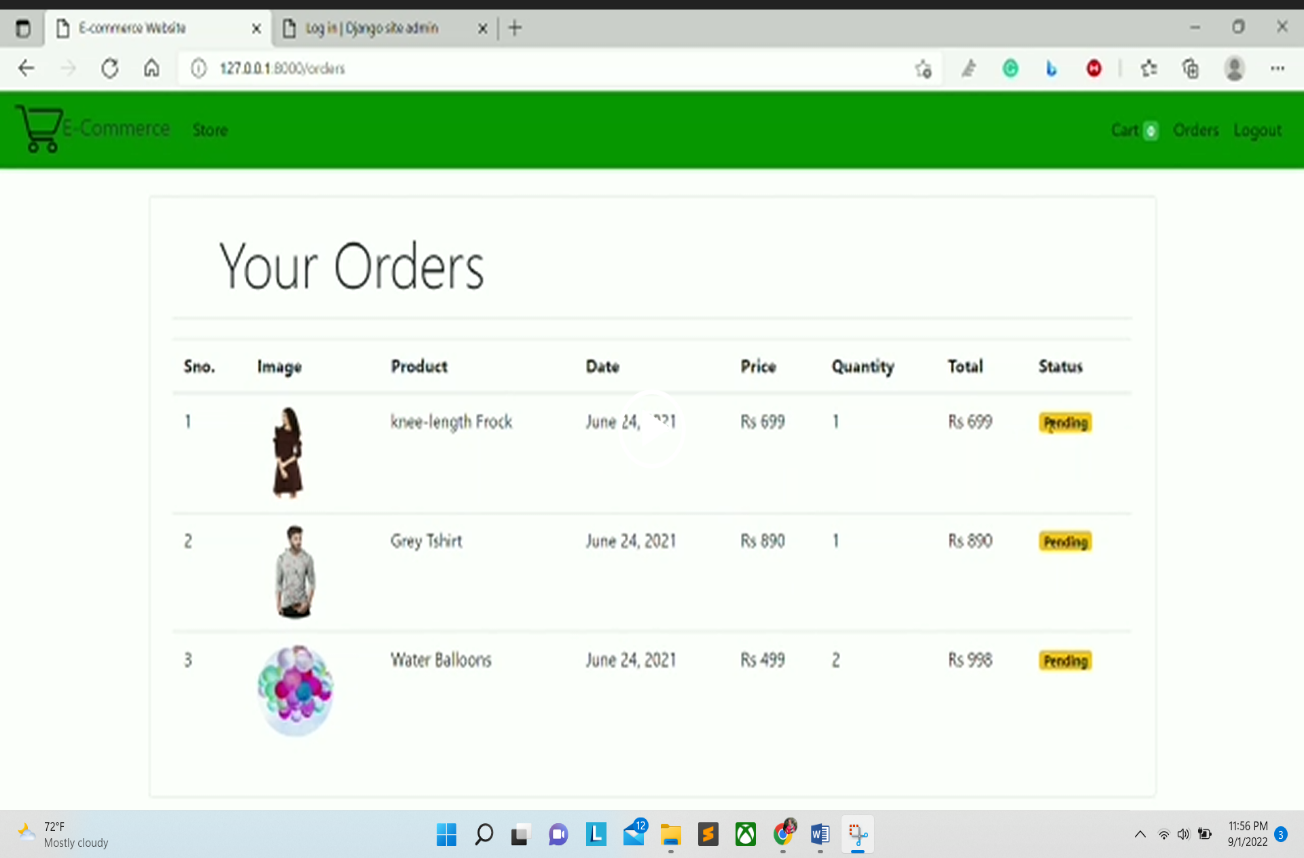
* 1. **Create Acount**

****

* 1. **Login**

****

* 1. **Your Order**

****

The project already includes a lot of features. The main beneficiaries are both customers and administrators who take longer to behave online. In addition, additional features can be identified and incorporated in the future. It will take more time and effort to understand the need and adjust it to a computerized system to accommodate additional features.