Cart folder

1) __init__.py

2) Admin.py

```
from django.contrib import admin
from .models import OrderItem
# Register your models here.
admin.site.register(OrderItem)
```

3) Apps.py

```
from django.apps import AppConfig
class CartConfig(AppConfig):
   name = 'cart'
```

4) Models.py

```
from django.db import models
from product.models import Product
from django.conf import settings

# Create your models here.
class OrderItem(models.Model):
        item = models.ForeignKey(Product, on_delete=models.CASCADE)
        quantity = models.IntegerField(default=1)
        start_date = models.DateTimeField(auto_now_add=True)
        ordered = models.BooleanField(default=False)
        user = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
```

5) Tests.py

```
from django.test import TestCase
```

6) Views.py

```
from django.shortcuts import render, get_object_or_404
from product.models import Product
from .models import OrderItem
from django.contrib import messages
from UserAuthentication import views
# Create your views here.
def cart_page(request):
   queryset = OrderItem.objects.all()

   query = request.GET.get('quantity')
```

```
    print(query)

    sum1 = 0
    for i in queryset:
        sum1 = sum1 + (i.item.price*i.quantity)

    tot_price_list = []
    for i in queryset:
        a = (i.quantity)*(i.item.price)
        tot_price_list.append(a)



    label = OrderItem.objects.all()
    no_label = len(label)
    context = {'cart_list': queryset, 'label':no_label, 'sum':sum1, 'price':tot_price_list}
    return render(request, 'cart.html', context)

def add_to_cart(request, obj_id):
    if views.authenticated == True:
        item = get_object_or_404(Product, id=obj_id)
      if len(OrderItem.objects.all()) == 0:
                        order_item = OrderItem.objects.create(item=item,
user=request.user)

        index = OrderItem.objects.all()

        found = 0

        for i in index:
            if i.item.name == item.name:
                found = 1
                break
            else:
                found = 0

      if found == 1:
            i.quantity += 1
        else:
```

```
                order_item = OrderItem.objects.create(item=item,
user=request.user)

        order_qs = OrderItem.objects.filter(user=request.user, ordered=False)
        obj = get_object_or_404(Product, id=obj_id)
      label = OrderItem.objects.all()
        no_label = len(label)
      context = {'object': obj, 'label':no_label}
        return render(request, "product_detail.html", context)
   else:
      return render(request, "logout.html")

def delete_from_cart(request, obj_id):
        queryset = OrderItem.objects.all()
        item_to_delete = OrderItem.objects.filter(id=obj_id)
        if item_to_delete.exists():
                item_to_delete[0].delete()
                messages.info(request, "Item has been deleted")
        label = OrderItem.objects.all()
        no_label = len(label)

        sum1 = 0
        for i in queryset:
                sum1 = sum1 + (i.item.price*i.quantity)

        context = {'cart_list': queryset, 'label':no_label, 'sum':sum1}
        return render(request, 'cart.html', context)

def increase_quantity(request, obj_id):
        queryset = OrderItem.objects.all()

        obj = get_object_or_404(OrderItem, id=obj_id)
        print(obj.quantity)
        obj.quantity = obj.quantity + 1
        obj.save()

        label = OrderItem.objects.all()
        no_label = len(label)
        sum1 = 0
        for i in queryset:
```

```
            sum1 = sum1 + (i.item.price*i.quantity)

        context = {'cart_list': queryset, 'label':no_label, 'sum':sum1}
        return render(request, 'cart.html', context)

def decrease_quantity(request, obj_id):
        queryset = OrderItem.objects.all()

        obj = get_object_or_404(OrderItem, id=obj_id)
        print(obj.quantity)
        if obj.quantity > 0:
            obj.quantity = obj.quantity - 1
            obj.save()

        label = OrderItem.objects.all()
        no_label = len(label)
        sum1 = 0
        for i in queryset:
            sum1 = sum1 + (i.item.price*i.quantity)

        context = {'cart_list': queryset, 'label':no_label, 'sum':sum1}
        return render(request, 'cart.html', context)
```

Categories folder:

1) __init__.py

2) Admin.py

```
from django.contrib import admin
```

3) Apps.py

```
from django.apps import AppConfig

class CategoriesConfig(AppConfig):
   name = 'categories'
```

4) Models.py

```
from django.db import models
```

5) Tests.py

```
from django.test import TestCase
```

6) Views.py

```
from django.shortcuts import render
```

```
from product.models import Product
from cart.models import OrderItem


def category_page(request, cat_no):
    queryset = Product.objects.all()
    label = OrderItem.objects.all()
    no_label = len(label)
    category_list = ['Action and adventure', 'Autobiographies', "Biographies",
"Children's", 'Crime', 'Fantasy', 'Mystery', 'Romance', 'Science fiction', 'Other']
    object_list = []
    print(cat_no)
    for i in queryset:
        if i.category == category_list[(cat_no-1)]:
            object_list.append(i)
        context = {'object_list': object_list, 'label':no_label}
    return render(request, "display_all.html", context)
```

Pages folder:

1) __init__.py

2) Admin.py

```
from django.contrib import admin
```

3) Apps.py

```
from django.apps import AppConfig
class PagesConfig(AppConfig):
    name = 'pages'
```

4) Models.py

```
from django.db import models
```

5) Tests.py

```
from django.test import TestCase
```

6) Views.py

```
from django.shortcuts import render
from django.http import HttpResponse
```

```
from product.models import Product
from django.db.models import Q
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from cart.models import OrderItem

# Create your views here.

def list_all_view(request):
    queryset_list = Product.objects.all()
    paginator = Paginator(queryset_list, 10) # Show 25 contacts per page.
    page = request.GET.get('page')
    try:
        queryset = paginator.page(page)
    except PageNotAnInteger:
        queryset = paginator.page(1)
    except EmptyPage:
        queryset = paginator.page(paginator.num_pages)

    page_obj = paginator.get_page(page)
    #return render(request, 'list.html', {'page_obj': page_obj})
    context = {'page_obj': page_obj, 'object_list': queryset}
    return render(request, "display_all.html", context)

def home_page(request, *args, **kwargs):
    queryset_list = Product.objects.all()
    label = OrderItem.objects.all()
    no_label = len(label)
    paginator = Paginator(queryset_list, 20)
    page = request.GET.get('page')
    try:
```

```
        queryset = paginator.page(page)
    except PageNotAnInteger:
        queryset = paginator.page(1)
    except EmptyPage:
        queryset = paginator.page(paginator.num_pages)

    page_obj = paginator.get_page(page)

    context = {'object_list': queryset, 'page_obj': page_obj, 'label': no_label }

    return render(request, "home.html", context)

def about_page(request, *args, **kwargs):
    label = OrderItem.objects.all()
    no_label = len(label)
    return render(request, "about.html", {'label': no_label })

def logout_page(request, *args, **kwargs):
    label = OrderItem.objects.all()
    no_label = len(label)
    return render(request, "logout.html", {'label': no_label })

def Categories_page(request, *args, **kwargs):

    return render(request, "categories.html", {})

def cart_page(request, *args, **kwargs):
    label = OrderItem.objects.all()
    no_label = len(label)
    return render(request, "cart.html", {'label': no_label })
```

```
def already_logged_in_page(request, *args, **kwargs):
    return render(request, "already_logged_in.html", {})


def login_page(request, *args, **kwargs):
    label = OrderItem.objects.all()
    no_label = len(label)
    return render(request, "UserAuthentication/templates/login.html", {'label':
no_label })
```

Product folder:

1) __init__.py

2) Admin.py

```
from django.contrib import admin
from .models import Product
admin.site.register(Product)
```

3) Apps.py

```
from django.apps import AppConfig
class ProductConfig(AppConfig):
    name = 'product'
```

4) Models.py

```
from django.db import models
# Create your models here.
class Product(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=10, decimal_places=2,
default=50.00)
    author = models.CharField(max_length=50, default='Anonymous')
    description = models.TextField(default='Hello')
```

```
    publisher = models.CharField(max_length=100, default='random')
    img = models.ImageField(default="D:\\omkar\\midoriya.jpg")
    img_location = models.CharField(max_length=255, default="Doesnt exist")
    category = models.CharField(max_length=200, default='Other')
    in_cart = models.BooleanField(default=False)
```

5) Tests.py

```
from django.test import TestCase
```

6) Views.py

```
from django.shortcuts import render, get_object_or_404
from .models import Product
from cart.models import OrderItem
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from django.db.models import Q

label = OrderItem.objects.all()
no_label = len(label)

def dynamic_product_view(request, my_id):
    label = OrderItem.objects.all()
    no_label = len(label)
    obj = get_object_or_404(Product, id=my_id)
    context = {'object': obj, 'label':no_label}
    return render(request, "product_detail.html", context)


def list(request):
    queryset = Product.objects.all()
    context = {'object_list': queryset}
```

```
def list_all_view(request):
    label = OrderItem.objects.all()
    no_label = len(label)
    queryset_list = Product.objects.all()
    paginator = Paginator(queryset_list, 10) # Show 25 contacts per page.
    page = request.GET.get('page')
    try:
        queryset = paginator.page(page)
    except PageNotAnInteger:
        queryset = paginator.page(1)
    except EmptyPage:
        queryset = paginator.page(paginator.num_pages)

    page_obj = paginator.get_page(page)
    #return render(request, 'list.html', {'page_obj': page_obj})
    context = {'page_obj': page_obj, 'object_list': queryset, 'label':no_label}
    return render(request, "display_all.html", context)


def search(request):
    queryset_list = Product.objects.all()
    print(queryset_list)
    query = request.GET.get('search')
    print(query)
    All_Prods = []
    for i in queryset_list:
        if (query in i.name) or (query in i.author) or (query in i.author.lower()) or
(query in i.name.lower()):
            All_Prods.append(i)


    print(All_Prods)
```

```
    context = {'object_list': All_Prods, 'label':no_label}
    return render(request, "search.html", context)
```

Projectone folder:

1) __init__.py
2) Asgi.py

```
"""
ASGI config for projectone project.
It exposes the ASGI callable as a module-level variable named ``application``.
For more information on this file, see
https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/
"""
import os
from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'projectone.settings')
application = get_asgi_application()
```

3) Settings.py

```
import os
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = '^b!8c=@30e0wx-&4tn^u^hh(+34-dp!s+)jrt%me_a8ch-l&8q'
DEBUG = True

ALLOWED_HOSTS = []

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
STATIC_URL = '/static/'
# Application definition

INSTALLED_APPS = [
```

```
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'product',
    'pages',
    'categories',
    'cart'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'projectone.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, "Templates")],
        'APP_DIRS': True,
        'OPTIONS': {
```

```
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'projectone.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
```

```
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

STATIC_URL = '/static/'

from django.contrib.messages import constants as messages

MESSAGE_TAGS = {
    messages.DEBUG: 'alert-info',
    messages.INFO: 'alert-info',
    messages.SUCCESS: 'alert-success',
    messages.WARNING: 'alert-warning',
    messages.ERROR: 'alert-danger',
}
```

4) Urls.py

```
from django.contrib import admin
from django.urls import path
```

```
from pages.views import home_page
from pages.views import about_page
from pages.views import logout_page
from pages.views import Categories_page
from pages.views import already_logged_in_page
from product.views import dynamic_product_view, list_all_view, search
from categories.views import category_page
from UserAuthentication import views as user_views
from cart.views import cart_page, add_to_cart, delete_from_cart,
increase_quantity, decrease_quantity


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', home_page, name='home'),
    path('home/about/', about_page, name='About'),
    path('home/logout/', user_views.logout, name='logout'),
    path('home/categories/', Categories_page, name='Categories'),
    path('home/cart/', cart_page, name='cart'),
    path('product/<int:my_id>/', dynamic_product_view, name='Product'),
    path('all_products/', list_all_view, name='All Products'),
    path('already_logged_in/', already_logged_in_page, name='Sign in'),
    path('search/', search, name='Search'),
    path('categories/<int:cat_no>/', category_page, name='Category'),
    path('register/', user_views.register),
    path('login/', user_views.login),
    path('home/', home_page),
    path('cart/', cart_page, name='cart'),
    path('add_cart/<int:obj_id>/', add_to_cart, name='cart'),
    path('cart/<int:obj_id>/', delete_from_cart, name='cart'),
    path('cart_q/<int:obj_id>/', increase_quantity, name='cart'),
```

```
    path('cart_qd/<int:obj_id>/', decrease_quantity, name='cart'),
]
```

5) Wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'projectone.settings')
application = get_wsgi_application()
```

UserAuthentication folder:

1) __init.py

2) Admin.py

```
from django.contrib import admin
from django.contrib.admin.models import LogEntry
LogEntry.objects.all().delete()
```

3) Apps.py

```
from django.apps import AppConfig
class UserauthenticationConfig(AppConfig):
    name = 'UserAuthentication'
```

4) Forms.py

```
from django import forms

class RegisterForm(forms.Form):
    username = forms.CharField(max_length=200)
    password = forms.CharField(max_length=32)
    email = forms.EmailField()
    password_confirm = forms.CharField(max_length=32)

class LoginForm(forms.Form):
    username = forms.CharField(max_length=200)
    password = forms.CharField(max_length=32)
```

5) Models.py

```
from django.db import models
```

6) Tests.py

```
from django.test import TestCase
```

7) Views.py

```
from django.shortcuts import render, redirect

from .forms import RegisterForm,LoginForm

from django.contrib.auth import authenticate

from django.contrib.auth import login as auth_login

from django.contrib.auth import logout as auth_logout

from django.contrib import messages

from pages import views as pages_views

from django.contrib.auth.models import User


authenticated = False

# Create your views here.

def register(request):

   form = RegisterForm(request.POST)

   if form.is_valid() and form.cleaned_data['password']==
form.cleaned_data['password_confirm'] :

      username = form.cleaned_data['username']

      email = form.cleaned_data['email']

      password = form.cleaned_data['password']

      user = User.objects.create_user(username,email,password)

      user.save()

      messages.success(request, 'Your account was created successfully!')

      return redirect('/login')

   return render(request, 'register.html', {'form': form})



def login(request):
```

```
    global authenticated
    form = LoginForm(request.POST)
    if authenticated == False:
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            u = authenticate(username= username, password=password)
            if u is not None:
                authenticated = True
                auth_login(request,u)
                request.session['username'] = username
                return redirect('/home')
            else:
                messages.warning(request, 'Please check your username and
password!')
    else:
        return redirect('/home')
        messages.warning(request, 'You are already logged in!')

    return render(request, 'login.html', {'form': form})

def home(request):
    if request.session.has_key('username'):
      username = request.session['username']
      return render(request, 'home.html')
    else:
      return render(request, 'home.html')
def logout(request):
    auth_logout(request)
    try:
```

```
    del request.session['username']
  except:
    pass
  return redirect('/login')
```

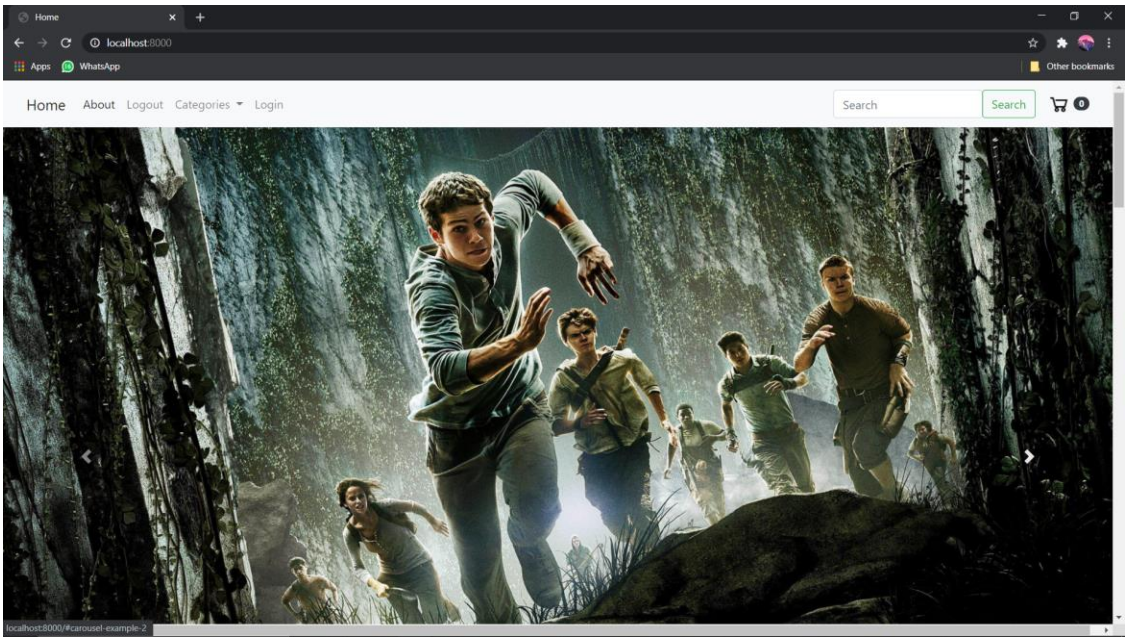
Static folder

1) Projectone folder
   a) Images folder
      All the images that are to be used in the templates are to be stored here.

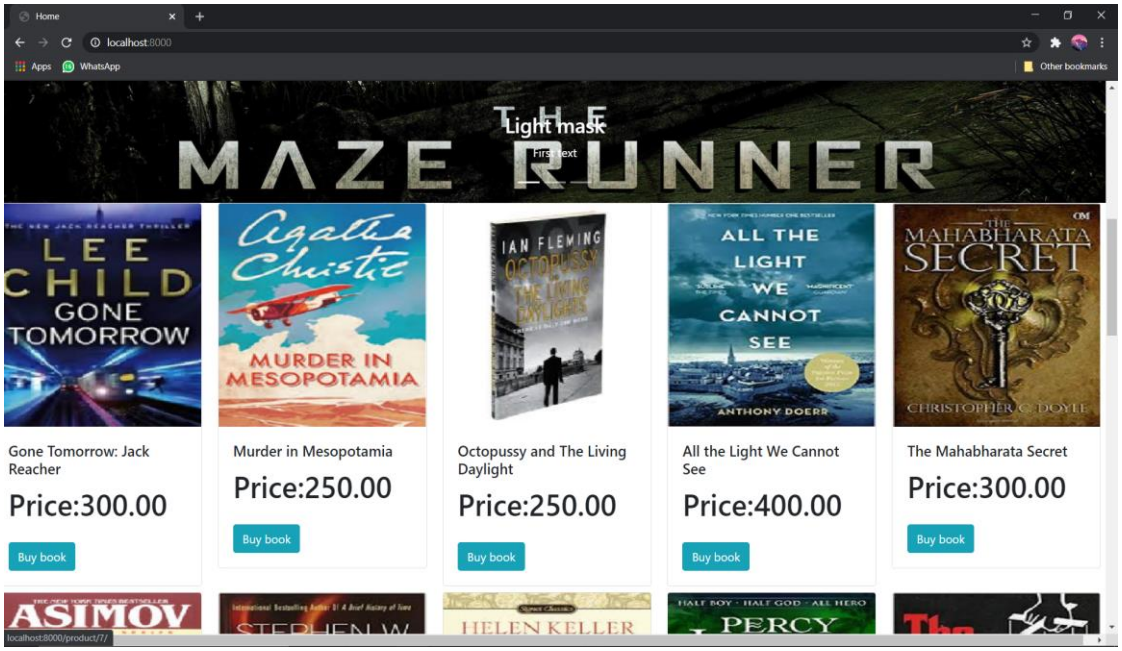
Templates folder

All the HTML files that are to be used in the programs are created here.

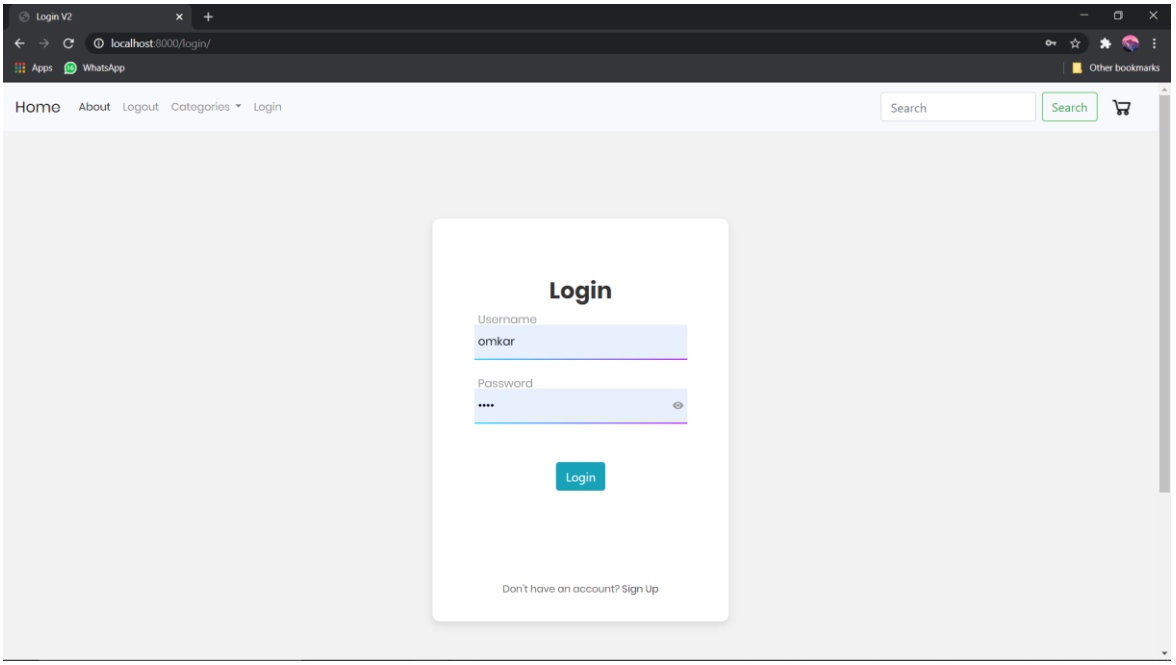
Screenshots



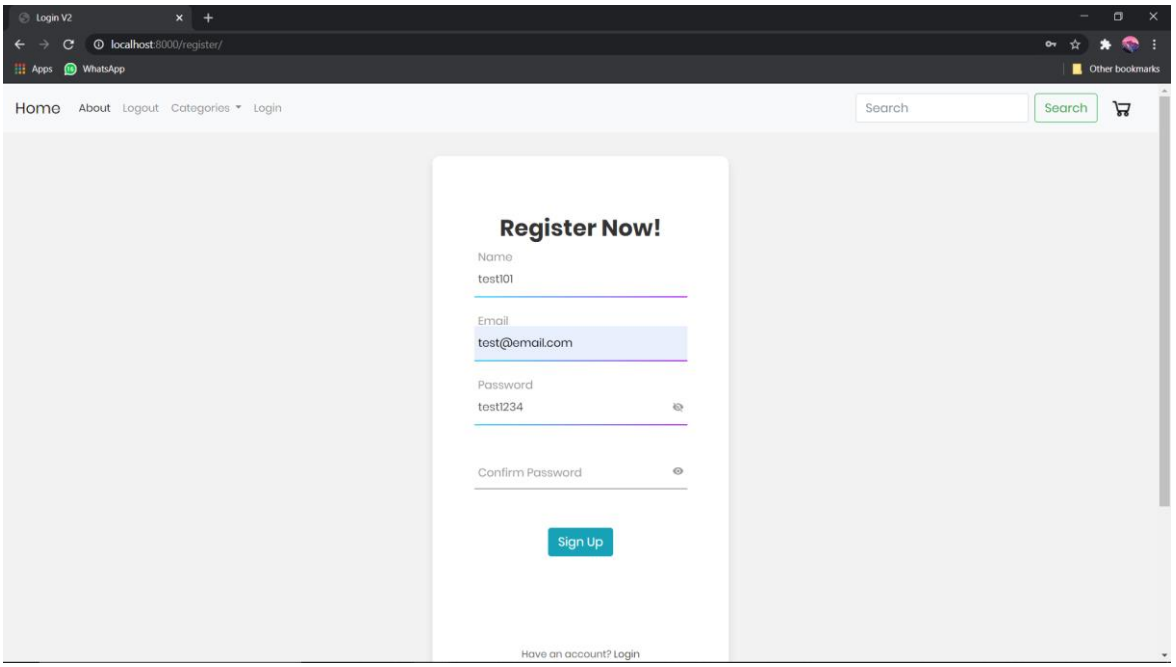
Home page



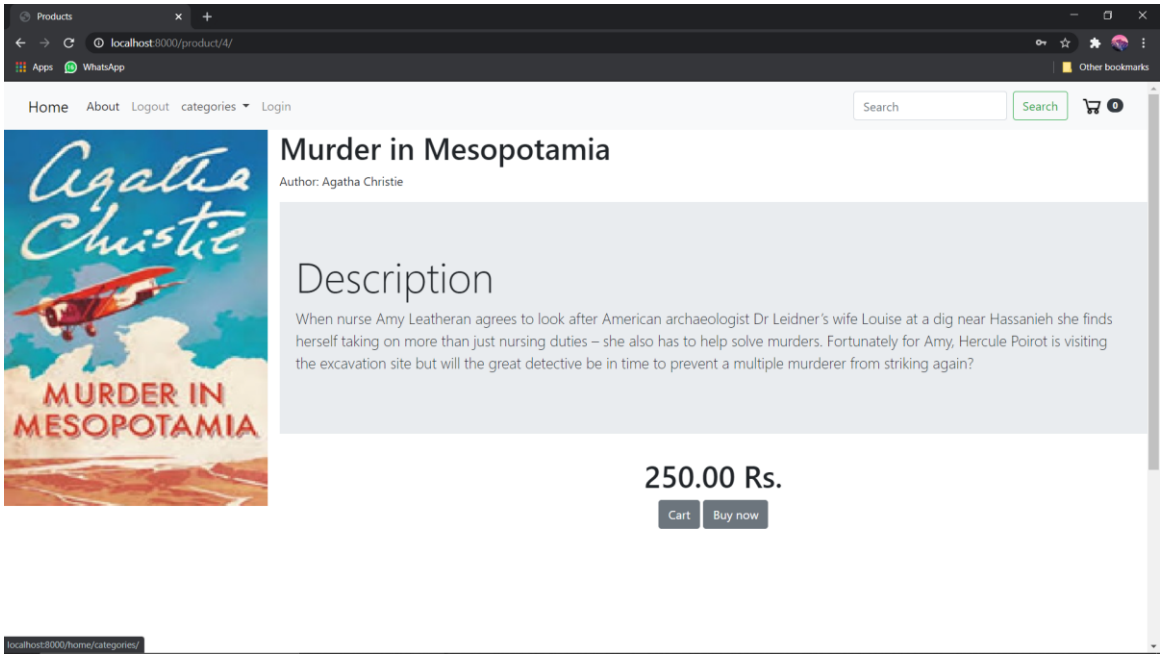
Homepage(2)



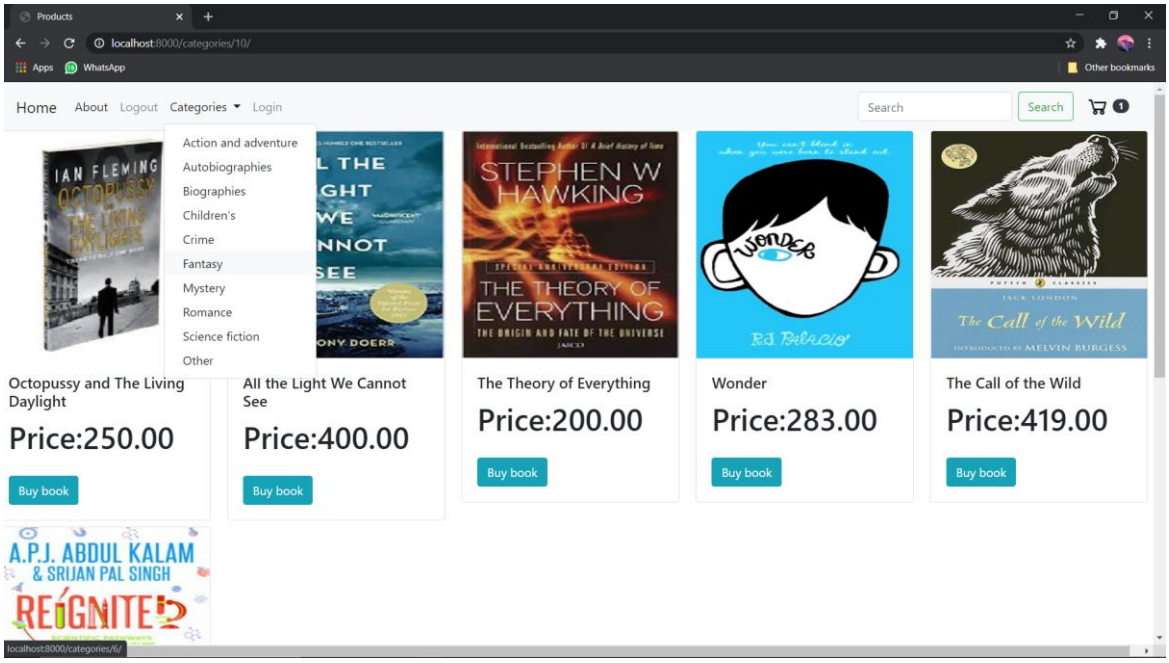
Login page



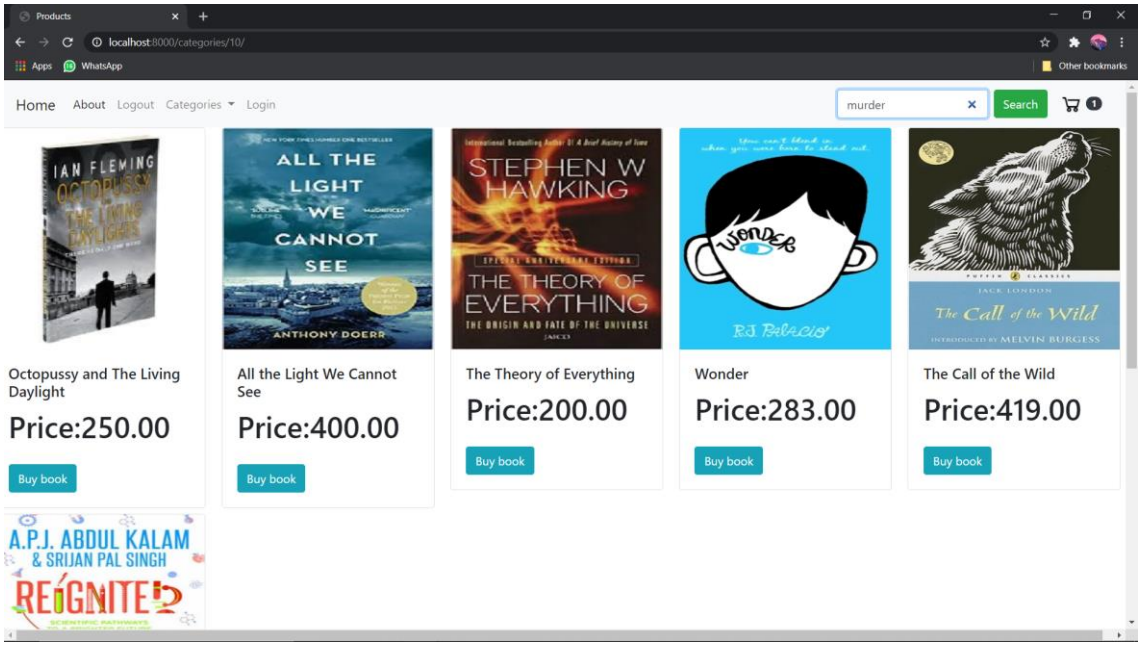
Registration page



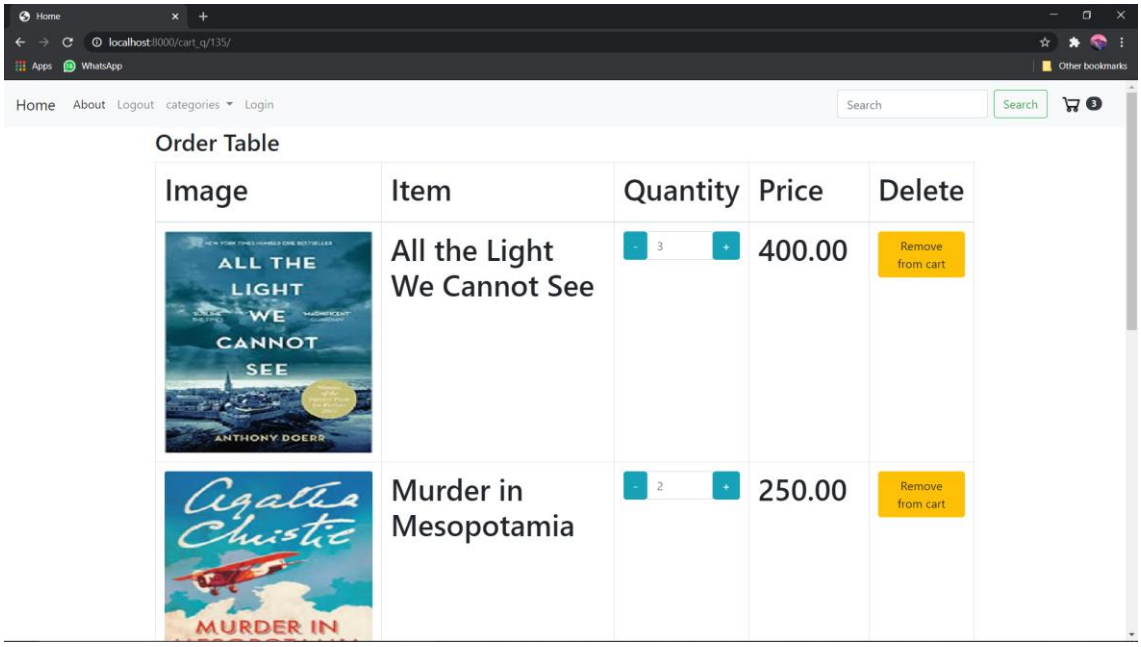
Product View



Categories


Products
localhost:8000/categories/10/
Apps
WhatsApp
Other bookmarks
Home
About
Logout
Categories
Login
murder
Search
Octopussy and The Living Daylight
Price:250.00
Buy book
All the Light We Cannot See
Price:400.00
Buy book
The Theory of Everything
Price:200.00
Buy book
Wonder
Price:283.00
Buy book
The Call of the Wild
Price:419.00
Buy book

Search


Home
localhost:8000/cart_q/135/
Apps
WhatsApp
Other bookmarks
Home
About
Logout
categories
Login
Search
Search
Order Table
Image
Item
Quantity
Price
Delete
All the Light We Cannot See
3
400.00
Remove from cart
Murder in Mesopotamia
2
250.00
Remove from cart

Cart

Future enhancements

- A payment gateway can be created
- A larger variety of books can be added.
- The User Interface can be made for friendly
- Reviews can be added for each book
- More pictures can also be added for each book

Bibliography

1. https://en.wikipedia.org/wiki/E-commerce
2. https://www.djangoproject.com/
3. https://en.wikipedia.org/wiki/SQL