"Book Store Management System – Python Django Project

Create a simple book store management system project using python django

Project Prerequisites

We will use the following technologies:

Front-end technologies:

HTML

CSS

Bootstrap

Back-end technologies:

Python

Django framework

To install the django, you can use the pip installer from cmd/terminal.

pip install Django

Download Book Store Management System Python Code

Please download the source code of book store project: Book Store Management System Code

Steps to Build the Book Store Project

To start the work on book store project, please run below commands in cmd/terminal

django-admin startproject bookstore

cd bookstore

django-admin startapp book

1. Writing Models

We are using sqlite3 database (the default database in Django). So we just have to define the classes in models.py file and the tables will be automatically created.

Code:

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Book(models.Model):

title=models.CharField(max\_length=200,null=True)

Author=models.CharField(max\_length=200,null=True)

Price=models.IntegerField()

Edition=models.IntegerField()

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

return str(self.title)

class Customer(models.Model):

user = models.OneToOneField(User, null=True, on\_delete=models.CASCADE)

name=models.CharField(max\_length=200,null=True)

phone=models.CharField(max\_length=200,null=True)

email=models.CharField(max\_length=200,null=True)

date\_created=models.DateTimeField(auto\_now\_add=True,null=True)

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

return str(self.name)

class Cart(models.Model):

customer=models.OneToOneField(Customer, null=True, on\_delete=models.CASCADE)

books=models.ManyToManyField(Book)

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

return str(self.customer)

Explanation:

Book: This model stores the information related to books like author, title, price, and edition.

Customer: This model stores customer’s name, phone number, email, and date.

Cart: This stores all the books added by a particular customer in his/her cart.

To create tables, run the following commands on cmd/terminal

Py manage.py makemigrations

Py manage.py migrate

2. forms.py

Code:

from django.forms import ModelForm

from .models import \*

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

class createuserform(UserCreationForm):

class Meta:

model=User

fields=['username','password']

class createcustomerform(ModelForm):

class Meta:

model=Customer

fields='\_\_all\_\_'

exclude=['user']

class createbookform(ModelForm):

class Meta:

model=Book

fields='\_\_all\_\_'

This code is simple because django does it all for us, we are just importing and calling the inbuilt functions in book store project.

3. admin.py

To access, add, delete or modify database entries using admin panel, add the following code in admin.py

Code:

from django.contrib import admin

from .models import \*

# Register your models here.

admin.site.register(Book)

admin.site.register(Cart)

admin.site.register(Customer)

We also need a superuser to access the admin panel. Command to create superuser

py manage.py createsuperuser

It will ask for the superuser name, email and password.

4. urls.py

We will create separate urls.py for each app. Thus, we have to include the new urls.py in the book store project’s urls.py.

Code:

"""bookstore URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/3.1/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

path('',include('book.urls')),

]

book/urls.py:

Code:

from django.urls import path

from django.contrib.auth import views as auth\_views

from .views import \*

urlpatterns =[

path('', home,name='home'),

path('login/', loginPage,name='login'),

path('viewcart/', viewcart,name='viewcart'),

path('addbook/', addbook,name='addbook'),

path('register/', registerPage,name='register'),

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

path('addtocart/<str:pk>', addtocart,name='addtocart'),

]

5. views.py:

Code:

from django.shortcuts import render,redirect

from django.http import HttpResponse

from .models import \*

from django.contrib.auth import login,logout,authenticate

from .forms import \*

# Create your views here.

def home(request):

books=Book.objects.all()

context={'books':books}

if request.user.is\_staff:

return render(request,'book/adminhome.html',context)

else:

return render(request,'book/home.html',context)

def logoutPage(request):

logout(request)

return redirect('/')

def loginPage(request):

if request.user.is\_authenticated:

return redirect('home')

else:

if request.method=="POST":

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

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

user=authenticate(request,username=username,password=password)

if user is not None:

print("working")

login(request,user)

return redirect('/')

context={}

return render(request,'book/login.html',context)

def registerPage(request):

form=createuserform()

cust\_form=createcustomerform()

if request.method=='POST':

form=createuserform(request.POST)

cust\_form=createcustomerform(request.POST)

if form.is\_valid() and cust\_form.is\_valid():

user=form.save()

customer=cust\_form.save(commit=False)

customer.user=user

customer.save()

return redirect('login')

context={

'form':form,

'cust\_form':cust\_form,

}

return render(request,'book/register.html',context)

def addbook(request):

form=createbookform()

if request.method=='POST':

form=createbookform(request.POST)

if form.is\_valid():

form.save()

return redirect('/')

context={'form':form}

return render(request,'book/addbook.html',context)

def viewcart(request):

cust=Customer.objects.filter(user=request.user)

for c in cust:

carts=Cart.objects.all()

for cart in carts:

if(cart.customer==c):

context={

'cart':cart

}

return render(request,'book/viewcart.html',context)

else:

return render(request,'book/emptycart.html')

def addtocart(request,pk):

book=Book.objects.get(id=pk)

cust=Customer.objects.filter(user=request.user)

for c in cust:

carts=Cart.objects.all()

reqcart=''

for cart in carts:

if(cart.customer==c):

reqcart=cart

break

if(reqcart==''):

reqcart=Cart.objects.create(

customer=c,

)

reqcart.books.add(book)

return redirect('/')

We have created seven views in this file:

1. Home

This is the home page for both admin and customer of book store management system. It renders all the ‘book’ objects to templates which displays all the details of the books based on the access rights of the user.

Admin’s home page:

The navbar is also different for both the users. For customers, the view cart option is available and for admin add books option is present. Also the add to cart button is only for customers.

2. Logout:

This view provides the logout functionality.

3. Login:

It first checks whether the user is authenticated or not. If the user is authenticated then it redirects to the home page otherwise it will check the user credentials and then after verifying it will change the state to logged in.

4. Register:

It is very similar to login view. It just creates a new Customer and then it redirects to the login page.

5. Addbook:

This view is just for admin. We are using the createbookform() here which we have already discussed in forms.py section. After adding a book in the database, it redirects to the home page.

6. Viewcart:

It is a customer view. Here we are just rendering the cart of the current user to viewcart.html which displays all the books in the cart. It also checks whether the cart is empty.

7.Addtocart:

This is also a customer view. If the cart of the current user is empty, it will create a new cart for the user. After that it adds the requested book in the cart.

6. templates

Addbook.html

{% extends 'book/main.html' %}

{% load static%}

{% block content %}

<div class="container jumbotron">

<form method="POST" action="" >

{% csrf\_token %}

{{form.as\_p}}

<input class="btn btn-success" type="submit" value="Add book">

</form>

</div>

{% endblock %}

Emptycart.html

{% extends 'book/main.html' %}

{% load static%}

{% block content %}

<div class="container jumbotron">

<h1>Your cart is empty!!!</h1>

</div>

{% endblock %}

Home.html

{% extends 'book/main.html' %}

{% block content %}

<br>

<div class="container">

<div class="container row">

<div class="col-lg-10">

<h3>BOOKS:</h3>

<div class="card card-body">

<table class="table table-sm">

<tr>

<th><Title></Title></th>

<th>Author</th>

<th>Price</th>

<th>Edition</th>

<th></th>

</tr>

{% for book in books %}

<tr>

<td><b>{{book.title}}</b> </td>

<td>{{book.Author}} </td>

<td>{{book.Price}} </td>

<td> {{book.Edition}} </td>

<td><a class="btn btn-sm btn-info" href="{% url 'addtocart' book.id %}">Add to cart</a> </td>

</tr>

{% endfor %}

</table>

</div>

</div>

</div>

{% endblock %}

Login.html

{% extends 'book/main.html' %}

{% load static%}

{% block content %}

<div class="container jumbotron">

<form method="POST" action="">

{% csrf\_token %}

<p><input type="text" name="username" placeholder="Username..."></p>

<p><input type="password" name="password" placeholder="Password..." ></p>

<input class="btn btn-success" type="submit" value="Login">

<p>Do not have an account<a href='{% url 'register' %}'>Register</a></p>

</form>

</div>

{% endblock %}

Main.html

{% load static %}

<html>

<head>

<title>

Bookstore Management System

</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">

</head>

<body>

{% include 'book/navbar.html' %}

{% block content %}

{% endblock %}

<br>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" crossorigin="anonymous"></script>

</body>

</html>

Navbar.html

{% load static %}

<style>

.hello-msg{

font-size: 18px;

color: #fff;

margin-right: 20px;

}

</style>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNav">

<ul class="navbar-nav">

<li class="nav-item active">

<a class="nav-link" href="{% url 'home' %}">Home</a>

</li>

{% if request.user.is\_staff %}

</li>

<li class="nav-item active">

<a class="nav-link" href="{% url 'addbook' %}">Add Books</a>

</li>

{% else %}

<li class="nav-item">

<a class="nav-link" href="{% url 'viewcart' %}">Cart</a>

</li>

{% endif %}

<li class="nav-item">

<a class="nav-link" href="{% url 'login' %}">Login</a>

</li>

</ul>

</div>

<span class="hello-msg">Hello, {{request.user}}</span>

<span ><a class="hello-msg" href="{% url 'logout' %}">Logout</a></span>

</nav>

Register.html

{% extends 'book/main.html' %}

{% load static%}

{% block content %}

<div class="container jumbotron">

<form method="POST" action="" >

{% csrf\_token %}

{{form.as\_p}}

{{cust\_form.as\_p}}

<input class="btn btn-success" type="submit" value="Register Account">

</form>

</div>

{% endblock %}

Viewcart.html

{% extends 'book/main.html' %}

{% load static%}

{% block content %}

<div class="container">

<div class="row">

<div class="col-lg-10">

<br>

<h5>BOOKS:</h5>

<div class="card card-body">

<table class="table table-sm">

<tr>

<th>Title</th>

<th>Author</th>

<th>Price</th>

<th>Edition</th>

</tr>

{% for book in cart.books.all %}

<tr>

<td>{{book.title}} </td>

<td>{{book.Author}} </td>

<td>{{book.Price}} </td>

<td>{{book.Edition}} </td>

</tr>

{% endfor %}

</table>

</div>

</div>

</div>

</div>

{% endblock %}"