**Developing an Inventory management system with Django and bootstrap.**

14 min read

**Table of contents**

[Introduction.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-introduction)

[Environment setup.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-environment-setup)

[Installing python](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-installing-python)

[Installing git](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-installing-git)

[Installing Django.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-installing-django)

[Structure of a Django application.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-structure-of-a-django-application)

[Creating the Inventory app](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-the-inventory-app)

[A project vs an app](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-a-project-vs-an-app)

[Creating the models.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-the-models)

[Creating forms from models.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-forms-from-models)

[Creating views.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-views)

[Adding the inventory/urls.py to IMS/urls.py](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-adding-the-inventoryurlspy-to-imsurlspy)

[Creating the admin.py file](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-the-adminpy-file)

[Creating migrations](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-migrations)

[Creating a super user.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-creating-a-super-user)

[Building the interface.](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-building-the-interface)

[Additional settings](https://mwass.hashnode.dev/developing-an-inventory-management-system-with-django-and-bootstrap" \l "heading-additional-settings)

**Introduction.**

**A full working project code is hosted on this**[**repository**](https://github.com/mwass/inventory)

Inventory => Inventory, also known as stock, refers to goods a business owns, with the aim to sell or use them in production.

Inventory management system => Inventory management software is a computer system that helps manage inventory, from keeping stock levels to issuing and receiving stock, among many other tasks.

Django => Django is a high-level web framework developed using python. Django comes preloaded with standard tools to handle common web development, such as user management, admin site, content management, etc., making it ideal for developing web systems with minimal effort.

Bootstrap => Bootstrap is a popular CSS framework for building beautiful and responsive web pages.

**Environment setup.**

**Installing python**

To begin with, you'll need to have a working python installation on your computer. Go to <https://www.python.org/downloads/> and get it for your PC/Mac. The instructions to install it are available on the official python website.

**Installing git**

Git is a popular version control system. It keeps track of changes you make to your code, which is handy when you want to revert to a previous version (mostly after a mess-up). Git also helps us push our code to online code hosting services such as Github and Gitlab, which allow developers to collaborate on a project.

In addition to keeping track of changes in our code, git on Windows allows you to use commands such as ls, execute bash scripts, etc., making development easy.

You can get git at <https://git-scm.com/downloads>. Instructions for installation are on the site as well.

**Installing Django.**

Once you have installed python and git, create a folder named inventory-management-system anywhere on your machine. Change to the inventory folder, then right-click and select git bash here. A git bash will open in the inventory folder.

1. Create a virtual environment:-

Type python -m venv env. This command creates a virtual environment env where we will install the project's dependencies.

1. Activate the virtual environment:-

On Windows

$ env\Scripts\activate

On Linux/Mac

$ source /env/bin/activate

1. Install Django:-

pip install django~=4.0.0

1. Install Django crispy form, which integrates Django nicely with Bootstrap

$ pip install django-crispy-forms~=2.0

$ pip install crispy-bootstrap5~=0.7

We can start developing our system now that we have installed all the required packages.

**Structure of a Django application.**

Django has tools to help the developer manage tasks such as generating files, interacting with the ORM, running the development server, etc.

To generate our project, we will use the django-admin command as follows:

$ django-admin startproject IMS

The above command will create a Django project called inventory with the necessary python files, as shown below.



The \_\_init\_\_.py indicates that the directory is a python package.

The asgi.py file is used to configure Django to run using the ASGI, which is an emerging for asynchronous web servers, by default django uses WSGI.

You can read more about asgi here, <https://asgi.readthedocs.io/en/latest/>

The settings.py contains setting for your django project.

The urls.py contain the URL declarations for your project.

The wsgi.py file is used as entry point to web servers such as Nginx and Apache.

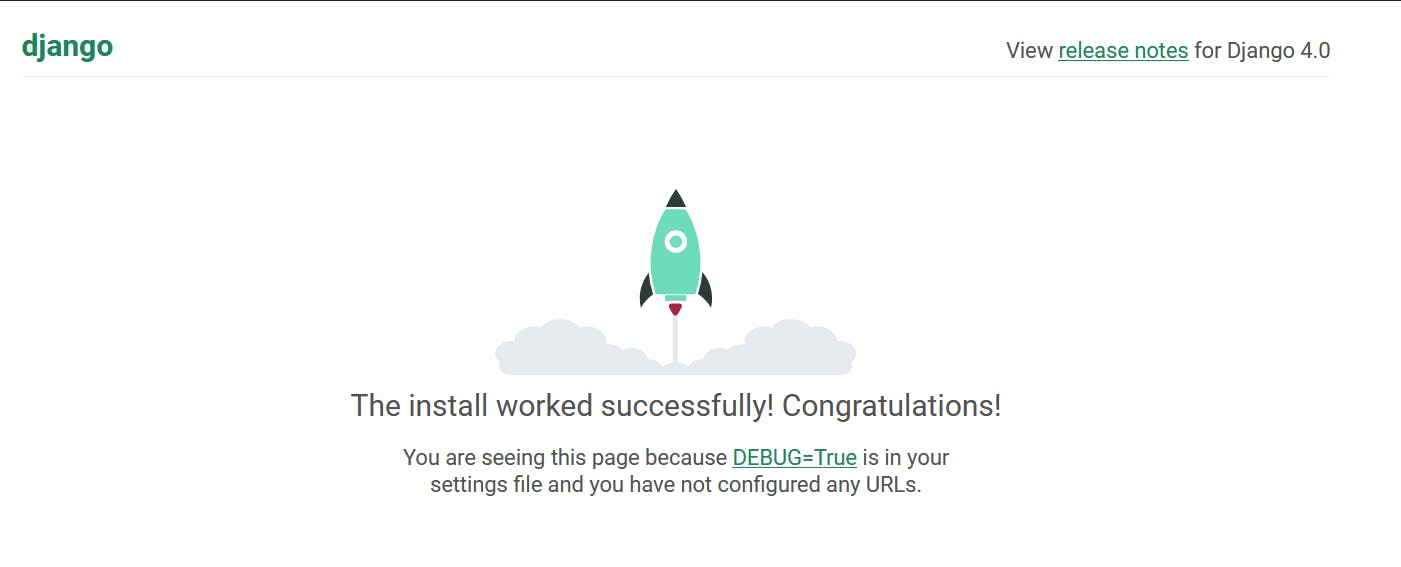
You'll also note that there is a manage.py file, this is a command line utility that helps you interact with your django project.

Now that everything is fully setup, change directory to the IMS directory.

$ cd IMS

To run the development server to test out our application, type the command below, then visit 127.0.0.1:8000 on your browser.

$ python manage.py runserver



You should see a cool site similar to the one above.

**Creating the Inventory app**

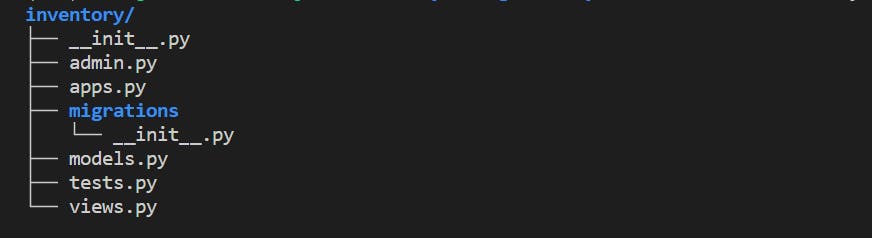
Now that our project setup is complete, let us create the inventory app.

$ python manage.py startapp inventory

**A project vs an app**

As per the django documentation *"An app is a web application that does something – e.g., a blog system, a database of public records or a small poll app. A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects."*

The command generates an app with the following structure.



**Creating the models.**

*"A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you’re storing. Generally, each model maps to a single database table.",* Django documentation.

Django models are written in the [models.py](http://models.py/) file, we will create four models for our project, namely UserRegistry, ProductForm and OrderForm.

Copy

Copy

# inventory/models.py

from email.policy import default

from django.db import models

from django.contrib.auth.models import User

CATEGORY = (

("Stationary", "Stationary"),

("Electronics", "Electronics"),

("Food", "Food"),

)

class UserProfile(models.Model):

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

physical\_address = models.CharField(max\_length=40, null=True)

mobile = models.CharField(max\_length=12, null=True)

picture = models.ImageField(default="avatar.jpeg", upload\_to="Pictures")

def \_\_str\_\_(self) -> str:

return self.user.username

class Product(models.Model):

name = models.CharField(max\_length=100, null=True)

category = models.CharField(max\_length=20, choices=CATEGORY, null=True)

quantity = models.PositiveIntegerField(null=True)

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

def \_\_str\_\_(self) -> str:

return self.name

class Order(models.Model):

product = models.ForeignKey(Product, on\_delete=models.CASCADE, null=True)

created\_by = models.ForeignKey(User, models.CASCADE, null=True)

order\_quantity = models.PositiveIntegerField(null=True)

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

def \_\_str\_\_(self) -> str:

return f"{self.product} ordered quantity {self.order\_quantity}"

**Creating forms from models.**

*"If you’re building a database-driven app, chances are you’ll have forms that map closely to Django models. For instance, you might have a BlogComment model, and you want to create a form that lets people submit comments. In this case, it would be redundant to define the field types in your form, because you’ve already defined the fields in your model.",* Django documentation.

Create the forms.py file inside the inventory folder and add the following code.

Copy

Copy

# inventory/forms.py

from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

from inventory.models import Product, Order

class UserRegistry(UserCreationForm):

email = forms.EmailField()

class Meta:

model = User

fields = [

"username",

"first\_name",

"last\_name",

"email",

"password1",

"password2",

]

class ProductForm(forms.ModelForm):

class Meta:

model = Product

fields = ["name", "category", "quantity", "description"]

class OrderForm(forms.ModelForm):

class Meta:

model = Order

fields = ["product", "order\_quantity"]

**Creating views.**

*"A view function, or view for short, is a Python function that takes a web request and returns a web response. This response can be the HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image . . . or anything, really. The view itself contains whatever arbitrary logic is necessary to return that response. This code can live anywhere you want, as long as it’s on your Python path. There’s no other requirement–no “magic,” so to speak."*, Django documentation.

Copy

Copy

# inventory/views.py

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.contrib.auth.decorators import login\_required

from django.contrib.auth.models import User

from inventory.forms import UserRegistry, ProductForm, OrderForm

from inventory.models import Product, Order

@login\_required

def index(request):

orders\_user = Order.objects.all()

users = User.objects.all()[:2]

orders\_adm = Order.objects.all()[:2]

products = Product.objects.all()[:2]

reg\_users = len(User.objects.all())

all\_prods = len(Product.objects.all())

all\_orders = len(Order.objects.all())

context = {

"title": "Home",

"orders": orders\_user,

"orders\_adm": orders\_adm,

"users": users,

"products": products,

"count\_users": reg\_users,

"count\_products": all\_prods,

"count\_orders": all\_orders,

}

return render(request, "inventory/index.html", context)

@login\_required

def products(request):

products = Product.objects.all()

if request.method == "POST":

form = ProductForm(request.POST)

if form.is\_valid():

form.save()

return redirect("products")

else:

form = ProductForm()

context = {"title": "Products", "products": products, "form": form}

return render(request, "inventory/products.html", context)

@login\_required

def orders(request):

orders = Order.objects.all()

print([i for i in request])

if request.method == "POST":

form = OrderForm(request.POST)

if form.is\_valid():

instance = form.save(commit=False)

instance.created\_by = request.user

instance.save()

return redirect("orders")

else:

form = OrderForm()

context = {"title": "Orders", "orders": orders, "form": form}

return render(request, "inventory/orders.html", context)

@login\_required

def users(request):

users = User.objects.all()

context = {"title": "Users", "users": users}

return render(request, "inventory/users.html", context)

@login\_required

def user(request):

context = {"profile": "User Profile"}

return render(request, "inventory/user.html", context)

def register(request):

if request.method == "POST":

form = UserRegistry(request.POST)

if form.is\_valid():

form.save()

return redirect("login")

else:

form = UserRegistry()

context = {"register": "Register", "form": form}

return render(request, "inventory/register.html", context)

Inside the inventory directory, we are going to create a urls.py file, the file will contain URL definitions for our app, which we will later link with the project's urls.py file in the IMS directory.

$ touch urls.py

Add the following code to the file

Copy

Copy

# inventory/urls.py

from django.urls import path

from inventory import views

urlpatterns = [

path("dash/", views.index, name="dash"),

path("products/", views.products, name="products"),

path("orders/", views.orders, name="orders"),

path("users/", views.users, name="users"),

path("user/", views.user, name="user"),

path("register/", views.register, name="register"),

]

**Adding the**inventory/urls.py**to**IMS/urls.py

Modify the project [urls.py](http://urls.py/) file as follows.

Copy

Copy

# IMS/urls.py

from django.conf import settings

from django.conf.urls.static import static

from django.contrib import admin

from django.contrib.auth import views as auth

from django.urls import include, path

urlpatterns = [

path("/", include("inventory.urls")),

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

path(

"", auth.LoginView.as\_view(template\_name="inventory/login.html"), name="login"

),

path(

"logout/",

auth.LogoutView.as\_view(template\_name="inventory/logout.html"),

name="logout",

),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**Creating the**[admin.py](http://admin.py/)**file**

The [admin.py](http://admin.py/) contains instructions on how to display the models in the admin site, we will register all our models in the admin site.

Copy

Copy

# inventory/admin.py

from django.contrib import admin

from inventory.models import Product, Order, UserProfile

admin.site.site\_header = "Inventory Admin"

class ProductAdmin(admin.ModelAdmin):

model = Product

list\_display = ("name", "category", "quantity")

list\_filter = ["category"]

search\_fields = ["name"]

class OrderAdmin(admin.ModelAdmin):

model = Order

list\_display = ("product", "created\_by", "order\_quantity", "date")

list\_filter = ["date"]

search\_fields = ["product"]

class UserProfileAdmin(admin.ModelAdmin):

model = UserProfile

list\_display = ("user", "physical\_address", "mobile", "picture")

list\_filter = ["user"]

search\_fields = ["user"]

admin.site.register(Product, ProductAdmin)

admin.site.register(Order, OrderAdmin)

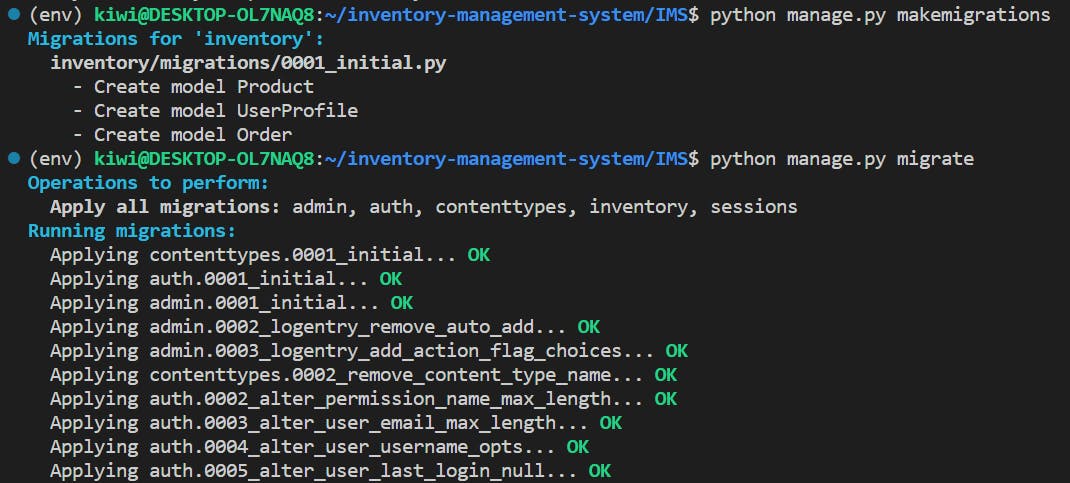
admin.site.register(UserProfile, UserProfileAdmin)

**Creating migrations**

*"Migrations are Django’s way of propagating changes you make to your models (adding a field, deleting a model, etc.) into your database schema. They’re designed to be mostly automatic, but you’ll need to know when to make migrations, when to run them, and the common problems you might run into."*, Django documentation

$ python manage.py makemigrations

$ python manage.py migrate

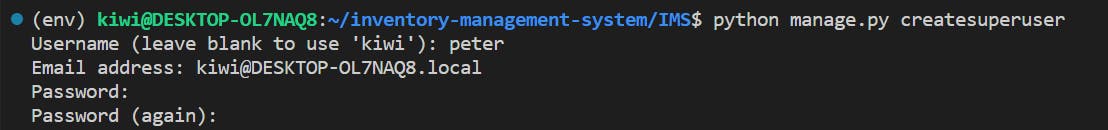


**Creating a super user.**

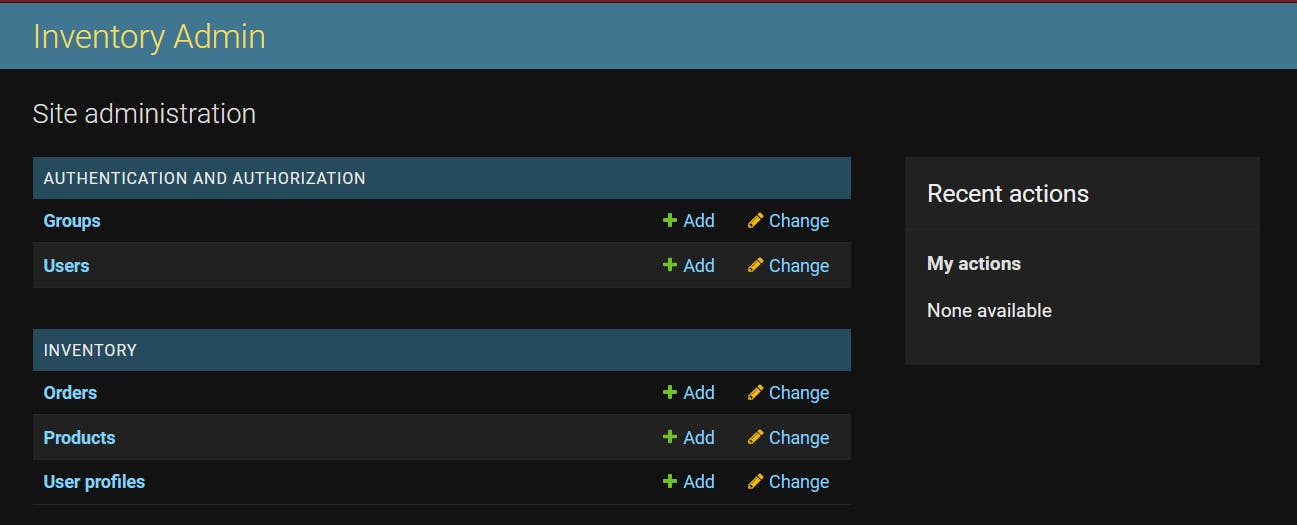
A super user account allows us to login and manage other users through the admin interface.

The super user account is created as follows.

$ python manage.py createsuperuser



Now if you run the development server and navigate to <http://127.0.0.1:8000/admin/> you will be able to see login with your created user.

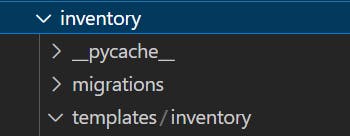


**Building the interface.**

If you go to <http://127.0.0.1:8000/> you will note that the site is displaying an error message saying that the template is not found.



Create a folder called templates inside inventory, and then inside the templates create another folder called inventory.



Inside the inventory folder, create the following HTML files base.html, index.html, login.html, logout.html, orders.html, products.html, register.html, user.html, users.html.

base.html

Copy

Copy

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>{% block title %}{% endblock %}</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet"

integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor" crossorigin="anonymous">

</head>

<body>

{% if user.is\_authenticated and user.is\_staff and user.is\_superuser%}

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

<div class="container">

<a class="navbar-brand" href="{% url 'dash' %}">Dashboard</a>

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

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'products' %}">Products</a>

</li>

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'orders' %}">Orders</a>

</li>

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'users' %}">Users</a>

</li>

</ul>

<ul class="navbar-nav ml-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'user' %}">Profile</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="{% url 'logout' %}">Logout</a>

</li>

</ul>

</div>

</div>

</nav>

{% elif user.is\_authenticated %}

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

<div class="container">

<a class="navbar-brand" href="{% url 'dash' %}">Dashboard</a>

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

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'products' %}">Products</a>

</li>

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'orders' %}">Orders</a>

</li>

</ul>

<ul class="navbar-nav ml-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'user' %}">Profile</a>

</li>

<li class="nav-item">

<a class="nav-link active" href="{% url 'logout' %}">Logout</a>

</li>

</ul>

</div>

</div>

</nav>

{% else %}

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

<div class="container">

<ul class="navbar-nav ml-auto mb-2 mb-lg-0">

<li class="nav-item">

<a class="nav-link active" aria-current="page" href="{% url 'register' %}">Register</a>

</li>

<li class="nav-item">

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

</li>

</ul>

</div>

</div>

</nav>

{% endif %}

{% block content %}

{% endblock %}

<div class="bg-light">

<footer class="py-3 my-4">

<p class="text-center text-muted">&copy; 2022 Inventory App, Inc.</p>

</footer>

</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/js/bootstrap.bundle.min.js"

integrity="sha384-pprn3073KE6tl6bjs2QrFaJGz5/SUsLqktiwsUTF55Jfv3qYSDhgCecCxMW52nD2"

crossorigin="anonymous"></script>

</body>

</html>

index.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ home }} {% endblock %}

{% block content %}

{% if user.is\_authenticated and user.is\_staff and user.is\_superuser %}

<div class="container my-4">

<div class="row align-items-start">

<div class="col-md-12">

<div class="card">

<div class="card-header">

<h1 class="text-center">Statistics</h1>

</div>

<div class="card-body">

<div class="row">

<div class="col-md-4">

<div class="card text-center p-3">

<h4>Registered Users</h4>

<h3><span class="badge bg-primary rounded-pill">{{ count\_users }}</span></h3>

</div>

</div>

<div class="col-md-4">

<div class="card text-center p-3">

<h4>Total Orders</h4>

<h3><span class="badge bg-primary rounded-pill">{{ count\_orders }}</span></h3>

</div>

</div>

<div class="col-md-4">

<div class="card text-center p-3">

<h4>All Products <i class="fa-thin fa-users"></i></h4>

<h3><span class="badge bg-primary rounded-pill">{{ count\_products }}</span></h3>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="container my-4">

<div class="row align-items-start">

<div class="col-md-4">

<div class="card">

<div class="card-header">

<h1 class="text-center">Users</h1>

</div>

<div class="card-body">

<table class="table">

<thead class="table-light">

<tr>

<th scope="col">User Name</th>

<th scope="col">First Name</th>

<th scope="col">Last Name</th>

</tr>

</thead>

<tbody>

{% for user in users %}

<tr>

<td>{{ user.username }}</td>

<td>{{ user.first\_name }}</td>

<td>{{ user.last\_name }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<a class="nav-link text-primary active" aria-current="page" href="{% url 'users' %}">View All

Staff</a>

</div>

</div>

</div>

<div class="col-md-4">

<div class="card">

<div class="card-header">

<h1 class="text-center">Orders</h1>

</div>

<div class="card-body">

<table class="table">

<thead class="table-light">

<tr>

<th scope="col">Product</th>

<th scope="col">Quantity</th>

<th scope="col">Date</th>

</tr>

</thead>

<tbody>

{% for order in orders\_adm %}

<tr>

<td>{{ order.product }}</td>

<td>{{ order.order\_quantity }}</td>

<td>{{ order.date.date }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<a class="nav-link text-primary active" aria-current="page" href="{% url 'orders' %}">Go to

Orders</a>

</div>

</div>

</div>

<div class="col-md-4">

<div class="card">

<div class="card-header">

<h1 class="text-center">Products</h1>

</div>

<div class="card-body">

<table class="table">

<thead class="table-light">

<tr>

<th scope="col">Name</th>

<th scope="col">Quantity</th>

</tr>

</thead>

<tbody>

{% for product in products %}

<tr>

<td>{{ product.name }}</td>

<td>{{ product.quantity }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<a class="nav-link text-primary active" aria-current="page" href="{% url 'products' %}">Go to

Products</a>

</div>

</div>

</div>

</div>

</div>

</div>

{% else %}

<div class="container my-4">

<div class="row">

<div class="col-md-4">

<div class="card">

<div class="card-header">

<h4 class="text-center">Make an Order</h4>

</div>

<div class="card-body">

<a class="btn btn-info" href="{% url 'orders' %}">Create Order</a>

</div>

</div>

</div>

<div class="col-md-8">

<div class="card">

<div class="card-header">

<h4 class="text-center">Here is a list of your Orders</h4>

</div>

</div>

<table class="table">

<thead>

<tr>

<th scope="col">Ordered Product</th>

<th scope="col">Created by</th>

<th scope="col">Quantity</th>

<th scope="col">Order Date</th>

</tr>

</thead>

<tbody>

{% for order in orders %}

{% if order.created\_by.username == user.username %}

<tr>

<td>{{ order.product }}</td>

<td>{{ order.created\_by.username }}</td>

<td>{{ order.order\_quantity }}</td>

<td>{{ order.date }}</td>

</tr>

{% endif %}

{% endfor %}

</tbody>

</table>

</div>

</div>

</div>

{% endif %}

{% endblock %}

login.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ register }} {% endblock %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container">

<div class="row mt-5">

<div class="col-md-3">

<h4>Login</h4>

<hr>

<form method="POST">

{% csrf\_token %}

{{ form | crispy }}

<br>

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

</form>

</div>

</div>

</div>

{% endblock %}

logout.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ register }} {% endblock %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container">

<div class="alert">

<h4>Logged out</h4>

<a href="{% url 'login' %}">Login</a>

</div>

</div>

{% endblock %}

orders.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ title }} {% endblock %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-md-8">

<table class="table">

<thead>

<tr>

<th scope="col">Ordered Product</th>

<th scope="col">Created by</th>

<th scope="col">Quantity</th>

<th scope="col">Order Date</th>

</tr>

</thead>

{% if user.is\_staff and user.is\_superuser %}

<tbody>

{% for order in orders %}

<tr>

<td>{{ order.product }}</td>

<td>{{ order.created\_by.username }}</td>

<td>{{ order.order\_quantity }}</td>

<td>{{ order.date }}</td>

</tr>

{% endfor %}

</tbody>

{% else %}

<tbody>

{% for order in orders %}

{% if user.username == order.created\_by.username %}

<tr>

<td>{{ order.product }}</td>

<td>{{ order.created\_by.username }}</td>

<td>{{ order.order\_quantity }}</td>

<td>{{ order.date }}</td>

</tr>

{% endif %}

{% endfor %}

</tbody>

{% endif %}

</table>

</div>

<div class="col-md-4">

<form method="POST">

<h4>Create a New Order</h4>

<hr>

{% csrf\_token %}

{{ form|crispy}}

<br>

<button type="submit" class="btn btn-primary">Create Order</button>

</form>

</div>

</div>

</div>

{% endblock %}

products.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ header }} {% endblock %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-md-8">

<table class="table">

<thead>

<tr>

<th scope="col">Product Name</th>

<th scope="col">Quantity</th>

<th scope="col">Category</th>

<th scope="col">Description</th>

</tr>

</thead>

<tbody>

{% for product in products %}

<tr>

<td>{{ product.name }}</td>

<td>{{ product.quantity }}</td>

<td>{{ product.category }}</td>

<td>{{ product.description }}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

{% if user.is\_staff and user.is\_superuser %}

<div class="col-md-4">

<form method="POST">

<h4>Add a New Product</h4>

<hr>

{% csrf\_token %}

{{ form|crispy}}

<br>

<button type="submit" class="btn btn-primary">Add</button>

</form>

</div>

{% else %}

<div class="col-md-4">

</div>

{% endif %}

</div>

</div>

{% endblock %}

register.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ register }} {% endblock %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container">

<div class="row">

<div class="col-md-6 offset-md-3">

<h3>Create Account</h3>

<hr>

<form method="POST">

{% csrf\_token %}

{{ form | crispy }}

<input class="btn btn-info" type="submit" value="Signup">

</form>

</div>

</div>

</div>

{% endblock %}

user.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ profile }} {% endblock %}

{% block content %}

<div class="container my-4">

<div class="row align-items-start">

<div class="col-md-8">

<div class="card">

<div class="card-header">

<h1 class="text-center">User's Profile</h1>

</div>

<div class="card-body">

<div class="row">

<div class="col-md-8">

<div class="card text-center p-3">

<h4>Profile Information</h4>

<br>

<table class="table bg-white table-borderless">

<tbody>

<tr>

<th scope="col">User Name</th>

<td scope="col">{{ user.username }}</td>

</tr>

<tr>

<th scope="col">Full Name</th>

<td scope="col">{{ user.first\_name }} {{user.last\_name}}</td>

</tr>

<tr>

<th scope="row">Email</th>

<td>{{ user.email }}</td>

</tr>

<tr>

<th scope="row">Phone</th>

<td>{{ user.userprofile.mobile }}</td>

</tr>

<tr>

<th scope="row">Address</th>

<td>{{ user.userprofile.physical\_address }}</td>

</tr>

</tbody>

</table>

</div>

</div>

<div class="col-md-4">

<img style="border-radius: 50%; width: auto; height: auto;" class="img-thumbnail" src="{{ user.userprofile.picture.url }}" alt="profile picture">

</div>

</div>

</div>

</div>

</div>

</div>

</div>

{% endblock %}

users.html

Copy

Copy

{% extends 'inventory/base.html' %}

{% block title %}{{ title }} {% endblock %}

{% block content %}

<div class="container">

<table class="table">

<thead>

<tr>

<th scope="col">First Name</th>

<th scope="col">Last Name</th>

<th scope="col">Email</th>

<th scope="col">Address</th>

</tr>

</thead>

<tbody>

{% for user in users %}

<tr>

<td>{{ user.first\_name}}</td>

<td>{{ user.last\_name }}</td>

<td>{{ user.email }}</td>

<td>{{ user.userprofile.physical\_address }}</td>

</tr>

{% endfor %}

</tbody>

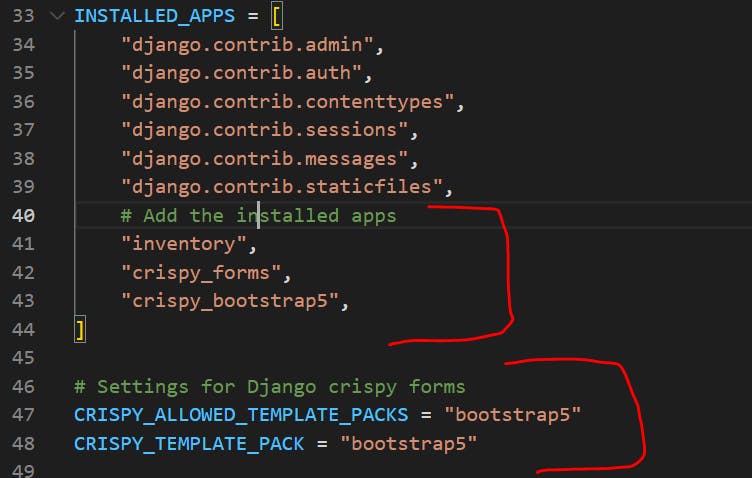
</table>

</div>

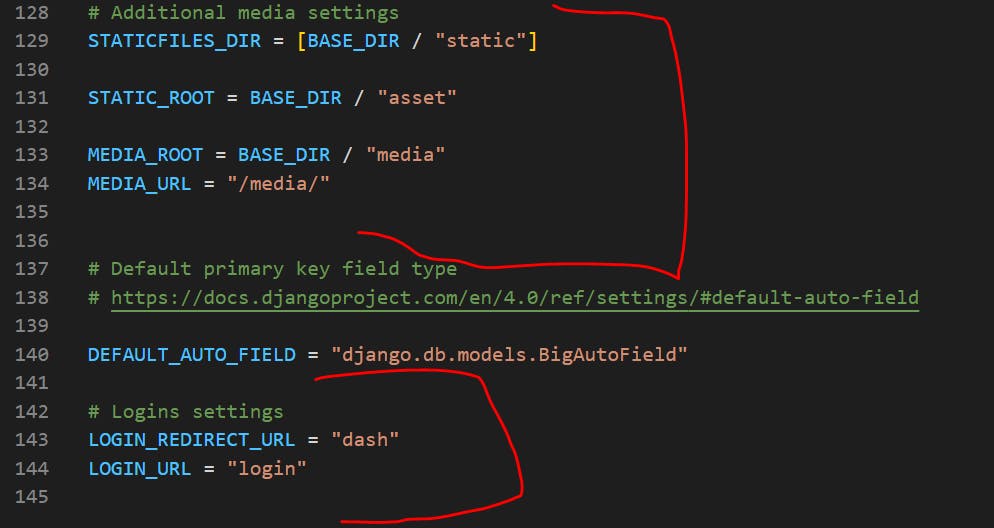
{% endblock %}

**Additional settings**

Add inventory, crispy\_forms and crispy\_bootstrap5 under the INSTALLED\_APPS and add the corresponding settings as shown below in settings.py

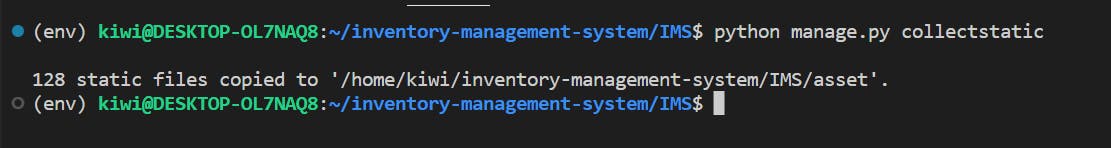


Next add settings to handle static media files as shown



Next, collect the static files, this is done to allow for easy serving of the files when hosting the app

$ python manage.py collectstatic



Now if you run the server again and visit the homepage again you will be met with a cool looking login page.

