1.Partials>>Topnav.html

Add the details of topnav from bootstrap.com

1. Dashboard(main)>>urls.py

Change name= ‘index’ to name= ‘dashboard-index’. and Change name= ‘staff’ to name= ‘dashboard-staff’.

1. Templates>>dashboard>>staff.html

Change staff.html to

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

{% block title %}Staff Page{% endblock %}

And index.html to

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

{% block title %}Home Page{% endblock %}

1. Dashboard>>views.py
2. Create ‘product’ and ‘order’ templates in views.py

def product(request):

    return render(request, 'dashboard/product.html')

def order(request):

    return render(request, 'dashboard/order.html')

1. Dashboard>>urls.py

path('product/', views.product, name= 'product'),

    path('order/', views.order, name= 'order'),

]

1. Templates>>dashboard>>product.html…Templates>>dashboard>>order.html

Create the corresponding templates for product.html and order.html

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

{% block title %}Order Page{% endblock %}

1. stockinventory\_project>>css

Create a css folder under stockinventory. Create style.css and enter

body{

    background: rgb(230, 227, 227);

}

.my-card:hover{

    transform: scale(1.1);

    transition: 0.2s ease-in-out;

}

1. Inventory(main)>>asert

Create asert folder in inventory(main)

1. Stockinventory\_project(main)>>settings.py

STATIC\_ROOT = (BASE\_DIR/"asert")

STATITICFILES\_DIRS = [

    BASE\_DIR/ "static"

]

1. Run python manage.py collectstatic
2. Make necessary changes to order.html and product.html
3. Dashboard(main)>>models.py

Add the below to models.py

from django.db import models

# Create your models here.

CATEGORY= (

    ('Stationary', 'Stationary'),

    ('Electronics', 'Electronics'),

    ('Food','Food')

)

class Product(models.Model):

    name= models.CharField(max\_length=100)

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

    quantity= models.PositiveIntegerField()

But this is not a thing to do when you want to deploy.

1. python manage.py migrate
2. Python manage.py createsuperuser
3. Create a product on the dashboard
4. Dashboard(main)>>admin.py

Register product model in the admin

from django.contrib import admin

from .models import Product

# Register your models here.

admin.site.register(Product)

1. Define a string representation of the product

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)

    def \_\_str\_\_(self):

        return f'{self.name}-{self.quantity}'

1. Inventory(main) >> User

Create User App: python manage.py startapp user

1. Register User app in settings.py

   'user.apps.UserConfig'

]

1. Templates>>user

Create user folder in templates.

Add three files in user:

Login.html, logout.html, and register.html

1. Templates>>user>>register.html

In register.html add:

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

1. Invemtory>>user>>views.py

Add views for the html files created for user in templates.

from django.shortcuts import render

# Create your views here.

def register(request):

    return render(request, 'user/register.html')

1. inventory>>urls.py

Handle the routing of each view in user’s views.py in inventory urls.py.

from django.contrib import admin

from django.urls import path, include

from user import views as user\_view

urlpatterns = [

    path('admin/', admin.site.urls),

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

    path('register',user\_view.register, name= 'user-register'),

]

1. Templates>>user>>register.html

Update to create the registration page

1. Update user>>views.py accordingly
2. Pip install django-crispy-forms
3. Add ‘crispy\_forms’ to installed apps in settings.py
4. Add CRISPY\_TEMPLATE\_PACK= ‘bootstrap4’ to settings.py
5. Add {% load crispy\_forms\_tags %} to register.html and form|crispy

<h3>Registration Page</h3>

                <form method="POST">

                    {% csrf\_token %}

                    {{ form|crispy }}

Modify User Creation Form

1. Inventory(main)>>user>>forms.py

Create forms.py in user accordingly:

from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

class CreateUserForm(UserCreationForm):

    email= forms.EmailField()

    class Meta:

        model= User

        fields= '\_\_all\_\_'

1. Update fields variable of forms.py accordingly

    fields= {'username', 'email', 'password1', 'password2'}

1. Repeat the steps for register.html for login.html and logout.html
2. Update settings.py accordingly

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

LOGIN\_REDIRECT\_URL = 'dashboard-index'

1. Update nav.html for registeration, login, and logout pages.
2. Stocks\_inventory>>stocks\_inventory>>urls.py

Update the urls path

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

urlpatterns = [

    path('admin/', admin.site.urls),

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

    path('register',user\_view.register, name= 'user-register'),

    path('login/', auth\_views.LoginView.as\_view(template\_name='user/login.html'),

                                                name= 'user-login'),

    path('logout/', auth\_views.LogoutView.as\_view(template\_name= 'user/logout.html'),

         name= 'user-logout')

]

1. Stocks\_inventory>>dashboard>>urls.py

Update the dashboard urls.py accordingly

urlpatterns= [

    path('dashboard/',views.index, name= 'dashboard-index'),

1. Stocks\_inventory>>stocks\_inventory>>urls.py
2. Update login path urls accordingly

   path('register',user\_view.register, name= 'user-register'),

    path('', auth\_views.LoginView.as\_view(template\_name='user/login.html'),

                                                name= 'user-login'),

1. Stock\_inventory>>dashboard|>>views.py

Add @login\_required decorators to restrict access.

from django.shortcuts import render

from django.http import HttpResponse

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

from .models import Product

# Create your views here.

@login\_required

def index(request):

    return render(request, 'dashboard/index.html')

@login\_required

def staff(request):

    return render(request, 'dashboard/staff.html')

@login\_required

def product(request):

    return render(request, 'dashboard/product.html')

@login\_required

def order(request):

    return render(request, 'dashboard/order.html')

1. Stock\_inventory>>stock\_inventory>>settings.py

Add: LOGIN\_URL= ‘user-login’

LOGIN\_URL = 'user-login'

1. Stock\_inventory>>dashboard>>index.html

Update index.html to restrict access to authenticated staff.

1. Stock\_inventory>>dashboard>>templates

Create staff\_index.html in templates

Populate it

1. Update index.html to accommodate staff\_index.html
2. Stock\_inventory>>stock\_inventory>>templates>>user

Create profile.html

Create profile urls

Create profile views

   path('profile',user\_view.profile, name= 'user-profile'),

def profile(request):

    return render(request, 'user/profile.html')

1. Update nav.html
2. Stock\_inventory>>user>>models.py

Update models.py for profile

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Profile(models.Model):

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

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

    phone = models.CharField(max\_length=20, null=True)

    image= models.ImageField(default = 'avatar.jpg', upload\_to= 'Profile\_Images')

    def \_\_str\_\_(self):

        return f'{self.staff.username}-Profile'

1. Make migrations

Python manage.py makemigrations

1. Stocks\_inventory>>user>>admin.py

Update admin.py

from django.contrib import admin

from .models import Profile

# Register your models here.

admin.site.register(Profile)

1. Update settings.py

MEDIA\_ROOT= 'BASE\_DIR/media'

MEDIA\_URL= '/media'

1. Stocks\_inventory>>stocks\_inventory>>urls.py

Update urls.py accordingly

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

from django.conf import settings

from django.conf.urls.static import static

                                                name= 'user-login'),

    path('logout/', auth\_views.LogoutView.as\_view(template\_name= 'user/logout.html'),

         name= 'user-logout')

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

1. Update profile.html to accommodate user.username, user.email, user.profile.phone, user.profile.address, and uer.profile.image.url.

Django Signals

A signal will execute an action based on an action that was executed. For example, automating profile creation.

1. Stock\_inventory>>user>>signals.py

Create signals.py.

Populate signals.py

1. Stock\_inventory>>user>>apps.py

Update apps.py

1. Templates>>user>>profile\_update

Create profile\_update.html

Populate it

Update views.py

from django.shortcuts import render

from django.contrib.auth.forms import UserCreationForm

from .forms import CreateUserForm

# Create your views here.

def register(request):

    form= UserCreationForm

    context= {

        'form':form

    }

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

def profile(request):

    return render(request, 'user/profile.html')

def profile\_update(request):

    context= {

    }

    return render(request, 'user/profile\_update', context)

UPDATE USER PROFILE

1. Update profile.html to accommodate profile\_update.html.
2. User>>forms.py

Create UserUpdateForm and ProfileUpdateForm in forms.py

Also add

from .models import Profile

       model= User

        fields= {'username', 'email', 'password1', 'password2'}

class UserUpdateForm(forms.ModelForm):

    class Meta:

        model= User

        fields= ['usernane', 'email']

class ProfileUpdateForm(forms.ModelForm):

    class Meta:

        model= Profile

        fields= ['address', 'phone', 'image']

1. Update views.py

def profile\_update(request):

    if request.method== 'POST':

        user\_form= UserUpdateForm(request.POST, instance=request.user)

        profile\_form= ProfileUpdateForm(request.POST, request.FILES, instance= request.user.profile)

    else:

        user\_form= UserUpdateForm(instance= request.user)

        profile\_form= ProfileUpdateForm(instance= request.user.profile)

    context= {

       'user\_form': user\_form,

       'profile\_form': profile\_form,

    }

    return render(request, 'user/profile\_update', context)

1. Update profile\_update.html to stylize profile\_update form with crispy.
2. Create ‘update’ button in profile\_update.html.
3. Update views.py to save profile\_form, and update\_form if both are valid. Then redirect to ‘user-profile’.

        if user\_form.is\_valid() and profile\_form.is\_valid():

            user\_form.save()

            profile\_form.save()

            return redirect('user-profile')

1. Update views.py to include csrf\_token for profile\_html.

CREATE, READ, UPDATE, and DELETE

1. Queryset is variable that stores all the objects of a model.

To retrieve all objects:

Model.objects.all(). Its SQL format is items= Product.objects.raw(‘SELECT \* FROM dashboard\_product’).

To retrieve specific objects with a filter

Model.objects.filter()

1. Update Product model to have the items variable that fetches all the objects of model.
2. Stocks\_inventory>>dashboard|>>forms.py

Create forms.py

Populate forms.py for ProductForm

from django import forms

from .models import Product

class ProductForm(forms.ModelForm):

    class Meta:

        model= Product

        fields= ['name', 'category', 'quantity']

1. Stocks\_inventory>>dashboard>>views.py

Update views.py to import ProductForm

from django.shortcuts import render

from django.http import HttpResponse

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

from .models import Product

from .forms import ProductForm

#

And

To activate ProductForm

@login\_required

def product(request):

    items= Product.objects.all() #using ORM

    #items= Product.objects.raw('SELECT \* FROM dashboard\_product')

    if request.method == 'POST':

        form= ProductForm(request.POST)

    else:

        form= ProductForm()

    context= {

        'items': items,

        'form': form,

    }

    retur

1. Update product.html to reflect the product form and stylize it with crispy.  
   68. Update views.py to save the form . Ensure to import redirect too.

from django.shortcuts import render, redirect

def product(request):

    items= Product.objects.all() #using ORM

    #items= Product.objects.raw('SELECT \* FROM dashboard\_product')

    if request.method == 'POST':

        form= ProductForm(request.POST)

        if form.is\_valid():

            form.save()

1. Dashboard>>views.py

Create view for product\_delete

def product\_delete(request, pk):

    item= Product.objects.get(id=pk)

    return render(request, 'dashboard/product\_delete')

1. Update urls.py

urlpatterns= [

    path('dashboard/',views.index, name= 'dashboard-index'),

    path('staff/', views.staff, name= 'dashboard-staff'),

    path('product/', views.product, name= 'dashboard-product'),

    path('product/delete/<int:pk>', views.product\_delete, name= 'dashboard-product-delete')

    path('order/', views.order, name= 'dashboard-order'),

]

1. Templates>>dashboard>>product\_delete

Create product\_delete.html in templates. Populate it.

1. Update product.html to link product\_delete.html.
2. Dashboard>>views.py

Update views.py for product\_delete

def product\_delete(request, pk):

    item= Product.objects.get(id=pk)

    if request.method == 'POST':

        item.delete()

        return redirect('dashboard-product')

1. Create view, url, and others for product\_update

def product\_update(request, pk):

    item= Product.objects.get(id= pk)

    if request.method== 'POST':

        form = ProductForm(request.POST, instance= item)

        if form.is\_valid():

            form.save()

            return redirect('dashboard-product')

    else:

        form= ProductForm(instance= item)

    context= {

        'form': form

    }

LIST and DETAIL VIEW

1. dashboard>>urls.py

Update staff url to below

def staff(request):

    workers= User.objects.all()

    context= {

        'workers': workers

    }

    ret

1. Dashboard>>templates>>staff.html

Update staff.html to include worker username, worker email, and worker profile.phone

1. Create staff\_detail url, staff\_detail.html,

@login\_required

def staff\_detail(request, pk):

    workers= User.objects.get(id=pk)

    context= {

        'workers': worker,

    }

    return render(request, 'dashboard/staff\_detail', context)

Update order url

def order(request):

    orders= Order.objects.all()

    context= {

        'orders': orders,

    }

    return render(request, 'dashboard/order.html')

1. Templates>>dashboard>>order.html

Update order.html to include orders.product, orders.category, orders.quantity, orders,date

1. Stocks\_inventory>>dashboard>>models.py

Update models.py to include Order model

class Order(models.Model):

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

    staff= models.ForeignKey(User, models.CASCADE, null= True)

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

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

    class Meta:

        verbose\_name\_plural= 'Order'

    def \_\_str\_\_(self):

        return f'{self.product} ordered by {self.staff.username}'

1. Update for OrderForm too

@login\_required

def index(request):

    orders= Order.objects.all()

    if request.method == 'POST':

        form= OrderForm(request.POST)

        if form.is\_valid():

            instance= form.save(commit=False)

            instance.staff= request.user

            instance.save()

            return redirect('dashboard-index')

    else:

        form= OrderForm()

    context = {

        'orders': orders,

        'form': form,

    }

    return render(request, 'dashboard/index.html')

1. Update staff\_index.html to reflect order.date

FLASH MESSAGES

1. Stocks\_inventory>>dashboard>>views.py

Import messages to handle flash messages

from django.contrib import messages

1. Update product view to display alert message

product\_name= form.cleaned\_data.get('name')

messages.success(request, f'{product\_name} has been added')

1. Update product.html to display message in messages.
2. User>>views.py

Import django’s messages method

1. Update register view to display flash messages

form= CreateUserForm(request.POST)

        if form.is\_valid():

            form.save()

            username= form.cleaned\_data.get('username')

            messages.success(

                request, f'Account has been created for {username}. Continue to log in')

            return redirect('user-login')

1. Update login.html to reflect the update.
2. Dashboard>>views.py

Update index in views.py to get all in products

def index(request):

    orders= Order.objects.all()

    products= Product.objects.all()

Also add products in the context dictionary

    context = {

        'orders': orders,

        'form': form,

        'products': products,

    }

CHARTS and DATA VISUALIZATION

1. Update background colour and border colour of order in orders in index.html.
2. Dashboard>>views.py

Create workers\_count variable. Then add it to the context dictionary.

def staff(request):

    workers= User.objects.all()

    workers\_count= workers.count()

    context= {

        'workers': workers

        'workers\_count' : workers\_count,

    }

    return render(request, 'dashboard/staff.html')

1. Partials>>top\_nav.html\

Update top\_nav.html to include workers\_count.

1. Dashboard>>views.py

Add workers\_count to product view

1. Do same for order.

@login\_required

@login\_required

def order(request):

    orders= Order.objects.all()

    orders\_count= orders.count()

    workers\_count= User.objects.all().count()

    context= {

        'orders': orders,

        'workers\_count': workers\_count,

        'orders\_count': orders\_count,

    }

    return render(request, 'dashboard/order.html')

1. Update staff too.

@login\_required

def staff(request):

    workers= User.objects.all()

    workers\_count= workers.count()

    orders\_count= Order.objects.all().count()

    product\_count= Products.objects.all().count()

    context= {

        'workers': workers,

        'workers\_count' : workers\_count,

        'orders\_count': orders\_count,

        'product\_count': products\_count,

    }

    ret

1. Do same upgrades for other functions in views.

FORGET PASSWORD

1. Staff\_inventory>>urls.py
2. Create path for password\_reset using django’s inbuilt password reset function.
3. Do same for password\_reset\_done, password\_reset\_confirm, and password\_reset\_complete.
4. Staff\_inventory>>settings.py

Create EMAIL\_BACKEND, EMAIL\_HOST, EMAIL\_PORT, EMAIL\_USE\_TLS, EMAIL\_HOST\_USER, EMAIL\_HOST\_PASSWORD

1. Create password\_reset\_done.html, password\_reset.html, password\_reset\_done.html, password\_reset\_complete.html and populate.