

ISE 2 DBMS PROJECT**Topic : Sales And Inventory Management System**

Site is hosted on following web address :

<https://sales-inventory-management.herokuapp.com/>

Code for the site is available on following link :

<https://github.com/kamatyogesh7/sales-and-inventory-management-system>

PURPOSE

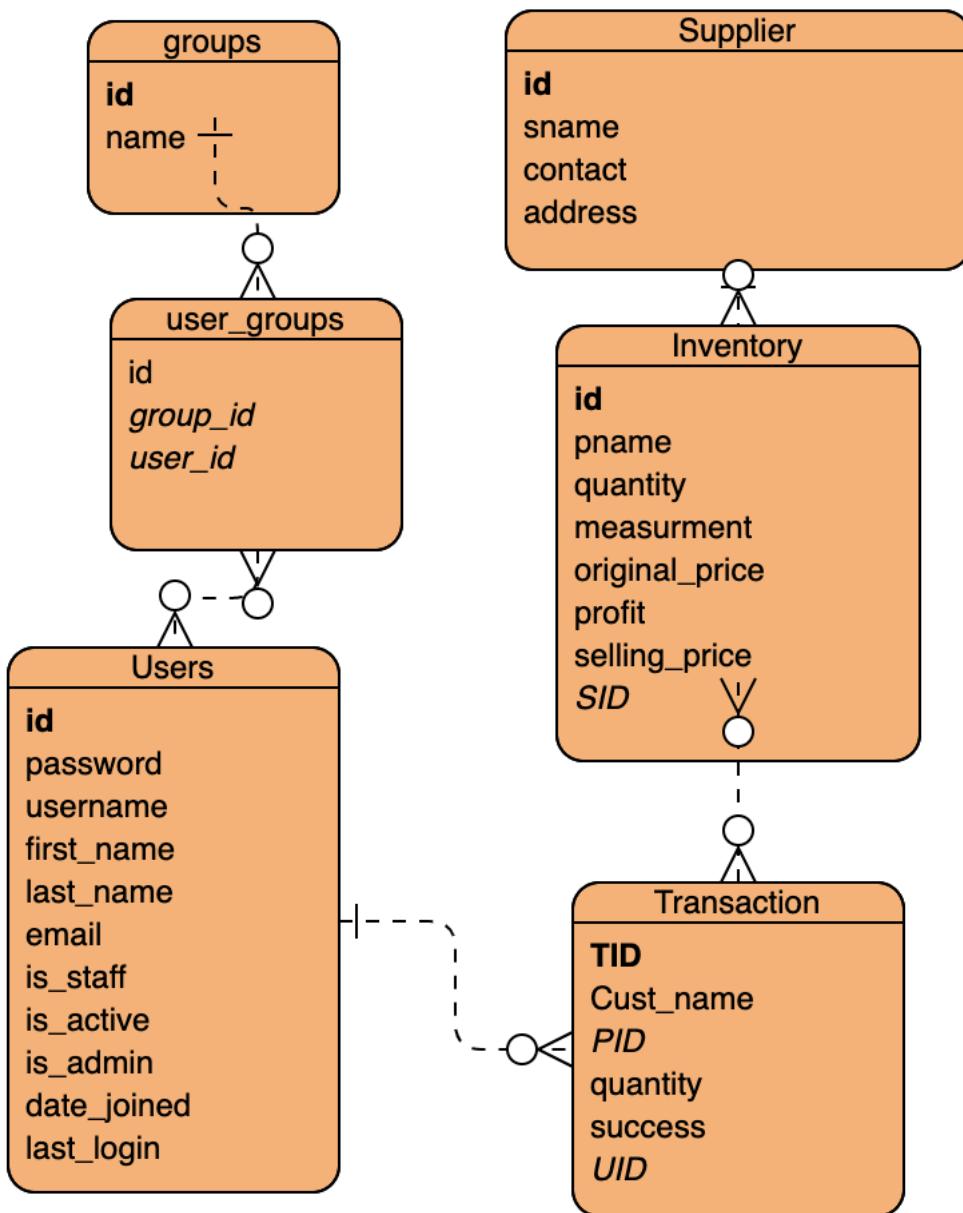
PURPOSE Inventory is the amount of tangible goods, products or services you offer your customers. These goods are itemized and catalogued into an inventory management system.

E-commerce businesses most likely deal with the latter, or finished goods that are currently in their possession. Inventory is a portion of a company's assets that are

either ready to order or will be ready for sale soon. Product inventory is a tangible asset because it's a real-time evaluation of the revenue a company is generating. If an e-commerce store can't move product from its inventory, it is clearly struggling to make sales and must adjust its operations, otherwise it won't obtain bottom-line revenue.

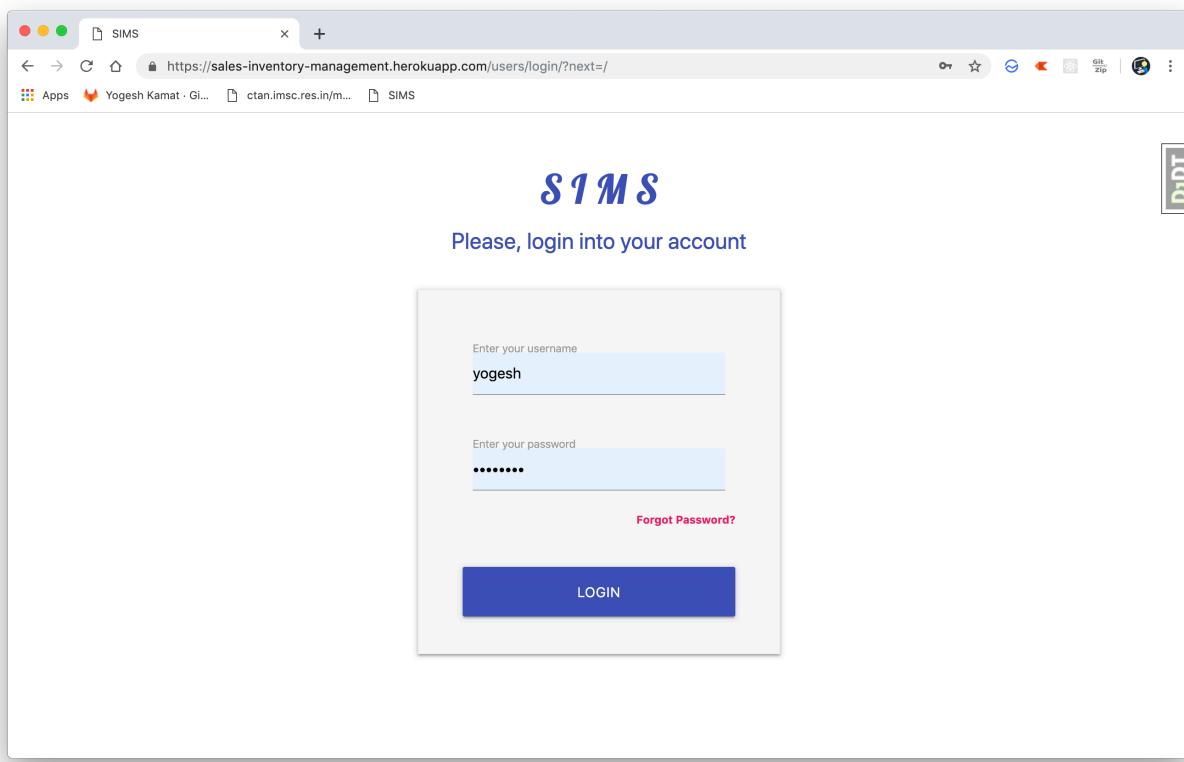
Inventory is often the largest item a business has in its current assets, meaning it must be accurately monitored. Inventory is counted and valued at the end of each accounting period to determine the company's profits or losses. In the e-commerce sector, a day-to-day management of inventory helps increase business intelligence and visibility. E-commerce business owners have better ongoing control when they have constant eyes on inventory. And that's why the need of "Inventory Management System".

E-R diagram

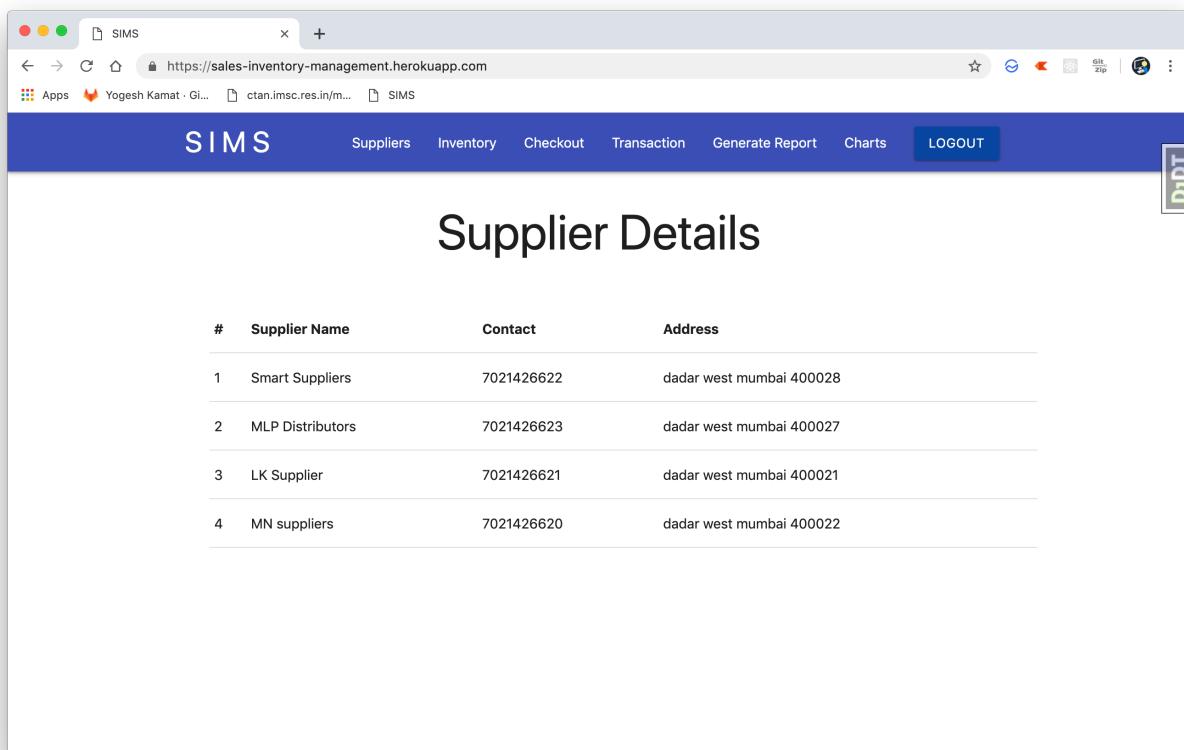


OUTPUT SCREENSHOTS WITH EXPLANATION

Login Page : (required without which we can not go to home page)



Home Page : (Shows supplier details)



Home Page contains following tabs as well :

1. Inventory (shows inventory details)
2. Checkout page for sales.
3. Transaction page for successful transactions.
4. Report generating page
5. Graph analytics
6. Debug tool to show row sql queries

1.

The screenshot shows the 'Inventory Details' page of the SIMS application. At the top, it displays 'Total Inventory Value : 7485' and 'Total Products : 5'. Below this is a table listing five products:

#	Product Name	Quantity	Measurement	Original Price	Profit	Selling Price	Supplier
1	Pen	35	Piece	25	10	35	SSmart Suppliers
2	Sugar	50	Kilogram	52	8	60	SMLP Distributors
3	Eraser	50	Piece	5	2	7	SSmart Suppliers
4	Pencil	47	Piece	7	3	10	SLK Supplier
5	Bread	61	Piece	32	8	40	SMLP Distributors

2.

The screenshot shows the 'Add to cart' page of the SIMS application. It has a form with fields for 'Customer Name' (suresh), 'Product' (Sugar), 'Quantity Available' (50), and 'Quantity Required' (10). To the right, there is a sidebar showing a list of suppliers with their respective selling prices and a '+' button next to each entry.

Supplier	Selling Price
SSmart Suppliers	25
SMLP Distributors	52
SSmart Suppliers	5
SLK Supplier	7
SMLP Distributors	32

Name : Yogesh D. Kamat

SPIT Batch E

UID : 2018240066

2.

The screenshot shows the SIMS application interface with a blue header bar containing the SIMS logo and navigation links: Suppliers, Inventory, Checkout, Transaction, Generate Report, Charts, and LOGOUT. A watermark 'DxDT' is visible in the top right corner. The main content area is titled 'Checkout'. On the left, there is a 'Cart' section with two rows of data:

#	Customer Name	Product	Quantity	Price	DELETE
2	suresh	2	10	600	<button>DELETE</button>
1	ramesh	2	10	600	<button>DELETE</button>

On the right, there is a table of products:

#	Product Name	Quantity	Measurement	Selling Price	Supplier	Add (+)
1	Pen	35	Piece	35	SSmart Suppliers	<button>(+)</button>
2	Sugar	50	Kilogram	60	SMLP Distributors	<button>(+)</button>
3	Eraser	50	Piece	7	SSmart Suppliers	<button>(+)</button>
4	Pencil	47	Piece	10	SLK Supplier	<button>(+)</button>
5	Bread	61	Piece	40	SMLP Distributors	<button>(+)</button>

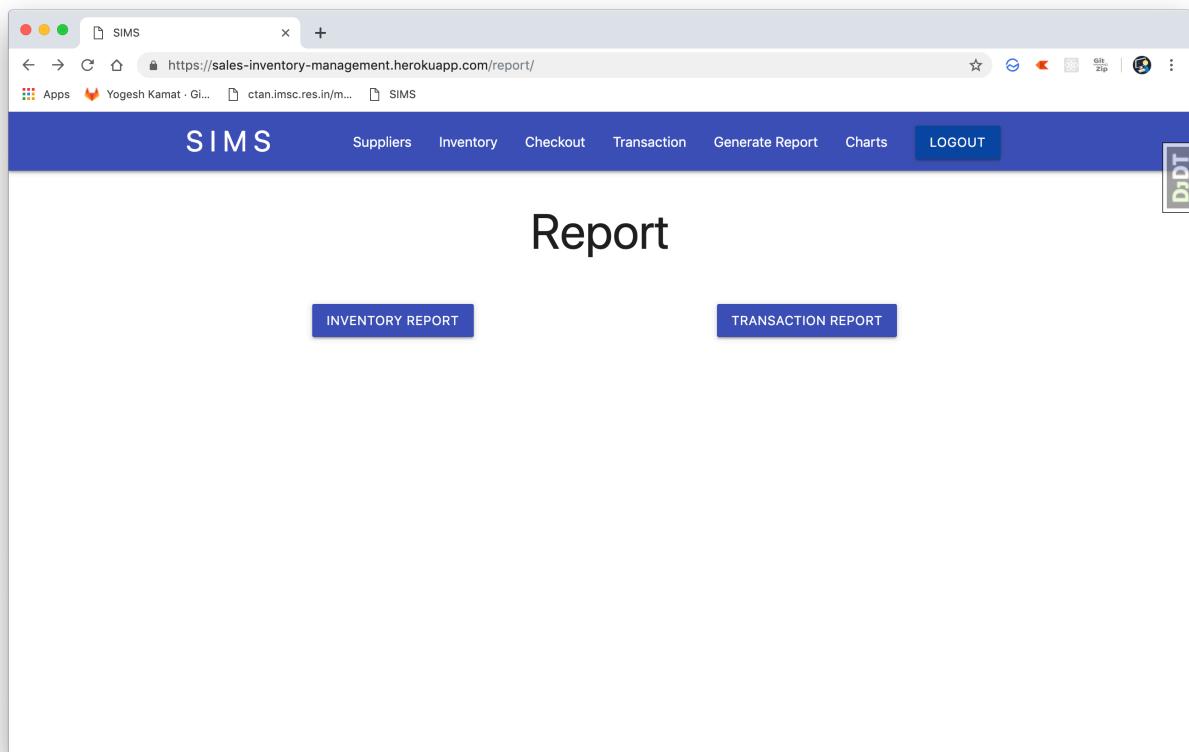
A large blue 'CHECKOUT' button is located at the bottom left of the cart area.

3.

The screenshot shows the SIMS application interface with a blue header bar containing the SIMS logo and navigation links: Suppliers, Inventory, Checkout, Transaction, Generate Report, Charts, and LOGOUT. A watermark 'DxDT' is visible in the top right corner. The main content area is titled 'Transaction Details'. There is a table of transaction items:

Customer Name	Quantity	Measurment	Price	Supplier	DELETE
suresh	10	Kilogram	600	MLP Distributors	<button>DELETE</button>
ramesh	10	Kilogram	600	MLP Distributors	<button>DELETE</button>

4.

A screenshot of a web browser window titled "SIMS Reports" at the top. The URL is https://sales-inventory-management.herokuapp.com/report/. The page shows a dark header with the "SIMS" logo and navigation links. Below the header, the text "1 / 1" is displayed. The main content area contains the following text:

S I M S Reports

Username : yogesh

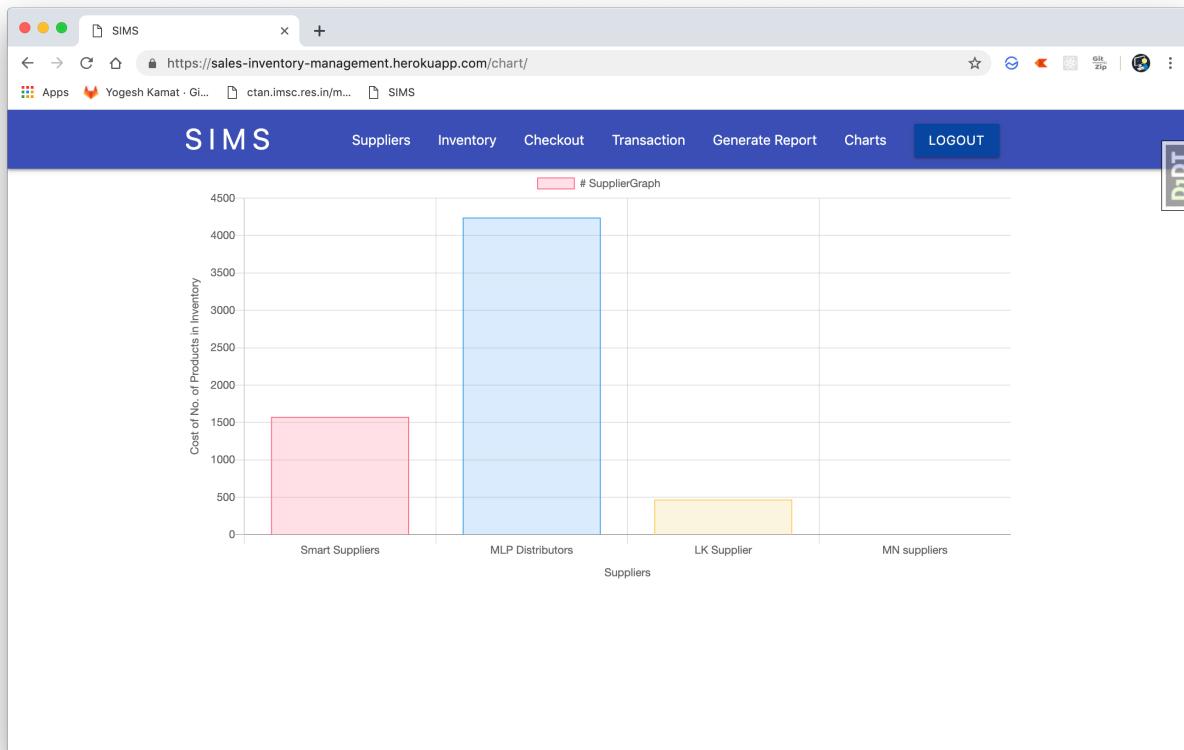
Inventory Details

14th of April

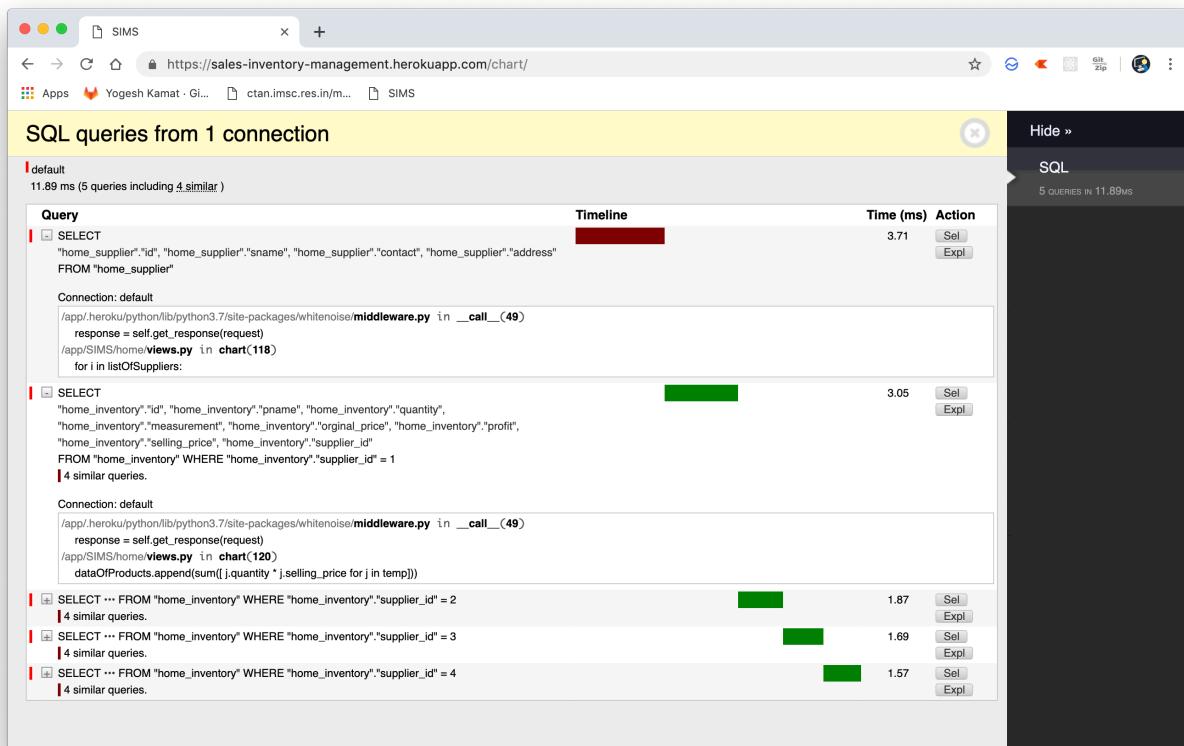
Product ID	Product Name	Quantity	Measurement	Original Price	Profit	Selling Price	Supplier
1	Pen	35	Piece	25	10	35	Smart Suppliers
3	Eraser	50	Piece	5	2	7	Smart Suppliers
4	Pencil	47	Piece	7	3	10	LK Supplier
5	Bread	61	Piece	32	8	40	MLP Distributors
2	Sugar	30	Kilogram	52	8	60	MLP Distributors

On the right side of the table, there are three small circular buttons with icons: a plus sign, a minus sign, and a double arrow.

5.



6.



Name : Yogesh D. Kamat

SPIT Batch E

UID : 2018240066

Admin Panel to add or delete inventories or suppliers or transactions.
In Admin Panel Only admin user can login.

HomePage for admin panel :

The screenshot shows the Django Admin interface for the SIMS application. The top navigation bar includes links for 'Site administration | Django site', 'https://sales-inventory-management.herokuapp.com/admin/', 'Apps', 'Yogesh Kamat · Gi...', 'ctan.imsc.res.in/m...', 'SIMS', 'WELCOME, YOGESH' (with options to 'VIEW SITE / CHANGE PASSWORD / LOG OUT'), and a 'DDT' icon. The main content area is divided into sections: 'AUTHENTICATION AND AUTHORIZATION' (Groups, Users) and 'HOME' (Inventory, Suppliers, Transactions). On the right, a sidebar displays 'Recent actions' with entries for 'Iphone Inventory' (deleted), 'Iphone Inventory' (added), 'sampleuser User' (added), and 'sejal User' (added).

Add Inventory form :

The screenshot shows the 'Add inventory' form within the Django Admin interface. The URL is https://sales-inventory-management.herokuapp.com/admin/home/inventory/add/. The page title is 'SIMS Admin Panel'. The breadcrumb navigation shows 'Home > Home > Inventories > Add inventory'. The form fields include: Pname (input field), Quantity (input field), Measurement (input field), Orginal price (input field), Profit (input field), Selling price (input field), and Supplier (a dropdown menu with a '+' button to add new suppliers). At the bottom, there are three buttons: 'Save and add another', 'Save and continue editing', and a large blue 'SAVE' button.

Supplier add form :

Add supplier

Sname:

Contact:

Address:

[Save and add another](#) [Save and continue editing](#) [SAVE](#)

Delete Transaction page (In similar manner every inventory and supplier object can be deleted as well)

Select transaction to change

Action: [Delete selected transactions](#) [Go](#) 2 of 2 selected

<input checked="" type="checkbox"/> TRANSACTION
<input checked="" type="checkbox"/> Transaction object (2)
<input checked="" type="checkbox"/> Transaction object (1)

2 transactions

[ADD TRANSACTION +](#)

TECHNOLOGIES USED :

1. PostgreSQL
2. Django web framework
3. Html and CSS
4. Heroku web hosting

CODE

(Complete code is available on GitHub link provided above.)

Home App :

Model :

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
```

```
class Supplier(models.Model):
    sname          = models.CharField(max_length=255, unique=True)
    contact        = models.CharField(max_length=255)
    address        = models.TextField(max_length=255)

    def __str__(self):
        return self.sname


class Inventory(models.Model):
    pname          = models.CharField(max_length=255)
    quantity       = models.PositiveIntegerField()
    measurement   = models.CharField(max_length=255)
    orginal_price = models.PositiveIntegerField()
    profit         = models.PositiveIntegerField()
    selling_price = models.PositiveIntegerField()
    supplier       = models.ForeignKey(Supplier, on_delete=models.CASCADE)

    def __str__(self):
        return self.pname


class Transaction(models.Model):
    cust_name      = models.CharField(max_length=255)
    pid            = models.ForeignKey(Inventory, on_delete=models.CASCADE)
    quantity_r     = models.PositiveIntegerField()
    success         = models.BooleanField(default=False)
    uid             = models.ForeignKey(User,
                                         on_delete=models.CASCADE)

    def actual_price(self):
        return int(self.quantity_r) * int(self.pid.selling_price)
```

Url for Routing :

```
from django.urls import path
from home import views
urlpatterns = [
    path('',views.index,name='home'),
    path('inventory/',views.inventory,name='inventory'),
    path('checkout/',views.checkout,name='checkout'),
    path('transaction/',views.transaction,name='transaction'),
    path('report/',views.report,name='report'),
    path('chart/',views.chart,name='chart'),
]
```

Views :

```
from django.shortcuts import render, HttpResponseRedirect
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.conf import settings
from easy_pdf import rendering
from django.http import JsonResponse
```

Create your views here.

```
from home.models import Supplier, Inventory, Transaction
```

```
@login_required
def index(request):
    suppliers      = Supplier.objects.all()
    context        = {
        'suppliers':suppliers,
    }
    return render(request, 'home/index.html', context)
```

```
@login_required
def inventory(request):
    inventory      = Inventory.objects.all()
    total_products = Inventory.objects.all().count()
    total_value    = [ i.quantity * i.selling_price for i in inventory]
    context        = {
        'inventory':inventory,
        'total_products':total_products,
        'total_value': sum(total_value)
    }
    return render(request, 'home/index.html', context)
```

```
@login_required
def checkout(request, *args, **kwargs):
    user          = request.user
    inventory    = Inventory.objects.all()
    cart         = Transaction.objects.all().filter(uid_id = user.id,success=0)
```

```

context      = {
    'checkout':1,
    'inventory_checkout':inventory,
    'cart':cart,
}

if request.method == 'POST' and request.POST.get('tiddel'):
    tid = int(request.POST.get('tiddel'))
    Transaction.objects.get(pk=tid).delete()
    cart      = Transaction.objects.all().filter(uid_id = user.id,success=0)
    context['cart'] = cart
    return render(request,'home/index.html', context)

if request.method == 'POST' and request.POST.get('checkoutrequest'):
    tobj      = Transaction.objects.all().filter(uid_id=user.id,success=False)
    for i in tobj:
        iobj = Inventory.objects.get(pk=i.pid_id)
        print(type(iobj.quantity),iobj.quantity)
        if iobj.quantity > 0 and (iobj.quantity - i.quantity_r) > 0:
            i.success = True
            i.save()
            iobj.quantity = iobj.quantity - i.quantity_r
            iobj.save()

    cart      = Transaction.objects.all().filter(uid_id = user.id,success=False)
    context['cart'] = cart
    return render(request,'home/index.html', context)

if request.method == 'POST':
    pid      = request.POST['pid']
    print("PID : ",pid)
    iobj      = Inventory.objects.get(pk=pid)
    cname     = request.POST['cname']
    qreq     = request.POST['qreq']
    if int(qreq) <= iobj.quantity:
        tobj      =
        Transaction.objects.create(cust_name=cname,pid_id=iobj.id,uid_id=user.id,quantity_r=qreq)
        tobj.save()

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

@login_required
def transaction(request):
    user = request.user
    tobj = Transaction.objects.all().filter(uid_id = user.id, success=True)
    context = {
        'transactions':tobj,
    }

    if request.method == 'POST' and request.POST.get('tiddel'):

```

```

tid = int(request.POST.get('tiddel'))
Transaction.objects.get(pk=tid).delete()
tobj = Transaction.objects.all().filter(uid_id = user.id, success=True)
context['transactions'] = tobj
return render(request,'home/index.html', context)

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

@login_required
def report(request):
    user = request.user
    inventory = Inventory.objects.all()
    tobj = Transaction.objects.all().filter(uid_id = user.id, success=True)
    context = {
        'report':1,
        'inventory_report':inventory,
        'transaction_report':tobj,
        'user':request.user
    }
    if request.method == 'POST' and request.POST.get('ireport'):
        return rendering.render_to_pdf_response(request, 'home/ipdf.html', context,
                                                using=None, download_filename=None, content_type='application/pdf',
                                                response_class=HttpResponse)

    if request.method == 'POST' and request.POST.get('treport'):
        return rendering.render_to_pdf_response(request, 'home/tpdf.html', context,
                                                using=None, download_filename=None, content_type='application/pdf',
                                                response_class=HttpResponse)
    return render(request,'home/index.html', context)

def chart(request):
    listOfSuppliers = Supplier.objects.all()
    dataOfProducts = []
    for i in listOfSuppliers:
        temp = Inventory.objects.all().filter(supplier_id = i.id)
        dataOfProducts.append(sum([ j.quantity * j.selling_price for j in temp]))
    print(dataOfProducts)
    listOfSuppliers = [i.sname for i in listOfSuppliers]
    context = {
        'listOfSuppliers':listOfSuppliers,
        'dataOfProducts':dataOfProducts,
    }
    return render(request, 'home/chart.html',context)

```

Users App :**Views :**

```

from django.shortcuts import render, redirect, reverse
from django.contrib.auth import authenticate, login, logout
# Create your views here.
from users.forms import LoginForm

```

```

def index(request):
    if request.method == 'POST':
        loginform = LoginForm(request.POST)
        if loginform.is_valid():
            username = loginform.cleaned_data['username']
            password = loginform.cleaned_data['password']
            user = authenticate(username = username, password = password)
            if user is not None:
                login(request, user)
                return redirect(reverse('home'))
            else:
                context = {
                    'form':loginform,
                    'error':'Could not login, Please try again...',
                }
                return render(request, 'users/index.html', context)
    loginform = LoginForm()
    context = {
        'form' : loginform,
    }
    return render(request,'users/index.html', context)

def logout_user(request):
    logout(request)
    return redirect(reverse('login'))

```

Url for routing :

```

from django.urls import path
from users import views

```

```

urlpatterns = [
    path('login/',views.index,name='login'),
    path('logout/', views.logout_user, name = "logout"),
]

```

CONCLUSION :

Hence, we successfully created sales and inventory management system using various things learned during the entire dbms course such as DDL, DML, DCL commands as well as View, Triggers etc. Our system can be used by Inventory manager to add or delete inventory items as well as supplier details.

And it can also be used by Sales persons to do various transactions.

Through chart section we can view graph analytics.

Name : Yogesh D. Kamat

SPIT Batch E

UID : 2018240066