PES UNIVERSITY EC Campus
1 KM before Electronic City, Hosur Road,
Bangalore-100

A Project Report on

SUPERMARKET MANAGEMENT SYSTEM

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

For the Academic year 2020

by

AJITESH NAIR (PES2201800681)
PURUSHOTHAM REDDY (PES2201800473)
NIKHIL KARLE (PES2201800642)
SAGAR S (PES2201800343)

CONTENTS

i. Title page

ii. Acknowledgement

iii. Certificate

iv. Contents

1. Introduction

   1.1 Purpose of this document

   1.2 Overview

   1.3 Business Context

2. General Description

   2.1 Product Functions

   2.2 Intended Users

3. Functional Requirements

4. User Interfaces

5. Limitations

6. Software Requirements

7.Code

v. Conclusion and Future Scope

vi. Bibliography/References

TEAM-1

WEB FRAMEWORKS PROJECT

SUPERMARKET MANAGEMENT SYSTEM

Acknowledgement

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this project gives us much pleasure. We would like to show our gratitude to Dr N Mehala, Course Instructor, PES University for giving us a good guideline for the project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in completing this project.

Many people, especially our classmates and team members themselves, have made valuable comment suggestions on this proposal which gave us inspiration to improve our project. We thank all the people for their help directly and indirectly to complete our project.

PES UNIVERSITY EC Campus

1 KM before Electronic City, Hosur Road,

Bangalore-100

Department of Computer Science Engineering

CERTIFICATE

Certified that the project work entitled Supermarket Management System carried out by AJITESH NAIR USN PES2201800681, a bonafide student of **IV semester** in partial fulfillment for the award of **Bachelor of Engineering** in PES University during the year 2020.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the course **Web Frameworks**.

……..	……..
GUIDE	**Dr. Sandesh**
Dr. N MEHALA	**HOD, CSE**

Declaration

I hereby declare that the project entitled **"SuperMarket Management System"** submitted for Bachelor of Computer Science Engineering degree is my original work and the project has not formed the basis of the awards of any degree, associate ship, fellowship or any other similar titles.

Signature of the Student: AJITESH NAIR

Place: BANGALORE

Date:

PES UNIVERSITY EC Campus

1 KM before Electronic City, Hosur Road,

Bangalore-100

Department of Computer Science Engineering

CERTIFICATE

Certified that the project work entitled Supermarket Management System carried out by

J.P PURUSHOTHAMA REDDY USN PES2201800473 a bonafide student of **IV semester** in partial fulfillment for the award of **Bachelor of Engineering** in PES University during the year 2020.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the course **Web Frameworks**.

……..	……..
GUIDE	**Dr. Sandesh**
Dr. N MEHALA	**HOD, CSE**

Declaration

I hereby declare that the project entitled **"SuperMarket Management System"** submitted for Bachelor of Computer Science Engineering degree is my original work and the project has not formed the basis of the awards of any degree, associate ship, fellowship or any other similar titles.

Signature of the Student: J.P PURUSHOTHAMA REDDY

Place: BANGALORE

Date:

PES UNIVERSITY EC Campus

1 KM before Electronic City, Hosur Road,

Bangalore-100

Department of Computer Science Engineering

CERTIFICATE

Certified that the project work entitled Supermarket Management carried out by NIKHIL KARLE USN PES2201800642, a bonafide student of **IV semester** in partial fulfillment for the award of **Bachelor of Engineering** in PES University during the year 2020.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the course **Web Frameworks**.

……..	……..
GUIDE	**Dr. Sandesh**
Dr. N MEHALA	**HOD, CSE**

Declaration

I hereby declare that the project entitled **"SuperMarket Management System"** submitted for Bachelor of Computer Science Engineering degree is my original work and the project has not formed the basis of the awards of any degree, associate ship, fellowship or any other similar titles.

Signature of the Student: NIKHIL KARLE

Place: BANGALORE

Date:

PES UNIVERSITY EC Campus

1 KM before Electronic City, Hosur Road,

Bangalore-100

Department of Computer Science Engineering

CERTIFICATE

Certified that the project work entitled Supermarket Management carried out by SAGAR S USN PES2201800343, a bonafide student of **IV semester** in partial fulfillment for the award of **Bachelor of Engineering** in PES University during the year 2020.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the course **Web Frameworks**.

……..	……..
GUIDE	**Dr. Sandesh**
Dr. N MEHALA	**HOD, CSE**

Declaration

I hereby declare that the project entitled **"SuperMarket Management System"** submitted for Bachelor of Computer Science Engineering degree is my original work and the project has not formed the basis of the awards of any degree, associate ship, fellowship or any other similar titles.

Signature of the Student: SAGAR S

Place: BANGALORE

Date:

1. Introduction

1.1 Purpose of this document

Overview of this project along with functionalities,intended users ,user interface and software requirements.

1.2 Overview

Supermarket management system is the system where all the aspects related to the proper management of supermarkets are done. These aspects involve managing information about the various products, staff, managers, customers, billing etc. This system provides an efficient way of managing the supermarket information. Also allows the customer to purchase and pay for the items purchased.

This project is based on the sales transaction and billing of items in a supermarket. The first activity is based on adding the items to the system along with the rate which are present in the supermarket and the name of the items which the supermarket will agree to sell. This authority is given only to the product manager. Any modifications to be done in the item name and the rate can be done only by him. He also has the right to delete any item. As the customer buys the products and comes to the billing counter, the user is supposed to enter the item name he purchased and the quantity of the item he had purchased.

This study is to produce software which manages the sales activity done in a supermarket, maintaining the stock details, maintaining the records of the sales done for a particular month/year,managing employees working for the store etc. The users will consume less time in calculation and the sales activity will be completed within a fraction of seconds whereas manual system will make the user to write it down which is a long procedure and so paperwork will be reduced and the user can spend more time on monitoring the supermarket. The project will be user friendly and easy to use.

1.3 Business Context

Supermarket chains or individual stores are the targeted audience.They could use it to manage their chain of stores and maintain clear records and analyse their sales and increase their profit.

Since everything is online it would be convenient to sync all the data across various stores at different locations.

2. General Description

2.1 Product Functions

-Manage employees working for a store.

-Net income and expenditure monitoring.

-Maintain record of all the products sold in the store.

-Monitor stock of different products.

-Add or remove products,remove expired products.

-Users can search for details of specific products.

-Bill generation.

Distinguishable feature from other products in market:

-Sales and revenue analytics

-Recommend Changes to increase profits

2.2 Intended Users

-Store Admin

-Product Managers

-End Users

3. Functional Requirements

-Store Admin

He should be able to

-Manage employees working for a store.

-Net income and expenditure monitoring.

-Product Managers

He should be able to

-Maintain record of all the products sold in the store.

-Monitor stock of different products.

-Add or remove products,remove expired products.

-End Users

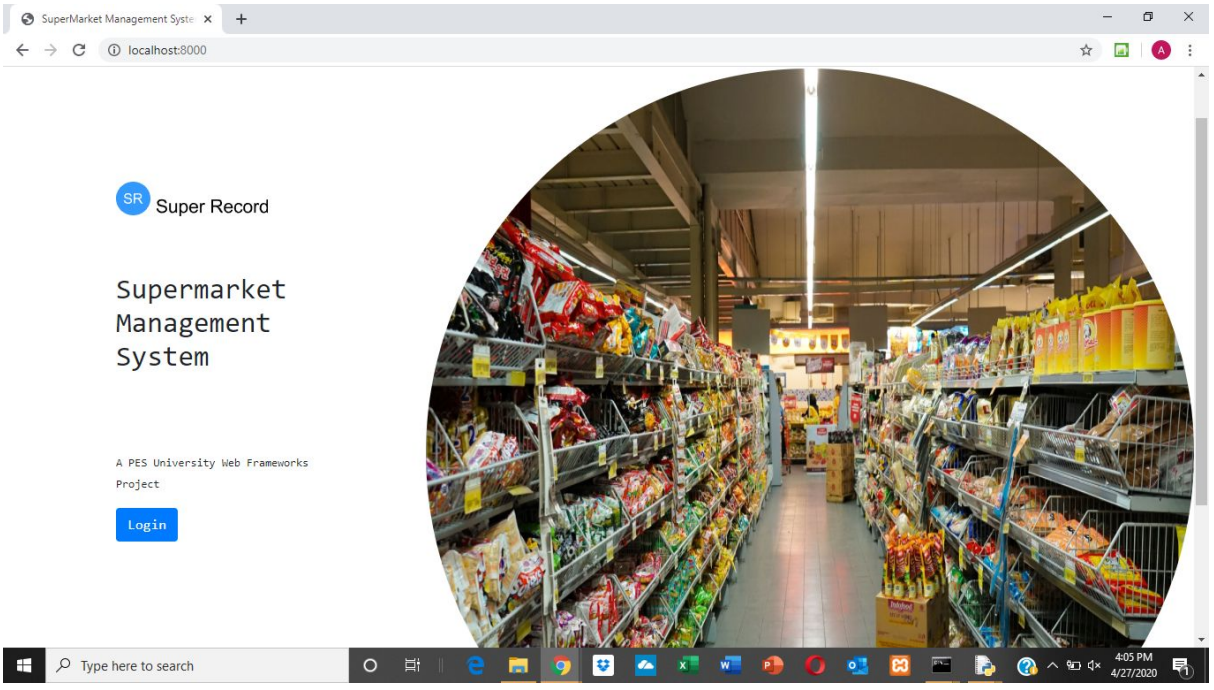
He should be able to

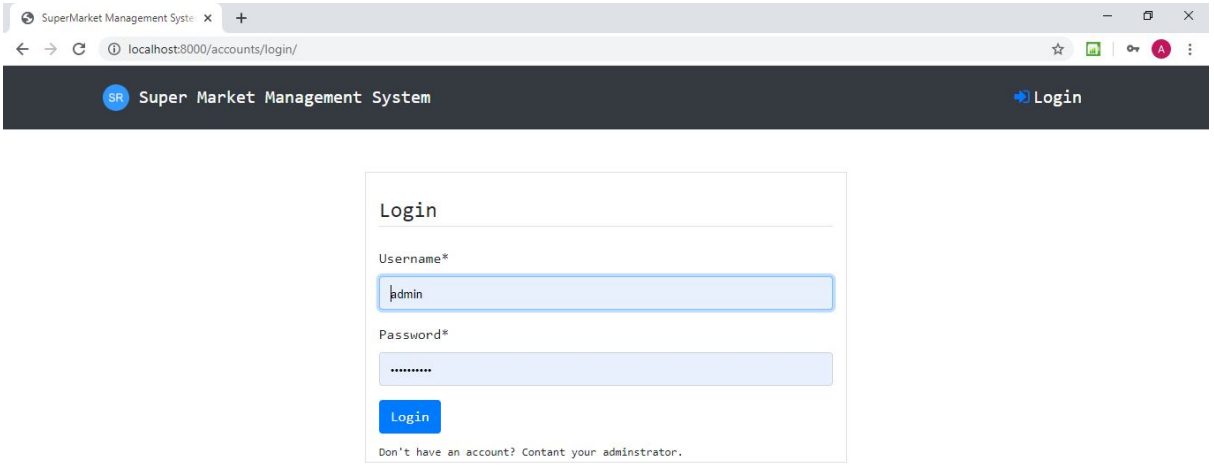
-Search for details of specific products.

-Bill generation.

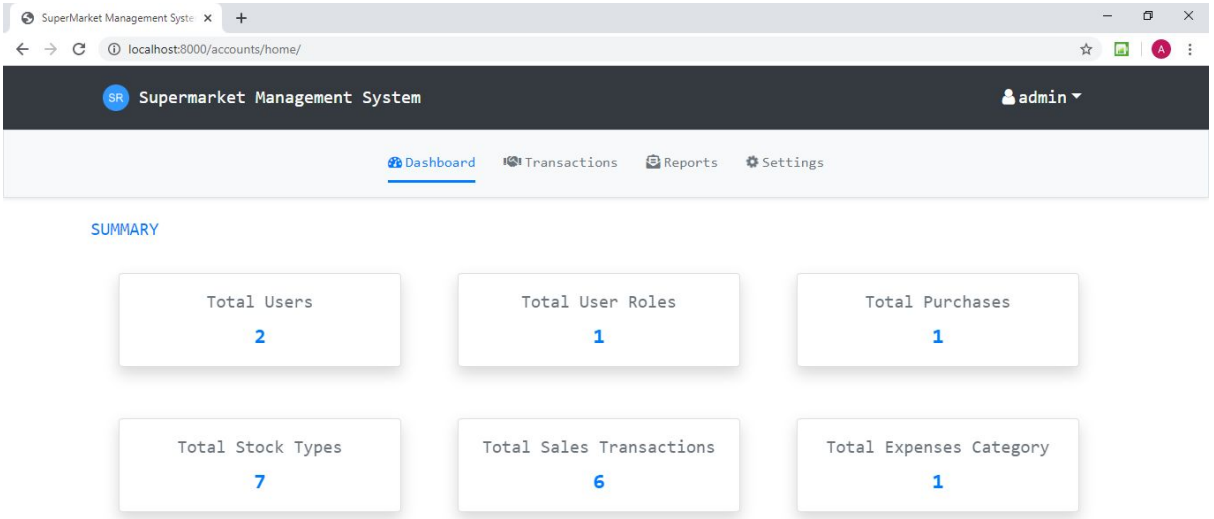
4. User Interface

Simple user interface which is easy to navigate and understand.A basic prototype would be

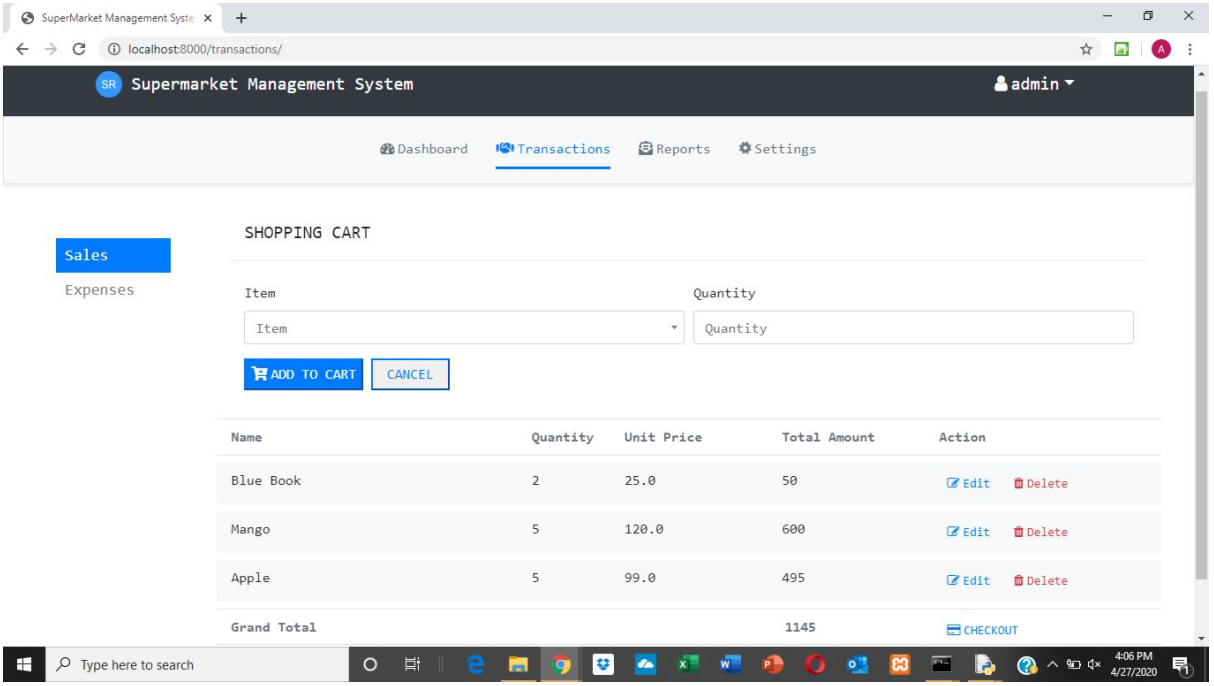


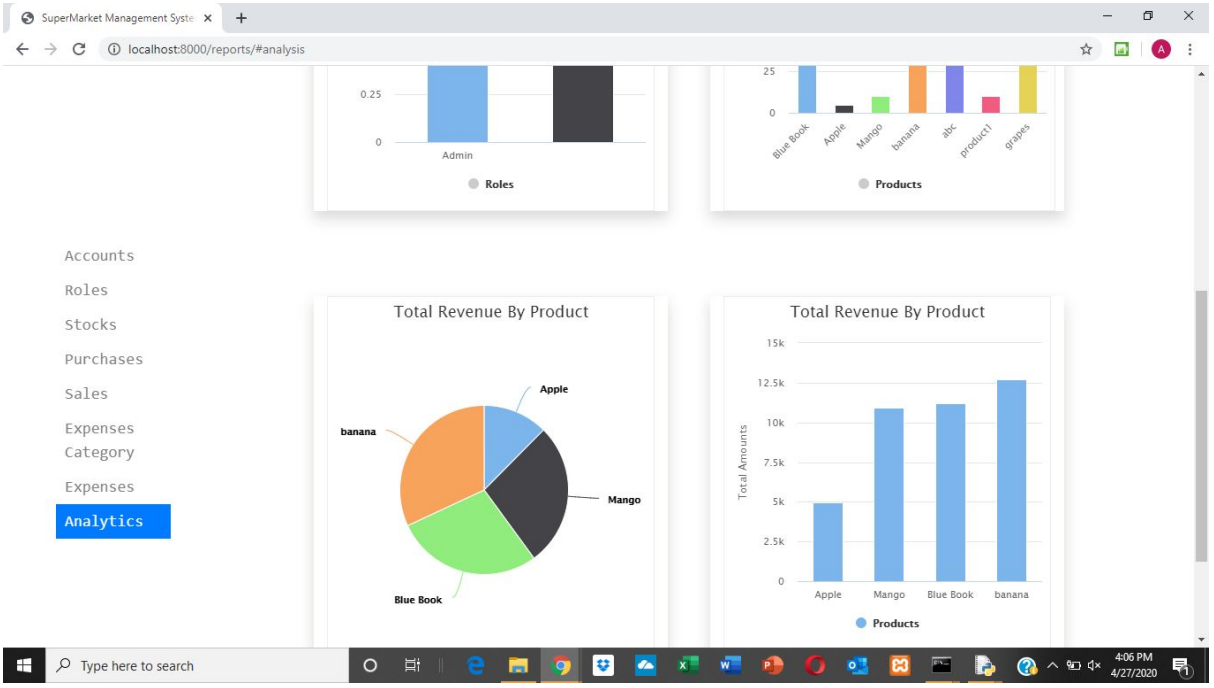

SuperMarket Management Syste
localhost:8000/accounts/login/
Super Market Management System
Login
Login
Username*
admin
Password*
Login
Don't have an account? Contant your adminstrator.

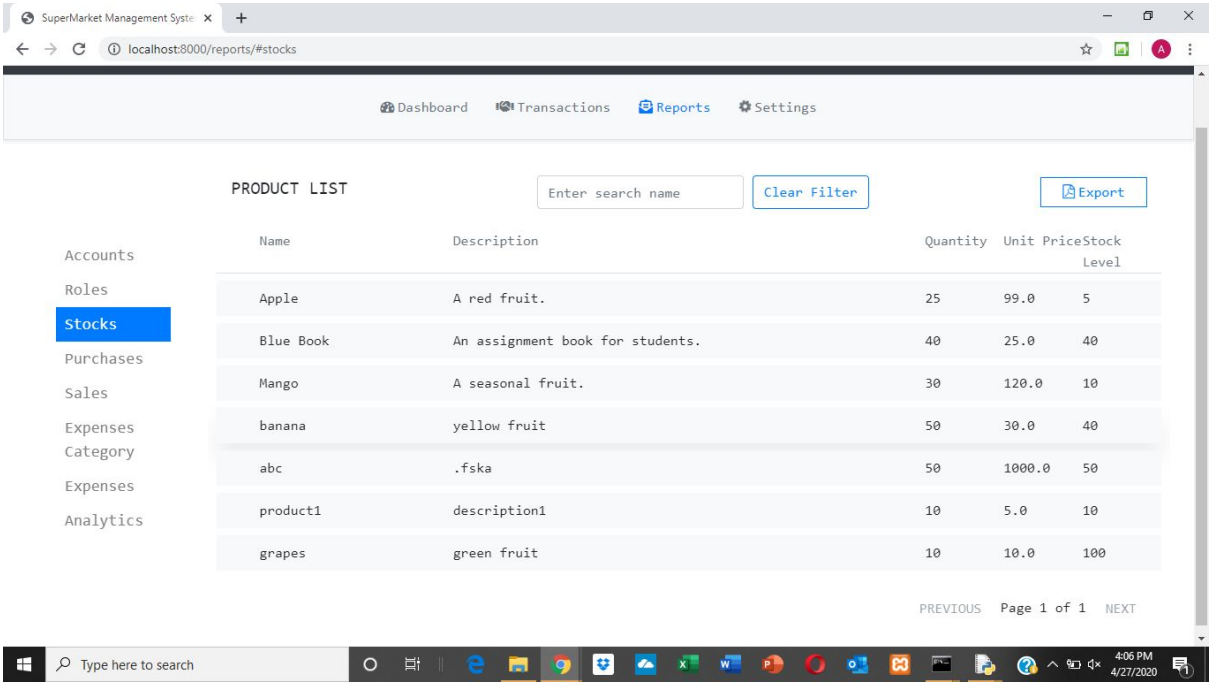

Type here to search
4:05 PM
4/27/2020

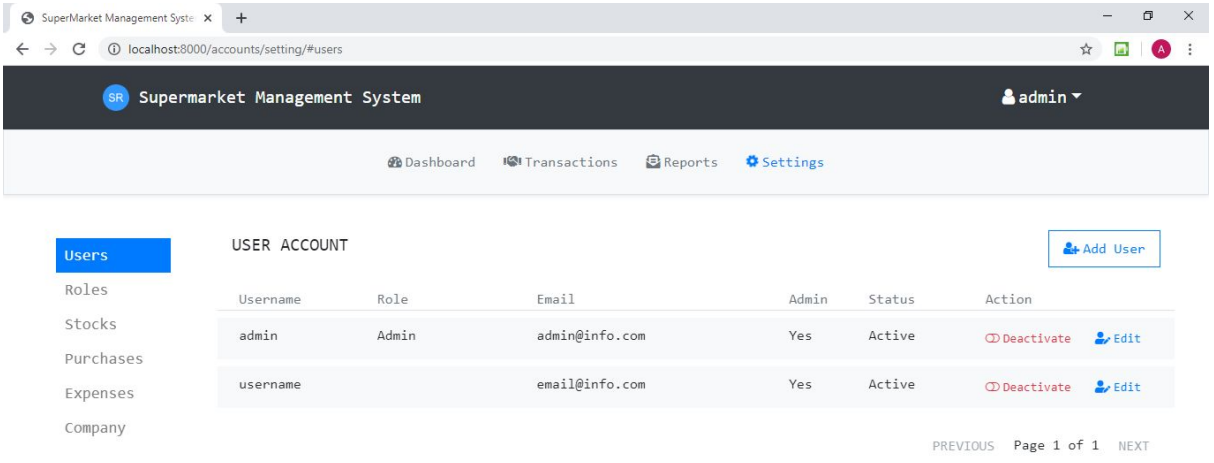

SuperMarket Management Syste
localhost:8000/accounts/home/
Supermarket Management System
admin
Dashboard
Transactions
Reports
Settings
SUMMARY
Total Users
2
Total User Roles
1
Total Purchases
1
Total Stock Types
7
Total Sales Transactions
6
Total Expenses Category
1

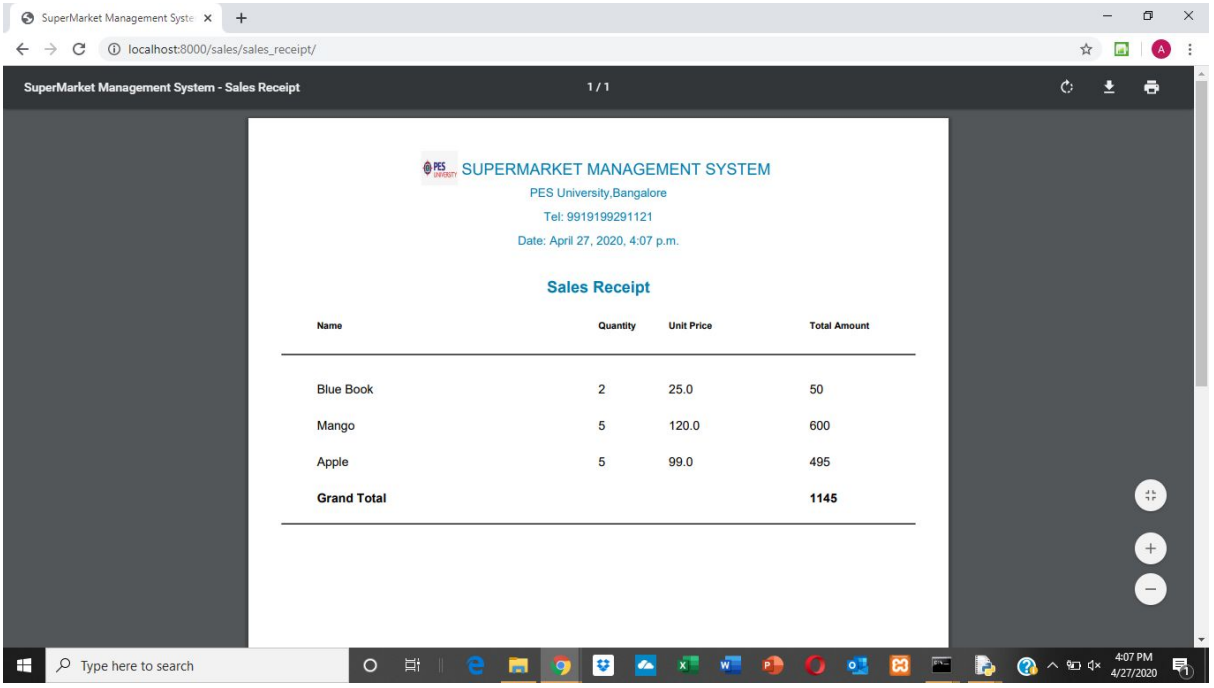

localhost:8000/reports/
Type here to search
4:05 PM
4/27/2020


SuperMarket Management Syste
localhost:8000/transactions/
Supermarket Management System
admin
Dashboard
Transactions
Reports
Settings
Sales
Expenses
SHOPPING CART
Item
Quantity
Item
Quantity
ADD TO CART
CANCEL
Name
Quantity
Unit Price
Total Amount
Action
Blue Book
2
25.0
50
Edit
Delete
Mango
5
120.0
600
Edit
Delete
Apple
5
99.0
495
Edit
Delete
Grand Total
1145
CHECKOUT
Type here to search
4:06 PM
4/27/2020

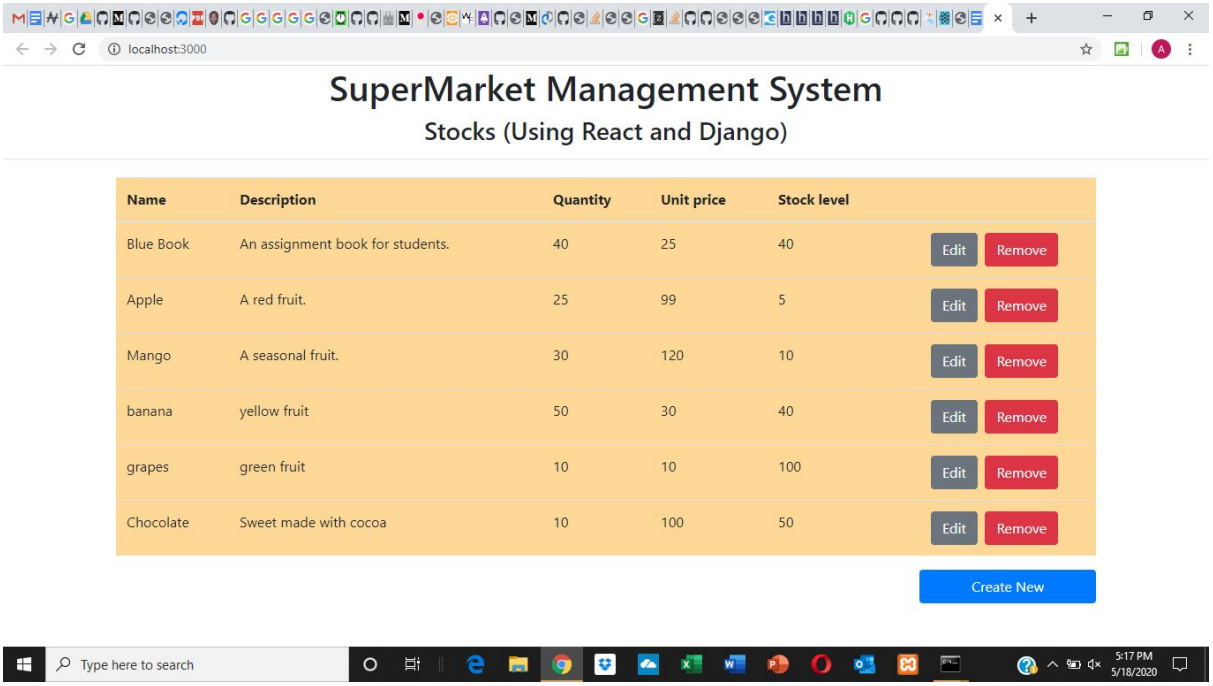

SuperMarket Management Syste
localhost:8000/reports/#analysis
0.25
0
Admin
Roles
25
0
Blue Book
Apple
Mango
banana
abc
product1
grapes
Products
Accounts
Roles
Stocks
Purchases
Sales
Expenses Category
Expenses
Analytics
Total Revenue By Product
Apple
banana
Mango
Blue Book
Total Revenue By Product
15k
12.5k
10k
7.5k
5k
2.5k
0
Total Amounts
Apple
Mango
Blue Book
banana
Products
Type here to search
4:06 PM
4/27/2020

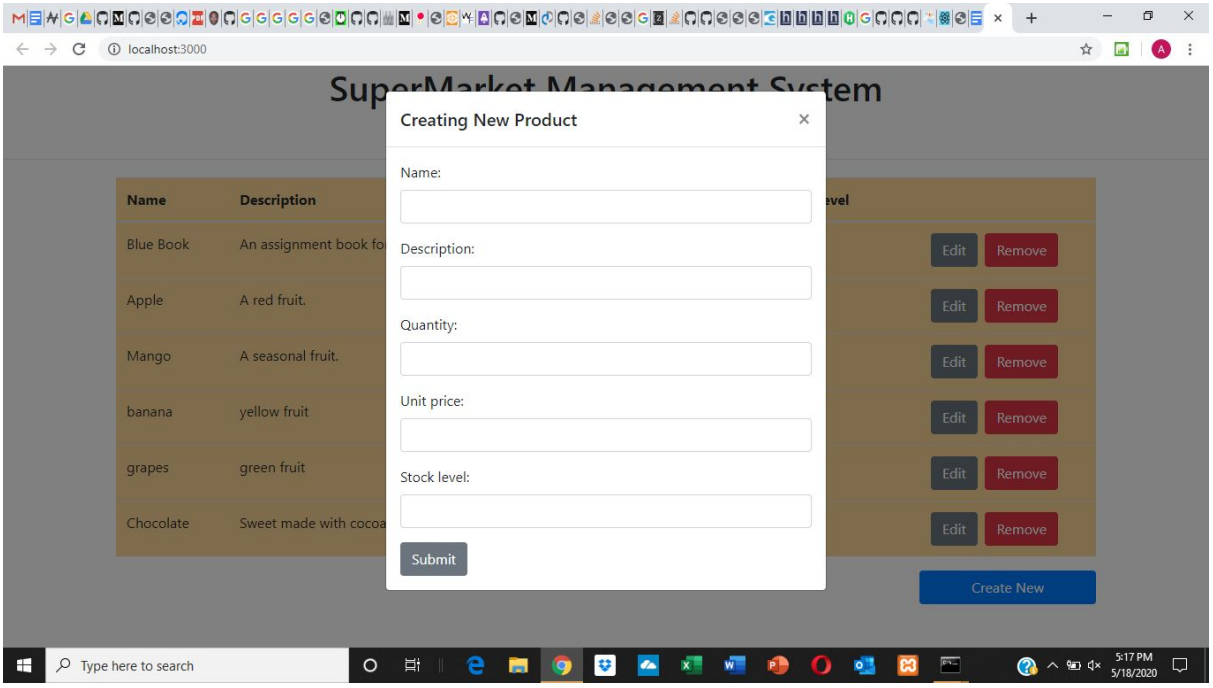

SuperMarket Management Syste
localhost:8000/reports/#stocks
Dashboard
Transactions
Reports
Settings
PRODUCT LIST
Enter search name
Clear Filter
Export
Accounts
Roles
Stocks
Purchases
Sales
Expenses
Category
Expenses
Analytics
Name
Description
Quantity
Unit Price
Stock Level
Apple
A red fruit.
25
99.0
5
Blue Book
An assignment book for students.
40
25.0
40
Mango
A seasonal fruit.
30
120.0
10
banana
yellow fruit
50
30.0
40
abc
.fska
50
1000.0
50
product1
description1
10
5.0
10
grapes
green fruit
10
10.0
100
PREVIOUS
Page 1 of 1
NEXT
Type here to search
4:06 PM
4/27/2020





React Component





5.Limitations

-Cannot manage promotions.

-Internet access required to access cross store details.

-Initial Setup required.

6.Software Requirements

-Windows 10 or newer

-Web browser

-Django

-Python 3.7 or newer

-Postgresql

-PgAdmin4

-React

-django rest framework

7.Code (only main parts are included)

Django + React Component

Header.js

```
import React, { Component } from "react";

class Header extends Component {
  render() {
    return (
      <div className="text-center">
        <h1>SuperMarket Management System </h1>
        <h3>Stocks (Using React and Django)</h3>
        <hr />
      </div>
    );
  }
}
```

```
export default Header;
```

Home.js

```
import React, { Component } from "react";
import { Col, Container, Row } from "reactstrap";
import ProductList from "./ProductList";
import NewProductModal from "./NewProductModal";

import axios from "axios";

import { API_URL } from "../constants";

class Home extends Component {
  state = {
    products: []
  };

  componentDidMount() {
    this.resetState();
  }

  getProducts = () => {
    axios.get(API_URL).then(res => this.setState({ products: res.data }));
  };

  resetState = () => {
    this.getProducts();
  };

  render() {
    return (
      <Container style={{ marginTop: "20px" }}>
```

```
        <Row>
          <Col>
            <ProductList
              products={this.state.products}
              resetState={this.resetState}
            />
          </Col>
        </Row>
        <Row>
          <Col>
            <NewProductModal create={true} resetState={this.resetState} />
          </Col>
        </Row>
      </Container>
    );
  }
}

export default Home;
```

NewProductForm.js

```
import React from "react";
import { Button, Form, FormGroup, Input, Label } from "reactstrap";

import axios from "axios";

import { API_URL } from "../constants";

class NewProductForm extends React.Component {
  state = {
    pk: 0,
    name: "",
```

```
  description: "",
  quantity: "",
  unit_price: "",
  stock_level: ""
};

componentDidMount() {
  if (this.props.product) {
    const { pk, name, description, quantity, unit_price, stock_level } = this.props.product;
    this.setState({ pk, name, description, quantity, unit_price, stock_level });
  }
}

onChange = e => {
  this.setState({ [e.target.name]: e.target.value });
};

createproduct = e => {
  e.preventDefault();
  axios.post(API_URL, this.state).then(() => {
    this.props.resetState();
    this.props.toggle();
  });
};

editProduct = e => {
  e.preventDefault();
  axios.put(API_URL + this.state.pk, this.state).then(() => {
    this.props.resetState();
    this.props.toggle();
  });
};
```

```
defaultIfEmpty = value => {
  return value === "" ? "" : value;
};

render() {
  return (
    <Form onSubmit={this.props.product ? this.editProduct : this.createProduct}>
      <FormGroup>
        <Label for="name">Name:</Label>
        <Input
          type="text"
          name="name"
          onChange={this.onChange}
          value={this.defaultIfEmpty(this.state.name)}
        />
      </FormGroup>
      <FormGroup>
        <Label for="description">Description:</Label>
        <Input
          type="text"
          name="description"
          onChange={this.onChange}
          value={this.defaultIfEmpty(this.state.description)}
        />
      </FormGroup>
      <FormGroup>
        <Label for="quantity">Quantity:</Label>
        <Input
          type="text"
          name="quantity"
          onChange={this.onChange}
```

```
          value={this.defaultIfEmpty(this.state.quantity)}
        />
      </FormGroup>
      <FormGroup>
        <Label for="unit_price">Unit price:</Label>
        <Input
          type="text"
          name="unit_price"
          onChange={this.onChange}
          value={this.defaultIfEmpty(this.state.unit_price)}
        />
      </FormGroup>
       <FormGroup>
        <Label for="stock_level">Stock level:</Label>
        <Input
          type="text"
          name="stock_level"
          onChange={this.onChange}
          value={this.defaultIfEmpty(this.state.stock_level)}
        />
      </FormGroup>
      <Button>Submit</Button>
    </Form>
  );
 }
}

export default NewProductForm;
```

Django Component

Stocks Folder

Models.py

```
from django.db import models
from accounts.models import User

class Product(models.Model):
    name = models.CharField(max_length=50,unique=True)
    description = models.TextField(max_length=100,unique=True)
    quantity = models.PositiveIntegerField()
    unit_price = models.FloatField()
    stock_level = models.PositiveIntegerField()
    created_by_id=models.PositiveIntegerField(default='1')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        """Returns a string representation of this Product."""
        return self.name
```

Serializers.py

```
from rest_framework import serializers
from .models import Product

class ProductSerializer(serializers.ModelSerializer):

    class Meta:
        model = Product
        fields =
('pk','name','description','quantity','unit_price','stock_level','created_by_id','created_at','update
d_at')
```

Views.py

```
from django.http import HttpResponseRedirect
from django.utils.decorators import method_decorator
from django.urls import reverse_lazy
from django.views.generic import (
    ListView, UpdateView, DetailView, DeleteView, CreateView )
from easy_pdf.views import PDFTemplateView
from .forms import ProductCreationForm, EditProductForm
from .models import Product
from decorators.decorators import group_required
from helpers.generate_pdf import generate_report




from django.shortcuts import render
from rest_framework.response import Response
from rest_framework.decorators import api_view
from rest_framework import status
from rest_framework import generics
from .models import Product
from .serializers import *

@api_view(['GET', 'POST'])
def stocks_list(request):
    if request.method == 'GET':
        data = Product.objects.all()

        serializer = ProductSerializer(data, context={'request': request}, many=True)
        return Response(serializer.data)
    elif request.method == 'POST':
        serializer = ProductSerializer(data=request.data)
```

```
        if serializer.is_valid():
            serializer.save()
            return Response(status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['PUT', 'DELETE'])
def stocks_detail(request, pk):
    try:
        product = Product.objects.get(pk=pk)
    except Product.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method == 'PUT':
        serializer = ProductSerializer(product, data=request.data,context={'request': request})
        if serializer.is_valid():
          serializer.save()
          return Response(status=status.HTTP_204_NO_CONTENT)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    elif request.method == 'DELETE':
        product.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)

class ProductPDFView(PDFTemplateView):
    template_name = 'stocks/product_report.html'

    def get_context_data(self, **kwargs):
        dataset = Product.objects.values(
                            'name','description',
                            'quantity','unit_price',
                            'stock_level').order_by('id')
        context = super(ProductPDFView, self).get_context_data(
```

```
        pagesize='A4',
        title='Stock Report',
        **kwargs
    )

    return generate_report(context, dataset, 'Products List')
```

Project Folder

Views.py

```
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
from django.views.generic import TemplateView, ListView
from company.models import Company
from accounts.models import User, Role
from stocks.models import Product
from purchase.models import Purchase
from sales.models import Sales
from expenses.models import ExpenseCategory, Expenses

class IndexView(TemplateView):
    template_name = 'accounts/index.html'

@method_decorator(login_required, name='dispatch')
class AccountReportView(TemplateView):
    template_name = 'reports/reports.html'

@method_decorator(login_required, name='dispatch')
class AccountListView(ListView):
    queryset = User.objects.all().order_by('id')
    paginate_by = 10
```

```
    context_object_name = 'account_list'
    template_name = 'reports/accounts.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['company'] = Company.objects.all().values()[0]

        return context

class SearchUserView(ListView):
    queryset = User.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'account_list'
    template_name = 'reports/accounts.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['account_list'] =
User.objects.filter(username__icontains=self.request.GET.get('q', None))

        return context

@method_decorator(login_required, name='dispatch')
class RolesListView(ListView):
    queryset = Role.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'roles'
    template_name = 'reports/roles.html'

@method_decorator(login_required, name='dispatch')
class StocksListView(ListView):
    queryset = Product.objects.all().order_by('id')
```

```
    paginate_by = 10
    context_object_name = 'stocks'
    template_name = 'reports/stocks.html'

class SearchStockView(ListView):
    queryset = Product.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'stocks'
    template_name = 'reports/stocks.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['stocks'] = Product.objects.filter(name__icontains=self.request.GET.get('q',
None))

        return context

@method_decorator(login_required, name='dispatch')
class PurchaseListView(ListView):
    queryset = Purchase.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'purchases'
    template_name = 'reports/purchases.html'

class SearchPurchaseView(ListView):
    queryset = Purchase.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'purchases'
    template_name = 'reports/purchases.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
```

```
        context['purchases'] = Purchase.objects.filter(name__icontains=self.request.GET.get('q',
None))

        return context

@method_decorator(login_required, name='dispatch')
class SalesListView(ListView):
    queryset = Sales.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'sales_list'
    template_name = 'reports/sales.html'

class SearchSalesView(ListView):
    queryset = Sales.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'sales_list'
    template_name = 'reports/sales.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['sales_list'] = Sales.objects.filter(name__icontains=self.request.GET.get('q',
None))

        return context

@method_decorator(login_required, name='dispatch')
class CategoryListView(ListView):
    queryset = ExpenseCategory.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'category_list'
    template_name = 'reports/category.html'
```

```
@method_decorator(login_required, name='dispatch')
class ExpensesListView(ListView):
    queryset = Expenses.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'expenses_list'
    template_name = 'reports/expenses.html'

class SearchExpensesView(ListView):
    queryset = Expenses.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'expenses_list'
    template_name = 'reports/expenses.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['expenses_list'] =
Expenses.objects.filter(description__icontains=self.request.GET.get('q', None))

        return context

@method_decorator(login_required, name='dispatch')
class AnalysisView(TemplateView):
    template_name = 'reports/analysis.html'

@method_decorator(login_required, name='dispatch')
class TransactionsView(TemplateView):
    template_name = 'transactions/transactions.html'
```

Urls.py

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

```
from django.conf import settings
from django.conf.urls.static import static
from helpers import graphs as graph_view
from . import views as main_view
from django.urls import path,re_path
from stocks import views as product_view
from django.conf.urls import url

urlpatterns = [
    path('', include('stocks.urls')),
    path('admin/', admin.site.urls),
    path('', main_view.IndexView.as_view(), name='index'),
    path('reports/', main_view.AccountReportView.as_view(), name='reports'),
    path('reports/accounts/', main_view.AccountListView.as_view(), name='accounts'),
    path('reports/roles/', main_view.RolesListView.as_view(), name='roles'),
    path('reports/stocks/', main_view.StocksListView.as_view(), name='stock_list'),
    path('reports/purchases/', main_view.PurchaseListView.as_view(), name='purchases_list'),
    path('reports/sales/', main_view.SalesListView.as_view(), name='sales_list'),
    path('reports/categories/', main_view.CategoryListView.as_view(), name='categories'),
    path('reports/expenses/', main_view.ExpensesListView.as_view(), name='expenses_list'),
    path('reports/analysis/', main_view.AnalysisView.as_view(), name='analysis'),
    path('company/', include('company.urls')),
    path('accounts/', include('accounts.urls')),
    path('stocks/', include('stocks.urls')),
    path('purchases/', include('purchase.urls')),
    path('sales/', include('sales.urls')),
    path('expenses/', include('expenses.urls')),
    path('transactions/', main_view.TransactionsView.as_view(), name='transactions'),
    path('graphs/roles/', graph_view.roles_graph, name='roles_data'),
    path('graphs/stocks/', graph_view.stocks_graph, name='stocks_data'),
    path('graphs/sales/', graph_view.sales_graph, name='sales_data'),
    path('graphs/expenses/', graph_view.expenses_graph, name='expenses_data'),
```

```
    path('search/user/', main_view.SearchUserView.as_view(), name='search_user'),
    path('search/expense/', main_view.SearchExpensesView.as_view(),
name='search_expense'),
    path('search/sale/', main_view.SearchSalesView.as_view(), name='search_sale'),
    path('search/purchase/', main_view.SearchPurchaseView.as_view(),
name='search_purchase'),
    path('search/stock/', main_view.SearchStockView.as_view(), name='search_stock')
] + static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

Accounts App

Models.py

```
from django.db import models
from django.contrib.auth.models import (
    BaseUserManager,
    AbstractBaseUser,
    PermissionsMixin,
    Group,
)


class UserManager(BaseUserManager):
    """Class to manage the creation of user objects"""

    def create_user(self, username, email, password=None):
        """Creates and returns a user object
        Arguments:
        username: the string to use as username
        email: the string to use as email
        password: the string to use as password

        Optionals:
        is_staff: Boolean to indicate a user is staff or not
        is_admin: Boolean to indicate a user is an admin or not
        is_active: Boolean to indicate a user can login or not
```

```
    Return:
        A user object
    """
    if not username:
        raise ValueError('Users must have a username')

    if not email:
        raise ValueError('Users must have an email address')

    if not password:
        raise ValueError('Users must have a password')

    user = self.model(username=username,email = self.normalize_email(email),)
    user.set_password(password)
    user.is_active=True
    user.save(using=self._db)
    return user

def create_superuser(self, username, email, password):
    """Creates an admin user object
    Arguments:
    username: the string to use as username
    email: the string to use as email
    password: the string to use as password

    Return:
        A user object
    """
    user = self.create_user(username, email, password=password)
    user.is_admin=True
    user.save(using=self._db)
```

```
        return user


class User(AbstractBaseUser, PermissionsMixin):
    """
    Class for creating user implementing the abstract
    base user and the permission class
    """
    username = models.CharField(max_length=255, unique=True)
    email = models.EmailField(verbose_name='email address', max_length=255,unique=True)
    is_active = models.BooleanField(default=True)
    is_admin = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    USERNAME_FIELD = 'username'
    REQUIRED_FIELDS = ['email']

    objects = UserManager()

    def __str__(self):
        """Returns a string representation of this `User`."""
        return self.username

    def delete(self, using=None, keep_parents=False):
        self.is_active ^= True
        self.save()

    def get_full_name(self):
        # The user is identified by their email address
        return self.username
```

```
    def get_short_name(self):
        # The user is identified by their email address
        return self.username

    def has_perm(self, perm, obj=None):
        "Does the user have a specific permission?"
        return True

    def has_module_perms(self, app_label):
        "Does the user have permissions to view the app `app_label`?"
        return True

    @property
    def is_staff(self):
        "Is the user a member of staff?"
        return self.is_admin

class Role(Group):
    description = models.TextField(max_length=100, unique=True)

    def __str__(self):
        """Returns a string representation of this `Role`."""
        return self.name
```

Views.py

```
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
from django.urls import reverse_lazy
from django.views.generic import (
    ListView, UpdateView, DetailView, DeleteView, CreateView, TemplateView )
from .forms import ( UserRegisterForm, EditProfileForm,
```

```
    RoleCreationForm, EditRoleForm )
from django.contrib.auth.forms import PasswordChangeForm
from django.contrib.auth.views import PasswordChangeView
from django.contrib.auth.models import Group
from company.models import Company
from .models import User, Role
from stocks.models import Product
from sales.models import Sales
from purchase.models import Purchase
from expenses.models import ExpenseCategory, Expenses
from .permissions import assign_permissions, remove_permissions
from decorators.decorators import group_required
from easy_pdf.views import PDFTemplateView
from helpers.generate_pdf import generate_report

decorators = [group_required(['Admin','Manager','General Manager'])]
@method_decorator(decorators, name='dispatch')
class UserListView(ListView):
    queryset = User.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'user_list'
    template_name = 'accounts/user.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['company'] = Company.objects.all().values()[0]

        return context

@method_decorator(login_required, name='dispatch')
class EditProfileView(UpdateView, DetailView):
    template_name = 'accounts/edit.html'
```

```
    pk_url_kwarg = 'id'
    form_class = EditProfileForm
    queryset = User.objects.all()
    success_url = reverse_lazy('setting')
    old_role = []

    def get(self, request, *args, **kwargs):
        self.user = User.objects.get(id=kwargs['id'])

        return super().get(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['roles'] = Group.objects.all().values_list('name', flat=True)

        return context

    def post(self, request, *args, **kwargs):
        user = User.objects.get(id=kwargs['id'])
        self.old_role += user.groups.all().values_list('name', flat=True)
        new_role = request.POST.get('role', None)

        if len(self.old_role) > 0:
            user.groups.remove(Group.objects.get(name=self.old_role[0]))

        if new_role is not None:
            user.groups.add(Group.objects.get(name=new_role))

        return super().post(request, *args, **kwargs)

@method_decorator(login_required, name='dispatch')
class PasswordUpdateView(PasswordChangeView):
```

```
    template_name = 'accounts/change_password.html'
    pk_url_kwarg = 'id'
    form_class = PasswordChangeForm
    queryset = User.objects.all()
    success_url = reverse_lazy('setting')

@method_decorator(decorators, name='dispatch')
class DeactivateView(DeleteView):
    template_name = 'accounts/deactivate.html'
    pk_url_kwarg = 'id'
    queryset = User.objects.all()
    success_url = reverse_lazy('setting')

@method_decorator(login_required, name='dispatch')
class ActivateView(DeleteView):
    template_name = 'accounts/activate.html'
    pk_url_kwarg = 'id'
    queryset = User.objects.all()
    success_url = reverse_lazy('setting')

@method_decorator(decorators, name='dispatch')
class SignUpView(CreateView):
    form_class = UserRegisterForm
    template_name = 'accounts/signup.html'
    success_message = 'Success: Sign up succeeded.'
    success_url = reverse_lazy('setting')

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['roles'] = Group.objects.all().values_list('name', flat=True)

        return context
```

```
    def post(self, request, *args, **kwargs):
        form = self.form_class(request.POST)

        if form.is_valid():
            user = form.save()

            role_name = request.POST['role']
            user.groups.add(Group.objects.get(name=role_name))

            return HttpResponseRedirect(self.success_url)

        return super().post(request, *args, **kwargs)

@method_decorator(decorators, name='dispatch')
class RoleCreationView(CreateView):
    form_class = RoleCreationForm
    template_name = 'accounts/add_role.html'
    success_message = 'Success: Role creation succeeded.'
    success_url = reverse_lazy('setting')

    def post(self, request, *args, **kwargs):
        form = self.form_class(request.POST)

        if form.is_valid():
            role = form.save()

            perm = [request.POST['user_perm']]
            if 'full' in perm:
                assign_permissions(role, perm, full=True)
            else:
                assign_permissions(role, perm)
```

```
            return HttpResponseRedirect(self.success_url)

        return super().post(request, *args, **kwargs)

@method_decorator(decorators, name='dispatch')
class RoleListView(ListView):
    queryset = Role.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'role_list'
    template_name = 'accounts/roles.html'

@method_decorator(decorators, name='dispatch')
class EditRoleView(UpdateView, DetailView):
    template_name = 'accounts/edit_role.html'
    pk_url_kwarg = 'id'
    form_class = EditRoleForm
    queryset = Role.objects.all()
    success_url = reverse_lazy('setting')
    old_perms = []

    def get(self, request, *args, **kwargs):
        self.edit_role = Role.objects.get(id=kwargs['id'])

        return super().get(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['role_perms'] = self.edit_role.permissions.all().values_list('codename', flat=True)

        return context
```

```
    def post(self, request, *args, **kwargs):

        edit_role = Role.objects.get(id=kwargs['id'])
        self.old_perms += edit_role.permissions.all().values_list('codename', flat=True)
        new_perm = request.POST.getlist('user_perm')

        if not new_perm:
            raise ValueError("You must assign atleast 1 permission for the new role")

        if 'full' in new_perm or len(new_perm) == 4:
            edit_role.permissions.remove()
            assign_permissions(edit_role, new_perm, full=True)
        else:
            if len(new_perm) > 1:
                for perm in new_perm:
                    if perm not in self.old_perms:
                        assign_permissions(edit_role, [perm])
                remove_permissions(edit_role, new_perm, self.old_perms)
            else:
                if new_perm not in self.old_perms:
                    assign_permissions(edit_role, new_perm)
                remove_permissions(edit_role, new_perm, self.old_perms)

        return super().post(request, *args, **kwargs)

@method_decorator(decorators, name='dispatch')
class DeleteRoleView(DeleteView):
    template_name = 'accounts/delete_role.html'
    pk_url_kwarg = 'id'
    queryset = Role.objects.all()
    success_url = reverse_lazy('setting')
```

```
@method_decorator(login_required, name='dispatch')
class Home(ListView):
    queryset = User.objects.all().order_by('id')
    paginate_by = 10
    template_name = 'accounts/home.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['users'] = len(User.objects.all())
        context['roles'] = len(Group.objects.all())
        context['stocks'] = len(Product.objects.all())
        context['sales'] = len(Sales.objects.all())
        context['purchases'] = len(Purchase.objects.all())
        context['category'] = len(ExpenseCategory.objects.all())
        context['expenses'] = len(Expenses.objects.all())
        context['company'] = Company.objects.all().values()[0]

        return context

@method_decorator(decorators, name='dispatch')
class Setting(TemplateView):
    template_name = 'accounts/setting.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['company'] = Company.objects.all()

        return context

class AccountPDFView(PDFTemplateView):
    template_name = 'accounts/account_report.html'
```

```
    def get_context_data(self, **kwargs):
        dataset = User.objects.values(
                            'username','groups','email',
                            'is_admin','is_active').order_by('id')
        context = super(AccountPDFView, self).get_context_data(
            pagesize='A4',
            title='Account Report',
            **kwargs
        )
        for data in dataset:
            data['groups'] = Role.objects.get(id=data['groups'])

        return generate_report(context, dataset, 'Accounts List')
```

Urls.py

```
from django.urls import path
from django.contrib.auth import views as auth_views
from accounts import views as user_views

urlpatterns = [
    path('home/', user_views.Home.as_view(), name='home'),
    path('user/', user_views.UserListView.as_view(), name='user'),
    path('roles/', user_views.RoleListView.as_view(), name='roles'),
    path('role/', user_views.RoleCreationView.as_view(), name='role'),
    path('edit_role/<int:id>/', user_views.EditRoleView.as_view(), name='edit_role'),
    path('delete_role/<int:id>/', user_views.DeleteRoleView.as_view(), name='delete_role'),
    path('signup/', user_views.SignUpView.as_view(), name='signup'),
    path('edit/<int:id>/', user_views.EditProfileView.as_view(), name='edit'),
    path('change_password/', user_views.PasswordUpdateView.as_view(),
name='change_password'),
    path('deactivate/<int:id>/', user_views.DeactivateView.as_view(), name='deactivate'),
    path('activate/<int:id>/', user_views.ActivateView.as_view(), name='activate'),
```

```
    path('login/', auth_views.LoginView.as_view(template_name='accounts/login.html'),
name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='accounts/logout.html'),
name='logout'),
    path('setting/', user_views.Setting.as_view(), name='setting'),
    path('account_report/', user_views.AccountPDFView.as_view(), name='account_report')
]
```

Sales app

Forms.py

```
from django import forms
from sales.models import Sales

class SalesCreationForm(forms.ModelForm):
    """
    A form for adding new sales with all required field
    """
    class Meta:
        model = Sales
        fields = ('name','item', 'quantity', 'unit_price', 'total_amount', 'sold_by')

    def save(self, commit=True):
        """
        Save form data to database
        """
        sale = super(SalesCreationForm, self).save(commit=False)

        if commit:
            sale.name = self.cleaned_data.get('name')
            sale.unit_price = self.cleaned_data.get('unit_price')
            sale.total_amount = self.cleaned_data.get('total_amount')
            sale.sold_by = self.cleaned_data.get('sold_by')
            sale.save()
```

```
        return sale


class EditSalesForm(forms.ModelForm):
    """
    A form for editing sales record with all required field
    """
    class Meta:
        model = Sales
        fields = ['item', 'quantity']
```

Models.py

```
from django.db import models
from accounts.models import User
from stocks.models import Product


class Sales(models.Model):
    name = models.CharField(max_length=50)
    item = models.ForeignKey(Product, on_delete=models.DO_NOTHING)
    quantity = models.PositiveIntegerField()
    unit_price = models.FloatField(verbose_name='unit price')
    total_amount = models.PositiveIntegerField(verbose_name='total amount')
    status = models.CharField(max_length=50, default='pending')
    sold_by = models.ForeignKey(User, on_delete=models.DO_NOTHING)
    sold_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        """Returns a string representation of this sale."""
        return self.name
```

Views.py

```
from django.http import HttpResponseRedirect
```

```
from django.shortcuts import render
from django.db.models import Sum
from django.contrib.auth.decorators import login_required
from django.utils.decorators import method_decorator
from django.urls import reverse_lazy
from django.views.generic import (
    ListView, UpdateView, DetailView, DeleteView, CreateView )
from company.models import Company
from stocks.models import Product
from .forms import SalesCreationForm, EditSalesForm
from .models import Sales
from accounts.models import User
from easy_pdf.views import PDFTemplateView
from helpers.generate_pdf import generate_report

@method_decorator(login_required, name="dispatch")
class SalesListView(ListView):
    queryset = Sales.objects.all().order_by('-id')
    paginate_by = 10
    context_object_name = 'sales'
    template_name = 'sales/sales.html'

@method_decorator(login_required, name="dispatch")
class SalesCreationView(CreateView, ListView):
    model=Sales
    form_class = SalesCreationForm
    context_object_name = 'sales_list'
    object_list = []
    template_name = 'sales/add_sales.html'
    success_message = 'Success: Sales creation succeeded.'
    success_url = reverse_lazy('transactions')
```

```
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['sales'] = Sales.objects.filter(status='pending').filter(
        sold_by=self.request.user.id)
    context['total_sales'] = Sales.objects.filter(status='pending').filter(
        sold_by=self.request.user.id).aggregate(Sum('total_amount'))
    context['company'] = Company.objects.all().values()[0]

    return context

def post(self, request, *args, **kwargs):

    if request.method == 'POST' and request.POST.get('item'):
        sale = Product.objects.filter(name=request.POST.get('item')).values()[0]
        data = {
            'name': sale['name'],
            'item': int(sale['id']),
            'quantity': request.POST.get('quantity', 0),
            'unit_price': float(sale['unit_price']),
            'total_amount': float(int(request.POST.get('quantity', 0)) * float(
                sale['unit_price'])),
            'sold_by': request.user.id
        }
        form = self.form_class(data)
        form.name=sale['name']
        form.unit_price=[]

        if form.is_valid():
            form.save()

            return HttpResponseRedirect(self.success_url)
```

```
        return super().post(request, *args, **kwargs)

@method_decorator(login_required, name="dispatch")
class EditSalesView(UpdateView, DetailView):
    template_name = 'sales/edit_sale.html'
    pk_url_kwarg = 'id'
    form_class = EditSalesForm
    queryset = Sales.objects.all()
    success_url = reverse_lazy('transactions')

@method_decorator(login_required, name="dispatch")
class DeleteSalesView(DeleteView):
    template_name = 'sales/delete_sale.html'
    pk_url_kwarg = 'id'
    queryset = Sales.objects.all()
    success_url = reverse_lazy('transactions')

@method_decorator(login_required, name="dispatch")
class CheckoutView(ListView):
    queryset = Sales.objects.all().order_by('-id')
    context_object_name = 'sales'
    template_name = 'sales/checkout.html'
    success_url = reverse_lazy('transactions')
    item_list = []
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['total_sales'] = Sales.objects.filter(status='pending').filter(
            sold_by=self.request.user.id).aggregate(Sum('total_amount'))
        context['balance'] = float(0.0)
        self.item_list += Sales.objects.filter(status='pending').values()
        context['itmes'] = self.item_list
```

```
        return context

    def post(self, request, *args, **kwargs):

        if request.method == 'POST' and request.POST.get('total_amount') and
request.POST.get('amount_received'):
            Sales.objects.filter(status='pending').filter(
                sold_by=request.user.id).update(status='sold')
            for item in self.item_list:
                prod = Product.objects.get(name=item['name'])
                stock_balance = prod.stock_level - item['quantity']
                Product.objects.filter(name=item['name']).update(stock_level=stock_balance)

            return HttpResponseRedirect(self.success_url)

        return render(request, self.template_name)

class PrintReceiptView(PDFTemplateView):
    template_name = 'sales/sales_receipt.html'

    def get_context_data(self, **kwargs):
        dataset = Sales.objects.values(
                            'name','quantity','unit_price',
                            'total_amount',
                            'sold_by').filter(status='pending').filter(
            sold_by=self.request.user.id)
        context = super(PrintReceiptView, self).get_context_data(
            pagesize='A5',
            title='Sales Receipt',
            **kwargs
        )
        context['total_sales'] = Sales.objects.filter(status='pending').filter(
```

```
            sold_by=self.request.user.id).aggregate(Sum('total_amount'))
        for data in dataset:
            data['sold_by'] = User.objects.get(id=data['sold_by'])

        return generate_report(context, dataset, 'Sales Receipt')

class SalesPDFView(PDFTemplateView):
    template_name = 'sales/sales_report.html'

    def get_context_data(self, **kwargs):
        dataset = Sales.objects.values(
                            'name','quantity','unit_price',
                            'total_amount',
                            'sold_by',
                            'sold_at').order_by('id')
        context = super(SalesPDFView, self).get_context_data(
            pagesize='A4',
            title='Sales Report',
            **kwargs
        )
        for data in dataset:
            data['sold_by'] = User.objects.get(id=data['sold_by'])

        return generate_report(context, dataset, 'Sales List')
```

Purchases App

Forms.py

```
from django import forms
from .models import Purchase

class PurchaseCreationForm(forms.ModelForm):
```

```
    """
    A form for adding new purchase with all required field
    """
    class Meta:
        model = Purchase
        fields = ('name', 'description', 'quantity', 'cost_price', 'current_stock_level',
                'total_stock_level', 'supplier_tel', 'created_by')


class EditPurchaseForm(forms.ModelForm):
    """
    A form for editing existing purchase with all required field
    """
    class Meta:
        model = Purchase
        fields = ['name', 'description', 'quantity', 'cost_price','current_stock_level',
                'total_stock_level', 'supplier_tel',]
        exclude = ['created_by']
```

Models.py

```
from django.db import models
from accounts.models import User

class Purchase(models.Model):
    name = models.CharField(max_length=50,unique=True)
    description = models.TextField(max_length=100,unique=True)
    quantity = models.PositiveIntegerField()
    cost_price = models.FloatField(verbose_name='cost price')
    current_stock_level = models.PositiveIntegerField(verbose_name='current stock level')
    total_stock_level = models.PositiveIntegerField(verbose_name='total stock level')
    supplier_tel = models.CharField(max_length=13,unique=True)
    created_by = models.ForeignKey(User, on_delete=models.DO_NOTHING)
```

```
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        """Returns a string representation of this Purchase."""
        return self.name
```

Urls.py

```
from django.urls import path
from purchase import views as purchase_views

urlpatterns = [
    path('purchases/', purchase_views.PurchaseListView.as_view(), name='purchases'),
    path('purchase/', purchase_views.PurchaseCreationView.as_view(), name='purchase'),
    path('edit_purchase/<int:id>/', purchase_views.EditPurchaseView.as_view(),
name='edit_purchase'),
    path('delete_purchase/<int:id>/', purchase_views.DeletePurchaseView.as_view(),
name='delete_purchase'),
    path('purchase_report/', purchase_views.PurchasePDFView.as_view(),
name='purchase_report')
]
```

Views.py

```
from django.http import HttpResponseRedirect
from django.utils.decorators import method_decorator
from django.urls import reverse_lazy
from django.views.generic import (
    ListView, UpdateView, DetailView, DeleteView, CreateView )
from .forms import PurchaseCreationForm, EditPurchaseForm
from .models import Purchase
from decorators.decorators import group_required
from easy_pdf.views import PDFTemplateView
```

```
from helpers.generate_pdf import generate_report

decorators = [group_required(['Admin','Manager','General Manager'])]
@method_decorator(decorators, name="dispatch")
class PurchaseListView(ListView):
    queryset = Purchase.objects.all().order_by('id')
    paginate_by = 10
    context_object_name = 'purchase_list'
    template_name = 'purchase/purchase.html'

@method_decorator(decorators, name='dispatch')
class PurchaseCreationView(CreateView):
    form_class = PurchaseCreationForm
    template_name = 'purchase/add_purchase.html'
    success_message = 'Success: Purchase creation succeeded.'
    success_url = reverse_lazy('setting')

    def post(self, request, *args, **kwargs):
        data = {
            'name': request.POST.get('name', None),
            'description': request.POST.get('description', None),
            'quantity': request.POST.get('quantity', 0),
            'cost_price': request.POST.get('cost_price', 0),
            'current_stock_level': request.POST.get('current_stock_level', 0),
            'total_stock_level': int(request.POST.get('quantity', 0)) +
int(request.POST.get('current_stock_level', 0)),
            'supplier_tel': request.POST.get('supplier_tel', None),
            'created_by': request.user.id
        }
        if request.method == 'POST':
            form = self.form_class(data)
```

```
            if form.is_valid():
                form.save()

                return HttpResponseRedirect(self.success_url)

        return super().post(request, *args, **kwargs)


@method_decorator(decorators, name='dispatch')
class EditPurchaseView(UpdateView, DetailView):
    template_name = 'purchase/edit_purchase.html'
    pk_url_kwarg = 'id'
    form_class = EditPurchaseForm
    queryset = Purchase.objects.all()
    success_url = reverse_lazy('setting')


@method_decorator(decorators, name='dispatch')
class DeletePurchaseView(DeleteView):
    template_name = 'purchase/delete_purchase.html'
    pk_url_kwarg = 'id'
    queryset = Purchase.objects.all()
    success_url = reverse_lazy('setting')

class PurchasePDFView(PDFTemplateView):
    template_name = 'purchase/purchase_report.html'

    def get_context_data(self, **kwargs):
        dataset = Purchase.objects.values(
                            'name','description',
                            'quantity','cost_price',
                            'current_stock_level',
```

```
                    'total_stock_level',
                    'supplier_tel').order_by('id')
context = super(PurchasePDFView, self).get_context_data(
    pagesize='A4',
    title='Purchase Report',
    **kwargs
)

return generate_report(context, dataset, 'Purchases List')
```

Conclusion and Future Scope

The supermarket management system has been successfully implemented.However there are a lot of improvements that can be done .For example use of artificial intelligence to predict sales and make smart recommendations etc.

Bibliography / References

1. Django Documentation (https://docs.djangoproject.com/en/3.0/)
2. React Documentation (https://reactjs.org/)
3. Book: Learning React by Kirupa Chinnathambi (Pearson Education

   Release Date: December 27, 2016Imprint: Addison-Wesley Professional

   ISBN:9780134546537)