NIRMALA MEMORIAL FOUNDATION COLLEGE OF SCIENCE AND COMMERCE
KANDIVALI (EAST)

PROJECT REPORT ON

ONLINE SHOPPING WEBSITE FOR HEALTHCARE PRODUCTS

Submitted in Partial Fulfilment of the
Requirement for the Award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

Under the Esteemed Guidance Of

Professor.
Vandana Singh

OF

[USIT6P1 – PROJECT IMPLEMENTATION]

NAME: **SHARMA VIVEK OMPRAKASH**

CLASS/DIVISION: TYIT/A SEMESTER: VI

ROLL NO.: 56 UNIVERSITY SEAT NO: 18TIT110

Nirmala Memorial Foundation College of Commerce and Science

Affiliated to university of Mumbai

Accredited by NAAC with B++ CGPA:2.80

CERTIFICATE

This is to certify that **SHARMA VIVEK OMPRAKASH** Seat no / Div -Roll No.: **A-56** , student of T.Y.**B.Sc. Information Technology Semester VI** has completed the necessary Project, Titled **"Online Shopping Website For Healthcare Products"** in **USIT6P1 – PROJECT IMPLEMENTATION** during the academic year 2021-22.

_____ _____

Internal Guide Head of Department

_____

External Examiner

Seal &Date:_____

Abstract

I don’t think it’s too soon to say that the COVID-19 global pandemic will likely be one of the defining events of 2021, and that it will have implications that last well into the decade.The situation is rapidly changing. The amount of people deemed safe to gather in a single place has dwindled from thousands, to hundreds, to ten. Restaurants, bars, movie theaters, and gyms in many major cities are shutting down. Meanwhile many office workers are facing new challenges of working remotely full time.Essentially, people are coming to terms with the realities of our interconnected world and how difficult it is to temporarily separate those connections to others. To say that we are living in unprecedented times feels like an understatement. One of the responses we’ve seen to how people are approaching this period of isolation and uncertainty is in huge overnight changes to their shopping behaviors. From bulk-buying to online shopping, people are changing what they’re buying, when, and how. As more cities are going under lockdowns, nonessential businesses are being ordered to close, and customers are generally avoiding public places. Limiting shopping for all but necessary essentials is becoming a new normal. Brands are having to adapt and be flexible to meet changing needs.While survey data shows that women are more likely to be concerned about the effects of COVID-19, it also shows that men are more likely to have it impact their shopping behaviors. One-third of men, compared to 25% of women, reported the pandemic affecting how much they spend on products. Additionally, 36% of men, compared to 28% of women, reported it affecting how much they are spending on experiences (travel, restaurants, entertainment, etc.)

Acknowledgement

It is a great pleasure and a moment of immense satisfaction for me to express my profound gratitude to **my project guide Prof. VANDANA SINGH** , Department of Information Technology whose constant encouragement enabled me to work enthusiastically. Her perpetual motivation, patience and expertise in discussion during the progress of work have benefited me to an extent, which is beyond expression. Working under her guidance has been a fruitful and unforgettable experience. Despite her busy schedule, she was always available to give me advice, support and guidance during the entire period of my project. The completion of this project would not have been possible without her constant support and patient guidance. I am thankful to the Coordinator, Information Technology Department of Nirmala Memorial Foundation College of Commerce & Science, Prof. VAISHALI MISHRA for her encouragement, guidance and support for my project. I am thankful to the Principal of Nirmala memorial foundation college of commerce & science for her encouragement and providing an outstanding academic environment. I would like to express my gratitude to my family members for their constant support.

INDEX

CHAPTER 1

INTRODUCTION

Due to the worldwide COVID-19 coronavirus outbreak, the wearing of face masks in public is becoming more common. It infected over 5 million people in 188 countries in less than 6 months. The virus spreads by close contact, as well as in crowded and overcrowded environments. People used to wear masks to protect their health from air pollution before Covid-19. While others are self-conscious about their appearance, they conceal their feelings from the public by covering their ears. Scientists demonstrated that wearing face masks reduces COVID-19 transmission. The year 2020 has given humanity several mind-boggling sequences of events, the most life- changing is the COVID19 pandemic, which has startled the world since the year began.

In general during the coronavirus, ecommerce seems to be in a pretty good spot. After all, shoppers that can't go to a brick-and-mortar may turn instead to online shopping. But of course, it's not that simple.

Factor in economic uncertainty, which you can see reflected in stock market performance, and a shift away from many of the activities we typically enjoy (looking at you, Travel), and the impact on brick-and-mortar retail sales as well as online sales varies widely by vertical and even by business.

Ecommerce has proved itself essential in days of social distancing and shut-down storefronts. But some consumers have unfortunately found themselves with less or no work as a result of shelter-in-place orders and closures of nonessential businesses, and economists have estimated the U.S. unemployment rate may reach 32%. Others have tightened nonessential spending due to worldwide economic uncertainty.

It's difficult to predict the full impact of the coronavirus on online sales growth rates overall, but what is sure is that results won't be consistent across the board. It will depend on niche, changing shopper behavior, and how much longer communities are asked to socially distance — among other things.

The paper is organized as follows. In Section 2, we will go through the survey of technologies. In Section 3, the requirements and analysis of our proposed solution are discussed in detail. In Section 4, the system design is evaluated, and results discussed. Section 5 discusses the implementation and testing. Then Section 6 depicts the results and finally with Section 7, the report is concluded.

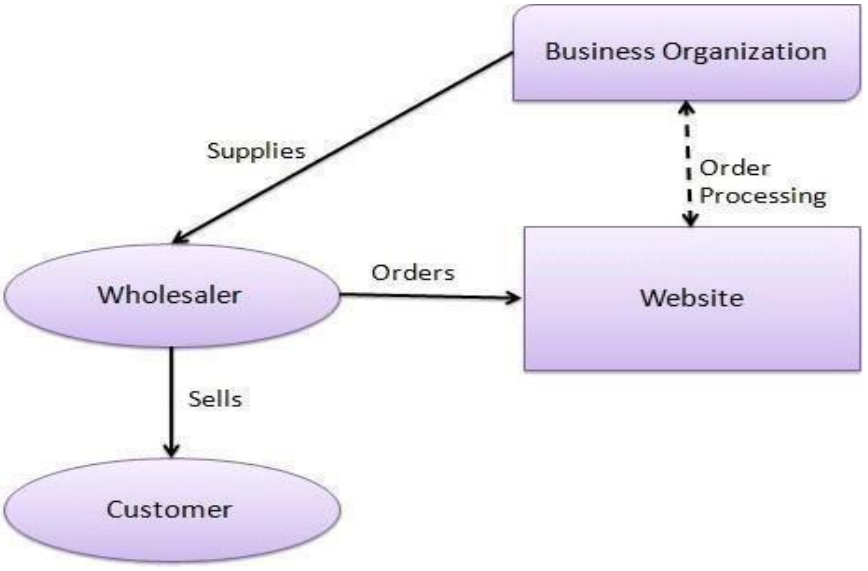
- **Background**

Business transactions that involve the exchange of money are covered by the term e-commerce. E-business includes all aspects of running a business that sells goods and services, including marketing, earning and retaining customers, procurement, developing business partners and customer education. In order to be successful, e- commerce and e-businesses must have quality storefronts that are simple to navigate and peruse, with accurate and thorough catalogue information. E- business became an extension of e-commerce to encompass all aspects of businesses that function online. E-business involves e-commerce, but e- Commerce does not cover all aspects of e-business.

- **BUSINESS MODELS** E-Commerce or Electronics Commerce business models can generally be categorized in the following categories:-

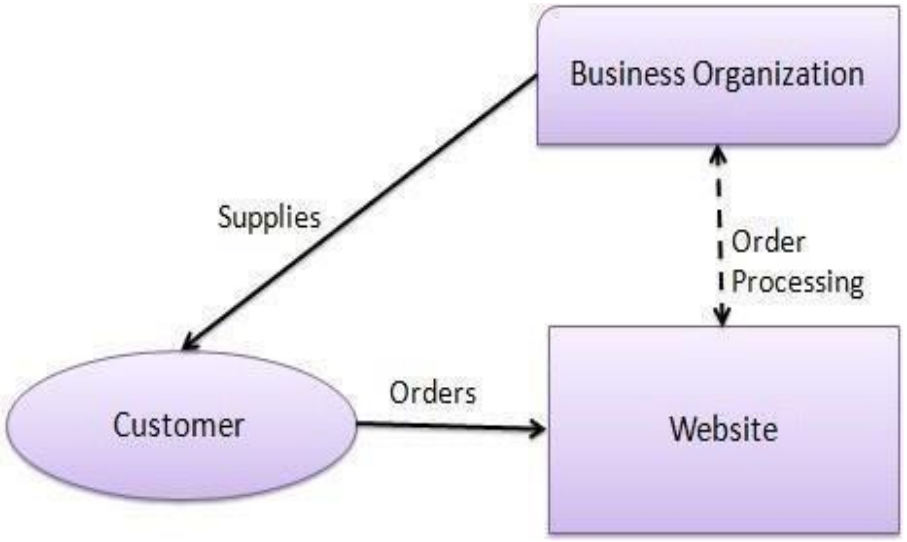
- **Business - to - Business (B2B)**

Website following B2B business model sells its product to an intermediate buyer who then sells the product to the final customer. As an example, a wholesaler places an order from a company's website and after receiving the consignment, sells the end product to final customer who comes to buy the product at wholesaler's retail outlet.



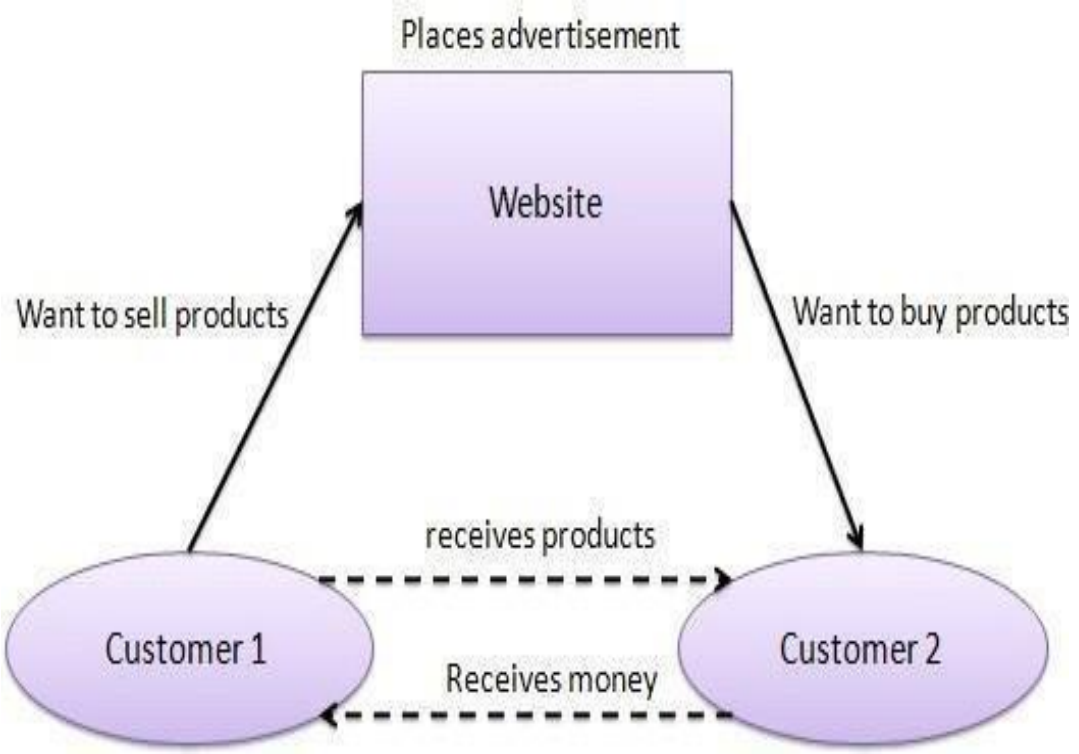
- **Business - to - Consumer (B2C)**

Website following B2C business model sells its product directly to a customer. A customer can view products shown on the website of business organization. The customer can choose a product and order the same. Website will send a notification to the business organization via email and organization will dispatch the product/goods to the customer.



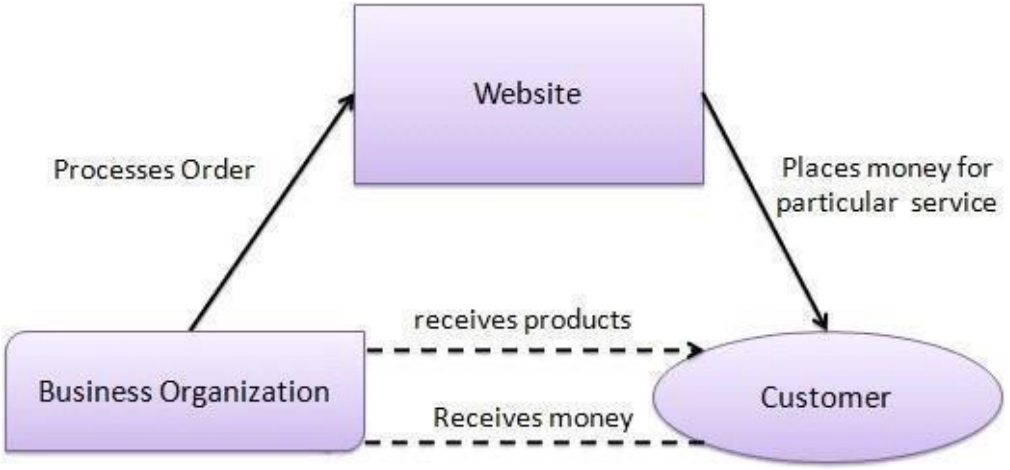
- **Consumer - to - Consumer (C2C)**

Website following C2C business model helps consumer to sell their assets like residential property, cars, motorcycles etc. or rent a room by publishing their information on the website. Website may or may not charge the consumer for its services. Another consumer may opt to buy the product of the first customer by viewing the post/advertisement on the website.



- **Consumer - to - Business (C2B)**

In this model, a consumer approaches website showing multiple business organizations for a particular service. Consumer places an estimate of amount he/she wants to spend for a particular service. For example, comparison of interest rates of personal loan/ car loan provided by various banks via website. Business organization who fulfills the consumer's requirement within specified budget approaches the customer and provides its services.

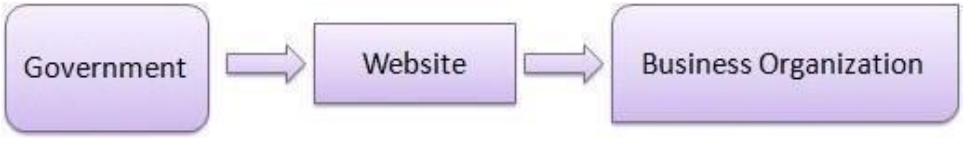


- **Business - to - Government (B2G)**

B2G model is a variant of B2B model. Such websites are used by government to trade and exchange information with various business organizations. Such websites are accredited by the government and provide a medium to businesses to submit application forms to the government.

- **Government - to - Business (G2B)**

Government uses B2G model website to approach business organizations. Such websites support auctions, tenders and application submission functionalities.



- **Government - to - Citizen (G2C)**

Government uses G2C model website to approach citizen in general. Such websites support auctions of vehicles, machinery or any other material. Such website also provides services like registration for birth, marriage or death certificates. Main objectives of G2C website are to reduce average time for fulfilling people requests for various government services.

- **Objectives**

In this project I have used Business - to - Consumer (B2C) model for online shopping for the Gents, Ladies and Kids wear where only Admins Can upload and take Order and track customer History.

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose

- The project's goal is based on previous research. It has been observed that shopping online will help prevent the spread of respiratory viruses. For instance, the efficiency of buying online and stop mass gathering prevents SARS transmission is 91% and 68%, respectively.
- Online Shopping interrupts airborne viruses and particles, preventing these organisms from entering another person's respiratory system.
- The system aims Get As many as customer and prevent loss of business from it.
- The system is simple to integrate into every current organizational system.
- It limits access to those who are not wearing masks while also alerting authorities.
- You can customize the orders as per your needs.

1.3.2 Scope

- It will help in collecting perfect data of citizens who are following the buying online and can get the perfect idea of the market.
- It is integrated with the loss and profit models and other business related data analysis.
- A real time monitoring dashboard which contains the database that helps the user to see who buys online and their track history of visit. Users can also create reports to import and integrate with third-party applications.

1.3.3 Applicability

Online Shopping always helps customer to get access to many items and can select as per their wishes.But the cons are they can not try at site buy but my application gives them 7 days trial period so that customer can decide wheater to buy or not.

The following are a few use cases where online shopping can be applicable: -

- **Homes**

Online shopping mainly uses in homes because people look at the product when they are free from all works and mainly during night.I have chosed only gents, ladies and kids clothing because our main customer will be ladies and kids wear. women usually visit online shopping.

- **Offices**

As Office going person usually dont have time for the shopping and during the panademic and the clothing is basic needs for human they can buy it online.
During the party office persons also dont have time to buy offline so they prefer online to buy clothes and save their valuable time.

CHAPTER 2

SURVEY OF TECHNOLOGIES

The summary of technologies that are used in the development of project is given below: -

- **PYTHON**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for Web Development (Server-Side), Software Development, Mathematics, System Scripting.

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

Python is widely considered one of the easiest programming languages for a beginner to learn, but it is also difficult to master. Anyone can learn Python if they work hard enough at it, but becoming a Python Developer will require a lot of practice and patience.

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It has fewer syntactic exceptions and special cases than C or Pascal.

- **DJANGO**

Django is a Python based free and open-source web framework that follows the model-template-views architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established in the US as a non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Some well-known sites that uses Django includes Instagram, Mozilla, Disqus, Bitbucket, Nextdoor and Clubhouse.

Django was created in the fall of 2003, when the web programmers at the Lawrence Journal-World newspaper, Adrian Holovaty and Simon Willison, began using Python to build applications. Jacob Kaplan-Moss was hired early in Django's development shortly before Simon Willison's internship ended. It was released publicly under a BSD license in July 2005. The framework was named after guitarist Django Reinhardt. Adrian Holovaty is a Romani jazz guitar player and a big fan of Django Reinhardt. In June 2008, it was announced that a newly formed Django Software Foundation (DSF) would maintain Django in the future.

Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses "views", the core Django framework can be seen as an MVC architecture. It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("Model"), a system for processing HTTP requests with a web templating system ("View"), and a regular-expression-based URL dispatcher ("Controller").

Also included in the core framework are:

- A lightweight and standalone web server for development and testing
- A form serialization and validation system that can translate between HTML forms and values suitable for storage in the database
- A template system that utilizes the concept of inheritance borrowed from object-oriented programming
- A caching framework that can use any of several cache methods
- support for middleware classes that can intervene at various stages of request processing and carry out custom functions
- An internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals
- An internationalization system, including translations of Django's own components into a variety of languages
- A serialization system that can produce and read XML and/or JSON representations of Django model instances
- A system for extending the capabilities of the template engine
- An interface to Python's built-in unit test framework

The main Django distribution also bundles a number of applications in its "contrib" package, include:

- An extensible authentication system
- The dynamic administrative interface
- Tools for generating RSS and Atom syndication feeds
- A "Sites" framework that allows one Django installation to run multiple websites, each with their own content and applications
- Tools for generating Google Sitemaps
- Built-in mitigation for cross-site request forgery, cross-site scripting, SQL injection, password cracking and other typical web attacks, most of them turned on by default[19][20]
- A framework for creating GIS applications

Django can be run in conjunction with Apache, Nginx using WSGI, Gunicorn, or Cherokee using flup (a Python module). Django also includes the ability to launch a FastCGI server, enabling use behind any web server which supports FastCGI, such as Lighttpd or Hiawatha. It is also possible to use other WSGI-compliant web servers. Django officially supports five database backends: PostgreSQL, MySQL, MariaDB, SQLite, and Oracle. Microsoft SQL Server can be used with django-mssql while similarly external backends exist for IBM Db2, SQL Anywhere and Firebird. There is a fork named django-nonrel, which supports NoSQL databases, such as MongoDB and Google App Engine's Datastore.

- **VISUAL STUDIO CODE**

  Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

  In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 70% of 82,000 respondents reporting that they use it.

  Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A preview build was released shortly thereafter.

  On November 18, 2015, the source of Visual Studio Code was released under the MIT License, and made available on GitHub. Extension support was also announced. On April 14, 2016, Visual Studio Code graduated from the public preview stage and was released to the Web. Microsoft has released most of Visual Studio Code's source code on GitHub under the permissive MIT License, while the releases by Microsoft are proprietary freeware.

- **SQLite 3**

  SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

  SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen().

  SQLite is a compact library. With all features enabled, the library size can be less than 750KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be faster than direct filesystem I/O.

  The SQLite code base is supported by an international team of developers who work on SQLite full-time. The developers continue to expand the capabilities of SQLite and enhance its reliability and performance while maintaining backwards compatibility with the published interface spec, SQL syntax, and database file format. The source code is absolutely free to anybody who wants it, but professional support is also available.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

1. Quality issues

The most common problems faced by shoppers who shop online, is the quality of the product. When you shop online, you do not have any guarantee on product quality. You can't just rely on product reviews as they are not reliable. Many e-commerce websites just function as an aggregator of sellers.

So, there are high chances of fraudulent sellers registering themselves on websites, selling fake products in the names of branded ones. As a huge volume of products is sold each day, it is not possible for e-commerce companies to check each and every product sold. Many a time sellers refuse to replace the faulty product and refund the money. Websites which sell fake products instead of genuine products do not accept cash on delivery and insist on payment upfront. Not receiving clothes and footwear of proper sizes, is another problem faced by consumers while shopping online.

2. Failures while making a digital transaction

Other than cash on delivery, e-commerce websites also allow you to pay through various modes like credit cards, debit cards, net banking and mobile wallets. Another important problem faced by online shoppers is the failure of digital payments.
Sometimes, due to a bad internet connection or technical glitch, the amount gets debited from the customer's bank account, but it does not get credited to the sellers account. In such a case, customers can get back this money by contacting the customer care of the company. It will take around 7-10 days for the amount to get refunded.

3. Unclear Website Policies

Before purchasing any product online, it is very important to go through the website policies. Some websites do not have a return or refund policy. In such a case, even if you receive a fake product, you will not be allowed to return it. Most of the websites which have unclear return policies end up selling low-quality products. You must also go through the policies vis-a-vis product warranty. In case you find any website having unclear policies, avoid making a purchase from them.

4. Delivery and logistics issues

Another important problem faced while shopping online is the delivery date. Most of the e-commerce websites have an option where you can track your order, but this doesn’t always hold true. Sometimes, delivery executives deliver the product when you are not at home. In such a case, product delivery will be rescheduled. If this happens say more than three times, your order will be returned to the point of origin. In some cases, the product will be lost in the transit.

Another major issue is logistics. As a large number of India's population lives in rural and tier III cities, they are not able to shop online, because most of the e-commerce companies do not deliver products to these places.

5. Additional charges

Another problem faced while shopping online, is additional charges incurred while making the payment. The amount you pay for the product will not be same as the one you see at the website. Price of the product will be displayed on the website without tax, shipping, and handling charges. When you make the payments, all these charges will be added and the price goes up.

6. Safety issues

As many people shop online these days, cyber criminals come up with innovative techniques to hack e-commerce websites and steal money from online shoppers. E-commerce sites will have important customer data like names, phone numbers, address, and bank details. If these details fall in the hand of fraudsters they can be misused.

3.2 Requirements Specification

There are some hardware and software requirements need to be fulfilled in accordance with the implementation of the Online ecommerce shopping. These requirements are

:

Software Requirement:-

Operating System	Windows XP, 7,10
RAM	Min 8GB
Processor	Intel Core i5 and above
Processor Speed	2 GHz or more
OS Capability	64 bits

Hardware Requirement: -

Monitor	LED, LCD, OLED
Hard Disk Memory	Min 256GB

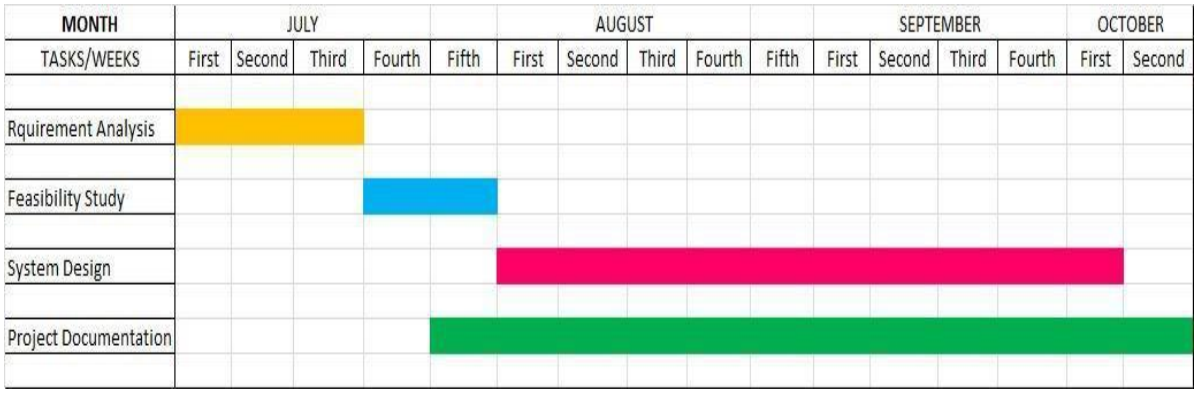
We also need to integrate programming languages and some datasets for training the model. The below list comprises of all the requirements: -

Programming Languages and Libraries: -

Language	PYTHON
Libraries	DJANGO
IDE (Integrated Development Environment)	VISUAL STUDIO CODE
Database management system	SQLite 3

3.3 Planning and Scheduling

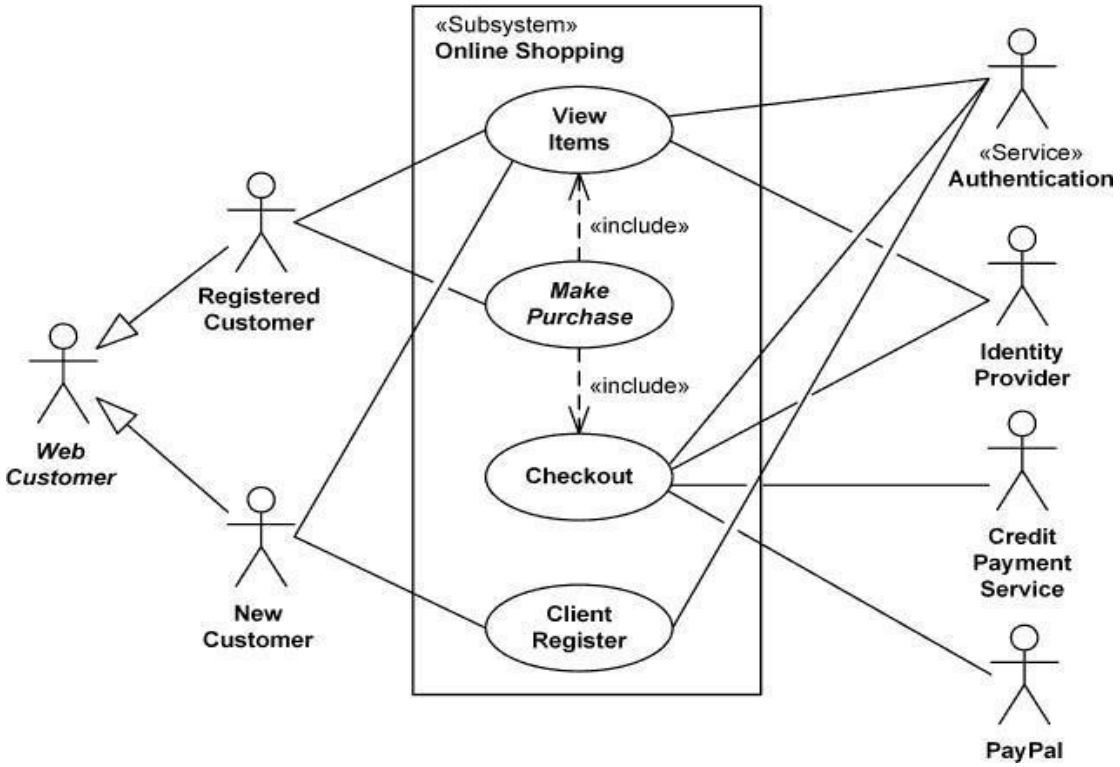
Gantt Chart:-



CHAPTER 4

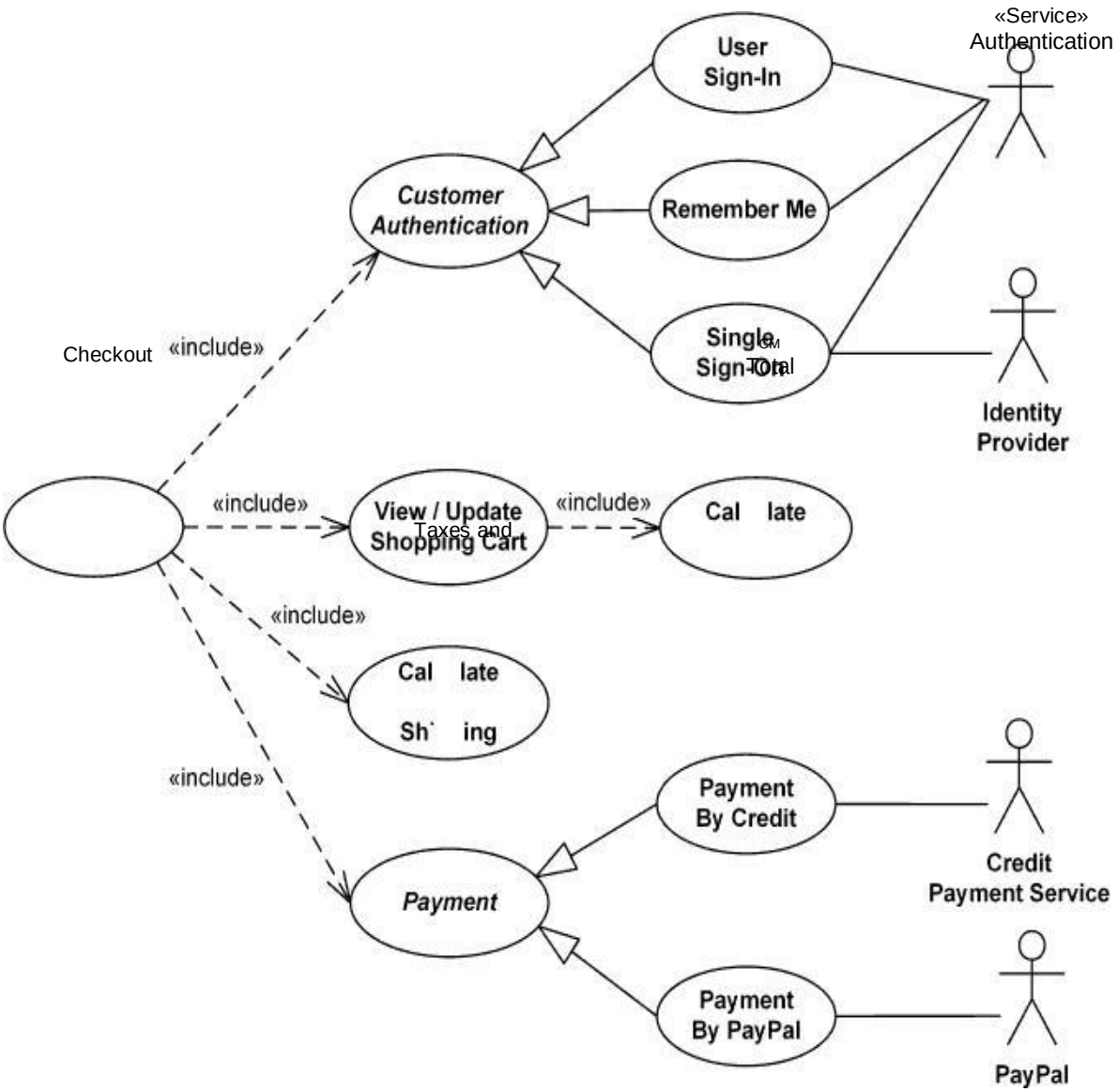
SYSTEM DESIGN

4.1 Basic Modules



4.2 Data Design

4.3 Procedural Design



CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Coding

Main:-

```
from django.shortcuts import render, redirect
from .forms import UserRegisterForm, UpdateUserDetailForm, UserUpdateForm,
UserAddressForm, UserAddressForm1
from django.http import HttpResponse, JsonResponse
from django.contrib import messages
from django.contrib.auth.models import User
from .models import UserDetail, Slider, Contact, Cart
from django.contrib.auth.decorators import login_required
from saler.models import Product, ProductSize, dow, category, Orders, trend,
ProductReview
from django.contrib.auth import update_session_auth_hash
from django.contrib.auth.forms import PasswordChangeForm
from django.views.decorators.csrf import csrf_exempt
from .PayTm import Checksum
from django.utils import timezone


def whywellbee(request):
    return render(request,'main/whywellbee.html')
def aboutus(request):
    return render(request,'main/aboutus.html')
def faq(request):
    return render(request,'main/faq.html')
def tnc(request):
    return render(request,'main/tnc.html')
def accrediation(request):
    return render(request,'main/accrediation.html')

def index(request):
    if request.user.is_superuser:
        return redirect('admin2')
    elif request.user.is_staff:
        return redirect("saler_home")
    else:
        pass

    prod = Product.objects.all()
    allProds = []
    catprods = Product.objects.values('category', 'product_id')
    cats = {item['category'] for item in catprods}
```

```
    for cat in cats:
        prod = []
        for p in [i for i in Product.objects.filter(category=cat)]:
            prod.append([p,[item for item in
ProductSize.objects.filter(product=p)]])
        n = len(prod)
        nSlides = 5
        allProds.append([prod[::-1], range(1, nSlides), nSlides])
    params = {
        'sliders':Slider.objects.all(),
        'allProds':allProds,
        'category':category.objects.all(),
        #'prod_men' : [i for i in prod if i.buyer_gender == 'Male'],
        #'prod_women' : [i for i in prod if i.buyer_gender == 'Female'],
        #'prod_other' : [i for i in prod if i.buyer_gender == 'All'],
        'dow' : dow.objects.all()[0:30],
        'trend': trend.objects.order_by('-number')[0:30],
        'cart_element_no' : len([p for p in Cart.objects.all() if p.user ==
request.user]),
    }
    return render(request, 'main/index.html', params)

def register(request):
    if request.user.is_authenticated:
        return redirect('home')
    else:
        if request.method == 'POST':
            form = UserRegisterForm(request.POST)
            if form.is_valid():
                form.save();
                username = form.cleaned_data.get('username')
                usr = User.objects.filter(username=username).first()
                if username.isdigit():
                    UserDetail(user=usr,mobile=username).save()
                else:
                    usr.email = username
                    usr.save()
                    UserDetail(user=usr).save()
                messages.success(request, f'Account is Created for
{username}')
                return redirect('login')
        else:
            form = UserRegisterForm()
    return render(request, 'main/signup.html', {'form':form, 'title':'Sign
Up','category':category.objects.all()})

@login_required
def account_settings(request):
```

```
    if request.method == 'POST':
        #User Details Update
        s_form = UpdateUserDetailForm(request.POST, request.FILES,
instance=request.user.userdetail)
        u_form = UserUpdateForm(request.POST, instance=request.user)
        if s_form.is_valid() and u_form.is_valid():
            s_form.save()
            u_form.save()
            messages.success(request, f'Your Account has been Updated!')
            return redirect("account_settings")

        #Change Password
        pass_change_form = PasswordChangeForm(request.user, request.POST)
        if pass_change_form.is_valid():
            user = pass_change_form.save()
            update_session_auth_hash(request, user)  # Important!
            messages.success(request, f'Your password was successfully
updated!')
            return redirect('account_settings')
        else:
            messages.error(request, 'Please correct the error below.')

    else:
        s_form = UpdateUserDetailForm(instance=request.user.userdetail)
        u_form = UserUpdateForm(instance=request.user)
        pass_change_form = PasswordChangeForm(request.user)
    detl = {
        'u_form':u_form,
        's_form':s_form,
        'pass_change_form':pass_change_form,
        'title':'User Account Settings',
        'cart_element_no' : len([p for p in Cart.objects.all() if p.user ==
request.user]),
        'category':category.objects.all(),
        }
    return render(request, 'main/account_settings.html', detl)

def productView(request, prod_id):
    if request.method == 'POST' and request.user.is_authenticated:
        prod = Product.objects.filter(product_id = prod_id).first()
        review = request.POST.get('review')
        ProductReview(user=request.user,product=prod,review=review).save()
        return redirect(f"/product/{prod_id}")

    prod = Product.objects.filter(product_id = prod_id).first()
    params = {
        'product':prod,
        'product_review': ProductReview.objects.filter(product = prod),
```

```
        'sizes':[item for item in
ProductSize.objects.filter(product=Product.objects.filter(product_id =
prod_id)[0])],
        'cart_element_no' : len([p for p in Cart.objects.all() if p.user ==
request.user]),
        'category':category.objects.all(),
    }
    return render(request, 'main/single.html', params)


def view_all(request, catg):
    if catg=='dow':
        params = {
            'product':[i for i in dow.objects.all()][::-1],
            'catg':'Deal of the Week',
            'cart_element_no' : len([p for p in Cart.objects.all() if p.user
== request.user]),
            'category':category.objects.all(),
            }
        return render(request, 'main/view_dow.html', params)
    elif catg=='trend':
        prod = []
        for p in trend.objects.order_by('number'):
            prod.append([p.product,[item for item in
ProductSize.objects.filter(product=p.product)]])
        params = {
            'product':prod,
            'catg':'Treanding',
            'cart_element_no' : len([p for p in Cart.objects.all() if p.user
== request.user]),
            'category':category.objects.all(),
            }
        return render(request, 'main/view_all.html', params)
    else:
        prod = []
        for p in [i for i in Product.objects.all() if str(i.category) ==
catg]:
            prod.append([p,[item for item in
ProductSize.objects.filter(product=p)]])
        params = {
            'product':prod,
            'catg':catg,
            'cart_element_no' : len([p for p in Cart.objects.all() if p.user
== request.user]),
            'category':category.objects.all(),
            }
    return render(request, 'main/view_all.html', params)
```

```
def search(request):
    query = request.GET.get('query', '')
    prods = []
    for prod in [i for i in Product.objects.all() ]:
        if query.lower() in prod.product_name.lower() or query.lower() in
prod.desc.lower() or query.lower() in prod.subcategory.lower():
            prods.append([prod,[item for item in
ProductSize.objects.filter(product=prod)]])
    params = {
        'product':prods,
        'cart_element_no' : len([p for p in Cart.objects.all() if p.user ==
request.user]),
        'category':category.objects.all(),
        }
    return render(request, 'main/view_all.html', params)

cart_item_local = []

def dummy_cart(request):
    if request.method == 'GET':
        prod_list = request.GET['prod_list']
        prod_list = prod_list.split(',')
        if request.user.is_authenticated:
            cart_prods = [p for p in Cart.objects.all() if p.user ==
request.user]
            card_prods_id =[i.product_id for i in cart_prods]
            if len(prod_list) >= 1 and prod_list != ['']:
                for item in prod_list:
                    pppp = item.split('|')
                    if pppp[0] in card_prods_id:
                        cart_prods[card_prods_id.index(pppp[0])].number =
int(pppp[1])
                        cart_prods[card_prods_id.index(pppp[0])].save()
                    else:

                        Cart(user = request.user, product_id = pppp[0], number
= int(pppp[1])).save()
        else:
            global cart_item_local
            cart_item_local = prod_list
    return HttpResponse("data sebd from py")

@login_required
def cart(request):
    if request.user.is_authenticated:
        allProds = []
        subtotal = 0.0
        delev = 0.0
```

```
        tax = 0.0
        cart_prods = [p for p in Cart.objects.all() if p.user == request.user]
        for p in cart_prods:
            tempTotal = p.number *
Product.objects.filter(product_id=p.product_id)[0].price
            subtotal += tempTotal
            tax +=
tempTotal*int(Product.objects.filter(product_id=p.product_id).first().gst)/100

        for cprod in cart_prods:
            prod = Product.objects.filter(product_id=cprod.product_id)[0]
            allProds.append([cprod, prod])
        params = {
                    'allProds':allProds,
                    'cart_element_no' : len([p for p in Cart.objects.all() if
p.user == request.user]),
                    'total':subtotal+tax+delev,
                    'subtotal':subtotal,
                    'tax':tax,
                    'delev':delev,
                    'category':category.objects.all(),
                }
        return render(request,'main/cart.html', params)

@login_required
def add_to_cart(request):
    cart_prods = [p for p in Cart.objects.all() if p.user == request.user]
    card_prods_id =[i.product_id for i in cart_prods]
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        prod_id = prod_id.split(',')
        for item in cart_prods:
            if prod_id[0] == item.product_id and prod_id[1] ==
item.product_size:
                item.number += 1
                item.save()
                return HttpResponse(len(cart_prods))
        Cart(user = request.user, product_id =
int(prod_id[0]),product_size=prod_id[1], number = 1).save()
        return HttpResponse(len(cart_prods)+1)
    else:
        return HttpResponse("")

@login_required
def plus_element_cart(request):
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        c = Cart.objects.get(id=prod_id)
```

```
        c.number+=1
        c.save()
        subtotal = 0.0
        delev = 0.0
        tax = 0.0
        cart_prods2 = [p for p in Cart.objects.all() if p.user ==
request.user]
        for p in cart_prods2:
            tempTotal = p.number *
Product.objects.filter(product_id=p.product_id)[0].price
            subtotal += tempTotal
            tax +=
tempTotal*int(Product.objects.filter(product_id=p.product_id).first().gst)/100

        datas = {
            'num':Cart.objects.get(id=prod_id).number,
            'tax':tax,
            'subtotal':subtotal,
            'delev' : delev,
            'total':subtotal+tax+delev,
            }
        return JsonResponse(datas)
    else:
        return HttpResponse("")

@login_required
def minus_element_cart(request):
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        c = Cart.objects.get(id=prod_id)
        c.number-=1
        c.save()
        subtotal = 0.0
        delev = 0.0
        tax = 0.0
        cart_prods2 = [p for p in Cart.objects.all() if p.user ==
request.user]
        for p in cart_prods2:
            tempTotal = p.number *
Product.objects.filter(product_id=p.product_id)[0].price
            subtotal += tempTotal
            tax +=
tempTotal*int(Product.objects.filter(product_id=p.product_id).first().gst)/100

        datas = {
            'num':Cart.objects.get(id=prod_id).number,
            'tax':tax,
            'subtotal':subtotal,
```

```
            'delev' : delev,
            'total':subtotal+tax+delev,
            }
        return JsonResponse(datas)
    else:
        return HttpResponse("")


@login_required
def delete_from_cart(request):
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        c = Cart.objects.get(id=prod_id)
        c.delete()
        subtotal = 0.0
        delev = 0.0
        tax = 0.0
        cart_prods2 = [p for p in Cart.objects.all() if p.user ==
request.user]
        for p in cart_prods2:
            tempTotal = p.number *
Product.objects.filter(product_id=p.product_id)[0].price
            subtotal += tempTotal
            tax +=
tempTotal*int(Product.objects.filter(product_id=p.product_id).first().gst)/100

        datas = {
            'num':len(cart_prods2),
            'tax':tax,
            'subtotal':subtotal,
            'delev' : delev,
            'total':subtotal+tax+delev,
            }
        return JsonResponse(datas)
    else:
        return HttpResponse("")

MERCHANT_KEY = 'zs_7mXxCH08pkhVR'
@login_required
def order_now(request):
    allProds =[]
    if request.method == 'GET':
        new_prod = request.GET.get('prod_id')
        prod_size = request.GET.get('prod_size')
        allProds = [[1,Product.objects.filter(product_id=int(new_prod))[0]]]
        #print(allProds)
    if request.method == 'POST':
        new_prod = request.GET.get('prod_id')
        prod_size = request.GET.get('prod_size')
```

```
        print(new_prod)
        print(prod_size)
        address_form = UserAddressForm(request.POST,
instance=request.user.userdetail)
        u_form2 = UserAddressForm1(request.POST, instance=request.user)
        if address_form.is_valid() and u_form2.is_valid():
            address_form.save()
            u_form2.save()
            pay_mode = request.POST.get('pay_mode')
            trends = [i.product.product_id for i in trend.objects.all()]
#            print(order)
            if pay_mode == 'on':
                a=str(timezone.now())
                a=a[:19]
                a=a.replace(" ", "")
                a=a.replace("-", "")
                a=a.replace(":","")
                if Orders.objects.all().last():
                    order_id
=str(request.user)+a+'C'+str((Orders.objects.all().last().pk)+1)
                else:
                    order_id = str(request.user)+a+'C'
                product1 = new_prod+'|'+str(1)+','
                delev = 0.0
                subtotal =
Product.objects.filter(product_id=int(new_prod)).first().price
                tax =
subtotal*int(Product.objects.filter(product_id=int(new_prod)).first().gst)/100
                totalprice1=int(subtotal+tax+delev)
                Orders(order_id=order_id,user=request.user,saler=Product.objec
ts.filter(product_id=int(new_prod)).first().shop,products=product1,size=prod_s
ize,itemprice=subtotal,gstprice=tax,totalprice=totalprice1).save()
                if int(new_prod) in trends:
                    t = trend.objects.filter(product =
Product.objects.filter(product_id=int(new_prod)).first())[0]
                    t.number += 1
                    t.save()
                else:
                    trend(product =
Product.objects.filter(product_id=int(new_prod)).first(), number=1).save()
                return redirect('/myorders')
            else:
                o_id = ''
                a=str(timezone.now())
                a=a[:19]
                a=a.replace(" ", "")
                a=a.replace("-", "")
                a=a.replace(":","")
```

```
                if Orders.objects.all().last():
                    order_id =
str(request.user)+a+'O'+str((Orders.objects.all().last().pk)+1)
                else:
                    order_id = str(request.user)+a+'O001'
                o_id = order_id
                product1 = new_prod+'|'+str(1)+','
                delev = 0.0
                subtotal =
Product.objects.filter(product_id=int(new_prod)).first().price
                tax =
subtotal*int(Product.objects.filter(product_id=int(new_prod)).first().gst)/100
                totalprice1=int(subtotal+tax+delev)
                Orders(order_id=order_id,user=request.user,saler=Product.objec
ts.filter(product_id=int(new_prod)).first().shop,products=product1,size=prod_s
ize,itemprice=subtotal,gstprice=tax,totalprice=totalprice1,order_type='Online
Payment').save()
                if int(new_prod) in trends:
                    t = trend.objects.filter(product =
Product.objects.filter(product_id=int(new_prod)).first())[0]
                    t.number += 1
                    t.save()
                else:
                    trend(product =
Product.objects.filter(product_id=int(new_prod)).first(), number=1).save()


                param_dict = {

                        'MID': 'sxSexJ66537719707415',
                        'ORDER_ID': str(o_id),
                        'TXN_AMOUNT': str(subtotal+tax+delev),
                        'CUST_ID': request.user.username,
                        'INDUSTRY_TYPE_ID': 'Retail',
                        'WEBSITE': 'DEFAULT',
                        'CHANNEL_ID': 'WEB',
                        'CALLBACK_URL':'http://127.0.0.1:8000/handlerequest2/'
,
                }
                param_dict['CHECKSUMHASH'] =
Checksum.generate_checksum(param_dict, MERCHANT_KEY)
                return render(request, 'main/paytm.html', {'param_dict':
param_dict})

    else:
        address_form = UserAddressForm(instance=request.user.userdetail)
        u_form2 = UserAddressForm1(instance=request.user)
```

```
    delev = 0.0
    subtotal = Product.objects.filter(product_id=int(new_prod)).first().price
    tax =
subtotal*int(Product.objects.filter(product_id=int(new_prod)).first().gst)/100
    totl = round(subtotal+tax+delev, 2)
    params = {
            'allProds':allProds,
            'cart_element_no' : len([p for p in Cart.objects.all() if p.user
== request.user]),
            'address_form': address_form,
            'u_form':u_form2,
            'total':totl,
            'category':category.objects.all(),
        }
    return render(request, 'main/checkout2.html', params)

@login_required
def checkout(request):
    temp = 0
    allProds = []
    cart_prods = [p for p in Cart.objects.all() if p.user == request.user]
    for cprod in cart_prods:
        prod = Product.objects.filter(product_id=cprod.product_id)[0]
        allProds.append([cprod, prod])
    if request.method == 'POST':
        address_form = UserAddressForm(request.POST,
instance=request.user.userdetail)
        u_form2 = UserAddressForm1(request.POST, instance=request.user)
        if address_form.is_valid() and u_form2.is_valid():
            address_form.save()
            u_form2.save()
            pay_mode = request.POST.get('pay_mode')
            trends = [i.product.product_id for i in trend.objects.all()]
#           print(order)
            if pay_mode == 'on':
                subtotal = 0.0
                delev = 0.0
                tax = 0.0
                totalamount = 0.0
                for p in cart_prods:
                    tempTotal = p.number *
Product.objects.filter(product_id=p.product_id)[0].price
                    subtotal += tempTotal
                    tax +=
tempTotal*int(Product.objects.filter(product_id=p.product_id).first().gst)/100
                totalamount=int(subtotal+tax+delev)
                for item in cart_prods:
                    a=str(timezone.now())
```

```
                    a=a[:19]
                    a=a.replace(" ", "")
                    a=a.replace("-", "")
                    a=a.replace(":","")
                    if Orders.objects.all().last():
                        order_id =
str(request.user)+a+'C'+str((Orders.objects.all().last().pk)+1)
                    else:
                        order_id = str(request.user)+a+'C001'
                    product1 = item.product_id+'|'+str(item.number)+','
                    totalitemprice = item.number *
Product.objects.filter(product_id=item.product_id)[0].price
                    totalitemtax=
totalitemprice*int(Product.objects.filter(product_id=p.product_id).first().gst
)/100
                    Orders(order_id=order_id,user=request.user,saler=Product.o
bjects.filter(product_id=int(item.product_id)).first().shop,products=product1,
size=item.product_size,itemprice=totalitemprice,gstprice=totalitemtax,totalpri
ce=totalamount).save()
                    item.delete()
                    if int(item.product_id) in trends:
                        t = trend.objects.filter(product =
Product.objects.filter(product_id=int(item.product_id)).first())[0]
                        t.number += 1
                        t.save()
                    else:
                        trend(product =
Product.objects.filter(product_id=int(item.product_id)).first(),
number=1).save()
                return redirect('/myorders')
            else:
                temp = 1
    else:
        address_form = UserAddressForm(instance=request.user.userdetail)
        u_form2 = UserAddressForm1(instance=request.user)
    subtotal = 0.0
    delev = 0.0
    tax = 0.0
    for p in cart_prods:
        tempTotal = p.number *
Product.objects.filter(product_id=p.product_id)[0].price
        subtotal += tempTotal
        tax +=
tempTotal*int(Product.objects.filter(product_id=p.product_id).first().gst)/100

    if temp == 1:
        o_id = ''
```

```
        for item in cart_prods:
            a=str(timezone.now())
            a=a[:19]
            a=a.replace(" ", "")
            a=a.replace("-", "")
            a=a.replace(":","")
            order_id =
str(request.user)+a+'O'+str((Orders.objects.all().last().pk)+1)
            o_id = order_id
            product1 = item.product_id+'|'+str(item.number)+','
            Orders(order_id=order_id,user=request.user,saler=Product.objects.f
ilter(product_id=int(item.product_id)).first().shop,products=product1,
size=item.product_size)

            if int(item.product_id) in trends:
                t = trend.objects.filter(product =
Product.objects.filter(product_id=int(item.product_id)).first())[0]
                t.number += 1
                t.save()
            else:
                trend(product =
Product.objects.filter(product_id=int(item.product_id)).first(), number=1)
        param_dict = {

                'MID': 'sxSexJ66537719707415',
                'ORDER_ID': str(o_id),
                'TXN_AMOUNT': str(subtotal+tax+delev),
                'CUST_ID': request.user.username,
                'INDUSTRY_TYPE_ID': 'Retail',
                'WEBSITE': 'WEBSTAGING',
                'CHANNEL_ID': 'WEB',
                'CALLBACK_URL':'http://127.0.0.1:8000/handlerequest/',

        }
        param_dict['CHECKSUMHASH'] = Checksum.generate_checksum(param_dict,
MERCHANT_KEY)
        print(param_dict)
        return render(request, 'main/paytm.html', {'param_dict': param_dict})

    params = {
            'allProds':allProds,
            'cart_element_no' : len(cart_prods),
            'address_form': address_form,
            'u_form':u_form2,
            'total':subtotal+tax+delev,
            'category':category.objects.all(),
        }
    return render(request, 'main/checkout.html', params)
```

```
@csrf_exempt
def handlerequest(request):


    # paytm will send your post request here
    form = request.POST
    response_dict = {}
    print(form)
    for i in form.keys():
        response_dict[i] = form[i]
        if i == 'CHECKSUMHASH':
            checksum = form[i]

    verify = Checksum.verify_checksum(response_dict, MERCHANT_KEY, checksum)
    customer=response_dict['ORDERID']
    customer=customer[:10]
#   cart_prods = [p for p in Cart.objects.all() if p.user == customer ]
    cart_prods=[]
    user1 = User.objects.get(username=customer)
#   print(user1)
    for p in Cart.objects.all():

        if(str(p.user) == str(customer)):
            cart_prods.append(p)

    trends = [i.product.product_id for i in trend.objects.all()]
#   print(customer)
#   print(Cart.objects.all())
    if verify:
        if response_dict['RESPCODE'] == '01':
            for item in cart_prods:
                print(item)
                a=str(timezone.now())
                a=a[:19]
                a=a.replace(" ", "")
                a=a.replace("-", "")
                a=a.replace(":","")
                order_id =
str(user1)+a+'O'+str((Orders.objects.all().last().pk)+1)
                o_id = order_id
                product1 = item.product_id+'|'+str(item.number)+','
                totalitemprice = item.number *
Product.objects.filter(product_id=item.product_id)[0].price
                totalitemtax=
totalitemprice*int(Product.objects.filter(product_id=p.product_id).first().gst
)/100
```

```
                Orders(order_id=order_id,user=user1,saler=Product.objects.filt
er(product_id=int(item.product_id)).first().shop,products=product1,
size=item.product_size,itemprice=totalitemprice,gstprice=totalitemtax,totalpri
ce=response_dict['TXNAMOUNT'],order_type='Online
Payment',payment_status='Success',payment_failreason=response_dict['RESPMSG'])
.save()

                item.delete()
                if int(item.product_id) in trends:
                    t = trend.objects.filter(product =
Product.objects.filter(product_id=int(item.product_id)).first())[0]
                    t.number += 1
                    t.save()
                else:
                    trend(product =
Product.objects.filter(product_id=int(item.product_id)).first(),
number=1).save()
            print('order successful')
        else:

            print('order was not successful because' +
response_dict['RESPMSG'])
    return render(request, 'main/paymentstatus.html', {'response':
response_dict})

@csrf_exempt
def handlerequest2(request):


    # paytm will send your post request here
    form = request.POST
    response_dict = {}
    print(form)
    for i in form.keys():
        response_dict[i] = form[i]
        if i == 'CHECKSUMHASH':
            checksum = form[i]

    verify = Checksum.verify_checksum(response_dict, MERCHANT_KEY, checksum)
    customer=response_dict['ORDERID']
    customer=customer[:10]
#   cart_prods = [p for p in Cart.objects.all() if p.user == customer ]
    cart_prods=[]
    user1 = User.objects.get(username=customer)
#   print(user1)
```

```
#    print(customer)
#    print(Cart.objects.all())
    if verify:
        if response_dict['RESPCODE'] == '01':
            order_object =
Orders.objects.filter(order_id=response_dict['ORDERID'])
            order_object.update(payment_status='Success',payment_failreason=re
sponse_dict['RESPMSG'])
            #Orders(order_id=order_id,user=user1,saler=Product.objects.filter(
product_id=int(item.product_id)).first().shop,products=product1,
size=item.product_size).save()
            print('order successful')
        else:
            order_object =
Orders.objects.filter(order_id=response_dict['ORDERID'])
            order_object.update(payment_status='Failure',payment_failreason=re
sponse_dict['RESPMSG'])
            print('order was not successful because' +
response_dict['RESPMSG'])
    return render(request, 'main/paymentstatus.html', {'response':
response_dict})


def MyOrders(request):
    if request.method == 'POST':
        order_id = request.POST.get('order_id')
        o = Orders.objects.filter(order_id=order_id)[0]
        o.status = 'Cancel'
        o.save()
    params = {
        'orders': [i for i in Orders.objects.all() if i.user == request.user
and i.status != 'Delivered' and i.status != 'Cancel'],
        'delivered': [i for i in Orders.objects.all() if i.user ==
request.user and i.status == 'Delivered'],
        'cancel': [i for i in Orders.objects.all() if i.user == request.user
and i.status == 'Cancel'],

    }
    return render(request,'main/myorders.html', params)

def MenuFilter(request, querys):
    print(querys.split(',')[0])
    prod = []
    for p in [i for i in Product.objects.all() if str(i.category).lower() ==
querys.split(',')[0].lower() and
str(i.subcategory).lower()==querys.split(',')[1].lower()]:
        prod.append([p,[item for item in
ProductSize.objects.filter(product=p)]])
    params = {
```

```
        'product':prod,
        'catg':querys,
        'cart_element_no' : len([p for p in Cart.objects.all() if p.user ==
request.user]),
        'category':category.objects.all(),
        }
    return render(request, 'main/view_all.html', params)


def search1(request):
    query = 'Medical Equipments'
    prods = []
    for prod in [i for i in Product.objects.all() ]:
        if query.lower() in prod.product_name.lower() or query.lower() in
prod.desc.lower() or query.lower() in prod.subcategory.lower():
            prods.append([prod,[item for item in
ProductSize.objects.filter(product=prod)]])
    params = {
        'product':prods,
        'cart_element_no' : len([p for p in Cart.objects.all() if p.user ==
request.user]),
        'category':category.objects.all(),
        }
    return render(request, 'main/view_all.html', params)

def contact(request):
    if request.method == 'POST':
        cont_name = request.POST.get('Name', default='')
        cont_email = request.POST.get('Email', default='')
        cont_subject = request.POST.get('Subject', default='')
        cont_mess = request.POST.get('Message', default='')
        con = Contact(name = cont_name, email = cont_email, subject =
cont_subject, message = cont_mess)
        con.save()
        messages.success(request, 'Your message has been sent. Thank you!')

    return render(request, 'main/contact.html',
{'category':category.objects.all(),'cart_element_no' : len([p for p in
Cart.objects.all() if p.user == request.user]),})
```

SALER:-

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, JsonResponse
from .models import SalerDetail, Product, ProductSize, SellerSlider, MyCart,
WholeSaleProduct, category, Orders,
WholeSaleProductOrders,DoctorRegistration,Petctscandetail,Ambulance
from django.contrib import messages
from django.contrib.auth.models import User
from .forms import SalerRegisterForm, SalerAddressForm, UpdateSalerDetailForm,
UpdateSalerAccountDetailForm,DoctorRegisterForm,PetctscanForm,AmbulanceBooking
Form
from main.forms import UserUpdateForm
from django.contrib.auth.decorators import login_required
from math import ceil
from django.contrib.auth import update_session_auth_hash
from django.contrib.auth.forms import PasswordChangeForm

#This is view of Index Page of Seller in which we Display Whole Sale Products
@login_required
def index(request):
    if request.user.is_superuser or request.user.is_staff:
        allProds = []
        catprods = WholeSaleProduct.objects.values('category', 'product_id')
        cats = {item['category'] for item in catprods}
        for cat in cats:
            prod = WholeSaleProduct.objects.filter(category=cat)
            n = len(prod)
            nSlides = n // 6 + ceil((n / 6) - (n // 6))
            allProds.append([prod, range(1, nSlides), nSlides])
        params = {
                'allProds':allProds,
                'prod':WholeSaleProduct.objects.all()[::-1][:4],
                'sliders':SellerSlider.objects.all(),
                'trending':WholeSaleProduct.objects.all(),
                'cart_element_no' : len([p for p in MyCart.objects.all() if
p.user == request.user]),

                }
        return render(request, 'saler/index.html', params)
    else:
        return redirect("/")

# This is View of Dashboard in which we display all orders of the seller and
ther status
@login_required
def dashboard(request):
    if request.user.is_superuser or request.user.is_staff:
```

```
        if request.method == 'GET':
            odrr = request.GET.get('odrr')
            st = request.GET.get('st')
            if st == 'Cancel':
                o = Orders.objects.filter(order_id=odrr).first()
                o.status = 'Cancel'
                o.save()
            if st == 'Accepted':
                o = Orders.objects.filter(order_id=odrr).first()
                o.status = 'Accepted'
                o.save()
            if st == 'Packed':
                o = Orders.objects.filter(order_id=odrr).first()
                o.status = 'Packed'
                o.save()
            if st == 'Delivered':
                o = Orders.objects.filter(order_id=odrr).first()
                o.status = 'On The Way'
                o.save()
        ordr = [i for i in Orders.objects.filter(saler=request.user) if
i.status != 'Cancel' and i.status != 'On The Way' and i.status !=
'Delivered'][::-1]
        params = {
                'orders':ordr,
                'dorders': [i for i in
Orders.objects.filter(saler=request.user) if i.status != 'Cancel' and i.status
== 'On The Way' or i.status == 'Delivered'][::-1],
                'cart_element_no' : len([p for p in MyCart.objects.all() if
p.user == request.user]),
                }
        return render(request, 'saler/newdashboard.html', params)
    else:
        return redirect("/")

# This is add to cart view of Seller means Whole Sale Products
@login_required
def add_to_cart(request):
    cart_prods = [p for p in MyCart.objects.all() if p.user == request.user]
    card_prods_id =[i.product_id for i in cart_prods]
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        if prod_id in card_prods_id:
            cart_prods[card_prods_id.index(prod_id)].number +=
WholeSaleProduct.objects.filter(product_id=prod_id)[0].min_Quantity
            cart_prods[card_prods_id.index(prod_id)].save()
            return HttpResponse(len(cart_prods))

        else:
```

```
            MyCart(user = request.user, product_id = prod_id, number =
WholeSaleProduct.objects.filter(product_id=prod_id)[0].min_Quantity).save()
            return HttpResponse(len(cart_prods)+1)
    else:
        return HttpResponse("")

# This is view for Increasing Quantity of any Product in Cart
@login_required
def plus_element_cart(request):
    cart_prods = [p for p in MyCart.objects.all() if p.user == request.user]
    card_prods_id =[i.product_id for i in cart_prods]
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        if prod_id in card_prods_id:
            cart_prods[card_prods_id.index(prod_id)].number +=
WholeSaleProduct.objects.filter(product_id=prod_id)[0].min_Quantity
            cart_prods[card_prods_id.index(prod_id)].save()
            subtotal = 0.0
            delev = 0.0
            cart_prods2 = [p for p in MyCart.objects.all() if p.user ==
request.user]
            for p in cart_prods2:
                subtotal += p.number *
WholeSaleProduct.objects.filter(product_id=p.product_id)[0].price
            tax = subtotal*5/100
            datas = {
                'num':MyCart.objects.filter(user=request.user,product_id=prod_
id)[0].number,
                'tax':tax,
                'subtotal':subtotal,
                'delev' : delev,
                'total':subtotal+tax+delev,
                }
            return JsonResponse(datas)
    else:
        return HttpResponse("")

# This is view for Decreasing Quantity of any Product in Cart
@login_required
def minus_element_cart(request):
    cart_prods = [p for p in MyCart.objects.all() if p.user == request.user]
    card_prods_id =[i.product_id for i in cart_prods]
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        if prod_id in card_prods_id:
            cart_prods[card_prods_id.index(prod_id)].number -=
WholeSaleProduct.objects.filter(product_id=prod_id)[0].min_Quantity
            cart_prods[card_prods_id.index(prod_id)].save()
```

```
            subtotal = 0.0
            delev = 0.0
            cart_prods2 = [p for p in MyCart.objects.all() if p.user ==
request.user]
            for p in cart_prods2:
                subtotal += p.number *
WholeSaleProduct.objects.filter(product_id=p.product_id)[0].price
            tax = subtotal*5/100
            datas = {
                'num':MyCart.objects.filter(user=request.user,product_id=prod_
id)[0].number,
                'tax':tax,
                'subtotal':subtotal,
                'delev' : delev,
                'total':subtotal+tax+delev,
                }
            return JsonResponse(datas)
    else:
        return HttpResponse("")

# This is view for Deleting a Product from Cart
@login_required
def delete_from_cart(request):
    cart_prods = [p for p in MyCart.objects.all() if p.user == request.user]
    card_prods_id =[i.product_id for i in cart_prods]
    if request.method == 'GET':
        prod_id = request.GET['prod_id']
        if prod_id in card_prods_id:
            cart_prods[card_prods_id.index(prod_id)].delete()
            subtotal = 0.0
            delev = 0.0
            cart_prods2 = [p for p in MyCart.objects.all() if p.user ==
request.user]
            for p in cart_prods2:
                subtotal += p.number *
WholeSaleProduct.objects.filter(product_id=p.product_id)[0].price
            tax = subtotal*5/100
            datas = {
                'len':len(cart_prods2),
                'tax':tax,
                'subtotal':subtotal,
                'delev' : delev,
                'total':subtotal+tax+delev,
                }
            return JsonResponse(datas)
    else:
        return HttpResponse("")
```

```
# This view for Display a Single Product
@login_required
def productView(request, prod_id):
    if request.user.is_superuser or request.user.is_staff:
        params = {
            'product':WholeSaleProduct.objects.filter(product_id =
prod_id)[0],
            'cart_element_no' : len([p for p in MyCart.objects.all() if p.user
== request.user]),
        }
        return render(request, 'saler/productview.html', params)
    else:
        return redirect("/")

# This is View of Display all Products of a perticular category
@login_required
def view_all(request, catg):
    if request.user.is_superuser or request.user.is_staff:
        params = {
            'product':[i for i in WholeSaleProduct.objects.all() if
str(i.category) == catg],
            'catg':catg,
            'cart_element_no' : len([p for p in MyCart.objects.all() if p.user
== request.user]),
            }
        return render(request, 'saler/view_all.html', params)
    else:
        return redirect("/")

# seller cart view
@login_required
def mycart(request):
    if request.user.is_superuser or request.user.is_staff:
        allProds = []
        subtotal = 0.0
        delev = 0.0
        cart_prods = [p for p in MyCart.objects.all() if p.user ==
request.user]
        for p in cart_prods:
            subtotal += p.number *
WholeSaleProduct.objects.filter(product_id=p.product_id)[0].price
        tax = subtotal*5/100

        for cprod in cart_prods:
            prod =
WholeSaleProduct.objects.filter(product_id=cprod.product_id)[0]
            allProds.append([cprod, prod])
        params = {
```

```
                    'allProds':allProds,
                    'cart_element_no' : len([p for p in MyCart.objects.all()
if p.user == request.user]),
                    'total':subtotal+tax+delev,
                    'subtotal':subtotal,
                    'tax':tax,
                    'delev':delev,
                }
        return render(request,'saler/cart.html', params)
    else:
        return redirect("/")

# seller checkout view means for whole sale products
def checkout(request):
    if request.user.is_superuser or request.user.is_staff:
        allProds = []
        cart_prods = [p for p in MyCart.objects.all() if p.user ==
request.user]
        for cprod in cart_prods:
            prod =
WholeSaleProduct.objects.filter(product_id=cprod.product_id)[0]
            allProds.append([cprod, prod])
        if request.method == 'POST':
            address_form = SalerAddressForm(request.POST,
instance=request.user.salerdetail)
            if address_form.is_valid():
                address_form.save()
                for item in cart_prods:
                    if WholeSaleProductOrders.objects.all().last():
                        order_id =
'WSPOrder'+str((WholeSaleProductOrders.objects.all().last().pk)+1)
                    else:
                        order_id = 'WSPOrder001'
                    product1 = item.product_id+'|'+str(item.number)+','
                    WholeSaleProductOrders(order_id=order_id,user=request.user
,products=product1).save()
                    item.delete()
                return redirect('seller_orders')
        else:
            address_form = SalerAddressForm(instance=request.user.salerdetail)

        subtotal = 0.0
        delev = 0.0
        for p in cart_prods:
            subtotal += p.number *
WholeSaleProduct.objects.filter(product_id=p.product_id)[0].price
        tax = subtotal*5/100
        params = {
```

```
                'allProds':allProds,
                'cart_element_no' : len(cart_prods),
                'address_form': address_form,
                'total':subtotal+tax+delev,
            }
        return render(request, 'saler/checkout.html', params)
    else:
        return redirect("/")

# Orders of Seller (Whole sale product)
def MyOrders(request):
    if request.method == 'POST':
        order_id = request.POST.get('order_id')
        o = WholeSaleProductOrders.objects.filter(order_id=order_id)[0]
        o.status = 'Cancel'
        o.save()
    params = {
        'orders': [i for i in WholeSaleProductOrders.objects.all() if i.user
== request.user and i.status != 'Delivered' and i.status != 'Cancel'],
        'delivered': [i for i in WholeSaleProductOrders.objects.all() if
i.user == request.user and i.status == 'Delivered'],
        'cancel': [i for i in WholeSaleProductOrders.objects.all() if i.user
== request.user and i.status == 'Cancel'],

    }
    return render(request,'saler/myorders.html', params)

# for adding a new product by seller
@login_required
def add_product(request):
    if request.user.is_superuser or request.user.is_staff:
        if request.method == 'POST':
            prod_name = request.POST.get('prod_name')
            desc = request.POST.get('desc')
            cat = request.POST.get('category')
            subcategory = request.POST.get('subcategory')
            price = request.POST.get('price')
            price_not = request.POST.get('price_not')
            gst_l = request.POST.get('gst')
            if Product.objects.all():
                prod_id2 = 'pr'+hex(Product.objects.all().last().product_id+1)
            else:
                prod_id2 = 'pr'+hex(0)
            image1 = request.FILES.get("image1")
            image2 = request.FILES.get("image2")
            image3 = request.FILES.get("image3")
            image4 = request.FILES.get("image4")
            image5 = request.FILES.get("image5")
```

```
            size_no = int(request.POST.get('size_no'))
            i=1
            sizes=[]
            while i <= size_no:
                if request.POST.get(f'size{i}'):
                    sizes.append([request.POST.get(f'size{i}'),int(request.POS
T.get(f'quantity{i}'))])
                i+=1
            Product(product_id2=prod_id2,shop=request.user,product_name=prod_n
ame,category=category.objects.get(id=int(cat)),subcategory=subcategory,price=p
rice,price_not=price_not,desc=desc,gst=gst_l,image1=image1).save()
            p = Product.objects.filter(product_id2=prod_id2)[0]
            if image2:
                p.image2=image2
            if image3:
                p.image3=image3
            if image4:
                p.image4=image4
            if image5:
                p.image5=image5
            p.save()
            for siz in sizes:
                ProductSize(product=p,size=siz[0],quantity=siz[1]).save()
            messages.success(request, f'Product Added !')
        prod = [p for p in Product.objects.all() if p.shop == request.user]

        if request.method == 'GET':
            pro_id = request.GET.get('pro_id', 0)
            del_prod = [p.product_id for p in Product.objects.all() if p.shop
== request.user]
            if int(pro_id) in del_prod:
                Product.objects.filter(product_id=int(pro_id))[0].delete()
                messages.success(request, f'The Product of id {pro_id} is
deleted !')
                return redirect('/seller/add_product/')

        subcat=[]
        for cat in category.objects.all():
            x = cat.sub_Categories.split(',')
            x.insert(0, cat)
            subcat.append(x)
        params = {
            'prod':prod[::-1][0:20],
            'cart_element_no' : len([p for p in MyCart.objects.all() if p.user
== request.user]),
            'subcat':subcat,
            }
        return render(request, 'saler/add_product.html',params)
```

```
    else:
        return redirect("/")

# it will display all products of a seller
@login_required
def view_products(request):
    if request.user.is_superuser or request.user.is_staff:
        prod = [p for p in Product.objects.all() if p.shop == request.user]

        if request.method == 'GET':
            pro_id = request.GET.get('pro_id')
            if pro_id:
                del_prod = [p.product_id for p in Product.objects.all() if
p.shop == request.user]
                if int(pro_id) in del_prod:
                    Product.objects.filter(product_id=int(pro_id))[0].delete()
                    messages.success(request, f'The Product of id {pro_id} is
deleted !')
                    return redirect('/seller/view_products/')

        params = {
            'prod': prod[::-1],
            'cart_element_no' : len([p for p in MyCart.objects.all() if p.user
== request.user]),
            }
        return render(request, 'saler/view_products.html',params)
    else:
        return redirect("/")

# Signup for Seller
def seller_signup(request):
    if request.user.is_authenticated:
        return redirect('home')
    else:
        if request.method == 'POST':
            form = SalerRegisterForm(request.POST)
            if form.is_valid():
                form.save();
                username = form.cleaned_data.get('username')
                shop = form.cleaned_data.get('shop')
                gst = form.cleaned_data.get('gst')
                usr = User.objects.filter(username=username).first()
                usr.is_staff=True
                usr.save()
                if username.isdigit():
                    SalerDetail(user=usr,mobile=username,gst_Number=gst,shop_N
ame=shop).save()
                else:
```

```
                    usr.email = username
                    usr.save()
                    SalerDetail(user=usr,gst_Number=gst,shop_Name=shop).save()
                messages.success(request, f'Account is Created for
{username}')
                return redirect('login')
        else:
            form = SalerRegisterForm()
    return render(request, 'saler/seller_signup.html', {'form':form,
'title':'Become a Seller'})

# Doctor Registration Form
def doctor_signup(request):
    if request.user.is_authenticated:
        return redirect('home')
    else:
        if request.method == 'POST':
            form = DoctorRegisterForm(request.POST,request.FILES)
            if form.is_valid():

                #form.save();
                username = form.cleaned_data.get('username')
                first_name = form.cleaned_data.get('first_name')
                last_name = form.cleaned_data.get('last_name')
                password1= form.cleaned_data.get('password1')
                #usr = User.objects.filter(username=username).first()
                is_staff=False
                #usr.save()
                clinic_Address = form.cleaned_data.get('clinic_Address')
                locality = form.cleaned_data.get('locality')
                pincode = form.cleaned_data.get('pincode')
                city = form.cleaned_data.get('city')
                state= form.cleaned_data.get('state')
                letterhead = request.FILES.get("letter_head")
                User(username=username,first_name=first_name,last_name=last_na
me,is_staff=is_staff).save()
                usr = User.objects.filter(username=username).first()
                usr.set_password(password1)
                usr.save()
                DoctorRegistration(user=usr,mobile=username,clinic_Address=cli
nic_Address,locality=locality,pincode=pincode,city=city,state=state,letter_hea
d=letterhead).save()
                messages.success(request, f'Account is Created for
{username}')
                return redirect('login')
        else:
            form = DoctorRegisterForm()
```

```
    return render(request, 'saler/doctor_signup.html', {'form':form,
'title':'Register As Doctor'})



#petctscan

def petctscan_booking(request):


    if request.method == 'POST':
        form = PetctscanForm(request.POST)
        if form.is_valid():

            #form.save();
            username = form.cleaned_data.get('username')
            residential_address = form.cleaned_data.get('residential_address')
            pincode = form.cleaned_data.get('pincode')
            city = form.cleaned_data.get('city')
            state= form.cleaned_data.get('state')
            patient_name=form.cleaned_data.get('patient_name')
            relative_name=form.cleaned_data.get('relative_name')
            relative_number=form.cleaned_data.get('relative_number')
            relation=form.cleaned_data.get('relation')
            services=form.cleaned_data.get('services')
            appointment_slot=form.cleaned_data.get('appointment_slot')
            appointment_date=form.cleaned_data.get('appointment_date')
            Petctscandetail(mobile=username,residential_address=residential_ad
dress,pincode=pincode,city=city,state=state,patient_name=patient_name,relative
_name=relative_name,relative_number=relative_number,relation=relation,services
=services,appointment_slot=appointment_slot,appointment_date=appointment_date)
.save()
            #messages.success(request, f'Data Submitted')
            return redirect('/')
    else:
        form = PetctscanForm()
    return render(request, 'saler/petctscan_booking.html', {'form':form,
'title':'Book Appointment '})

#ambulance
def ambulance_booking(request):


    if request.method == 'POST':
        form = AmbulanceBookingForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            residential_address = form.cleaned_data.get('residential_address')
            pincode = form.cleaned_data.get('pincode')
```

```
            city = form.cleaned_data.get('city')
            state= form.cleaned_data.get('state')
            patient_name=form.cleaned_data.get('patient_name')

            Petctscandetail(mobile=username,residential_address=residential_ad
dress,pincode=pincode,city=city,state=state).save()
            #messages.success(request, f'Data Submitted')
            return redirect('/')
    else:
        form = AmbulanceBookingForm()
    return render(request, 'saler/ambulance_booking.html', {'form':form,
'title':'Book Appointment '})

# Seller Account Settings
@login_required
def account_settings(request):
    if request.user.is_superuser or request.user.is_staff:
        if request.method == 'POST':
            #User Details Update
            s_form = UpdateSalerDetailForm(request.POST, request.FILES,
instance=request.user.salerdetail)
            u_form = UserUpdateForm(request.POST, instance=request.user)
            if s_form.is_valid() and u_form.is_valid():
                s_form.save()
                u_form.save()
                messages.success(request, f'Your Account has been Updated!')
                return redirect("saler_account_settings")

            #Change Password
            pass_change_form = PasswordChangeForm(request.user, request.POST)
            if pass_change_form.is_valid():
                user = pass_change_form.save()
                update_session_auth_hash(request, user)  # Important!
                messages.success(request, 'Your password was successfully
updated!')
                return redirect('saler_account_settings')
            else:
                messages.error(request, 'Please correct the error below.')

            #Account Settings
            acc_form = UpdateSalerAccountDetailForm(request.POST,
request.FILES, instance=request.user.salerdetail)
            if acc_form.is_valid():
                acc_form.save()
                messages.success(request, f'Account Settings has been
Updated!')
                return redirect("saler_account_settings")
```

```
        else:
            s_form = UpdateSalerDetailForm(instance=request.user.salerdetail)
            u_form = UserUpdateForm(instance=request.user)
            acc_form =
UpdateSalerAccountDetailForm(instance=request.user.salerdetail)
            pass_change_form = PasswordChangeForm(request.user)
        detl = {
            'u_form':u_form,
            's_form':s_form,
            'pass_change_form':pass_change_form,
            'acc_form':acc_form,
            'cart_element_no' : len([p for p in MyCart.objects.all() if p.user
== request.user]),
            'title':'User Account Settings',
            }
        return render(request, 'saler/account_settings.html', detl)
    else:
        return redirect("/")

# This is a part of admin view in which all ordered products will display with
address
def admin2(request):
    if request.user.is_superuser:
        ordr = [i for i in Orders.objects.all() if i.status != 'Cancel' and
i.status != 'On The Way' and i.status != 'Delivered'][::-1]
        params = {
                'orders':ordr,
                'dorders': [i for i in
Orders.objects.filter(saler=request.user) if i.status != 'Cancel' and i.status
== 'On The Way' or i.status == 'Delivered'],
                'cart_element_no' : len([p for p in MyCart.objects.all() if
p.user == request.user]),
                }
        return render(request, 'saler/admin2.html', params)
    else:
        return redirect("/")
```

SETTINGS:-

```
"""
Django settings for wrappers project.

Generated by 'django-admin startproject' using Django 3.0.2.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '$a^w$eh7g*&@dt%hfv7qu*a87u(b3$(5ng(f@bj@%9a*2_z+te'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ["*"]


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'crispy_forms',
    'main',
    'saler',
    'coupon',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
```

```
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'wrappers.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'wrappers.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_L10N = True

USE_TZ = False


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'

MEDIA_ROOT = os.path.join(BASE_DIR,'media')
MEDIA_URL = '/media/'

CRISPY_TEMPLATE_PACK = 'bootstrap4'

LOGIN_URL = 'login'

LOGIN_REDIRECT_URL = 'home'


EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'mstpatel@gmail.com'
EMAIL_HOST_PASSWORD = 'Ganpatipapa'
EMAIL_USE_TLS = True
```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 User Documentation

- User have to register to buy any of the products on my project so that I can get the Email and phone number to communicate with the customer.
- User can get quick access as they have register for the first time and their payment system will be saved for the future transaction
- I have used plugins for data analytics so that customer dont have to search for their collection
- You can get a lot of range or variety of products
- No need to visit any mall, just click and add stuff to your shopping cart any time.
- You can save time and money as you can compare the prices of different websites which in the case of markets, is not possible.
- when we buy an article online we can obtain detailed reviews about that article and as well as about the site which helps us by saving us from any kind of fraud.
- Different modes of payment are available. Some sites even provide Cash on delivery, which is very helpful for new users.

- **Some negative impacts are::**
  - Many times the product received are not of the same quality as promised to the customer. Most of the sites look appealing by the pictures of their stuff but sometimes they fool people by showing the HD pictures, however, in reality, the fabric is of very cheap quality.
  - Some sites have no Guarantees / Warranties/ Return Policies.
  - Issues like late delivery, unable to contact the delivery person, additional shipping charge on each item.
  - Improper customer care service.
  - In case of long distances, products get damaged on the way and then the customers fail to avail its benefits.

CHAPTER 7

CONCLUSIONS

7.1 Significance of the System

In general, today's businesses must always strive to create the next best thing that consumers will want because consumers continue to desire their products, services etc. to continuously be better, faster, and cheaper. In this world of new technology, businesses need to accommodate to the new types of consumer needs and trends because it will prove to be vital to their business' success and survival. E-commerce is continuously progressing and is becoming more and more important to businesses as technology continues to advance and is something that should be taken advantage of and implemented.

From the inception of the Internet and e-commerce, the possibilities have become endless for both businesses and consumers. Creating more opportunities for profit and advancements for businesses, while creating more options for consumers. However, just like anything else, e-commerce has its disadvantages including consumer uncertainties, but nothing that can not be resolved or avoided by good decision-making and business practices.

There are several factors and variables that need to be considered and decided upon when starting an e-commerce business. Some of these include: types of e-commerce, marketing strategies, and countless more. If the correct methods and practices are followed, a business will prosper in an e-commerce setting with much success and profitability.

7.2 Limitations of the System

- Although I have put my best efforts to make the software flexible, easy to operate but limitations cannot be ruled out even by me. Though the software presents a broad range to its user some intricate options could not be covered into it; partly because of the logistic and partly due to lack of sophistication.

- Though e-commerce offers many advantages to customers, business, society and nation, there are still some areas of concern that need to be addressed. The following are some of the limitations or disadvantages of e-commerce.: -

- **Security**
    - The biggest drawback of e-commerce is the issue of security. People fear to provide personal and financial information, even though several improvements have been made in relation to data encryption.
    - Certain websites do not have capabilities to conduct authentic transactions.
    - Fear of providing credit card information and risk of identity limit the growth of e-commerce.
- **Tax issue**
    - Sales tax is another bigger issue when the buyer and seller are situated in different locations.
    - Computation of sales tax poses problems when the buyer and seller are in different states.
    - Another factor is that physical stores will lose business if web purchases are free from tax.
- **Product suitability**
    - People have to rely on electronic images to purchase products. Sometimes, when the products are delivered, the product may not match with electronic images.
    - Finally, it may not suit the needs of the buyers. The lack of 'touch and feel' prevent people from online shopping.
- **Huge technological cost**
    - It is difficult to merge electronic business with traditional business. Technological infrastructure may be expensive and huge cost has to be incurred to keep pace with ever changing technology.
    - It is necessary to allocate more funds for technological advancement to remain competitive in the electronic world.

7.3 Future Scope of the Project

E-commerce has bloomed over the years and is one of the fastest-growing domains in the online world. Though it took some time for this to be accepted by the end-users, today we are at a point where the majority of the people love to shop online. There were numerous concerns revolving around online shopping at its launch, but over years people tend to have started trusting E-commerce for all their shopping needs.

In India, people prefer shopping online these days rather than having to visit the physical store. The payment features that are smart and secure as well as the cash on delivery (COD), which makes the payment, even more, safer with hassle-free shipping, easy returns and reach out.

Let us check out the development or growth of this e-commerce sector in India. We have specifically collected substantial data from across the web after analysis and inferences of information acquired from authentic sources. In the year 2013, around 8 million people have been shopping online. And, the most interesting factor is that they have done shopping from some of the major online shopping sites. And, the number 8 million had risen to around 100 million by the year 2016. The new shoppers (customer base) accounting to around 50% came from the tier one and tier two towns of India. Today, we can proudly say that India is one of the places where online shopping has been booming and will continue to do so. This means that online shopping has a lot of prospects in the future.

REFERENCES

1. http://www.google.com
2. http://www.youtube.com/Code with Harry/
3. http://www.github.com/django-project/
4. http://www.djangoproject.com
5. WITH THE HELP OF MY BROTHER.