Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B.Tech. CE Semester – IV

Subject:

Project Title: Online_Shopping_For_Accessories

Made By:

1) Name: Vrushika Makwana ; Roll No: CE062 ; ID: 19CEUSS117
2) Name: Shruti Makwana ; Roll No: CE061 ; ID: 19CEUSG018

Guided By: Pinkal Chauhan

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled as "**Online_Shopping_For_Accessories**" is a Bonafide report of the work carried out by

1) Name: **Vrushika Makwana**
   ID: **19CEUSS117**
2) Name: **Shruti Makwana**
   ID: 19CEUSG018

of Department of Computer Engineering, semester VI, under the guidance and supervision of Prof. **Pinkal Chauhan** for the subject System Design Practice during the academic year 2020-2021.

Project Guide
Assistance Professor
Prof. Jigar M. Pandya
Department of Computer Engineering,
Faculty of Technology,
Dharmsinh Desai University
Nadiad
Date: 30/04/2021

Head of the Department
Dr. C.K Bhensdadia
Prof. & Head,
Department of Computer Engineering,
Faculty of Technology,
Dharmsinh Desai University
Nadiad
Date: 30/04/2021

Contents

Abstract

Online of this word is used in almost every field. An online shopping system permits a customer to submit online order for items or services from a store. It aims to develop an online shopping of accessories for customers with the goal so that it is very easy to shop the things you want to buy from the online shopping websites available.

As we are very much aware about today's situation, going out in a crowed stores or shopping centres during festival season is very dangerous. With online shopping you can carry out your shopping from home. It also provide users with a good proposal to allow the user to get the appropriate selection of goods and gets the kind of high-quality services.

Introduction

Online shopping is a form of electronic commerce which allows consumers to directly buy goods or services from a seller over the Internet using a web browser or a mobile app. Consumers find a product of interest by visiting the website of the retailer directly or by searching among alternative vendors using a shopping search engine, which displays the same product's availability and pricing at different e-retailers. As of 2020, customers can shop online using a range of different computers and devices, including desktop computers, laptops, tablet computers and smartphones.

Technologies/tools used:

- Platform used: VS Studio 2019
- Platform used: MySQL(database)
- Language: Python with Django framework

Software Requirement Specifications:

This project aims to develop an online accessories shop for customers with the goal to buy the accessories of their wish and their likes.

Purpose:

- **Convenience.** The convenience is the biggest perk. Where else can you comfortably shop at midnight? There are no lines to wait in or shop assistants to wait on to help you with your purchases, and you can do your shopping in minutes.

- **No need to travel.** People don't usually like to move a lot to get what they want. Customers don't usually live near the shops they would want to visit, but today they have an option to visit the shops online by sitting in their homes.

Implements Details:

Modules

- Authentication module
  Admin & user must have to login to the system for using any of the functionality.

- Manage Products module
  In manage products admin can have access to add, update, remove products.

- Manage User module
  Admin can add or remove user

- Manage order
  Admin has access to order details to manage orders.

- Manage Cart module
  Users can add, update or remove products from cart.

- Manage Checkout and place order module
  Users can checkout and place orders.

Major Function Prototypes.

1. Signup and Login

   User has to first register themselves to see the products.

```
def home(request):
    if request.method == 'GET':
    return render(request,"login.html")

    else:
    return render(request,"registration.html")
```

```
def creation(request):
    if request.method=='POST':
        name=request.POST['uname']
        email_id=request.POST['id']
        number=request.POST['no']
        password=request.POST['pass']
        cpass=request.POST['cpass']

        if User.objects.filter(username=name).exists():
            return HttpResponse("<h1>User Already Exists</h1>")
        if User.objects.filter(email=email_id).exists():
            return HttpResponse("<h1>User Already Exists</h1>")
        if password!=cpass:
            return HttpResponse("<h1>Enter Password Again</h1>")

        info=register(name=name,phone_no=number,email_id=email_id,
        password=password,cpassword=cpass)
        info.save()
        info1=User.objects.create_user(username=name,password=password,
        email=email_id)
        info1.save()
```

```
        print("User Created")
        return render(request, "login.html")
        else:
            return render(request, "login.html")
    else:
        return render(request, "registration.html")

def login(request):
    if request.method=='POST':
        user_login=request.POST['name']
        password_login=request.POST['pass_login']
        user=auth.authenticate(username=user_login,password=password_login)
        if user is not None:
            auth.login(request,user)
            return render(request,"home.html")
        else:
            return render(request,'login.html')
    else:
        return render(request,'login.html')
```

```
class Login(View):
    def get(self, request):
        return render(request, 'login.html')

    def post(self, request):
        email = request.POST.get('email')
        password = request.POST.get('password')
        customer = Customer.get_customer_by_email(email)
        error_message = None
        if customer:
            flag = check_password(password, customer.password)
            if flag:
                # id and email is saved in session
                request.session['customer_id'] = customer.id
                request.session['email'] = customer.email
                return redirect("ShopHome")
            else:
                error_message = "Email or password is Invalid"
        else:
            error_message = "Email or password is Invalid"
        print(email, password)
        return render(request, 'login.html', {'error':error_message})
```

2. Add To Cart

   The cart function has a separate cart for each user.

```
def cart(request):
    if request.user.is_authenticated:
        user = request.user
        cart = Cart.objects.filter(user=user)
        print(cart)
        amount = 0.0
        cart_product = [p for p in Cart.objects.all() if p.user == user]
        if cart_product:
            for p in cart_product:
                temp_amt = (p.quantity * p.product.price)
                amount += temp_amt
            return render(request, 'addtocart.html', {'carts' : cart, 'amount' : amount})
        else:
            return render(request, 'emptycart.html')
```

   add_to_cart is used for adding items to the cart.

```
def add_to_cart(request):
    user = request.user
    prod_id = request.GET.get('prod_id')
    product = Product.objects.get(id=prod_id)
    Cart(user=user, product=product).save()
    return redirect('/cart')
```

3. Remove from cart
   remove_from_cart used to remove items from cart.

```
def remove_from_cart(request):
    cid = request.POST.get('cid')
    cart_item = Cart.objects.get(id = cid)
    cart_item.delete()
    return redirect(cart)
```

4. Update Cart item
   Update quantity of item added to the cart.

```
def update_cart_item(request):
    ciid = request.POST.get('cid')
    cart_item = Cart.objects.get(id = ciid)
    cart_item.quantity = request.POST.get('qty')
    cart_item.price = int(cart_item.quantity) * int(cart_item.product.price)
    cart_item.save()
    return redirect(cart)
```

5. Checkout function
   It is used for checkout where user can see the list of items they selected and have to fill some personal and shipping details.

```
def checkout(request):
        user = request.user
        cart_items = Cart.objects.filter(user=user)
        total = 0
        cart_product = [item for item in Cart.objects.all() if item.user == request.user]

        for item in cart_items:
            if(cart_product):
                total = total + (item.product.price * item.quantity)
        context = {
            'user': user,
            'cart_items': cart_items,
            'total': total,
        }
        return render(request, "checkout.html", context)
```

6. Product Page

   User can see the products and they register themselves.

```
class Index(View):
    def get(self, request):
        allProds = []
        catprods = Product.objects.values('category', 'id')
        cats = {item['category'] for item in catprods}
        for cat in cats:
            prod = Product.objects.filter(category=cat)
            n = len(prod)
            nSlides = n // 4 + ceil((n / 4) - (n // 4))
            allProds.append([prod, range(1, nSlides), nSlides])

        params = {'allProds':allProds}
        print('You are : ',request.session.get('email'))
        return render(request, 'index.html', params)

class Product_view(View):
    def get(self, request, myid):
        product=Product.objects.filter(id=myid)
        print(product)
        return render(request, 'product_view.html',  {'product' : product[0]})
```

7. Logout

   User can logout and redirected to login page

```
def logout_request(request):
    logout(request)
    messages.info(request, "You have successfully logged out.")
    return redirect("Login")
```

Testing:

Manual testing was performed in order to find and fix the bugs in development process.

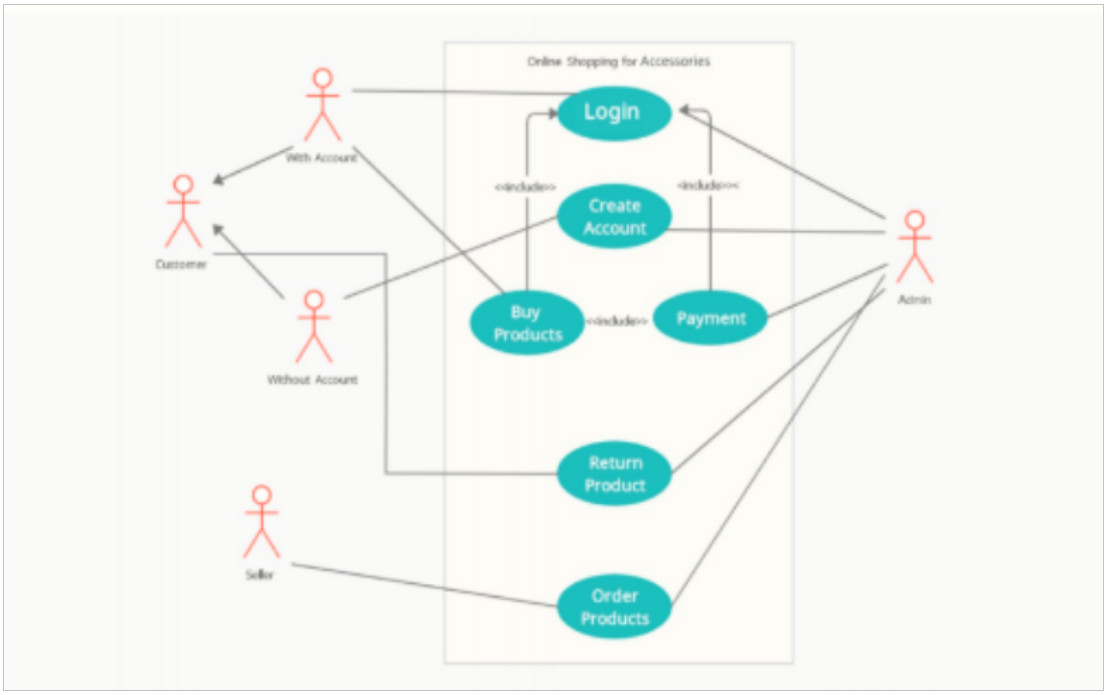
Testing Method: Manual Testing

Sr. No	Test Scenario	Expected Result	Actual Result	Status
1.	Register with incorrect credentials	User Should not able to register themselves	User will be provided with the message to fill correct information	Success
2.	Login with correct credentials	User should be able to login with correct credentials	User is Logged in.	Success
3.	Add to cart	Registered user should able to add items to the cart	User is able to add the products in the cart	Unsuccessful
4.	Remove Cart	User should able to remove the products from the cart	User are able to remove the products from the cart	Unsuccessful

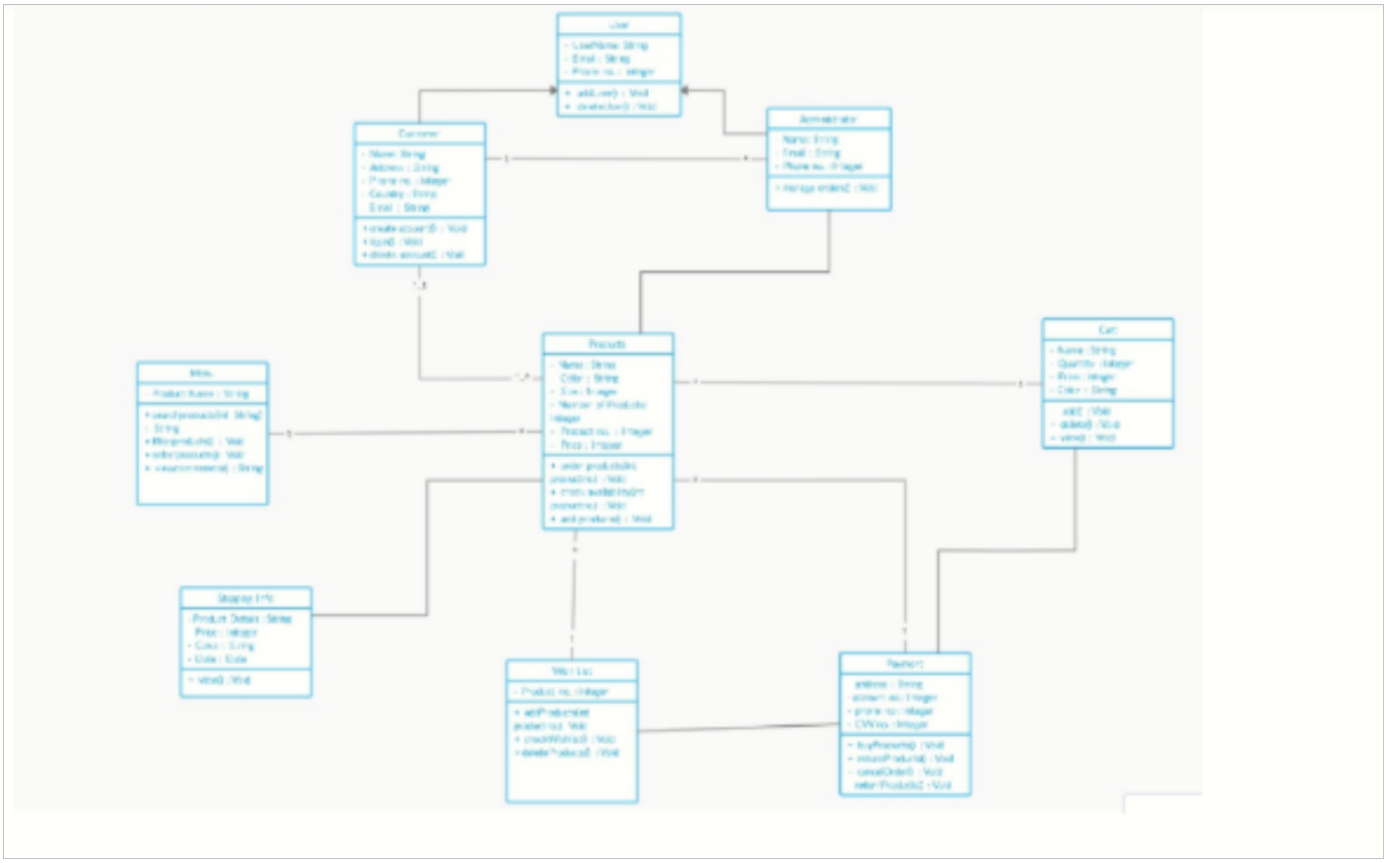
5.	Checkout	User should checkout and can get total price and even can fill the details	User are able to checkout and can get total price and even can fill the details	Unsuccessful
6.	Place Order	User should able to place order	User are able to place order	Unsuccessful
7.	Product Details	User should able to see the product details.	User are able to see the details with image.	Unsuccessful
8.	Logout	User should logout from the system	User are able to logout from the system	Unsuccessful

Design:

1. Use case diagram



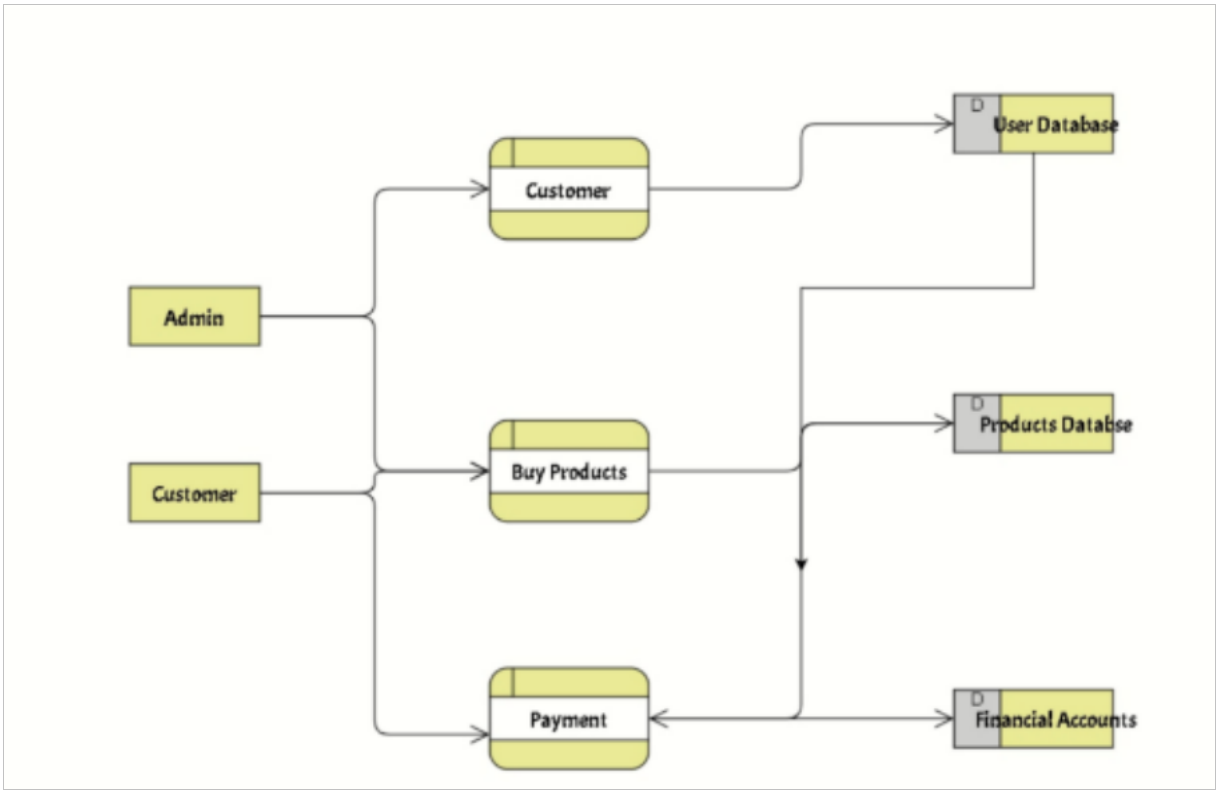
2. UML class diagram



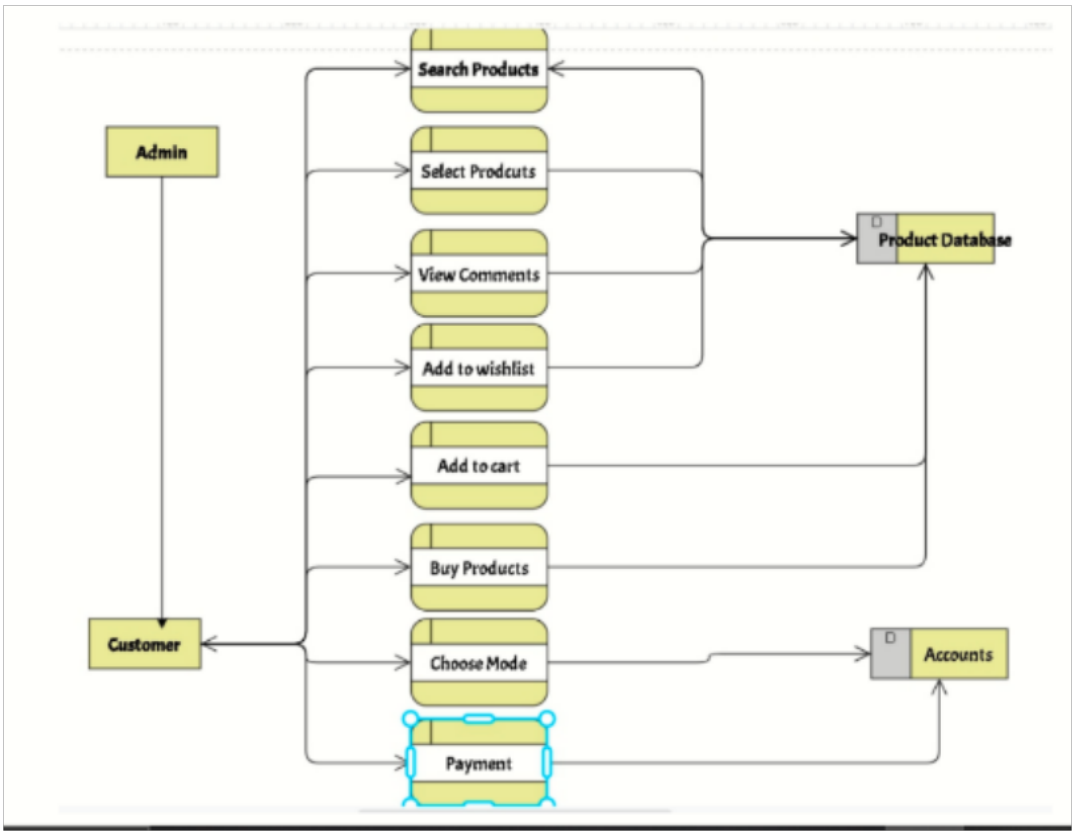
2. DFD structure

Level 0

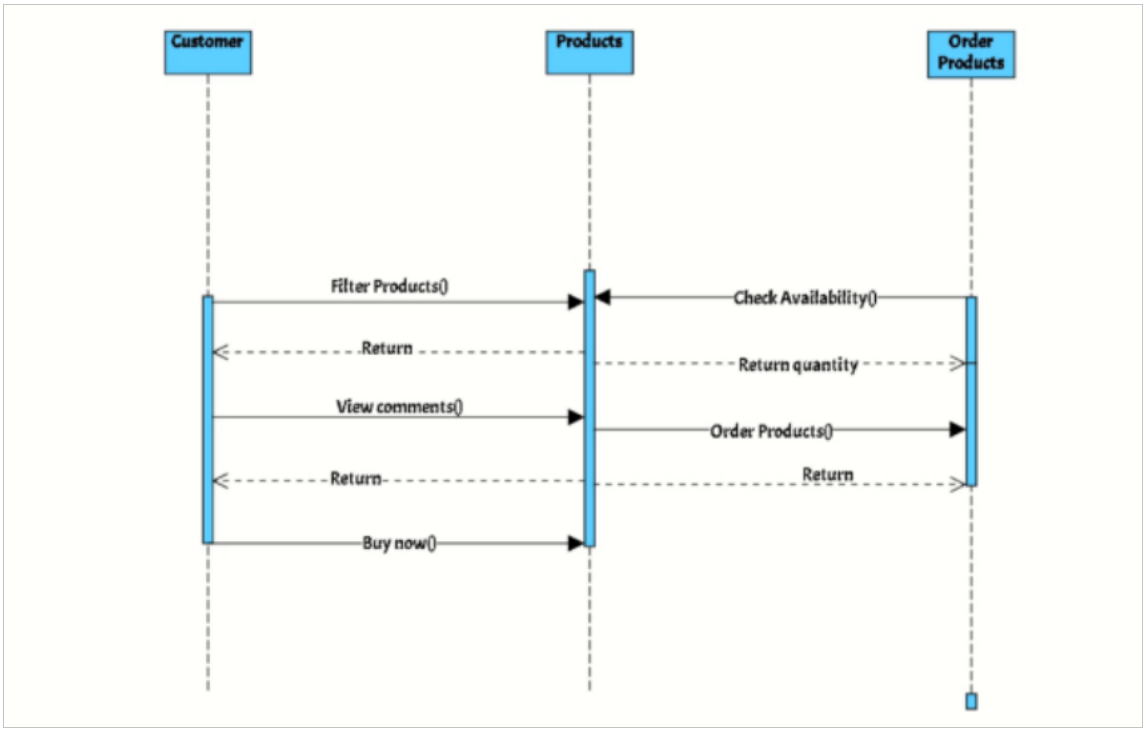
Level 1

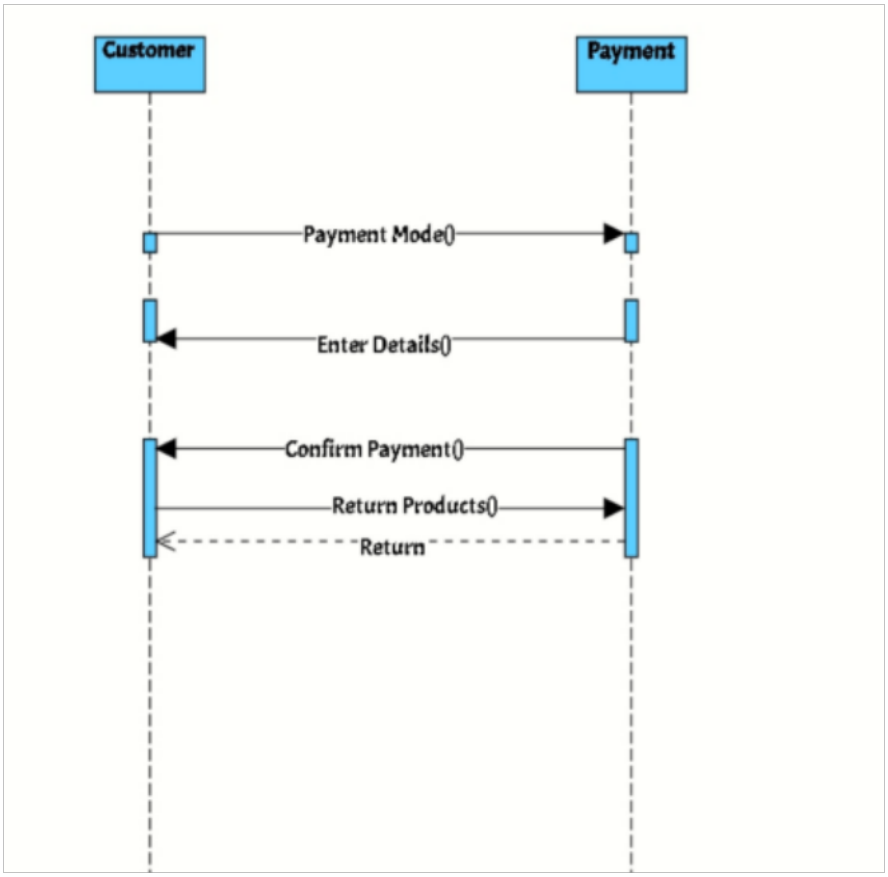


3. Structure chart



5. Sequence diagram





Conclusion:

This is to conclude that the project that we undertook was worked upon with sincere efforts. Most of the requirements have been fulfilled up to the mark and the requirements which have been remaining can be completed with short extension.

Functionalities that are successfully implemented in the system are:
User side functionalities:

- Login
- Signup
- Products fetched from database
- Add to cart
- Remove from cart
- Checkout
- Place order

- Logout
- Product detail page

Admin side:

- Customized UI
- Manage products
- Manage users

Limitations and Future Extensions:

We are able to implement most of the functionalities from the “fashioner” functionality module.

We aim to make this product ready to be used in practical use cases.

- Limitations:
  - In Django we have used MySQL database. It is stored with application files so size will increase as users increase.
  - Limited number of administration (we have only 1 admin)
  - Only cash on delivery option.
  - All users have to login first to see the homepage of the website.
  - Login is compulsory to visit the website.
  - Forgot password option is not given.

- Future Extensions:
  - Provide online payment options to users.
  - Provide multiple admin functionality
  - Product recommendations using machine learning
  - More interactive user interface
  - Search product functionality
  - More security options
  - Email verification on signup
  - Forgot password option
  - Admin can track the product

Bibliography:

Images and user interface related to this project is taken from the following resources.

- https://getbootstrap.com (styling)
- https:// google.com (images and product details)
- GitHub
- Django Documentation