

A PROJECT REPORT
ON
ONLINE SHOPPING SYSTEM

By

Goti Shreya P. (CE045) (19CEUOS156)
Kyada Sweta R. (CE056) (19CEUES118)

B.Tech CE Semester - IV

Subject: Software Engineering Principles and Practices

Guided by:

Prof. Pinkal Chauhan

Prof. Ankit Vaishnav



Faculty of Technology
Department of Computer Engineering
Dharm Singh Desai University

Abstract

Along with the growth of the Internet, the trend shows that e-commerce has been growing significantly in the last several years. E-commerce provides an easy way to sell products to a large customer base. This project aims to develop online shopping for customers with the goal so that it becomes very easy to shop your loved things from an online shopping site available on the web. With the help of this you can carry out an online shopping of your favourite clothes and fashion accessories from your home.

Introduction

E-commerce shopping System in which there are mainly two users Admin and Customers. Customers can purchase things such as clothes, footwear, etc. All the items are displayed category wise. Customers can pick their favourite items from our online shopping site looking at cost and quality. Customers can also add their favourite items in their wishlist. Admin can add item categories, items, manage orders. Admin can also see user profiles, carts, wishlists.

Tools/Technologies Used

Technologies :

- Django
- Python
- MySQL
- Bootstrap
- JavaScript
- HTML

Tools :

- Git
- Visual Studio Code

Software Requirement Specifications

Functional requirements :

User Requirements :

R.1 : Manage user

R.1.1 : Register user

Description: A new user can register to this site.

Input : User's name, Email, password, name, address and contact details.

Output : Registered successfully

R.1.2 : Login user

Description : A registered user can login to the site.

Input : User's email or username and password.

Output : Redirect to home page after successful login.

R.1.3 Update user info

Description : A registered user can update their account information.

Input : Click on edit info button.

Output : Information updated successfully.

R.1.4 Forgot password

Description : A User can reset the password if it is forgotten.

Input : Email Id which is registered.

Output : Verification code will be sent to entered email Id.

R.2 : Manage Orders

R.2.1 : Place order

Description: after adding the items to the cart, the user can place the order.

Input: customer have to provide shipping address and contact details.

Next: Payment Process

R.2.2 Payment Process

Description: It will display users to choose payment methods like Payment with card, Net Banking, Paytm.

Input : Select a payment option.

Output : If After payment is done, “order placed successfully” message will be printed.

R.2.3 Cancel order

Description: It will cancel the placed order before shipping

Input : Select cancel order option

Output : “Order cancelled” message will be shown

R.2.4 : Return or exchange a product

Description: after receiving the order if the customer is not satisfied with the products then they can also return or exchange their products. Customers will be provided a form for return or exchange.

Input: customers have to provide a description or reason why they want to return or exchange the product. If possible they should provide a picture of a product.

Output: if the reason given by the customer is valid, then he/she will be notified that they are can exchange or return the product.

R.2.5 : Reviews

Description : After users receive the order they will be able to review their purchased items.

Input : user can add photos of purchased item ,write their experience and also be able to give ratings to the particular product.

Output : Review will be added to the review section of the product.

R.3 : Cart Management

R.3.1 : Add to Cart

Description : User can add the item into the cart which he/she wants to buy and can also choose the size and color of the product.

Input : click on “Add to cart” button

Output : product will be added to the cart.

R.3.2 : Remove from Cart

Description : User can also remove the item from cart

Input : click on “Remove from cart” button

Output : product will be removed from the cart.

R.4 : Wishlist Management

R.4.1 : Add to Wishlist

Description : User can add the item into the wishlist which he/she likes and wants to buy in future.

Input : click on “Add to wishlist” button

Output : product will be added to the wishlist.

R.4.2 : Remove from Cart

Description : User can also remove the item from wishlist

Input : click on “Remove from wishlist” button

Output : product will be removed from the wishlist.

R.5 : Online tracking of shipments

Description : Users can track their order

Input : to track order, the user has to provide a tracking id.

Output : It will display the current tracking information about the order.

Admin Requirements :

R.1 : Product Management

R.1.1 : Add Product

Description : Admin can add new products.

Input : click on “Add new product” button and Enter the name, description, sizes, colors of product along with pictures of the product.

Output : product will be added to the database.

R.1.2 : Update Product Details

Description : Admin can update product details.

Input : click on “Edit product details” button and make the desired changes.

Output : product details will be updated into the database.

R.1.3 : Remove Product

Description : User can also remove the item from cart

Input : click on “Remove Product” button.

Output : product will be removed from the database.

R.2 : Product Category Management

R.2.1 : Add Category

Description : Admin can add a new category of the products.

Input : click on “Add new Category” button and then Enter the name of the new category and description along with pictures related to that category.

Output : A new category will be added in the database.

R.1.2 : Update Category Details

Description : Admin can update category details.

Input : click on “Edit category details” button and make the desired changes.

Output : Category details will be updated into the database.

R.2.2 : Remove Category

Description : Admin can remove the specific category of products.

Input : click on “Remove Category” button

Output : Category will be removed from the database if no product of that category is there.

R.3 : Monthly Sales Report

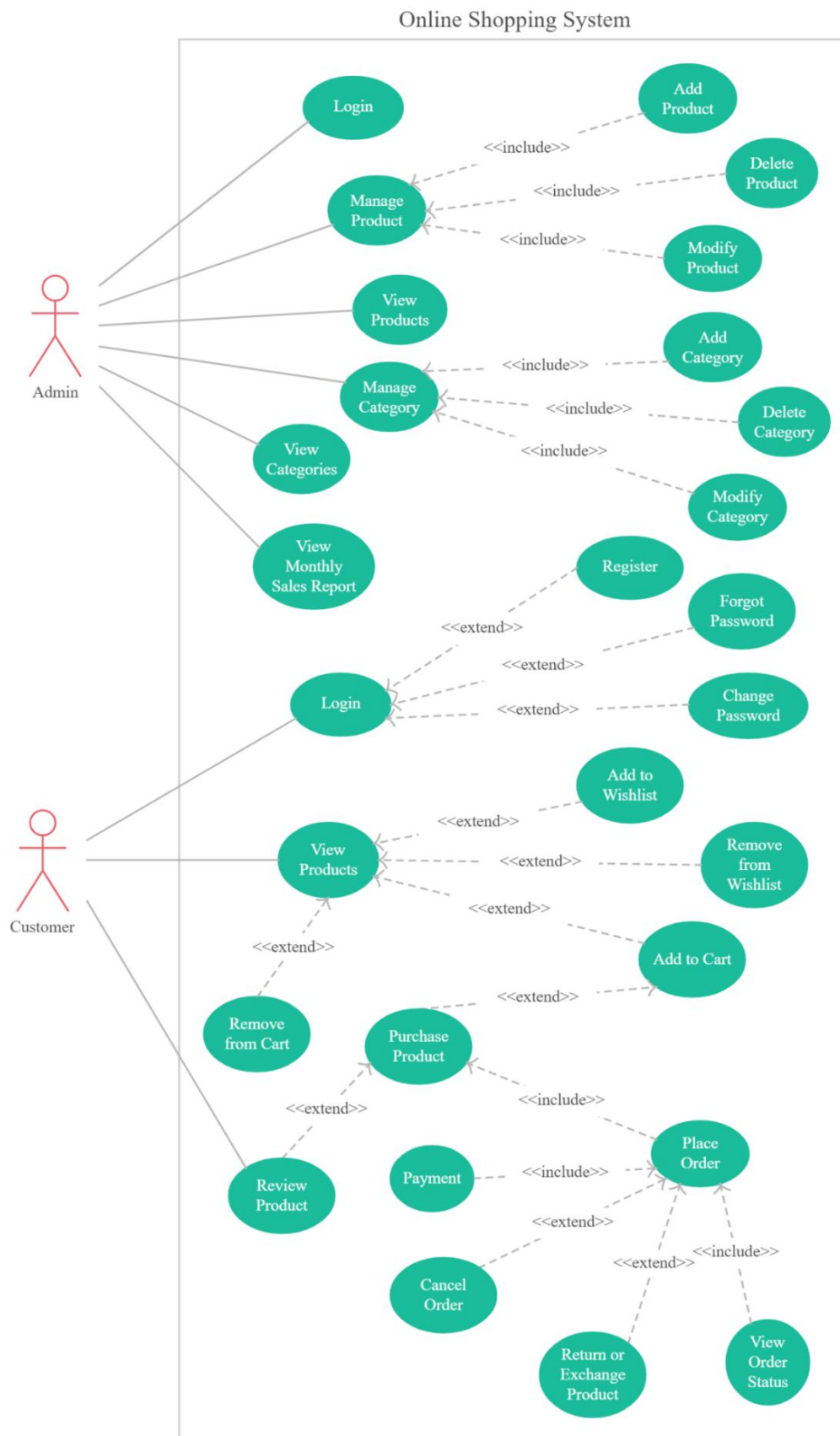
Description : Admin can see daily sales as well as monthly sales report of different products

Input : click on “show sell report” button

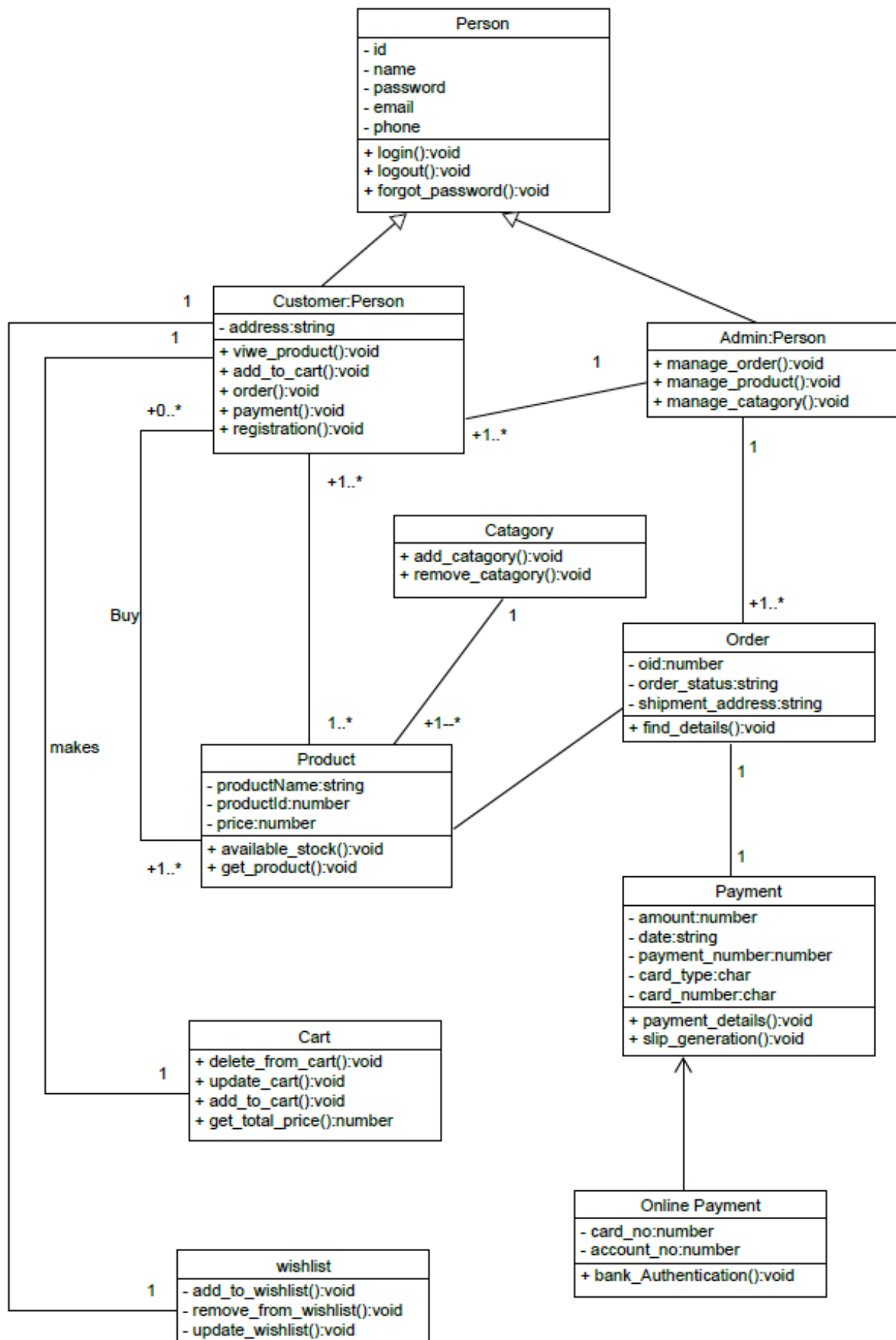
Output : It will show the monthly sales report of last month. It can also show the monthly sales reports of previous months.

Design

Use case diagram

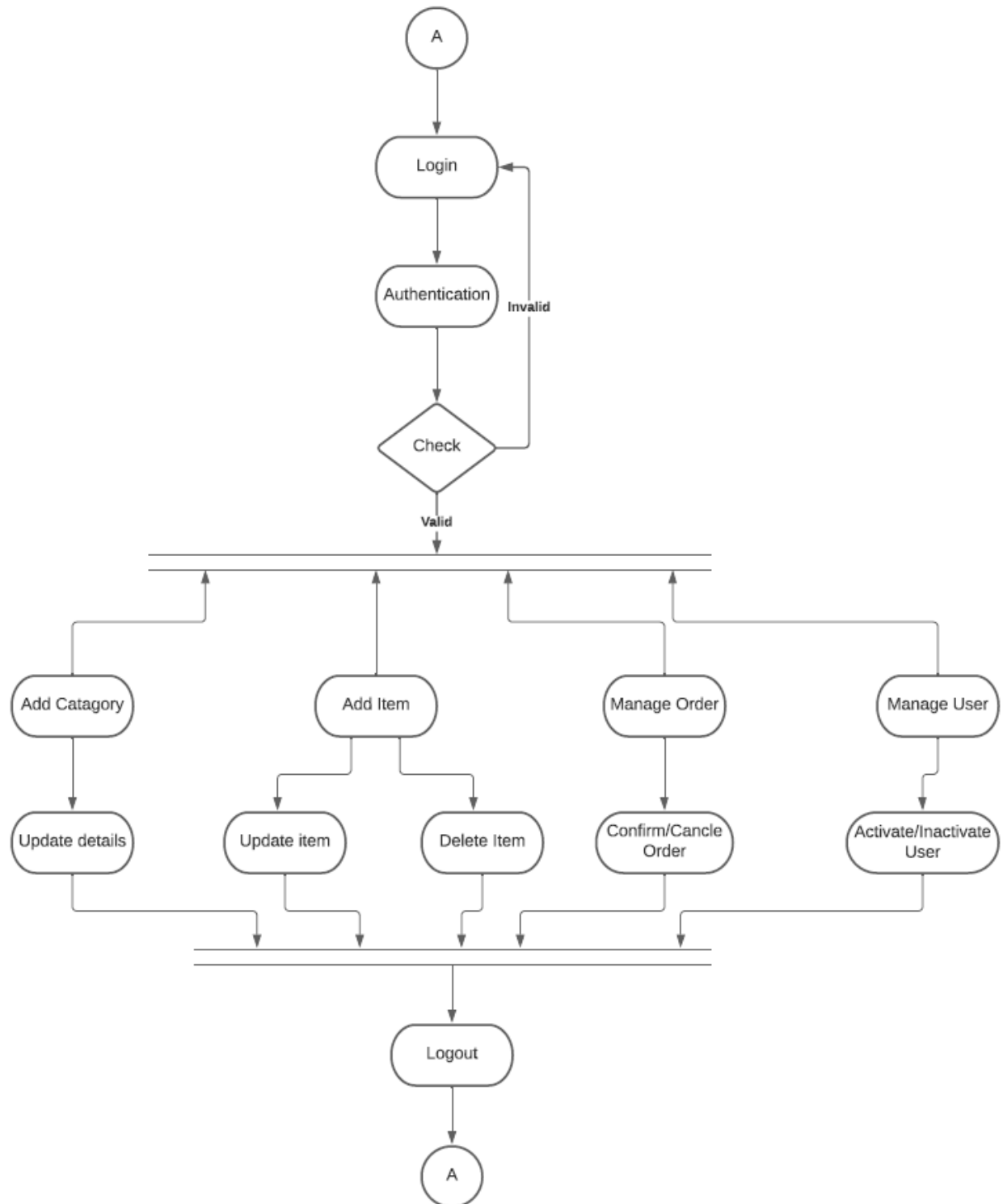


Class Diagram

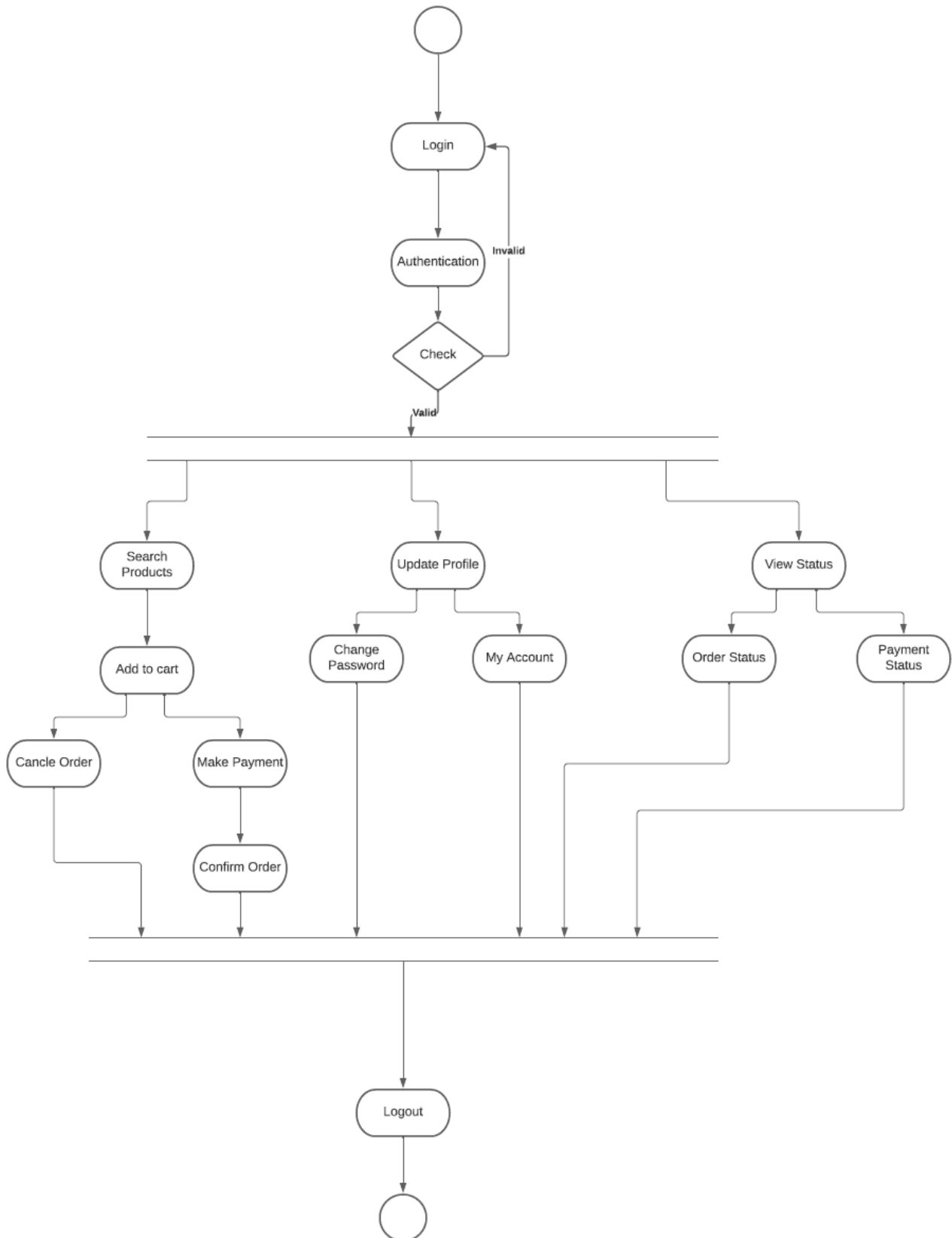


Activity Diagrams

Activity Diagram For Admin Side

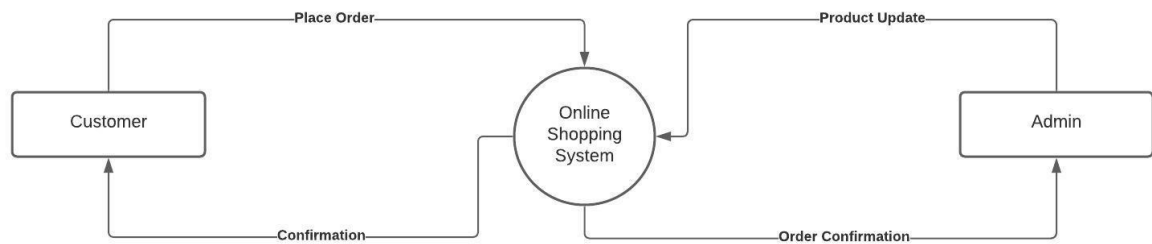


Activity Diagram for User Side

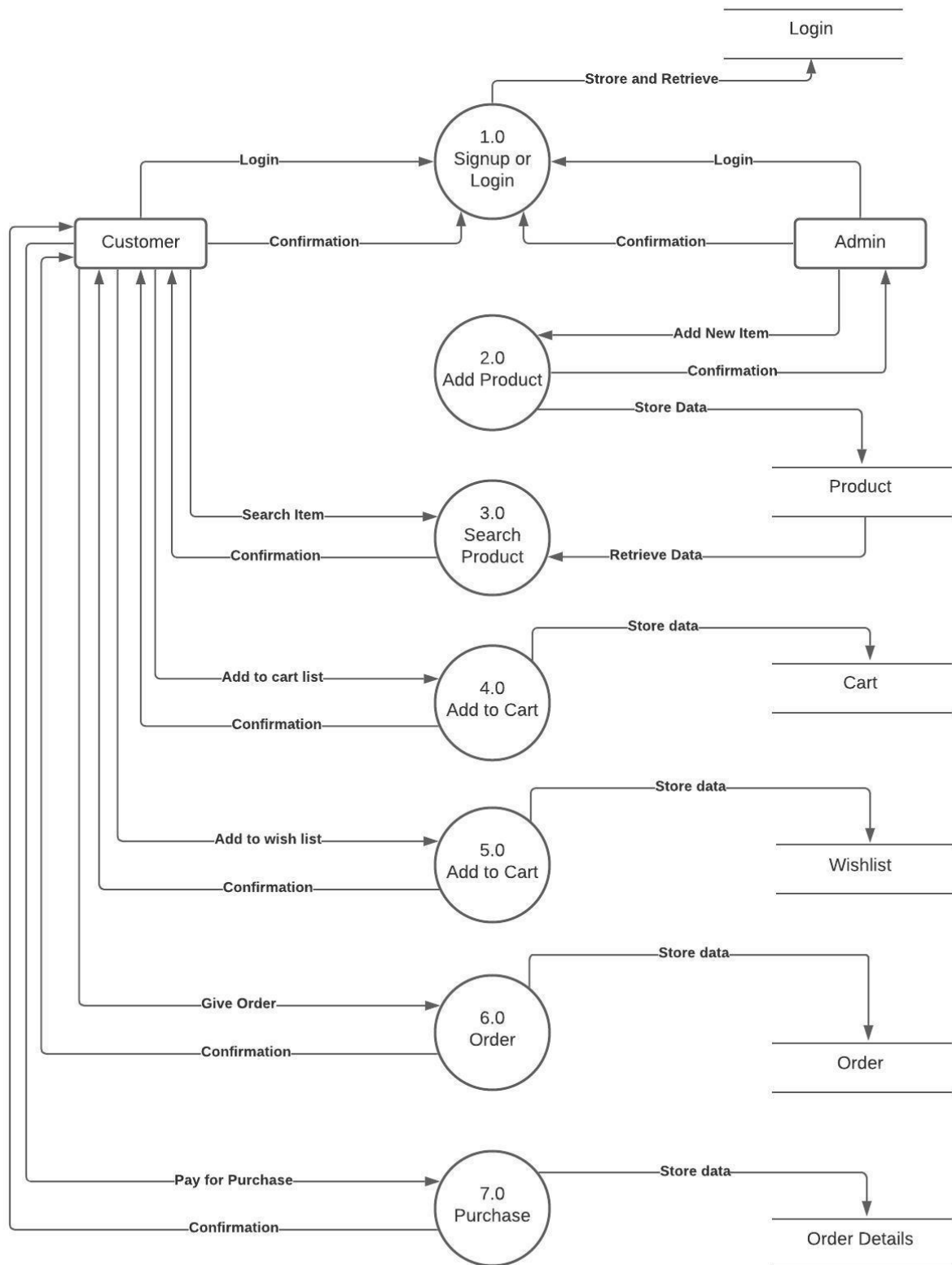


Data Flow Diagrams

Level 0 Data Flow Diagram

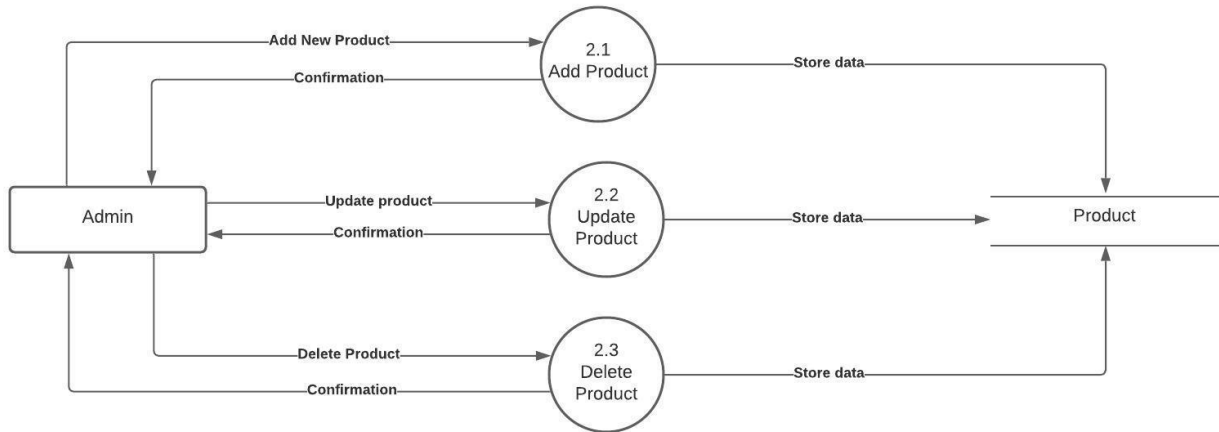


Level 1 Data Flow Diagram

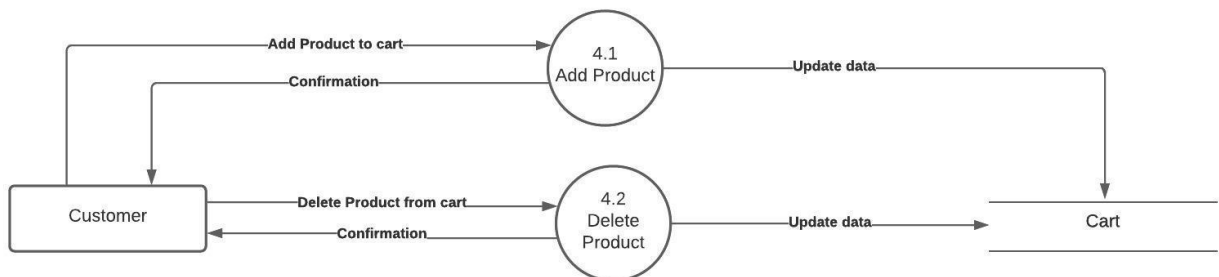


Level 2 Data Flow Diagram

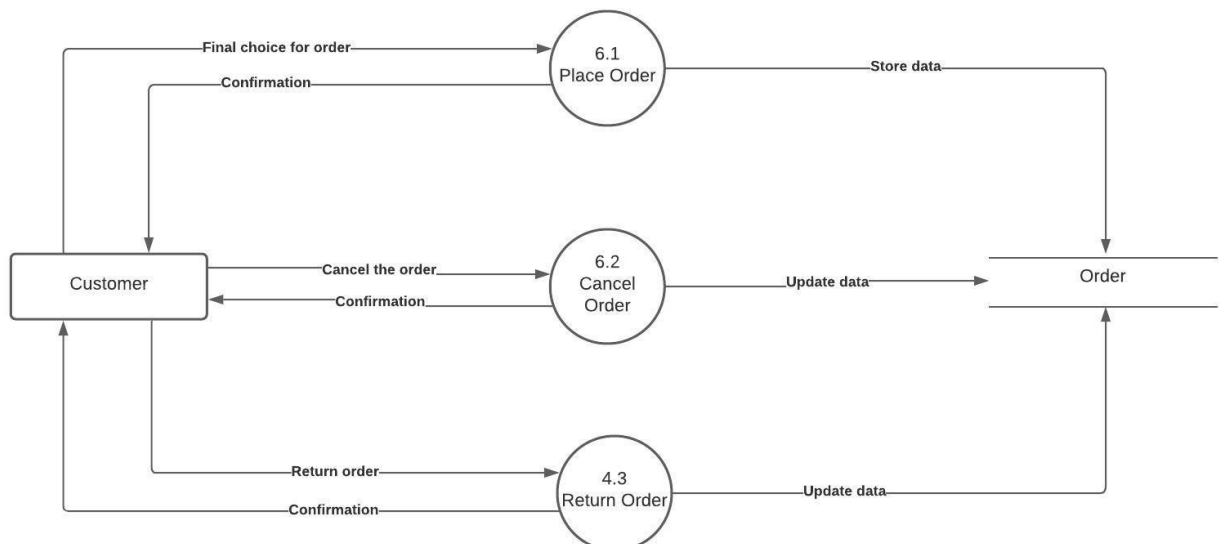
For Add Product Bubble

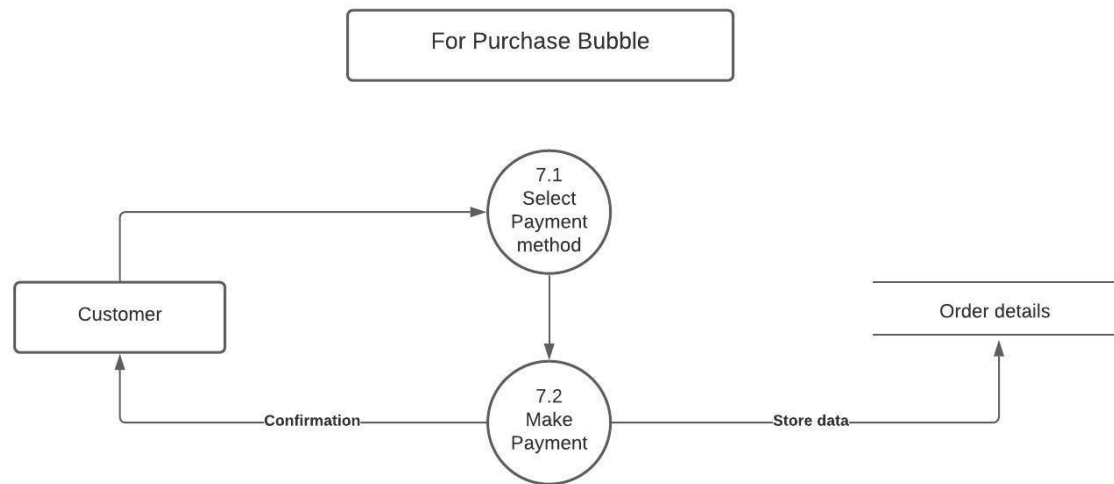


For Add to Cart Bubble



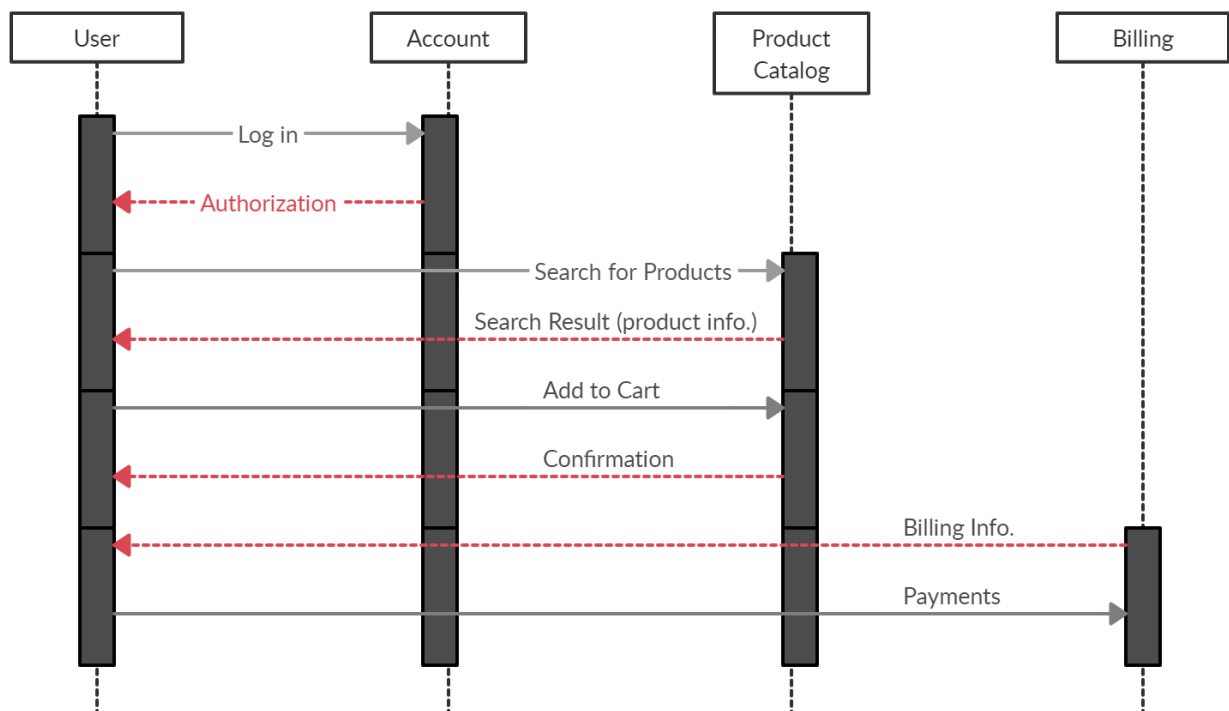
For Order Bubble



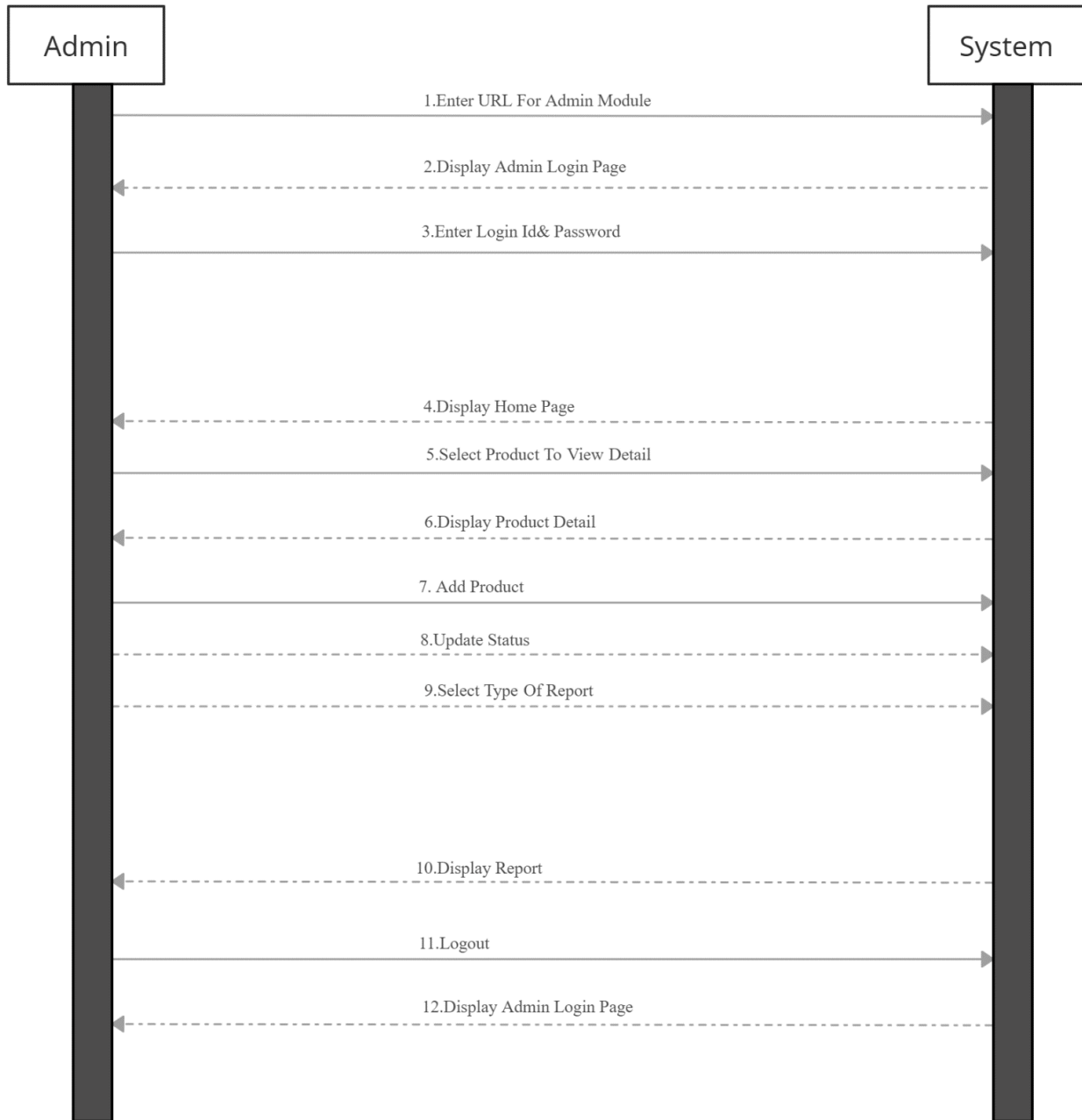


Sequence Diagrams

Sequence Diagram for User Side



Sequence Diagram for Admin Side



Implements Details

Modules

- 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.
- Authentication module
Admin & user must have to login to the system for using any of the Functionality.
- Manage Cart module
Users can add ,update or remove products from cart.
- Manage Wishlist
Users can add or remove products from wishlist.
- Manage Checkout and place order module
Users can checkout and place orders.
- Manage category module
Admin has access to add or remove categories from the database.

Major Function Prototypes

1. Add to cart

The `user_cart` function has a separate cart for each user.

```
def user_cart(request):
    user = request.user
    itms = Cart.objects.filter(user = user)
    length = len(itms)
    total = 0
    items = []
    for item in itms:
        if(item.ordered == False):
            total = total + item.price
            items.append(item)
    context = {
        'items': items,
        'length': length,
        'total': total,
        'men': men,
        'women': women,
        'girls': girls,
        'boys': boys,
    }
    return render(request, "cart-page.html", context)
```

`Add_to_cart` is used for adding items to the cart.

```
def add_to_cart(request):
    user = request.user
    itm = Product.objects.get(id=request.POST.get('pid'))
    item = Cart(user = user,
                product = itm,
                quantity = request.POST.get('qty'),
                size = request.POST.get('size'))
    item.price = int(itm.price) * int(item.quantity)
    item.save()
    return redirect(user_cart)
```

2. 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(user_cart)
```

3. Update Cart item

Update size and quantity of item added to the cart.

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

4. Place order function

To place an order.

```
def place_order(request):
    user = request.user
    items = Cart.objects.filter(user = user)
    length = len(items)
    total = 0
    for item in items:
        total = total + item.price
    order = Order(user = user, total_amount = total,
                  address = request.POST.get('address'),
                  address2 = request.POST.get('address2'),
                  country = request.POST.get('country'),
                  state = request.POST.get('state'),
                  pincode = request.POST.get('pincode'))
    order.save()
    for item in items:
        order.products.add(item)
        itm = Cart.objects.get(id = item.id)
        itm.ordered = True
        itm.save()
    return redirect(profile_page)
```

5. Checkout function

It is used for checkout.

```
def checkout(request):
    user = request.user
    items = Cart.objects.filter(user = user)
    length = len(items)
    total = 0
    for item in items:
        if(item.ordered == False):
            total = total + item.price
    context = {
        'user': user,
        'items': items,
        'length': length,
        'total': total,
        'men': men,
        'women': women,
        'girls': girls,
        'boys': boys,
    }
    return render(request, "checkout-page.html", context)
```

6. User wishlist and add to wishlist functions

The user_wishlist function has a separate cart for each user.

```
def user_wishlist(request):
    user = request.user
    items = Wishlist.objects.filter(user = user)
    length = len(items)
    context = {
        'items': items,
        'length': length,
        'men': men,
        'women': women,
        'girls': girls,
        'boys': boys,
    }
    return render(request, "wishlist-page.html", context)
```

Add_to_wishlist is used for adding items to a wishlist.

```
def add_to_wishlist(request):
    user = request.user
    product = Product.objects.get(id=request.POST.get('pid'))
    item = Wishlist(user = user, product = product)
    item.save()
    return redirect(user_wishlist)
```

7. Remove from wishlist function

```
def remove_from_wishlist(request):
    wid = request.POST.get('wid')
    wishlist_item = Wishlist.objects.get(id = wid)
    wishlist_item.delete()
    return redirect(user_wishlist)
```

8. Category page function

```
@login_required(login_url='/accounts/')
def category_page(request, cid):
    products = Product.objects.filter(category_id = cid)
    context = {
        'products': products,
        'men': men,
        'women': women,
        'girls': girls,
        'boys': boys,
    }
    return render(request, "category-page.html", context)
```

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	Login with incorrect credentials	User should not able to log in.	User is given a message. And redirected to login page.	Success
2	Login with correct credentials	User should be able to log in.	User is logged in and shown the home page.	Success
3	Signup with correct details	User should not be able to signup with incorrect fields.	User is not able to sign up with incorrect details.	Success
4	Add to cart	Logged in user should able to add items to the cart	User is able to add item to the cart	Success
5	Remove from cart	User should able to remove items from the cart	User is able to remove items from the cart	Success
6	Update cart	User should able to update size and quantity of the product	User is able to update cart	Success

7	Remove from cart	User should able to remove the product	User is able to remove from the cart	Success
8	Add to wishlist	Use should able to add items to wishlist	User is able to add items to the wishlist	Success
9	Remove form the wishlist	User should able to remove from the wishlist	User is able to remove items from the wishlist	Success
10	Checkout	User should be able to checkout and get total price	User is able to checkout and getting total price	success
11	Place order	User should be able to place order	User is able to place order	success
12	Category filtering	User should be able to filter products category wise	User is able to filter category wise	success
13	Show product details	User should be able to show product details	User is able to show product details with images.	success
14	Show profile	User should able to show profile	User is able to show profile	success
15	logout	User should be logged out and restricted from the system until next login.	User is successfully logged out and not able to access the system without signing again.	success

Automation testing was performed using selenium.

Testing Method: AutomationTesting

Only login is tested

```
from django.test import TestCase

# Create your tests here.
from django.test import LiveServerTestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time

class AccountTestCase(LiveServerTestCase):

    def setUp(self):
        module selenium.webdriver
        self.selenium = webdriver.Chrome(executable_path=r'E:\chromedriver_win32\chromedriver.exe')
        super(AccountTestCase, self).setUp()

    def tearDown(self):
        time.sleep(300)
        self.selenium.quit()
        super(AccountTestCase, self).tearDown()

    def test_register(self):
        selenium = self.selenium
        #Opening the link we want to test
        selenium.get('http://127.0.0.1:8000/accounts/login/')
        #find the form element
        username = selenium.find_element_by_name('username')
        password = selenium.find_element_by_name('password')
        submit=selenium.find_element_by_name('submit')

        #Fill the form with data
        username.send_keys('fashion')
```

env) 0 0 8 In 17, Col 24 Spaces

```
password.send_keys('fashion')

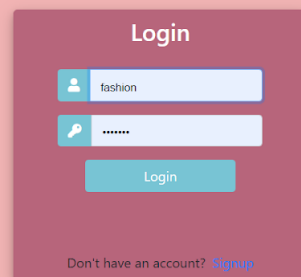
time.sleep(3000)
#submitting the form
submit.send_keys(Keys.RETURN)

#check the returned result

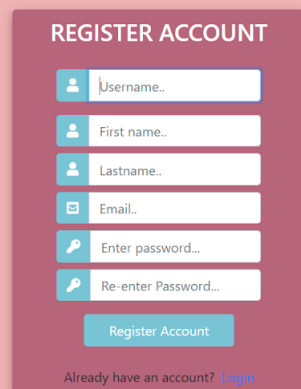
# assert 'Check your email' in selenium.page_source
```

Screenshots(layout)

Login and signup

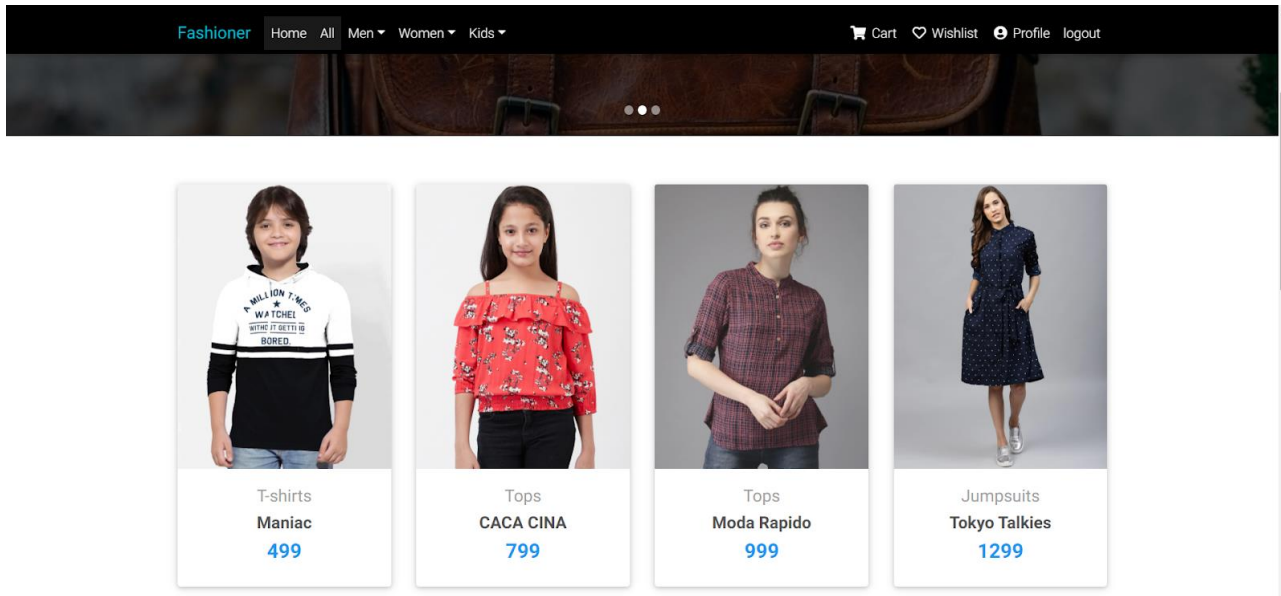


A login form titled "Login" with a teal header. It features two input fields: a username field with the text "fashion" and a password field with masked characters "*****". Below the fields is a teal "Login" button. At the bottom, there is a link that says "Don't have an account? [Signup](#)".

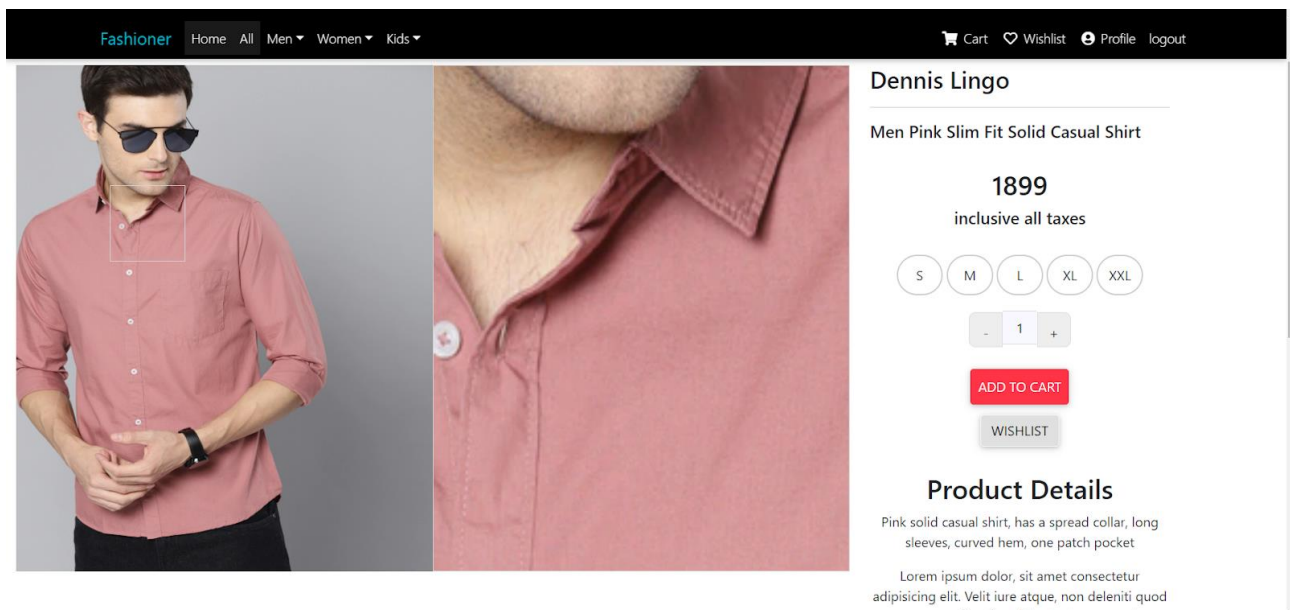


A registration form titled "REGISTER ACCOUNT" with a teal header. It contains six input fields: "Username..", "First name..", "Lastname..", "Email..", "Enter password...", and "Re-enter Password...". Each field has a corresponding icon (person, envelope, or key). Below the fields is a teal "Register Account" button. At the bottom, there is a link that says "Already have an account? [Login](#)".

Homepage



Product-detail page




Cart-page

[Fashioner](#) [Home](#) [All](#)

Shopping cart

Cart (11 items)



HIGHLANDER

Men Black Slim Fit Mid-Rise Clean Look Stretchable Cropped Jeans


SIZE : S

QUANTITY : 78

REMOVE ITEM

UPDATE ITEM

\$109122



GoSriKi

Beige Linen Blend Printed Chanderi Saree

SIZE : XL

QUANTITY : 16

The total amount of

Temporary amount

\$161979


Shipping

\$00




The total amount of (including VAT)

\$161979

GO TO CHECKOUT

Add a discount code (optional) 

Checkout page

[Fashioner](#) [Home](#) [All](#) [Men](#) [Women](#) [Kids](#)  Cart  Wishlist  Profile [logout](#)

Checkout form

Shipping address

First name

fashioner

Last name

style

Username

@fashion

Email (Optional)

fashioner@gmail.com

Address

Address 2 (Optional)

Country

State

Zip

Your cart

11

HIGHLANDER

Men Black Slim Fit Mid-Rise Clean Look Stretchable Cropped Jeans

1399

GoSriKi

Beige Linen Blend Printed Chanderi Saree

1999

CACA CINA

Girls Red Floral Printed Blouson Top

799

Moda Rapido

Women Navy & Red Checked Shirt-Style Top

999

Maniac

Boys White & Black Colourblocked Hood T-shirt

499

CACA CINA

Girls Red Floral Printed Blouson Top

799

Maniac

Boys White & Black Colourblocked Hood T-shirt

499

Wishlist page

Fashioner


Home


All


Men

Women




Kids

 Cart

 Wishlist

 Profile

logout

image	Product	Price	Remove
	CACA CINA	799	<div>Remove</div>
	GoSriKi	1999	<div>Remove</div>
	HIGHLANDER	1399	<div>Remove</div>

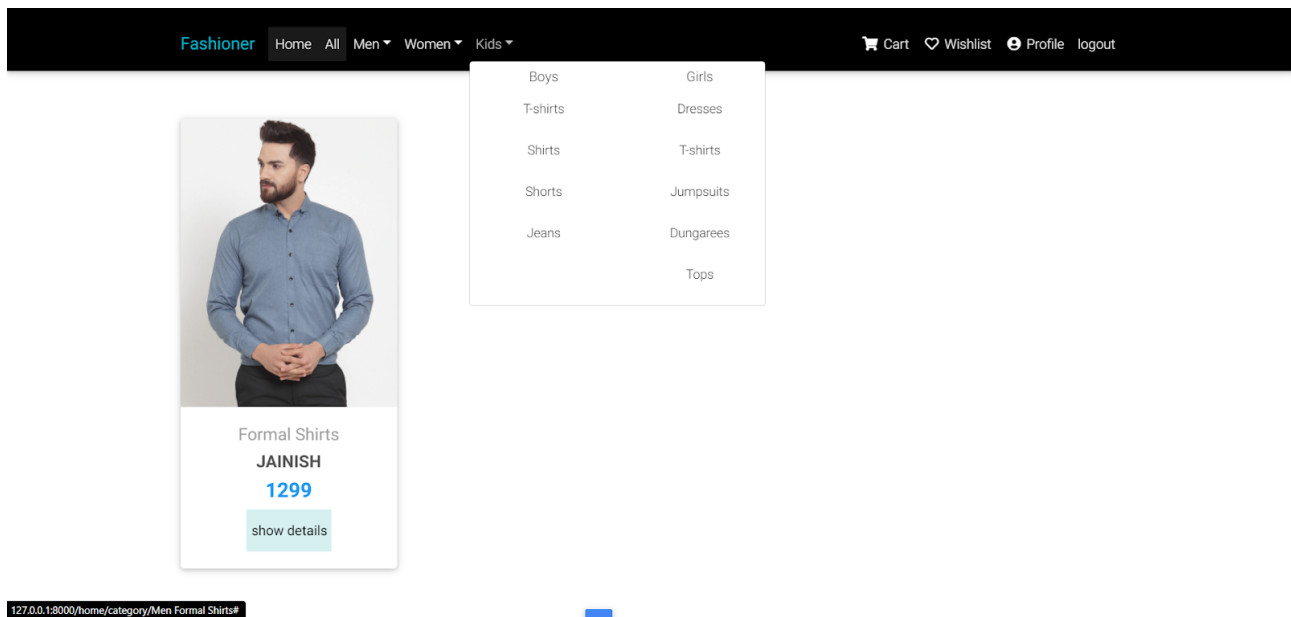
Profile page

Fashioner Home All Men Women Kids				Cart Wishlist Profile logout	
-----------------------------------	--	--	--	------------------------------	--

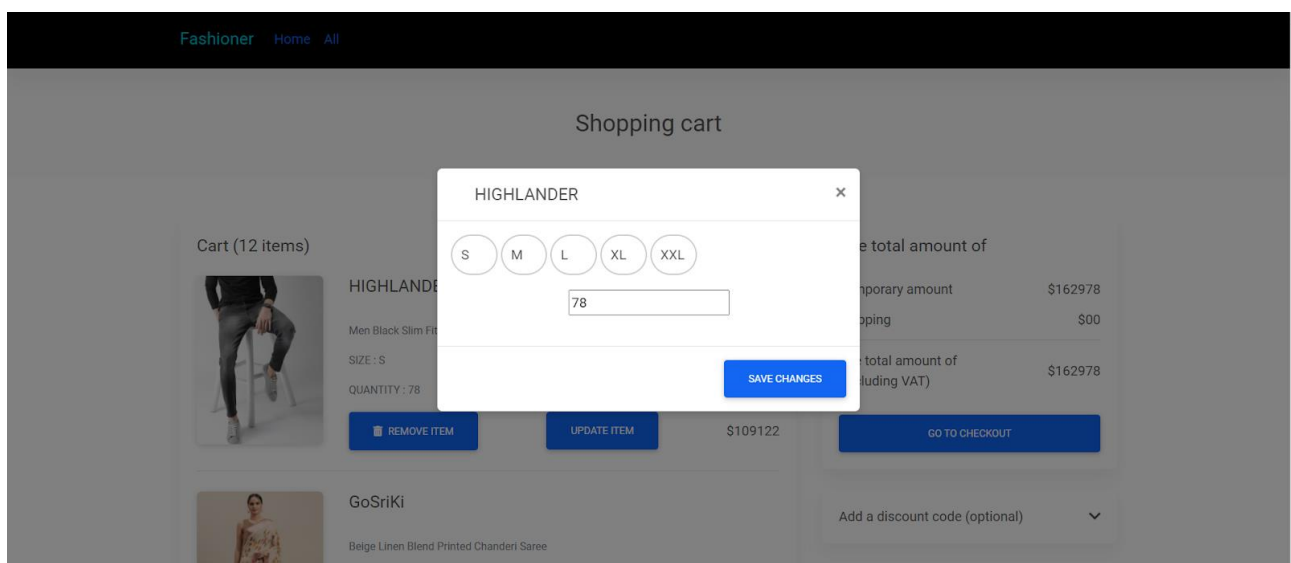
Userdetails

Username	fashion
Firstname	fashioner
Lastname	style
Email	fashioner@gmail.com

Category wise filtering

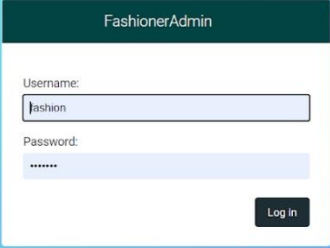


Update cart item



Admin

Login



The login form is titled "FashionerAdmin" and is set against a blue-to-teal gradient background. It contains two input fields: "Username:" with the value "fashion" and "Password:" with masked characters "*****". A "Log in" button is located at the bottom right of the form.

Admin panel

FashionerAdmin

WELCOME, FASHIONER. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Users

+ Add

Change

FASHIONERMAIN

Carts

+ Add

Change

Categories

+ Add

Change

Orders

+ Add

Change

Products

+ Add

Change

Wish lists

+ Add

Change

Recent actions

My actions

admin

User

+ admin

User

× dddddd

User

× dar44

User

× dar

User

admin

User

Cart object (1)

Cart

Cart object (2)

Cart

Cart object (2)

Cart

+ Cart object (2)

Cart

AUTHENTICATION
AND AUTHORIZATION

Users

+ Add

FASHIONERMAIN

Carts

+ Add

Categories

+ Add

Orders

+ Add

Products

+ Add

Wish lists

+ Add

Add category

Category name:

Category for:

Category id:

Save and continue editing

Save and add another

SAVE

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

- Signup, Login, Logout
- User authentication
- Products fetched from database
- Add to cart (with size and quantity)
- Remove from cart
- Update cart
- Add to wishlist
- Remove from wishlist
- Checkout and Place order
- Category filters(Men, Woman, kids(girls, boys))
- Product detail page
- User profile page

Admin side functionalities

- Customized UI
- Manage category, products
- Manage users and orders

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 sqlite 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

Bibliography

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

- <https://getbootstrap.com> (styling)
- <https://fontawesome.com> (fonts)
- <https://mdbootstrap.com> (basic templates)
- <https://fonts.google.com> (fonts)
- <https://unsplash.com> (photos)
- Clothes photos taken from internet(random resources)
- Github