S.P. Mandali's
Ramnarain Ruia Autonomous College
Matunga, Mumbai

Project Report
On
Online Grocery Store - Website

Project Guide
Mrs. Pooja Rasam

Project By

Asavari Vijay Akshekar

TYBSC
Department of Computer Science
(2021-2022)

Acknowledgement

It gives us pleasure in presenting the partial project report on ‘Online Grocery Store Website’.

Firstly, we would like to express our indebtedness appreciation to our internal guide Prof. Pooja Rasam. Her constant guidance and advice played very important role in making the execution of the report. She always gave us her suggestions, that were crucial in making this report as flawless as possible.

We would like to express our gratitude towards Prof. Dr. Mrs. Megha Sawant Head of Computer Science Department, Ramnarain Ruia Autonomous College for her kind co-operation and encouragement which helped us during the completion of this report.

Also, we wish to thank all faculty members for their whole hearted co-operation for completion of this report. We also thank our laboratory assistants for their valuable help in laboratory.

A special thank of mine goes to my classmates who helped me out in testing/debugging the project which made it possible to complete my project.

I am highly indebted to Ramnarain Ruia Autonomous College Computer Science Department for providing guidance and constant supervision as well as for keeping me on track regarding the project & also for their support in completing the project.

Last but not the least, the backbone of our success and confidence lies solely on blessings of dear parents and lovely friends.

Sincere Thanks from,

Asavari Vijay Akshekar.

INDEX

Preliminary Investigation

1. Description of Present System

Traditional Grocery Shopping

The majority of the Indian customers are still finding the walk-in store method for buying groceries convenient compared to the online.

The following are the reasons for practising the traditional method of buying groceries: -

- People get to examine every single product carefully.
- Many families in India have bought their groceries from a shop in their locality before the existence of these e-commerce websites and mutual trust has been established between the shopkeeper and the consumer.
- Also, people buy groceries by placing a telephonic order to the local shop every month.

Online Grocery Shopping

Online grocery shopping has come as a relief for everyone as it saves time and is incredibly easy to use. All one needs is an internet connection or the application on their phone.

No more hassles of sweating it out in crowded markets, grocery shops & supermarkets - now shop from the comfort of your home; office, or on the move.

- Also considering the **current situation** i.e.; Covid 19 and the norms related to it, why not shift ourselves to the Online method of purchasing groceries?
- Not only supermarkets but the local shops also can bring their business online so that the shops won’t be crowded and safety measures related to covid 19 are followed.

2. Limitations of Present System

Traditional Grocery Shopping

- Recent economic conditions have made cashless transactions a necessity and many local shops don’t promote this at a convenient level.
- In-store grocery shopping requires you to spend hours in stores.
- You can also be lured into the temptations of buying various products you don’t require (for eg: DMart). Thus, traditional grocery shopping is extremely time-consuming.
- Carrying grocery bags is a task because making a second round is a terrible idea.
- Also considering the pandemic there are new rules to be followed even in supermarkets.

Online Grocery Shopping

- The disadvantage of online grocery shopping is that people personally cannot examine the product. Fruits and vegetables need to be examined and there is no certain assurance of how fresh they are in online grocery shopping.
- Delivery charges are usually based on your location and because they are delivered to your house can make it slightly expensive.

3. Proposed System

Quick Reorder or Repeat Order

Grocery is meant to be the same each month, so instead of having a telephonic conversation where the customer would be recalling every product out of several products each time, it would be better if the shopkeeper is already having your list that can be updated according to the customer's need.

So, a website would help where there would be a list **(Purchase History)** based on the products purchased in the previous month.

Also, this system is useful post-Covid. Customers from the comfort of their homes, or anywhere can order things using that list in few clicks.

4. Advantages of Proposed System

- The local shopkeeper can take his business online to meet the customer's needs in the current situation.
- Easy to use for both the customer and the shopkeeper.
- Secure Admin Panel.
- Categories listings.
- Products listings.
- Product Variations. Because it is time consuming and impractical to add each product of different variations.
- Instead of adding each product again every month, there would be a list for the respective Customer that can be updated as well.
- Review and Rating System.

5.Feasibility Study

Economic Feasibility:

This project doesn’t cost much for development as all the resources needed are open sources.

Technical Feasibility:

The project is technically feasible as there isn’t any difficulty to get the required resources. All the technology used is simple. The devices used to develop the project don’t need to be of high specifications.

Operational Feasibility:

Anyone with the basic knowledge and understanding of how to create an account and carry out the checkout process can use this system.

Environment Feasibility:

The system is environmentally feasible.

6.Stakeholders

A stakeholder is an entity that has an interest in an organization and whose support is required for an organization to be successful.

So here the customer is a stakeholder and plays a vital role and a motivation to complete this project.

1)Admin:

A person to develop the website and its services and the one who adds or updates any new features and the one who's responsible for authentication of users.

2)Shopkeeper: A person who sells goods to the public.

- Categories Add/Update/Delete/Active/Inactive
- Product Add/Update/Delete/Active/Inactive
- Product Variations
- Orders
- Payments
- User listings

3)Customer:

Everyone can use this website who are interested in buying groceries online.

7.Project Requirements

Hardware and Software Requirements

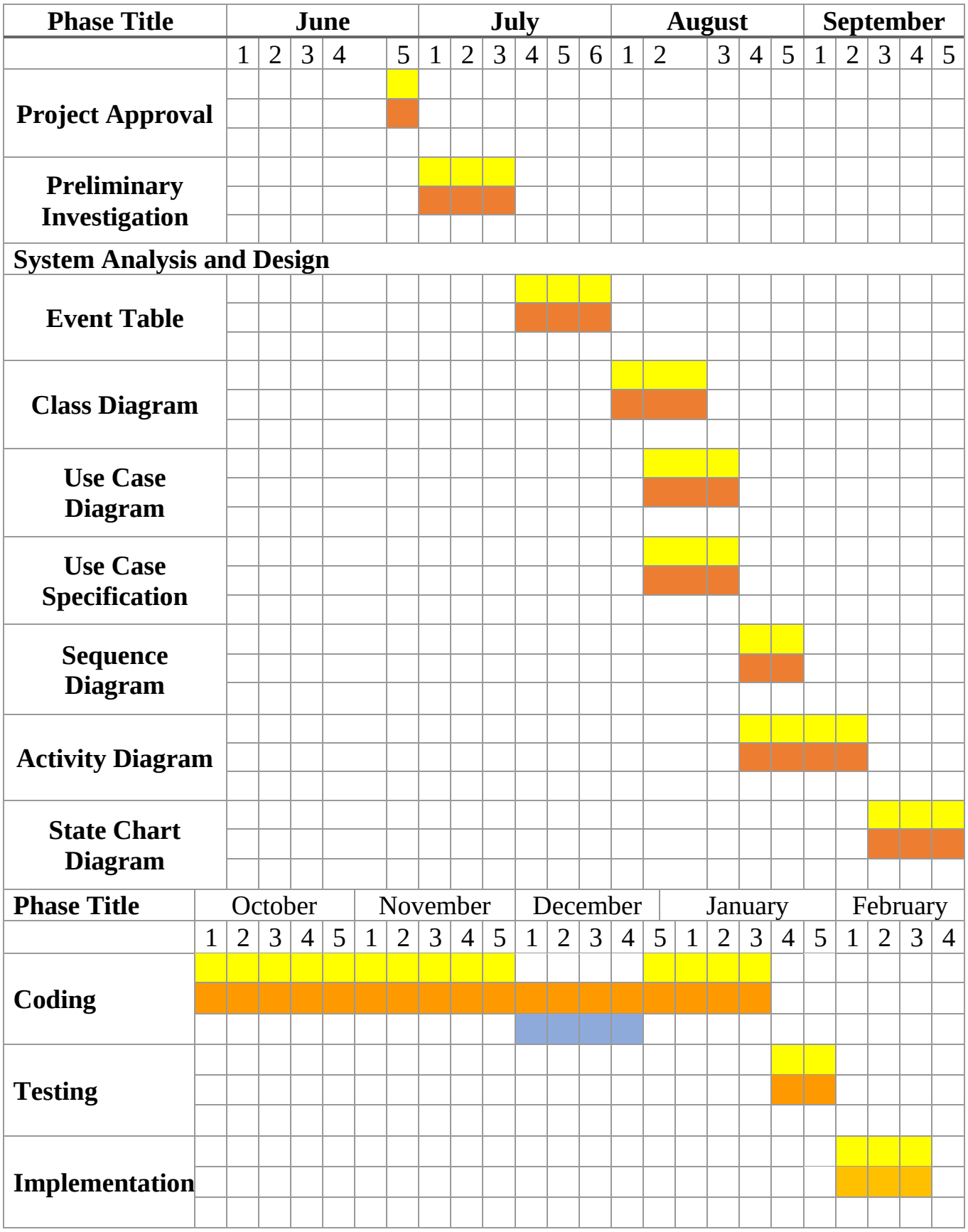
- Operating System: Windows 10 64-bit operating system, x64-based processor
- RAM: 4 GB RAM minimum,8 GB recommended
- Disk Space: Requires 2 GB of available disk space,4 GB recommended
- 1280 x 800 minimum screen resolution
- Processor: Intel i3, Intel i5 recommended
- Visual Studio IDE

8.Technologies Used

Html, CSS, JavaScript, Jquery, Bootstrap – Frontend development

Django (Python Framework), Sqlite3 – Backend development

Gantt Chart


Phase Title
June
July
August
September
1 2 3 4 5 1 2 3 4 5 6 1 2 3 4 5 1 2 3 4 5
Project Approval
Preliminary Investigation
System Analysis and Design
Event Table
Class Diagram
Use Case Diagram
Use Case Specification
Sequence Diagram
Activity Diagram
State Chart Diagram
Phase Title
October
November
December
January
February
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4
Coding
Testing
Implementation

Phase Title	March					
	1	2	3	4	5	
Report Submission						

	Actual Time of Completion		Expected Time of Completion		Break

System Analysis

Project Development Methodology

Agile is a software development methodology to build a software incrementally using short iterations of 1 to 4 weeks so that the development process is aligned with the changing business needs. Instead of a single-pass development of 6 to18 months where all the requirements and risks are predicted upfront, Agile adopts a process of frequent feedback where a workable product is delivered after one-to-four-week iteration. Agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s).

Event Table

1.User

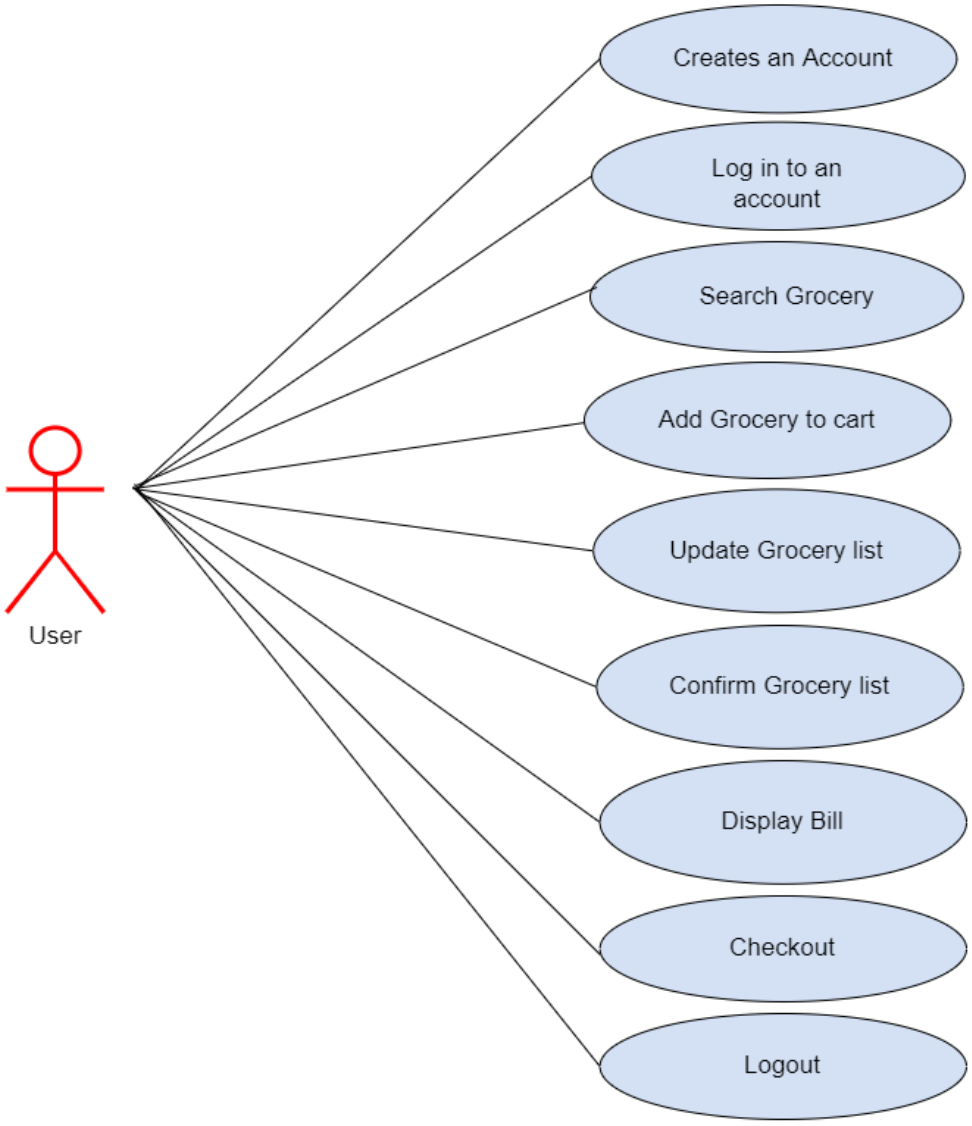
Sr. No.	Event	Trigger	Source	Activity	Response	Destination
1.	User creates an account	Requests to Sign Up	User	Enter details for Registration	Registered	User
2.	User Login	Requests to Log In	User	Verify Authentication	Logged In	User
3.	User searches for the desired product	Request to view the products	User	View the Products	Found the desired products	User
4.	User adds the product to cart	Product added to the cart	User	Clicks on Add to Cart Button	Increase in the No. of items inside the Cart	User
5.	User wants to remove the product from the Cart	Product is removed from the Cart	User	Clicks on Remove Item	Decrease in the No. of items inside the Cart	User
6.	User views the Purchase History	Request to Reorder or Repeat Order	User	Clicks on Order Items	Order is placed. Total is displayed.	Admin
7.	User reaches the Checkout Page	Requests to access the Checkout page	User	Payment Done	Payment received by the admin	Admin
8.	User wants to Logout	Requests to Logout	User	Logout	Logout	User

2.Admin/Retailer

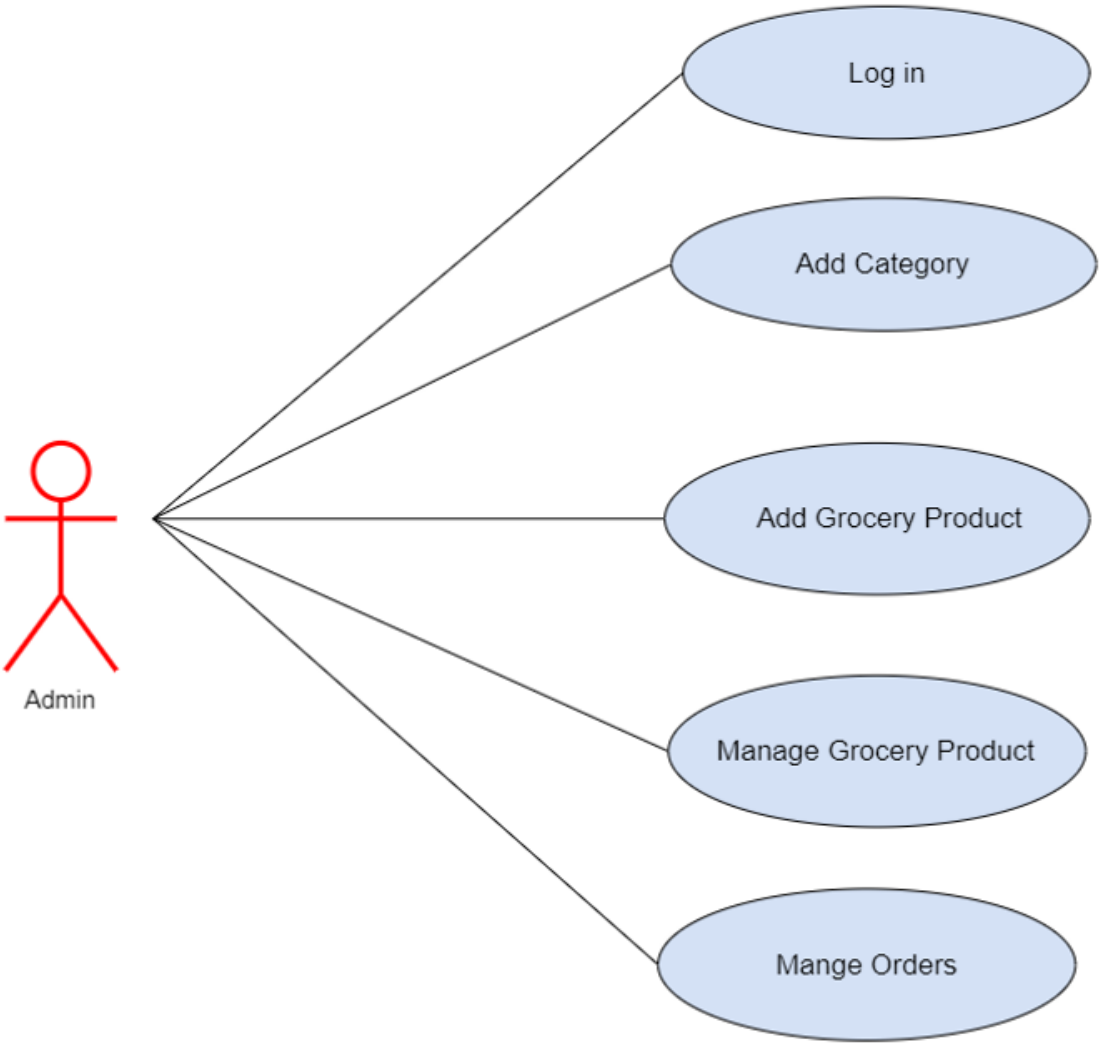
Sr. No.	Event	Trigger	Source	Activity	Response	Destination
1.	Retailer login	Request to Log In	Admin	Verify Authentication	Logged In	Admin
2.	Retailer adds product	Request to add product	Admin	Add products	Products saved	Admin
3.	Retailer deletes product	Request to delete product	Admin	Delete products	Products deleted	Admin
4.	Retailer update product details	Request to update product	Admin	Update Product details	Product details are updated	Admin
5.	Retailer views Orders	Request to view Orders	Admin	View Order details	Order details are viewed	Admin
6.	Retailer view User Listings	Request to view User Listings	Admin	View Users	User details viewed	Admin

Use Case Diagram:

1.User



2.Admin


Log in
Add Category
Add Grocery Product
Manage Grocery Product
Mange Orders
Admin

User Case Scenario:

1. Use Case Specification: User wants to Sign Up

This use case describes Sign Up of their account.

Invoking Actor: User

Flow of Events:

Basic Flow: The use case begins with opening the Sign-Up form. User must enter the username & password.

Pre-Condition:

Username & Password is required to Sign Up

Post-Condition:

Account is successfully created

2. Use Case Specification: User wants to Login

This use case describes login of their account

Invoking Actor: User

Flow of Events:

Basic Flow: The use case begins with opening the login form.

User must enter the Username & Password.

Pre-Condition:

Username & Password is required to login.

Post-Condition:

Account is successfully logged in.

3. Use Case Specification: User wants to view Groceries

This use case describes searching for the groceries.

Invoking Actor: User

Flow of Events:

Basic Flow:

The use case begins with opening the groceries page.

User can view the groceries

Pre-Condition:

No pre-condition

Post-Condition:

Grocery Items are viewed

4. Use Case Specification: User wants to add Grocery items to the Cart

The use case describes adding items to the Cart

Invoking Actor: User

Flow of Events:

Basic Flow:

The use case begins with adding the items to the Cart

User can view the groceries

Pre-Condition:

User must be logged in

Post-Condition:

Grocery Items are added to the Cart

5.UseCase Specification: User wants to save Cart items to the list

The use case describes adding items to the List

Invoking Actor: User

Flow of Events:

Basic Flow:

The use case begins with adding the items to the List

User can view the groceries

Pre-Condition:

User must have some items inside the Cart

Post-Condition:

Grocery Items are saved to the List

6. Use Case Specification: User confirm the items added to the Cart

The use case describes displaying the Bill after the Conformation

Invoking Actor: User

The flow of Events:

Basic Flow:

The use-case begins with conformation from the User

Users can view the Bill

Pre-Condition:

User must have some items inside the Cart

Post-Condition:

The Bill is displayed

7. Use Case Specification: User wants to checkout

The use-case describes paying for the items purchased

Invoking Actor: User

The flow of Events:

Basic Flow:

The use-case begins with selecting the payment gateway

Users can now pay the Bill

Pre-Condition:

User must fill the required details

Post-Condition:

The Payment is done

8.Use Case Specification: User wants to logout

This use case describes logging out of account

Invoking Actor: User

Flow of Events:

After carrying out all the functionality, user can logout.

Pre-Condition:

User should logout only if they are done with buying Groceries

Post-Condition:

Logged Out Successfully

9.UseCase Specification: Admin wants to add Categories Listings

This use case describes adding the Categories

Invoking Actor: Admin

Flow of Events:

This use case describes the opening of Product Categories Page

Admin can add Categories

Pre-Condition:

Admin must be logged in

Post-Condition:

Categories are added

10. Use Case Specification: Admin wants to add Grocery Products

This use case describes adding the Grocery Products

Invoking Actor: Admin

Flow of Events:

This use case describes the opening of Product Page

Admin can add Products to the respective Categories

Pre-Condition:

Admin must be logged in

Post-Condition:

Products are added

11. Use Case Specification: Admin wants to Update /Add / Delete Grocery Products

This use case describes managing the Product Page

Invoking Actor: Admin

Flow of Events:

This use case describes the opening of Product Page

Admin can add/update/delete the Products

Pre-Condition:

Admin must be logged in

Post-Condition:

Products are updated

12.Use Case Specification: Admin wants to View the orders

This use case describes managing the Orders

Invoking Actor: Admin

Flow of Events:

This use case describes the opening of Orders Page

Admin can view the Orders on the Order Page

Pre-Condition:

Admin must be logged in

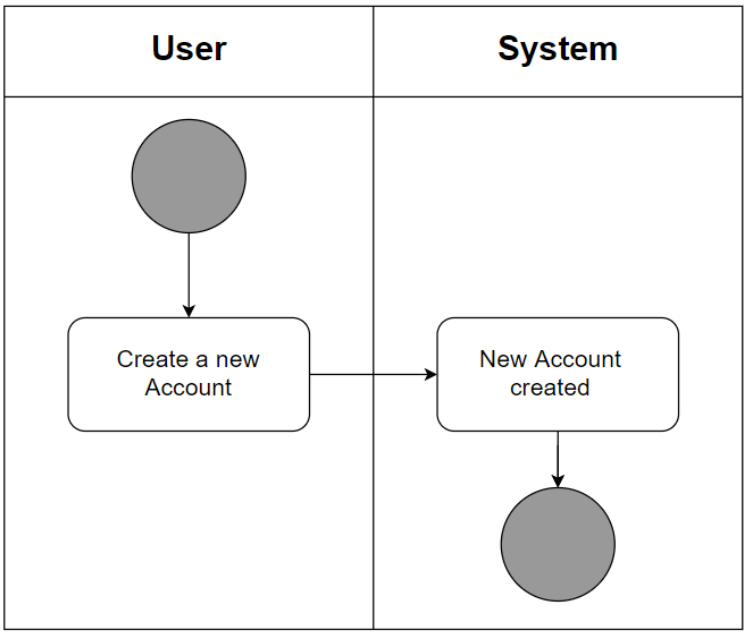
Post-Condition:

Orders are managed

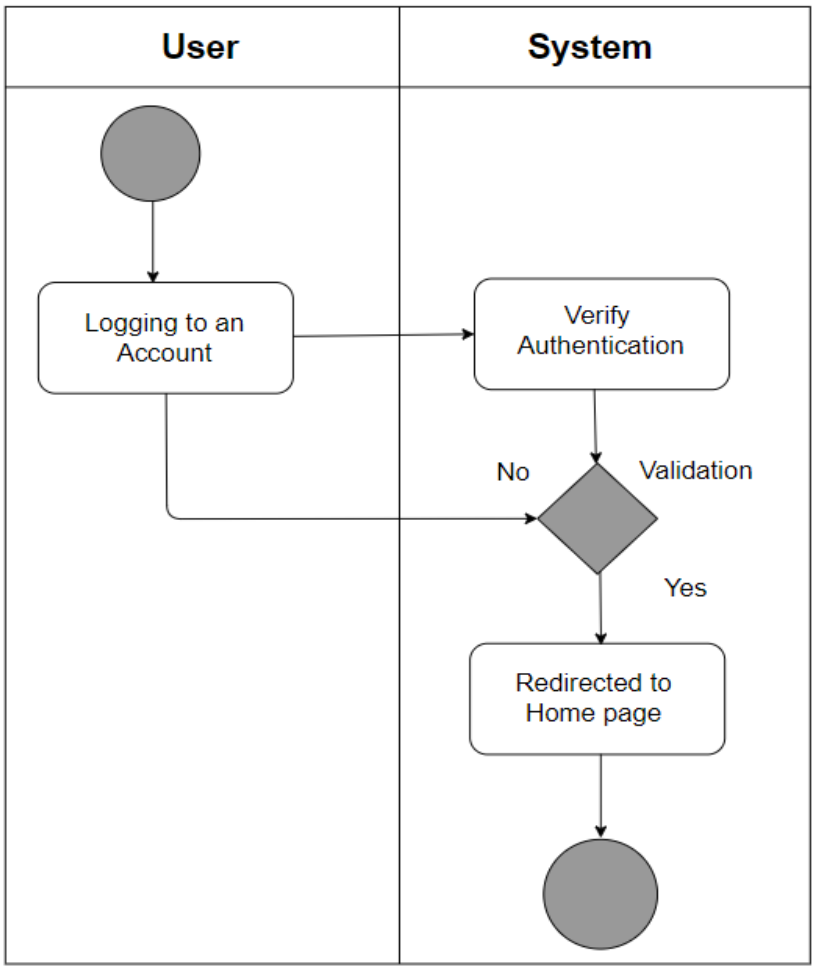
Activity Diagram

User

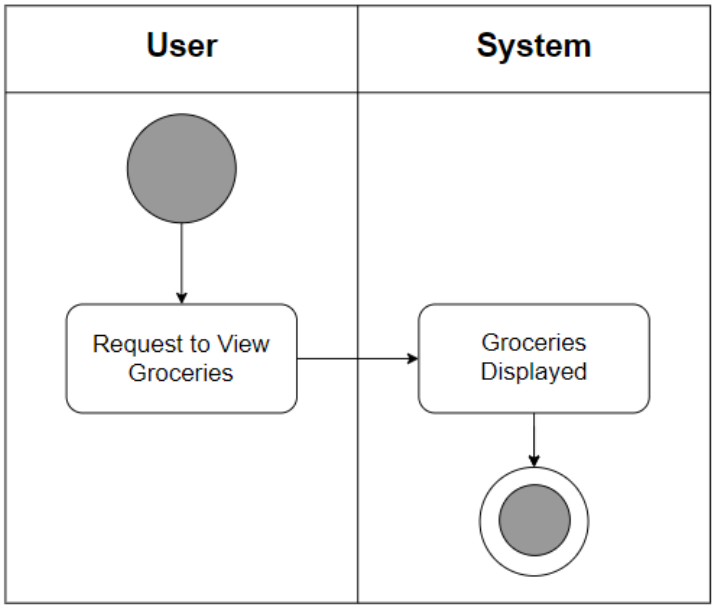
1.User Account Creation



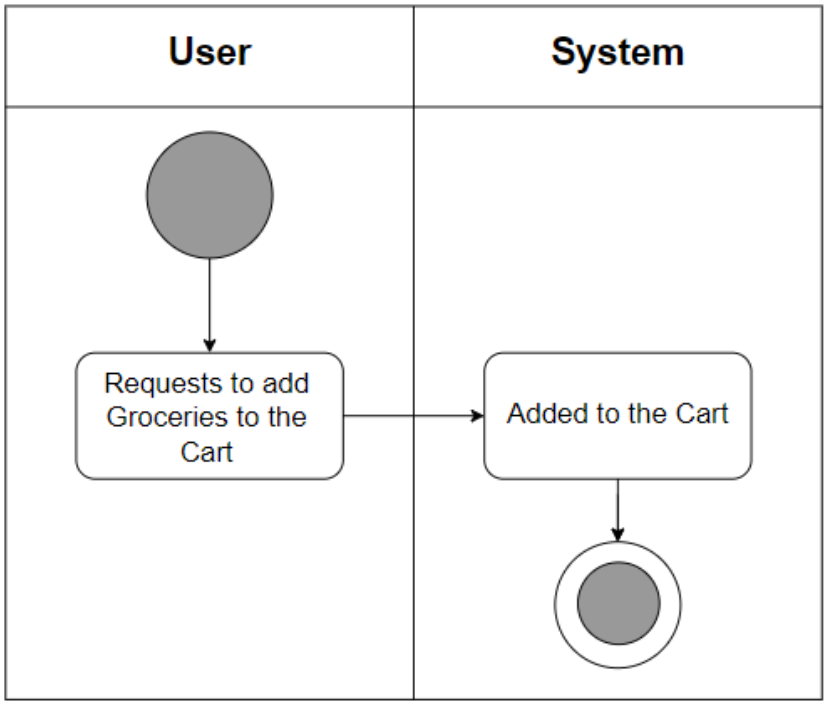
2.Verify Authentication



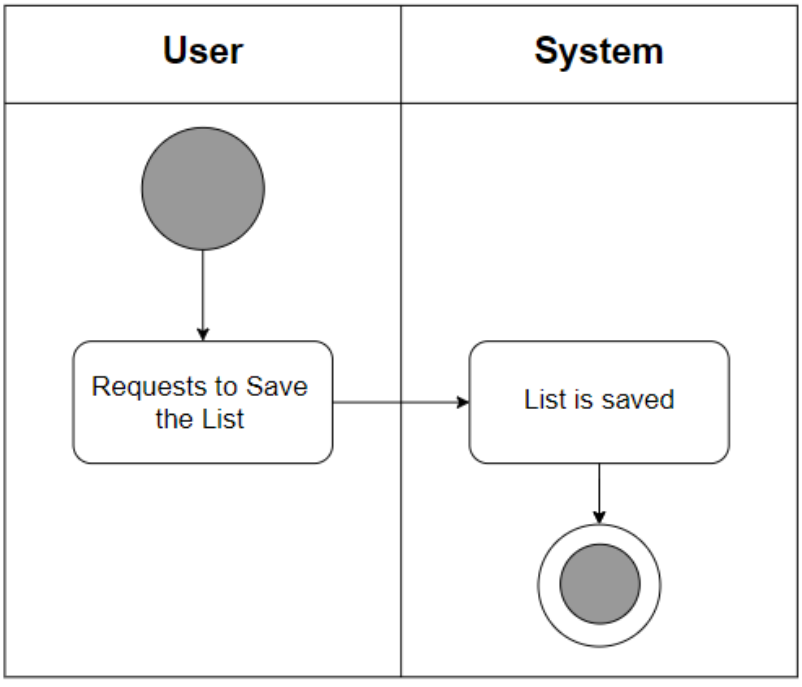
3.User Views Groceries



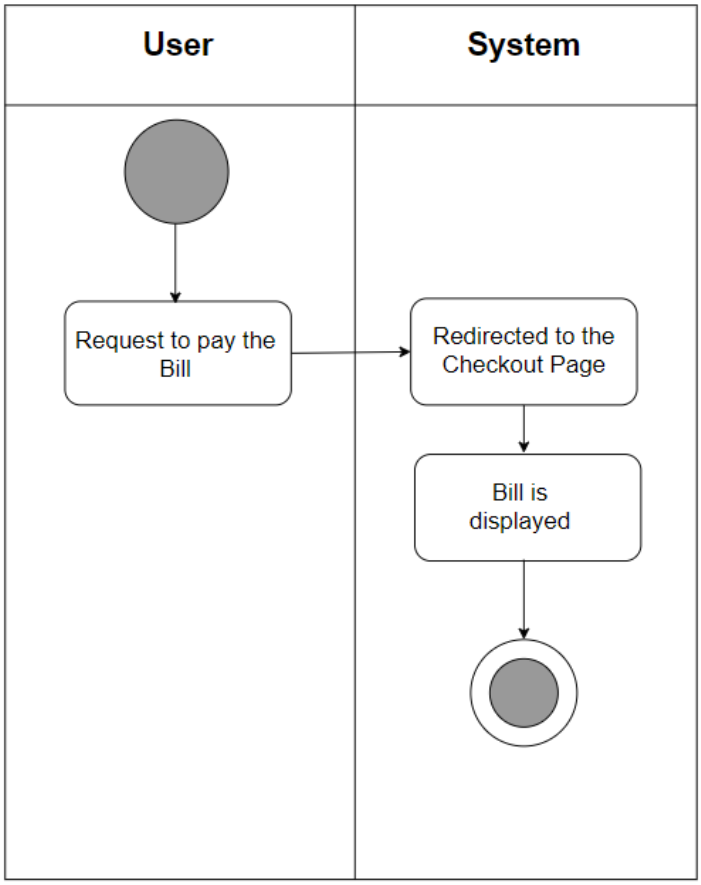
4.User Adds Groceries to Cart



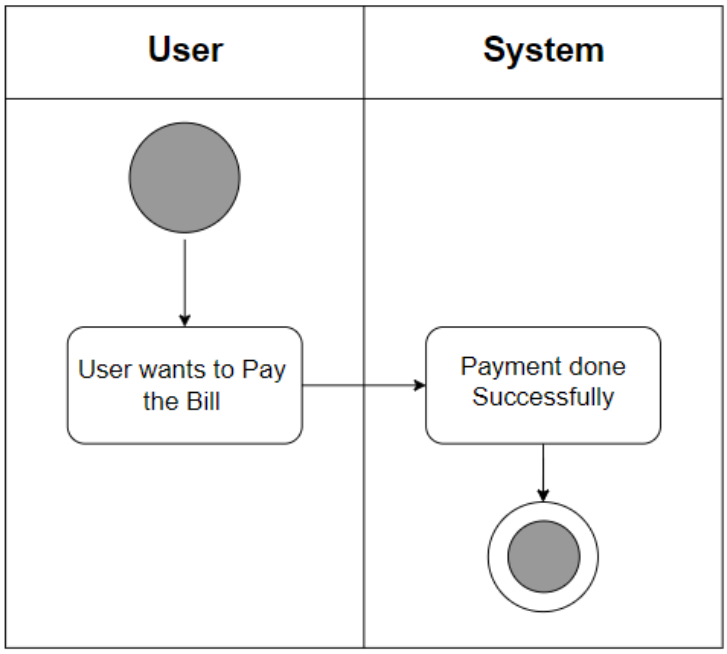
5.Saves the list for later Use



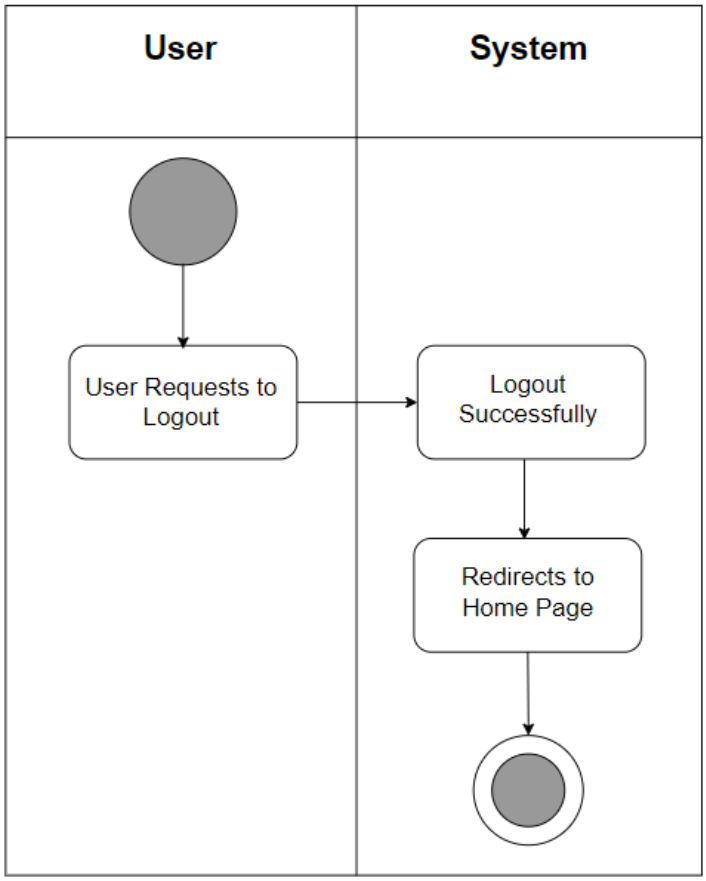
6. Checkout



7.User pays the Bill

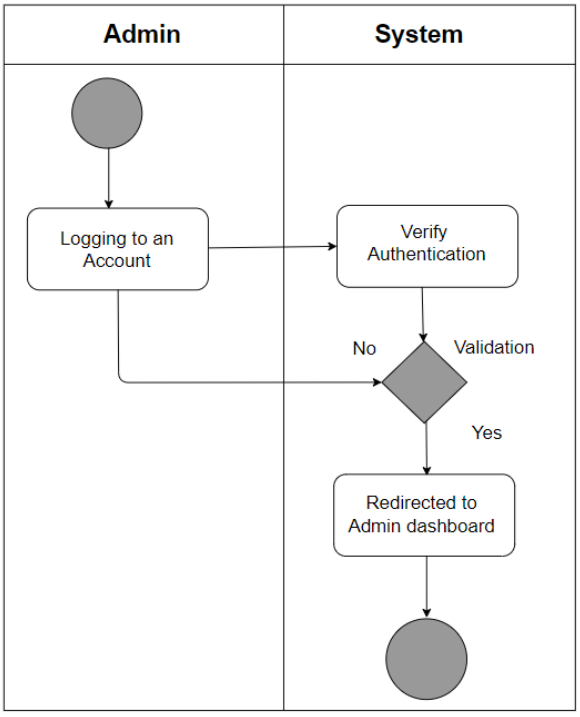


8.User wants to logout

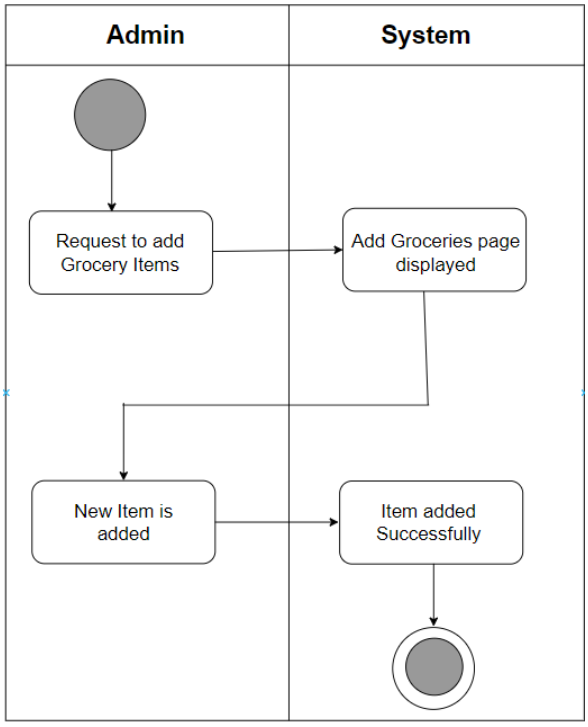


Admin

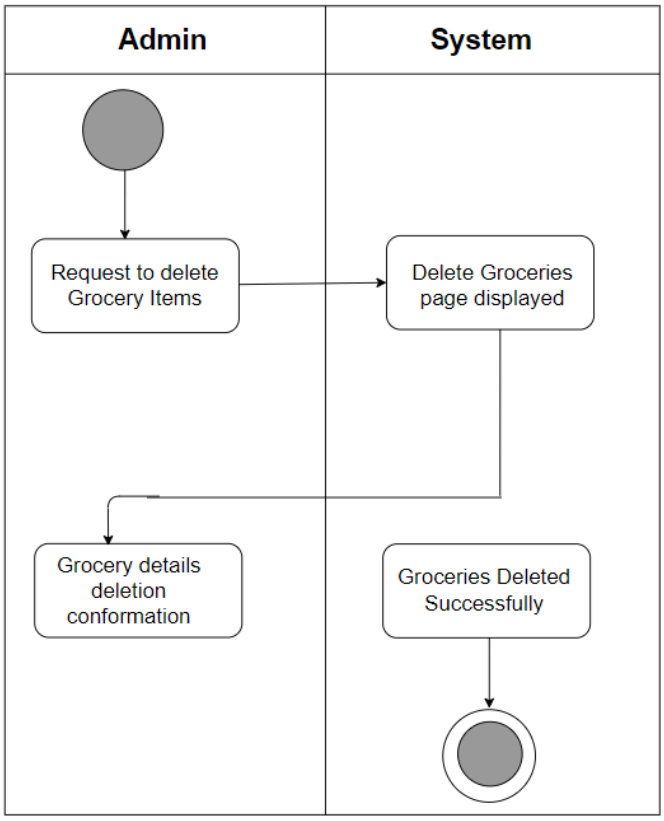
1. Admin wants to Login



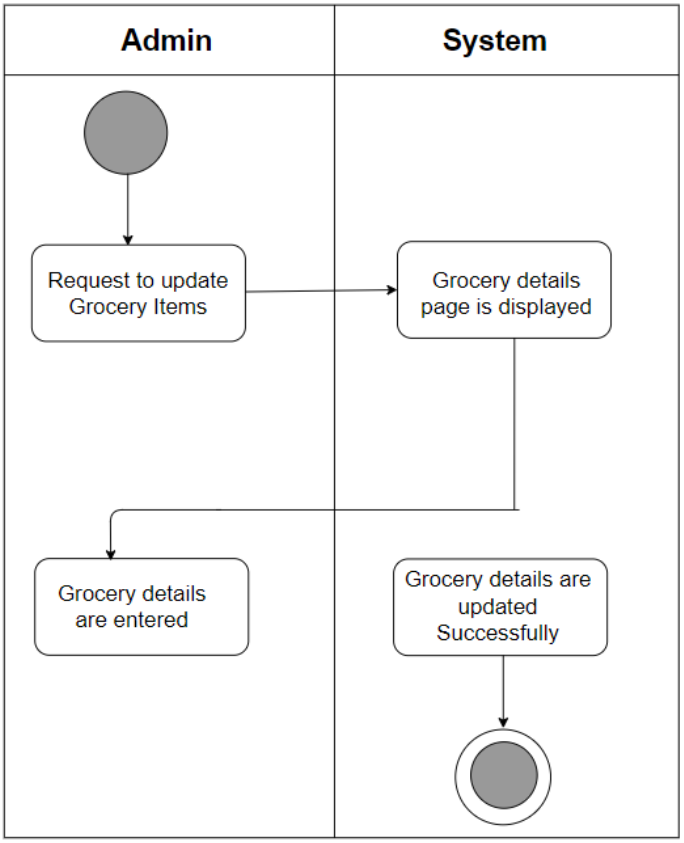
2.Admin wants to add Grocery Items



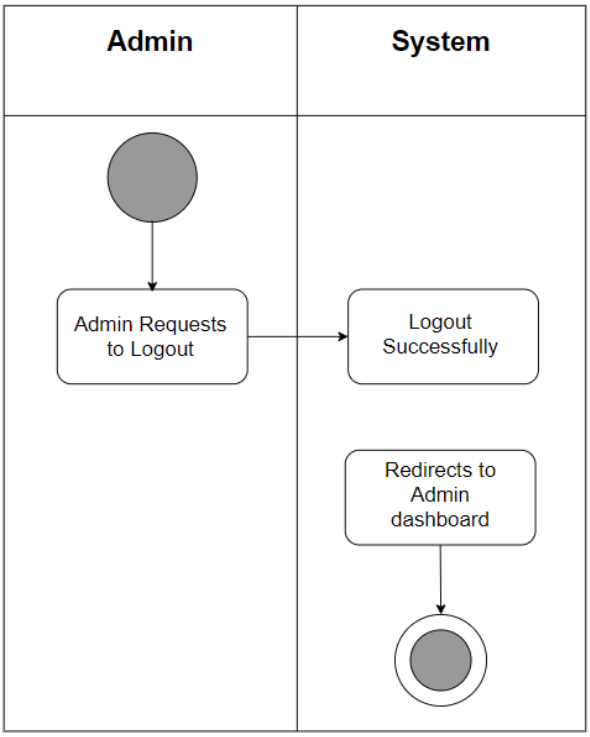
3.Admin wants to delete Grocery items



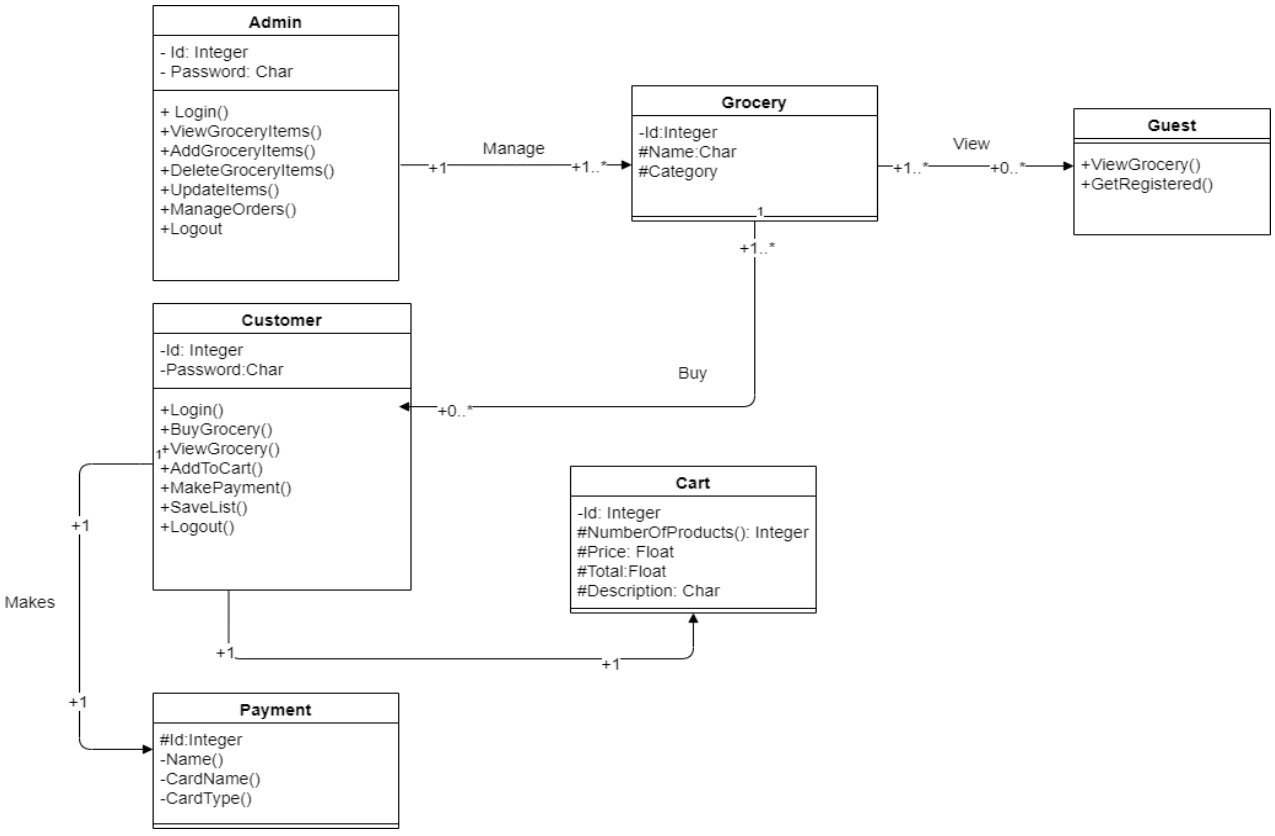
4.Admin wants to update Grocery items



5.Admin wants to logout

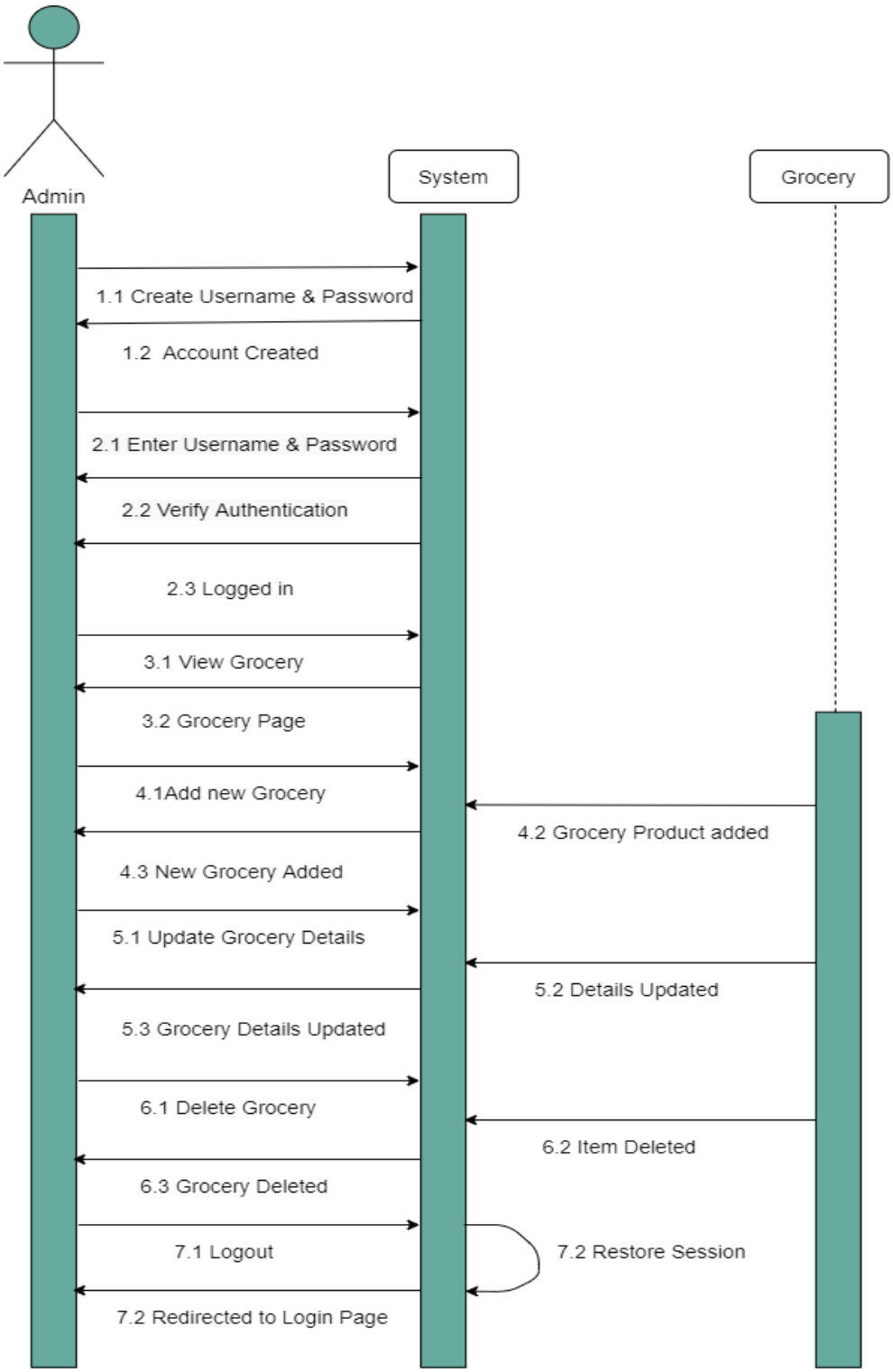


Class Diagram

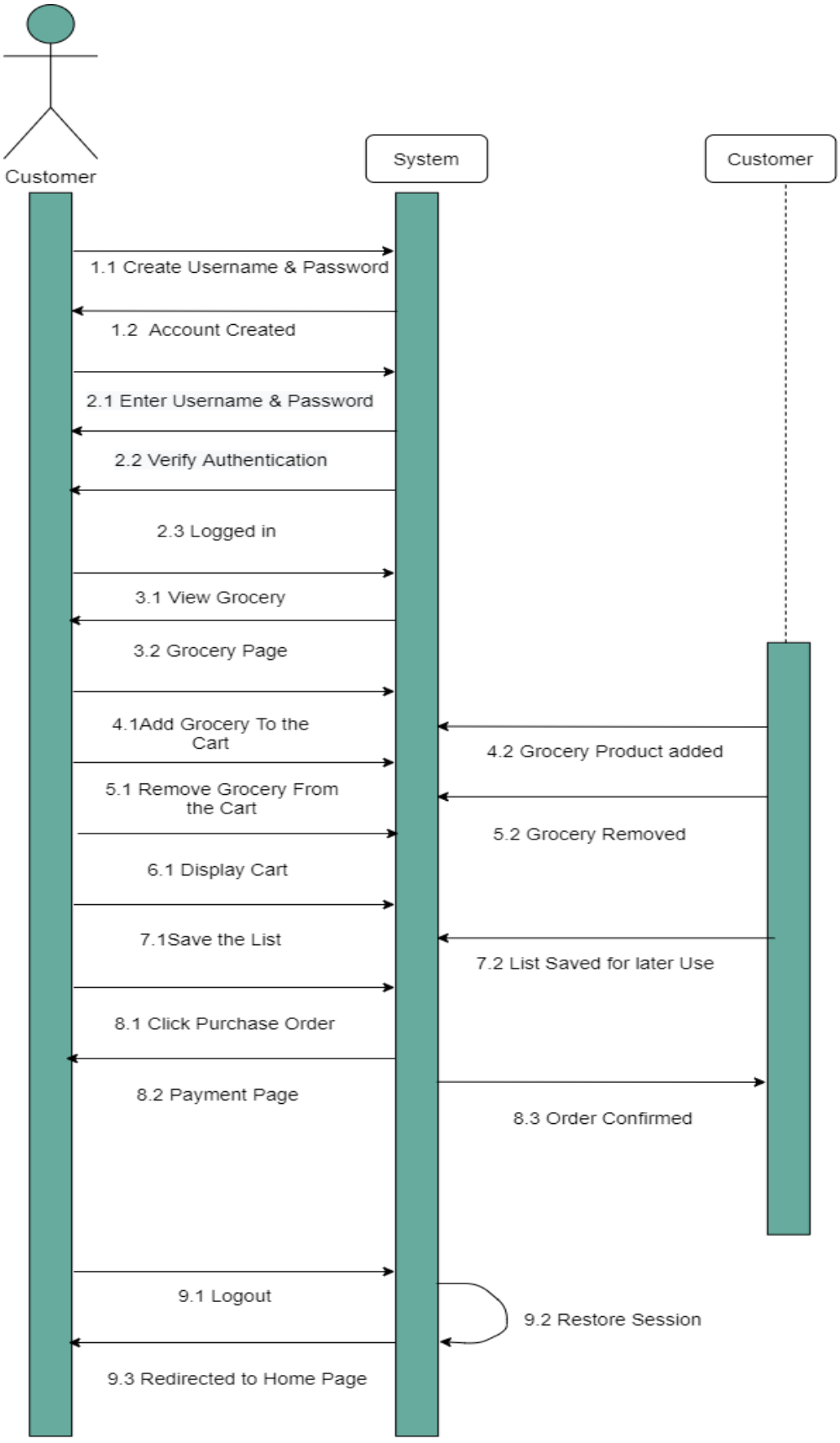


Sequence Diagram

Admin



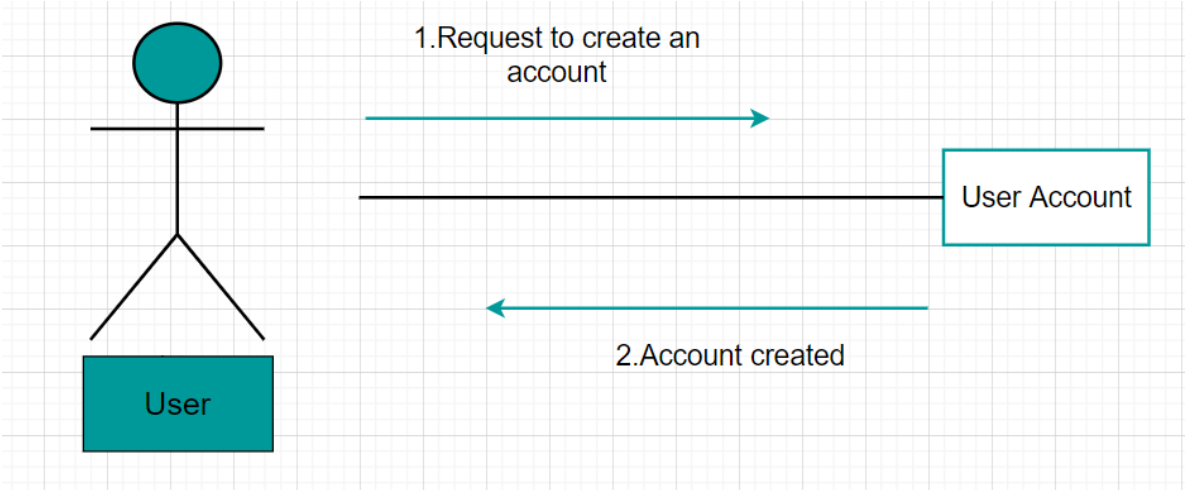
User


Customer
System
Customer
1.1 Create Username & Password
1.2 Account Created
2.1 Enter Username & Password
2.2 Verify Authentication
2.3 Logged in
3.1 View Grocery
3.2 Grocery Page
4.1Add Grocery To the Cart
4.2 Grocery Product added
5.1 Remove Grocery From the Cart
5.2 Grocery Removed
6.1 Display Cart
7.1Save the List
7.2 List Saved for later Use
8.1 Click Purchase Order
8.2 Payment Page
8.3 Order Confirmed
9.1 Logout
9.2 Restore Session
9.3 Redirected to Home Page

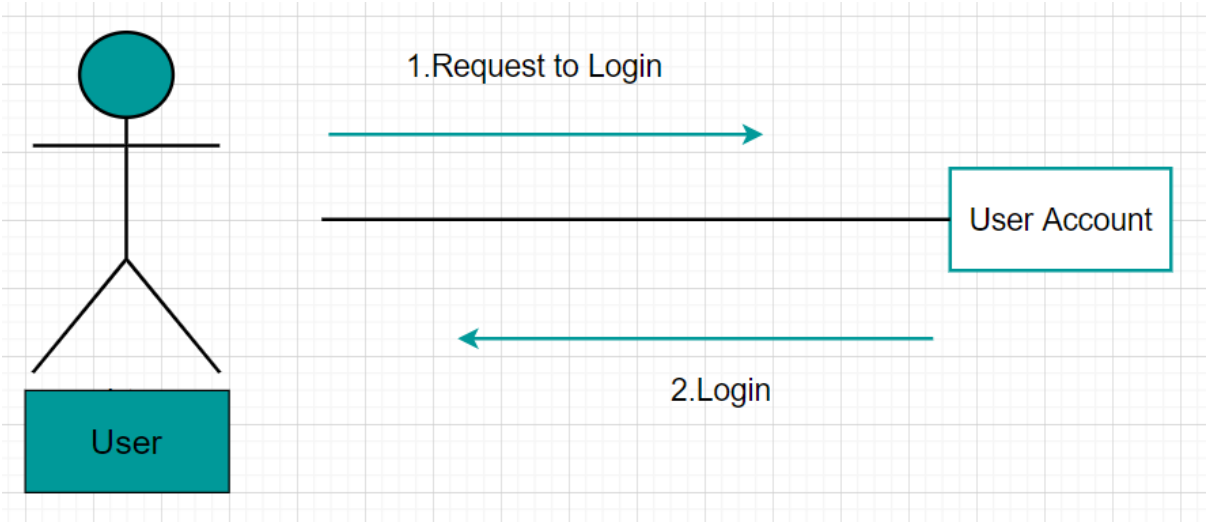
Collaboration Diagram

1. User

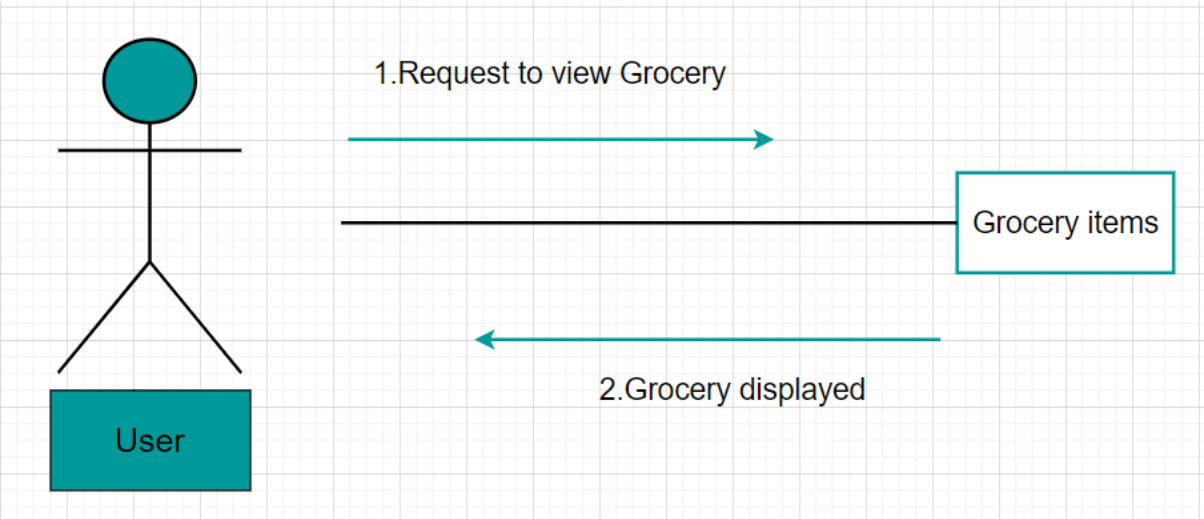
i. Create Account



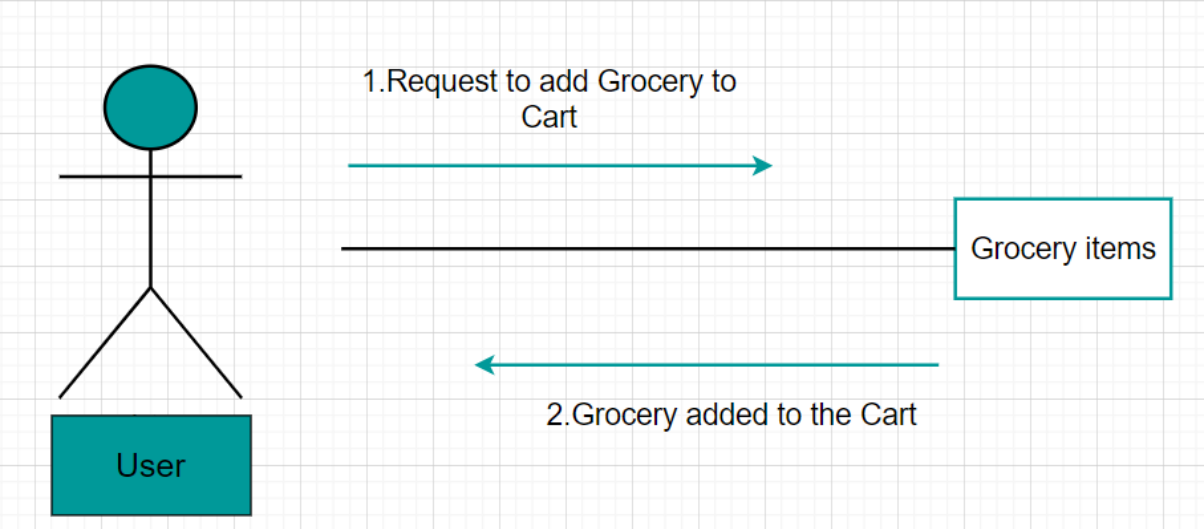
ii. Login



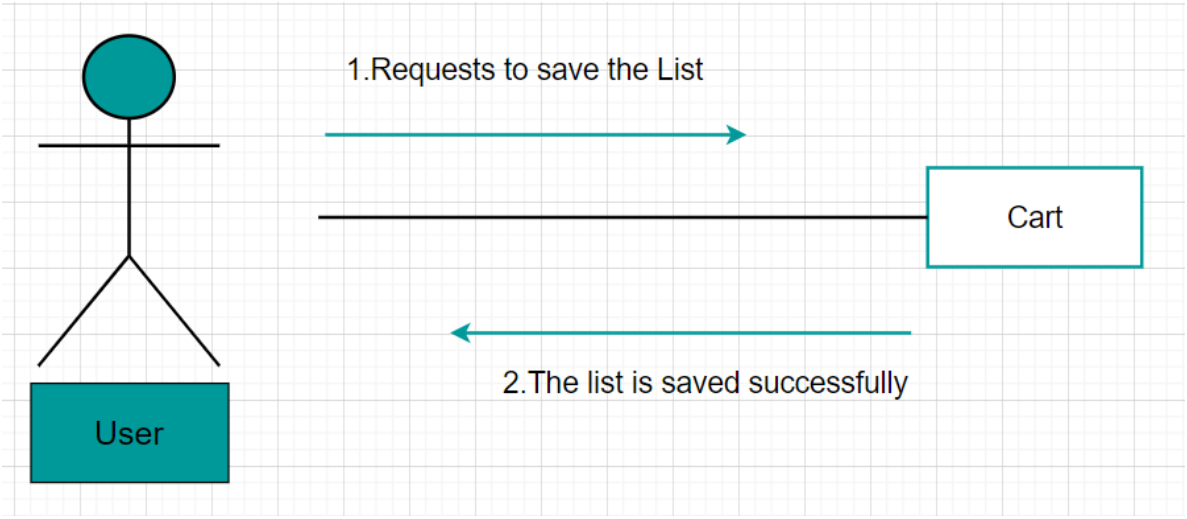
iii. View Grocery



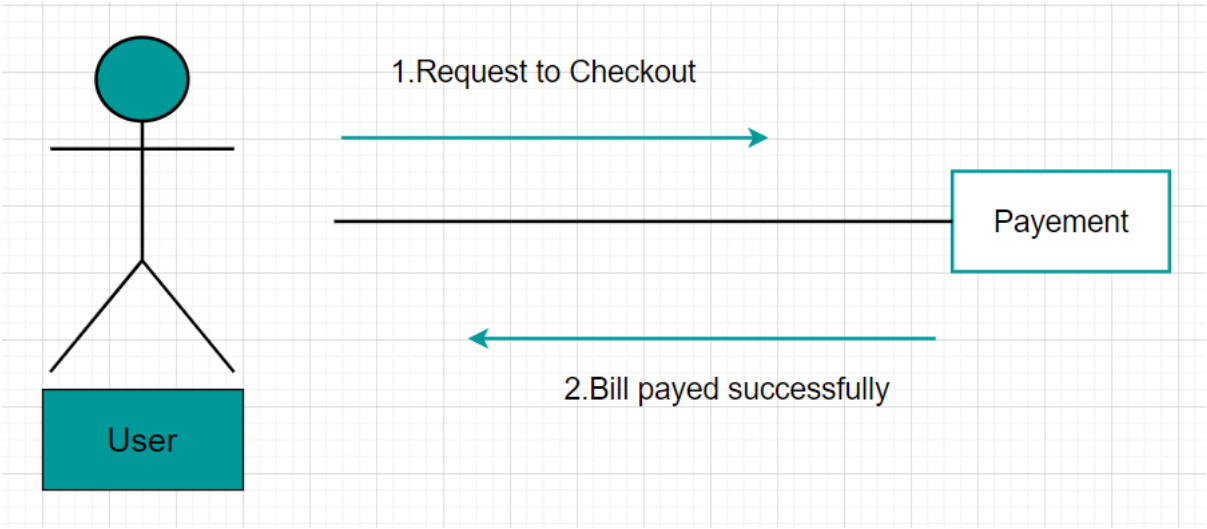
iv. Add Grocery Products



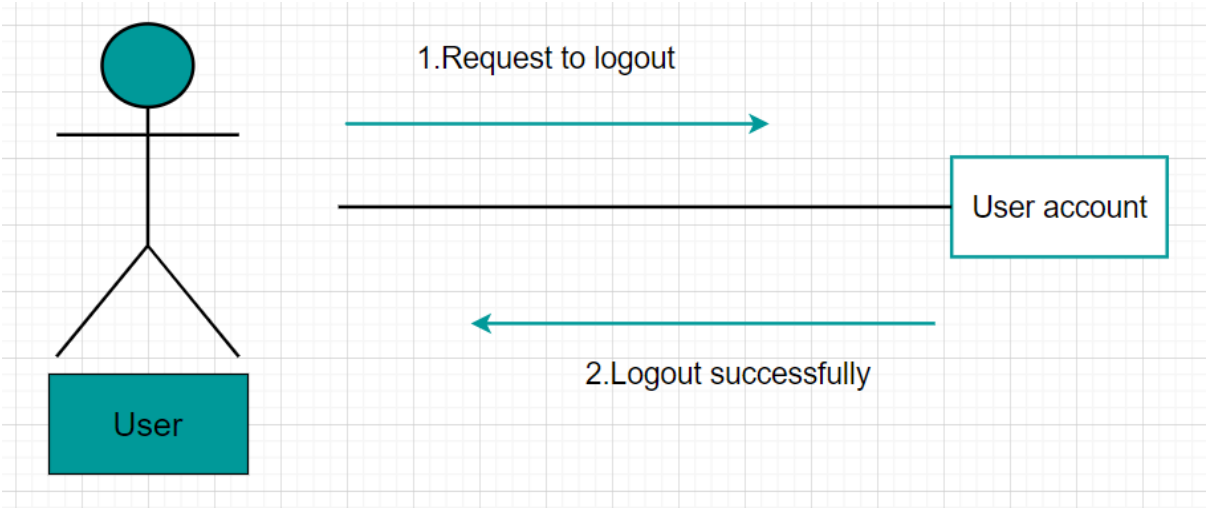
v. Requests to save the list



vi. Proceed to Checkout Page

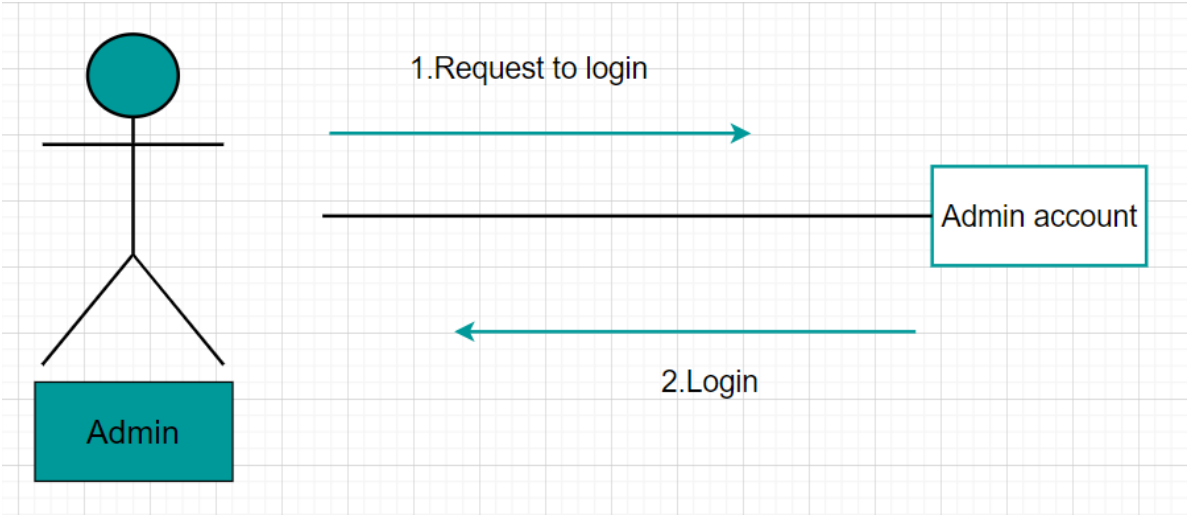


vii. Logout

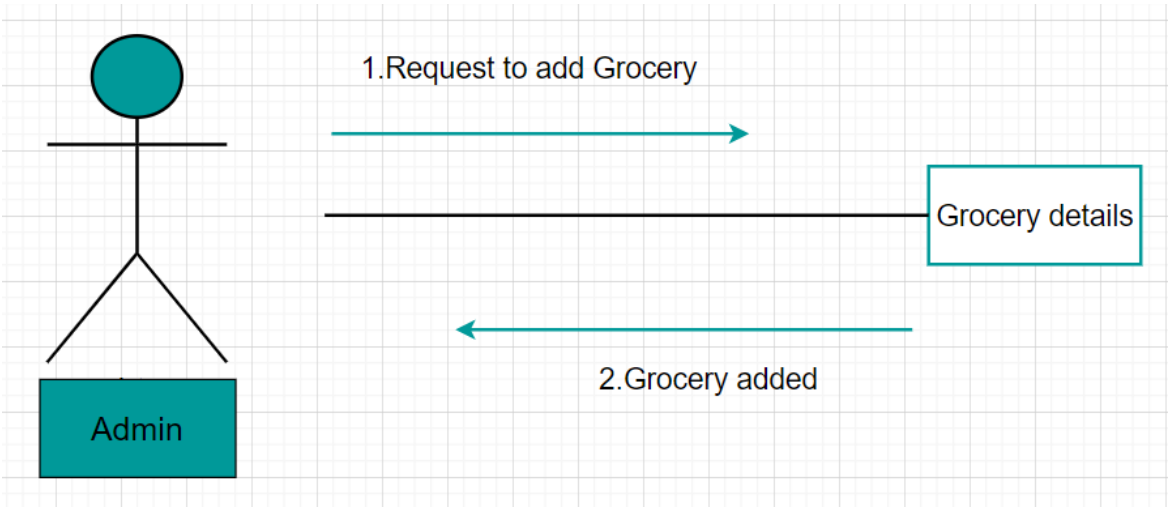


2. Admin

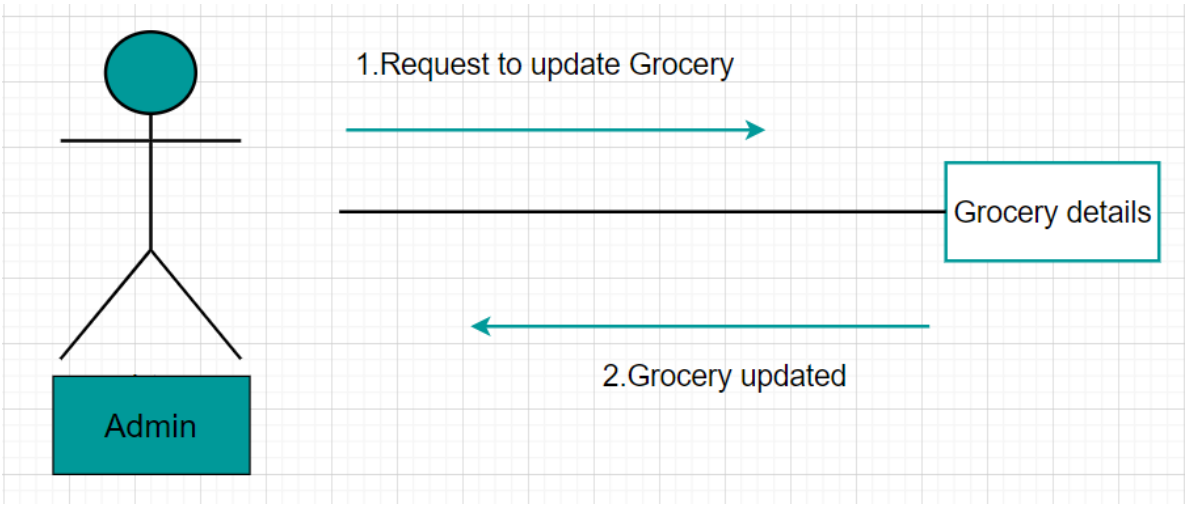
i. Admin Login



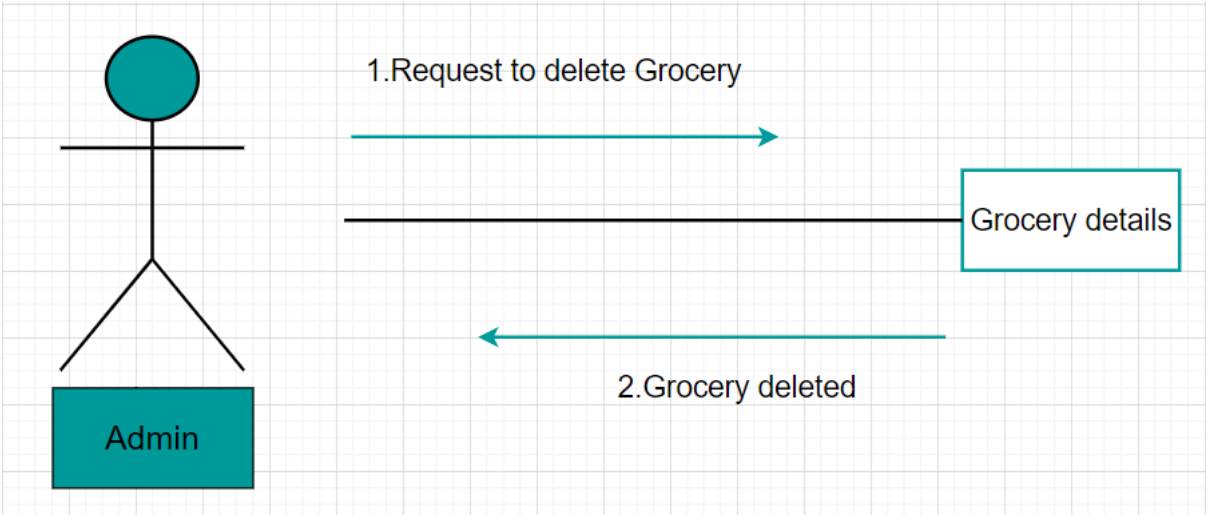
ii. Add Grocery



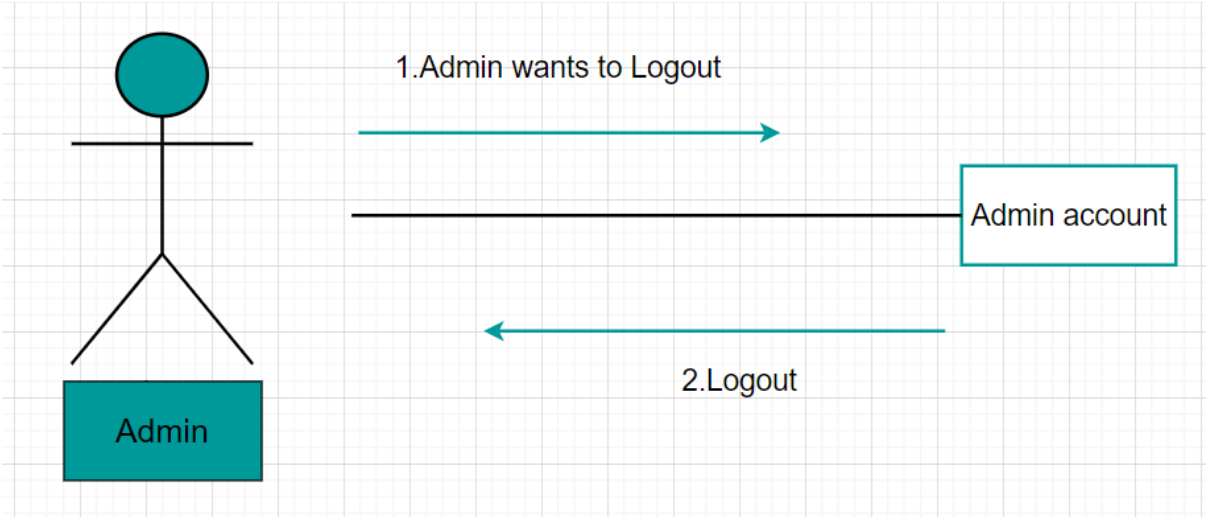
iii. Update Grocery



iv. Delete Grocery

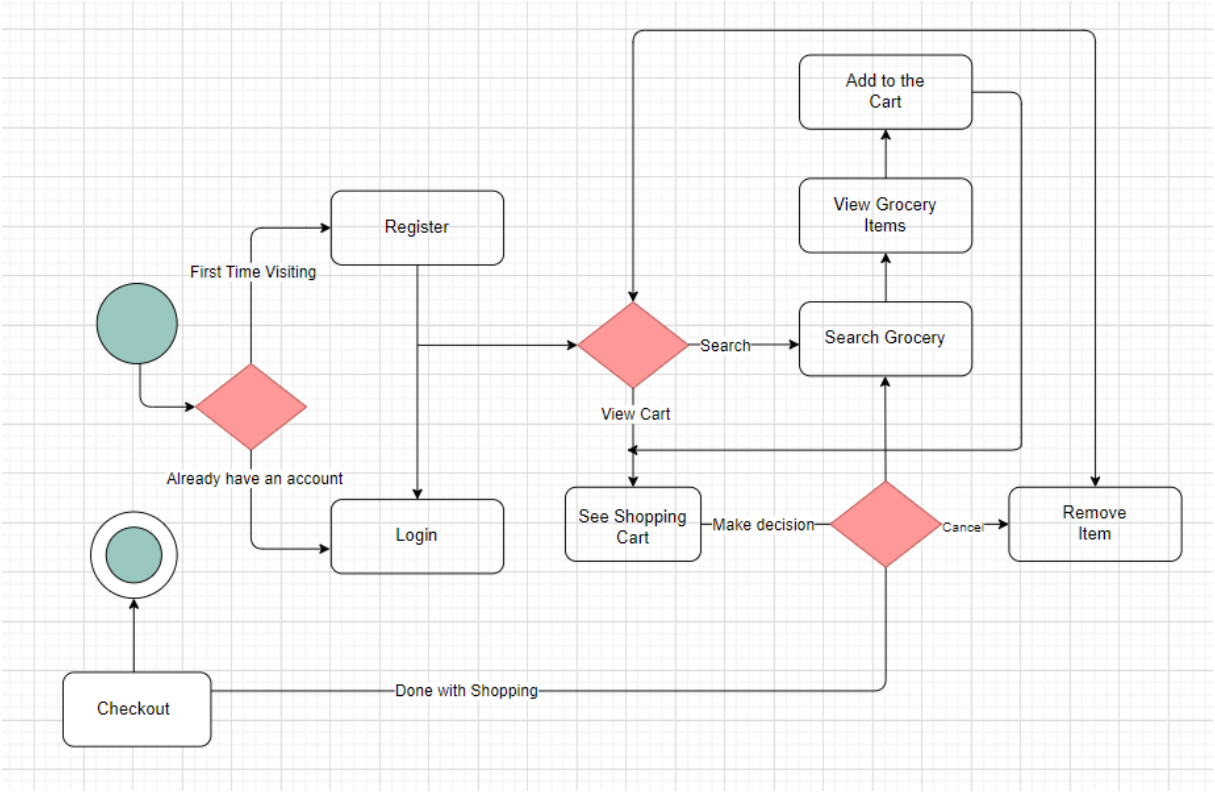


v. Logout

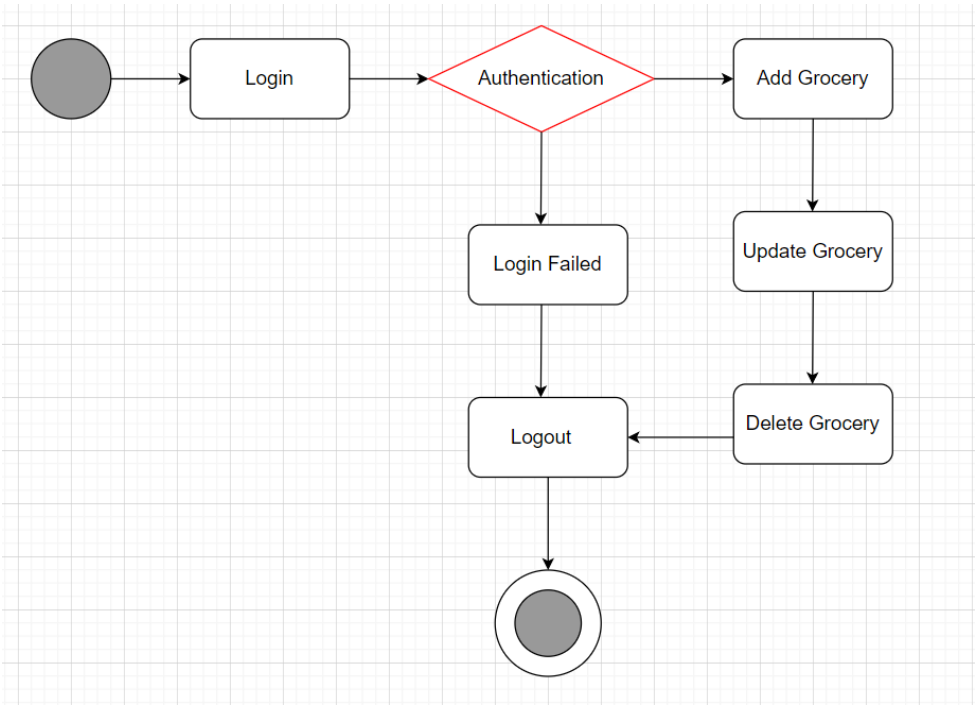


State Chart Diagram

1.User

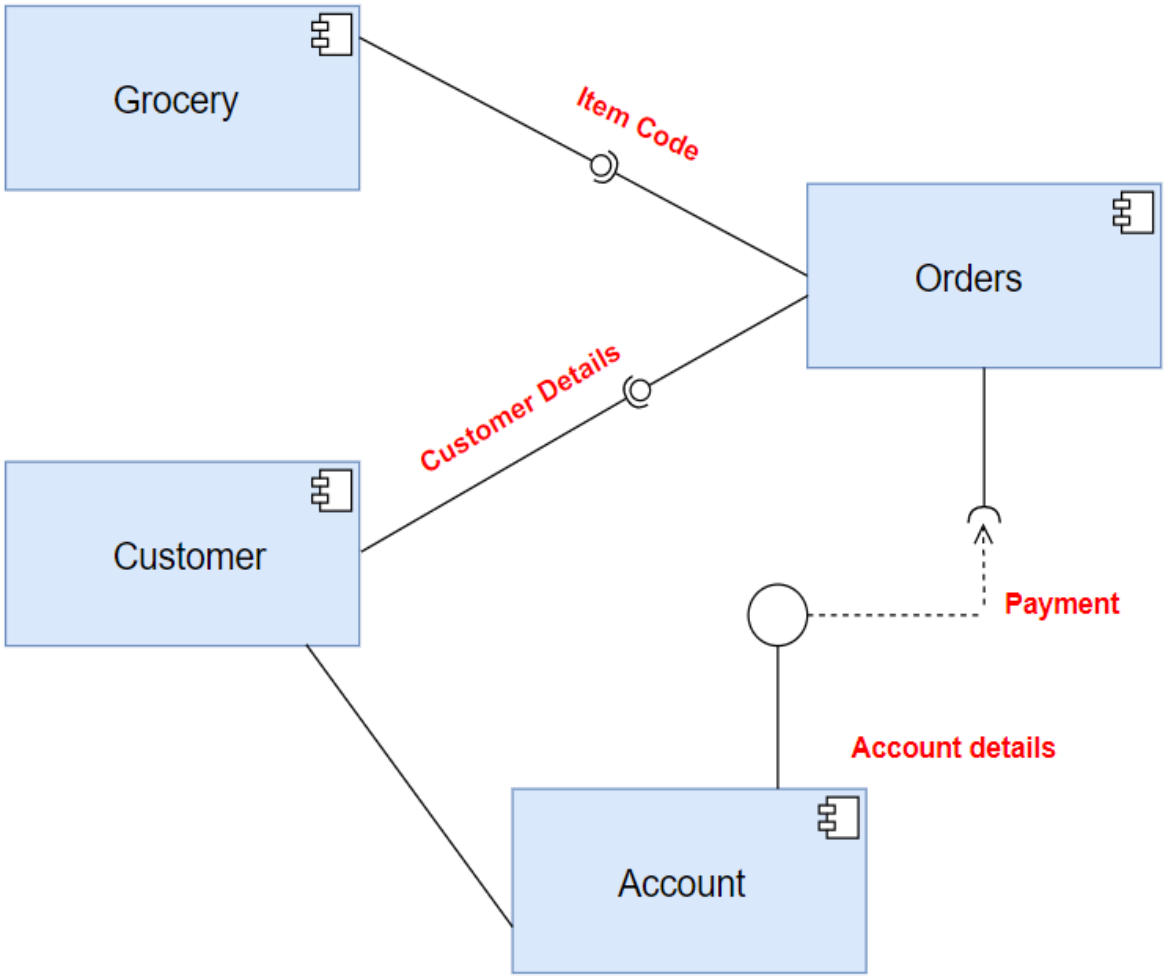


2.Admin

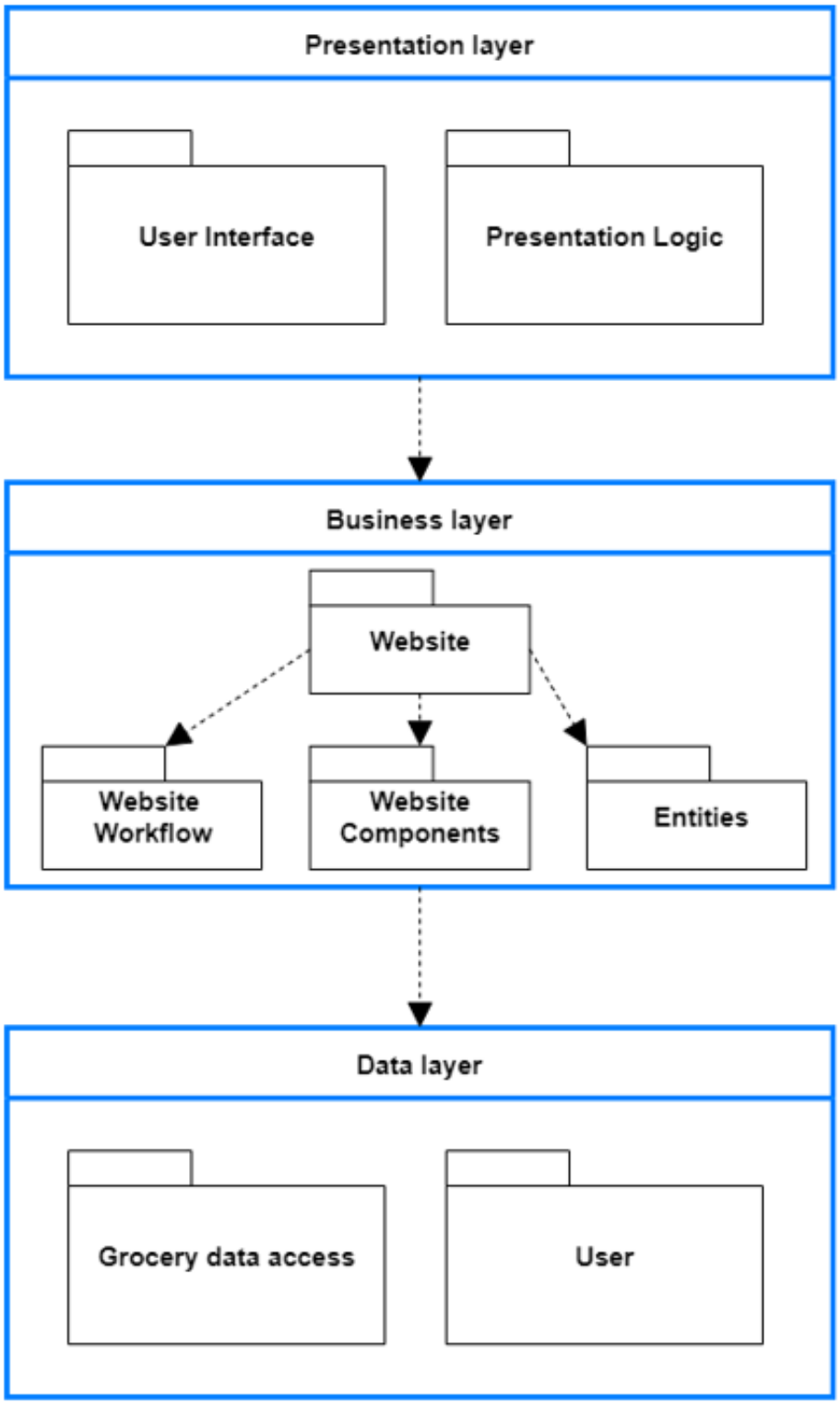


System Design

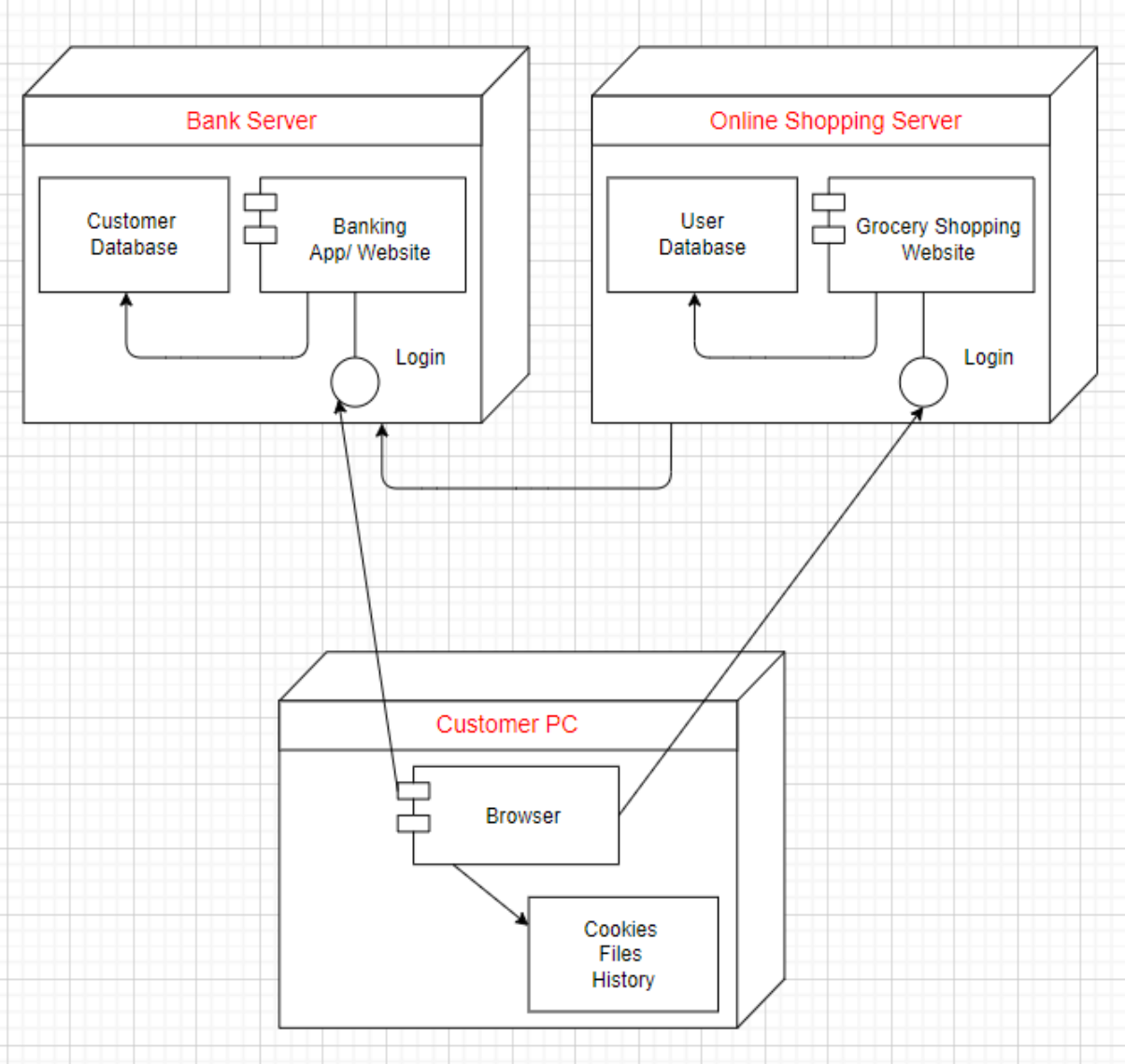
Component Diagram



Package Diagram



Deployment Diagram



List of Tables with Attributes and Constraints

Table Name	Field Name	Data Type	Description
Admin	Username	String	Not Null
	Password	String	Not Null

Table Name	Field Name	Data Type	Description
User	Id	Int	Not Null
	Username	String	Not Null
	Password	String	Not Null
	Details	String	Not Null

Table Name	Field Name	Data Type	Description
Categories	Id	Int	Not Null
	Title	String	Not Null

Table Name	Field Name	Data Type	Description
Grocery Products	Id	Int	Not Null
	Name	String	Not Null
	Price	Float	Not Null
	Quantity	Int	Not Null
	Description	String	Not Null
	Category Id	Int	Not Null

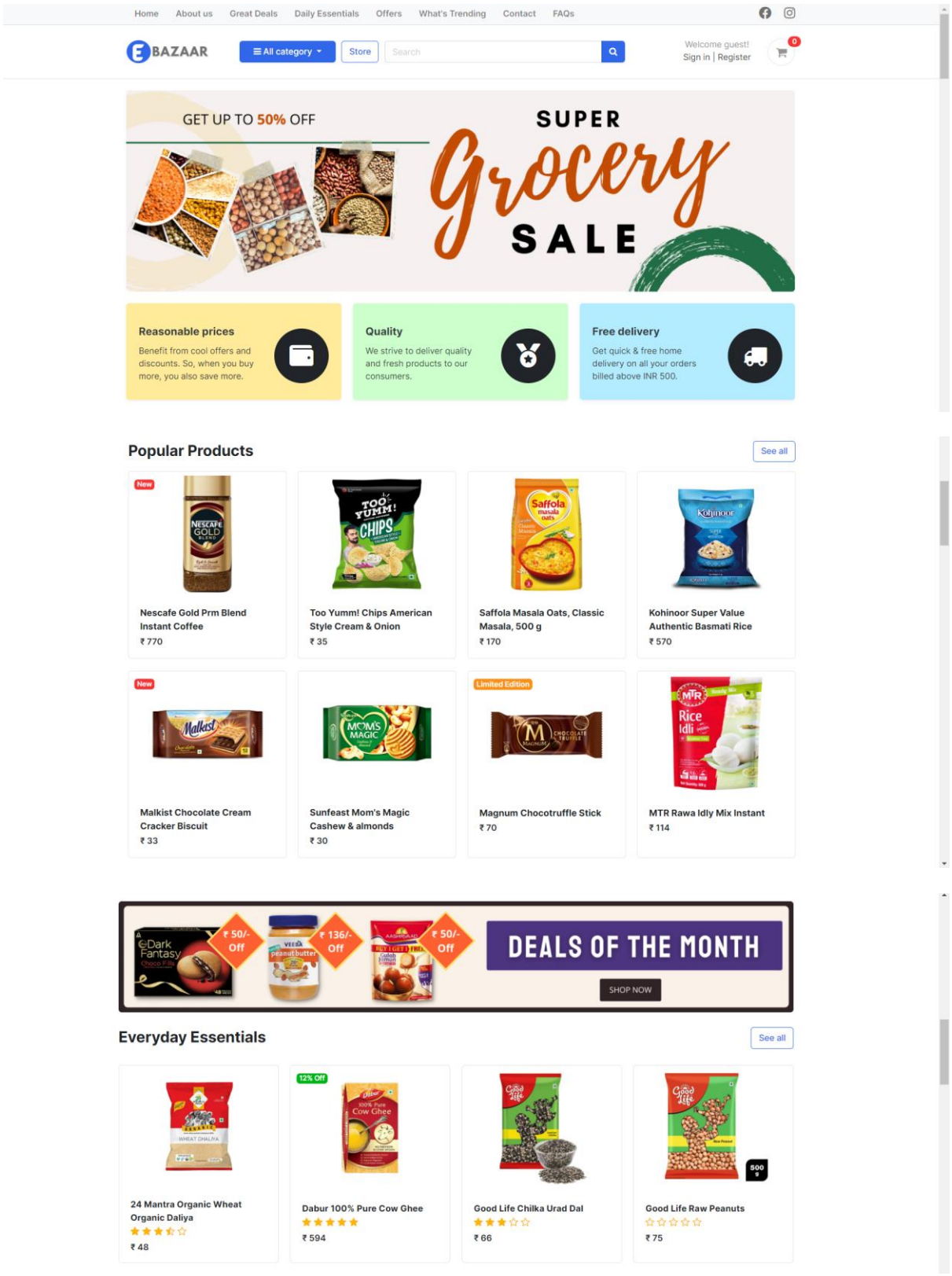
Table Name	Field Name	Data Type	Description
Order Details	Id	Int	Not Null
	Order Id	Int	Not Null
	Product Id	Int	Not Null
	Quantity	Int	Not Null
	Price	Float	Not Null

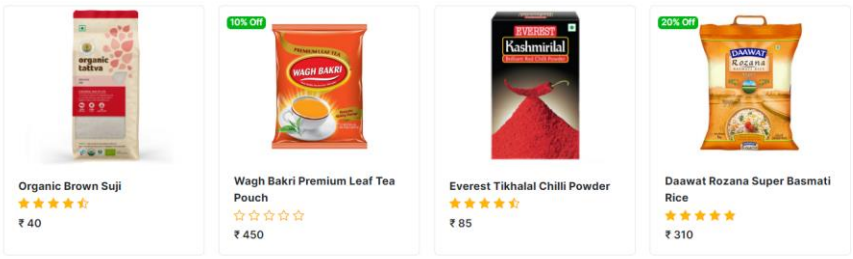
Table Name	Field Name	Data Type	Description
Order	Id	Int	Not Null
	User Id	String	Not Null

Table Name	Field Name	Data Type	Description
Cart	Id	Int	Not Null
	Checkout	Boolean	Not Null
	Quantity	Int	Not Null
	Status	String	Not Null

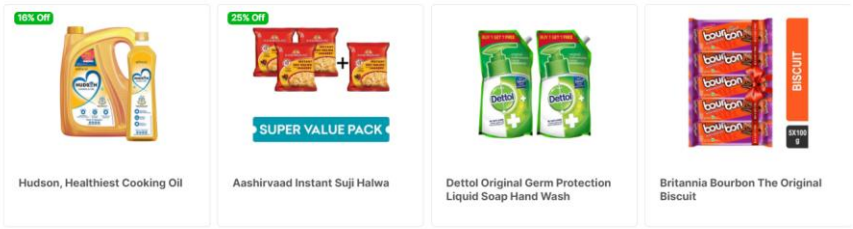
System Coding

Screen Layouts

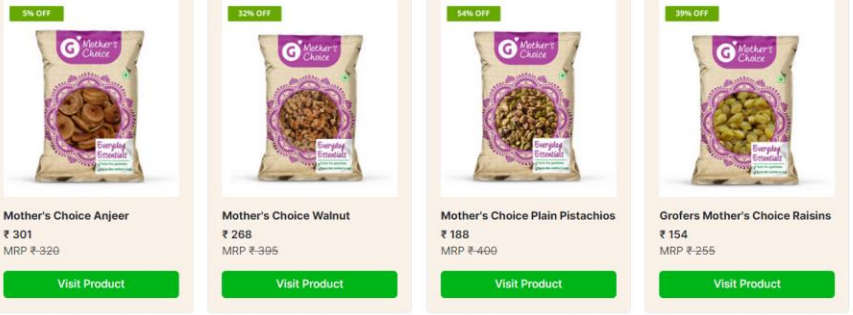




Great Deals on Products



Best Offers on Products



What's Trending


Britannia Cheese Slices


Cadbury Chocobakes Cake


Gadre Jus-Like - Lobster Bite


So Good Almond Milk

E-Bazaar -Online Grocery Store

Did you ever imagine that the freshest of fruits and vegetables, top quality pulses and food grains, dairy products and hundreds of branded items could be handpicked and delivered to your home, all at the click of a button? India's first comprehensive online megastore, E-Bazaar.com, brings a whopping 20000+ products with more than 1000 brands, to over 4 million happy customers. From household cleaning products to beauty and makeup, E-Bazzar has everything you need for your daily needs. E-Bazzar.com is convenience personified We've taken away all the stress associated with shopping for daily essentials, and you can now order all your household products and even buy groceries online without travelling long distances or standing in serpentine queues. Add to this the convenience of finding all your requirements at one single source, along with great savings.Online grocery shopping has never been easier. Need things fresh? Whether it's fruits and vegetables or dairy and meat, we have this covered as well!

Read More...

E-Bazaar -Online Grocery Store

Did you ever imagine that the freshest of fruits and vegetables, top quality pulses and food grains, dairy products and hundreds of branded items could be handpicked and delivered to your home, all at the click of a button? India's first comprehensive online megastore, E-Bazaar.com, brings a whopping 20000+ products with more than 1000 brands, to over 4 million happy customers. From household cleaning products to beauty and makeup, E-Bazzar has everything you need for your daily needs. E-Bazzar.com is convenience personified We've taken away all the stress associated with shopping for daily essentials, and you can now order all your household products and even buy groceries online without travelling long distances or standing in serpentine queues. Add to this the convenience of finding all your requirements at one single source, along with great savings.Online grocery shopping has never been easier. Need things fresh? Whether it's fruits and vegetables or dairy and meat, we have this covered as well!

Best quality products for our quality-conscious customers. When it comes to payment, we have made it easy for our customers can pay through multiple payment channels like Credit and Debit cards, Internet Banking, e-wallets and Sodexo passes or simply pay Cash on Delivery (COD). The convenience of shopping for home and daily needs, while not compromising on quality, has made E-Bazaar.com the online supermarket of choice for thousands of happy customers across India.

E-Bazaar.com believes in providing the highest level of customer service and is continuously innovating to meet customer expectations. Our On-time Guarantee is one such assurance where we refund 10% of the bill value if the delivery is delayed. For all your order values above Rs. 1000, we provide free delivery. A wide range of imported and gourmet products are available through our express delivery and slotted delivery service. If you ever find an item missing on delivery or want to return a product, you can report it to us within 48 hours for a 'no-questions-asked' refund.The convenience of shopping for home and daily needs, while not compromising on quality, has made E-Bazaar.com the online supermarket of choice for thousands of happy customers across India.E-Bazaar.com believes in providing the highest level of customer service and is continuously innovating to meet customer expectations.

Get Social With Us

Information
Terms & Conditions
FAQ
About Us

Customer Service
Contact Us
Returns
Sitemap

Payment Options

Home | About us | Great Deals | Daily Essentials | Offers | What's Trending | Contact | FAQs

EBAZAAR | All category | Store | Search | Welcome guest! Sign in | Register

Nescafe Gold Prm Blend Instant Coffee

4.5 ★ 1 reviews

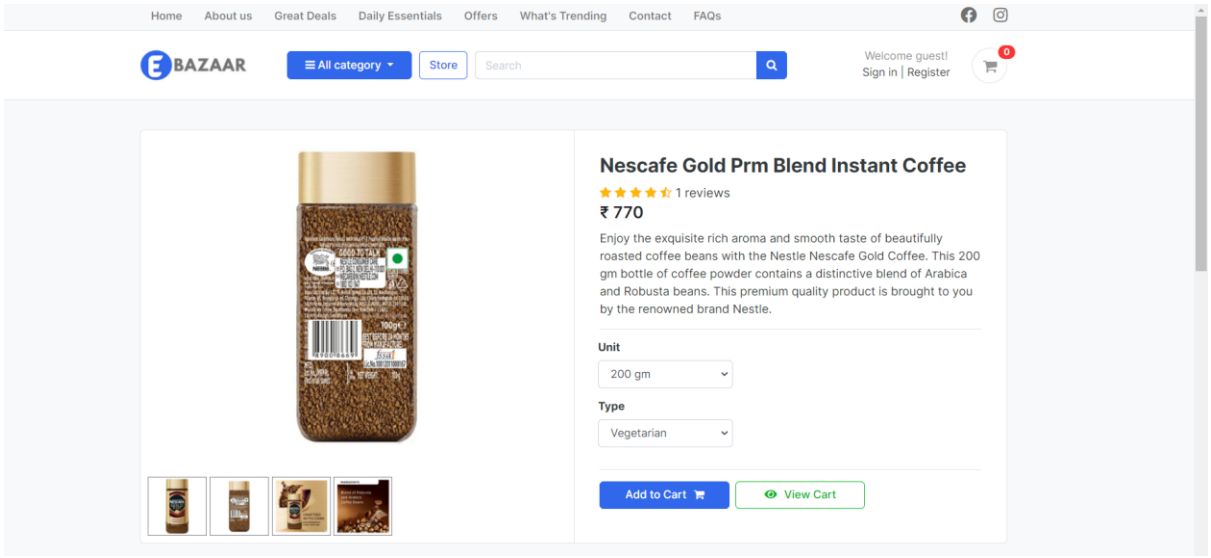
₹ 770

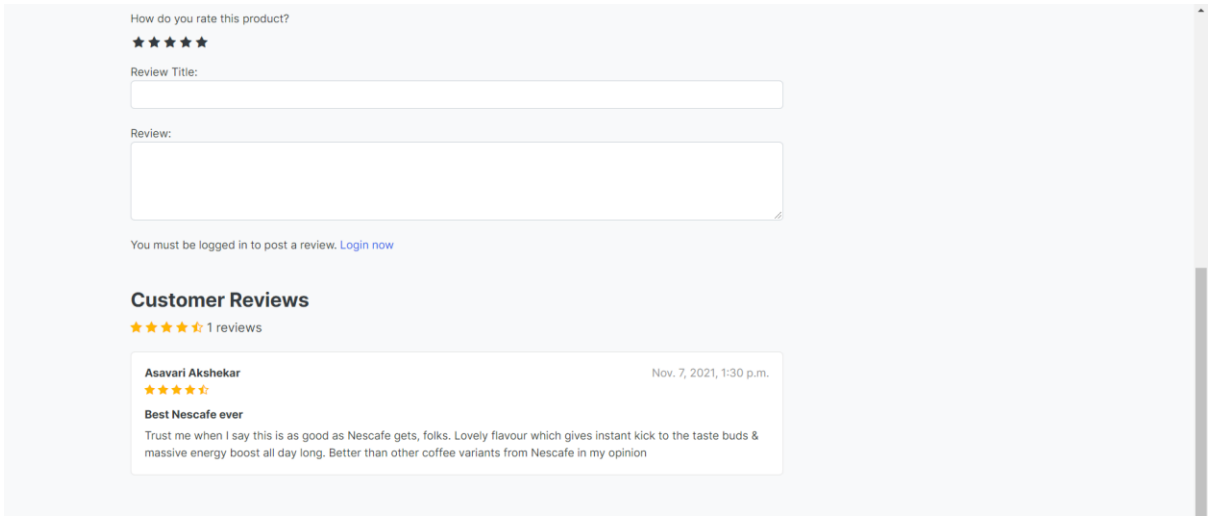
Enjoy the exquisite rich aroma and smooth taste of beautifully roasted coffee beans with the Nestle Nescafe Gold Coffee. This 200 gm bottle of coffee powder contains a distinctive blend of Arabica and Robusta beans. This premium quality product is brought to you by the renowned brand Nestle.

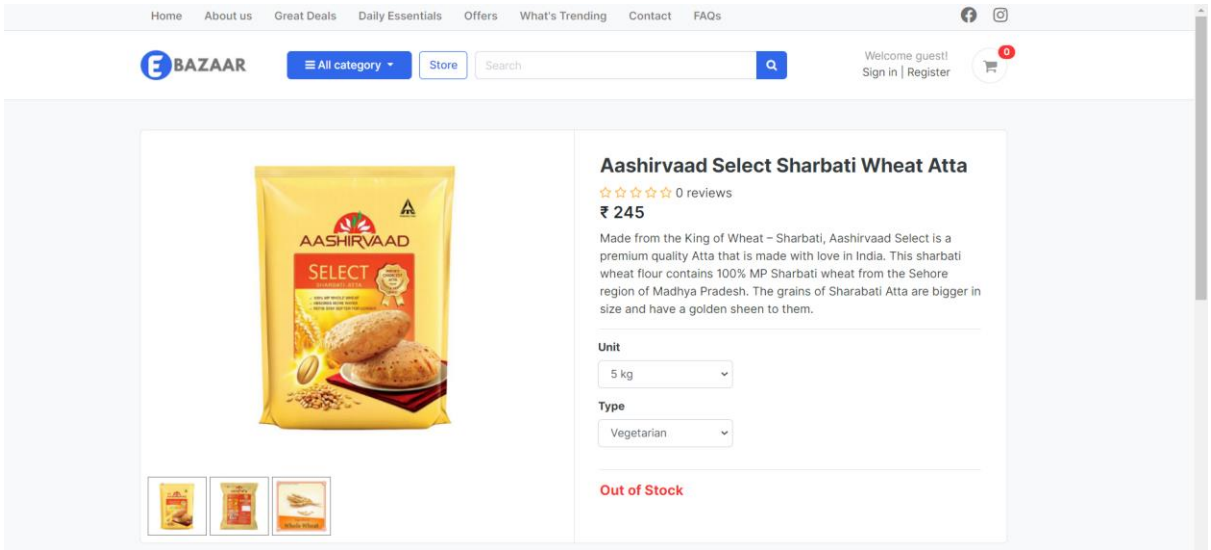
Unit
200 gm

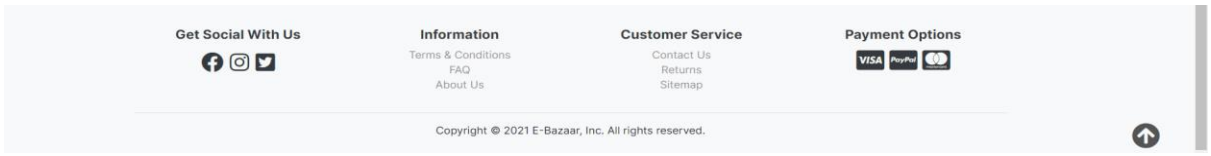
Type
Vegetarian

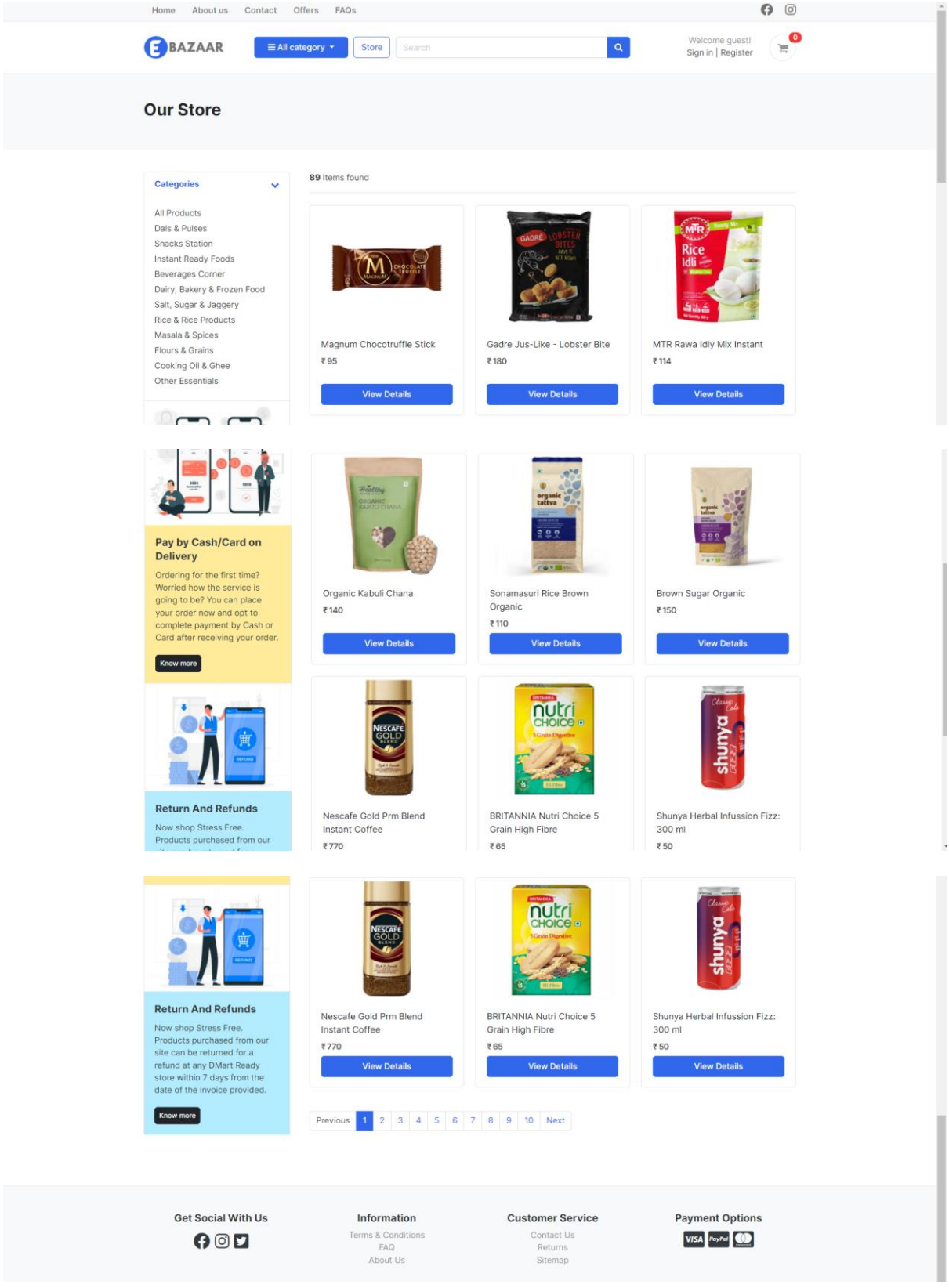
Add to Cart | View Cart


Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
EBAZAAR
All category
Store
Search
Welcome guest!
Sign in | Register
Nescafe Gold Prm Blend Instant Coffee
1 reviews
₹ 770
Enjoy the exquisite rich aroma and smooth taste of beautifully roasted coffee beans with the Nestle Nescafe Gold Coffee. This 200 gm bottle of coffee powder contains a distinctive blend of Arabica and Robusta beans. This premium quality product is brought to you by the renowned brand Nestle.
Unit
200 gm
Type
Vegetarian
Add to Cart
View Cart


How do you rate this product?
Review Title:
Review:
You must be logged in to post a review. Login now
Customer Reviews
1 reviews
Asavari Akshekar
Nov. 7, 2021, 1:30 p.m.
Best Nescafe ever
Trust me when I say this is as good as Nescafe gets, folks. Lovely flavour which gives instant kick to the taste buds & massive energy boost all day long. Better than other coffee variants from Nescafe in my opinion


Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
EBAZAAR
All category
Store
Search
Welcome guest!
Sign in | Register
AASHIRVAAD
SELECT
Aashirvaad Select Sharbati Wheat Atta
0 reviews
₹ 245
Made from the King of Wheat – Sharbati, Aashirvaad Select is a premium quality Atta that is made with love in India. This sharbati wheat flour contains 100% MP Sharbati wheat from the Sehore region of Madhya Pradesh. The grains of Sharabati Atta are bigger in size and have a golden sheen to them.
Unit
5 kg
Type
Vegetarian
Out of Stock


Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.


Home
About us
Contact
Offers
FAQs
BAZAAR
All category
Store
Search
Welcome guest!
Sign in | Register
Our Store
Categories
All Products
Dals & Pulses
Snacks Station
Instant Ready Foods
Beverages Corner
Dairy, Bakery & Frozen Food
Salt, Sugar & Jaggery
Rice & Rice Products
Masala & Spices
Flours & Grains
Cooking Oil & Ghee
Other Essentials
89 Items found
Magnum Chocotruffle Stick
₹ 95
View Details
Gadre Jus-Like - Lobster Bite
₹ 180
View Details
MTR Rawa Idly Mix Instant
₹ 114
View Details
Pay by Cash/Card on Delivery
Ordering for the first time? Worried how the service is going to be? You can place your order now and opt to complete payment by Cash or Card after receiving your order.
Know more
Organic Kabuli Chana
₹ 140
View Details
Sonamasuri Rice Brown Organic
₹ 110
View Details
Brown Sugar Organic
₹ 150
View Details
Return And Refunds
Now shop Stress Free. Products purchased from our
Nescafe Gold Prm Blend Instant Coffee
₹ 770
BRITANNIA Nutri Choice 5 Grain High Fibre
₹ 65
Shunya Herbal Infussion Fizz: 300 ml
₹ 50
Return And Refunds
Now shop Stress Free. Products purchased from our site can be returned for a refund at any DMart Ready store within 7 days from the date of the invoice provided.
Know more
Nescafe Gold Prm Blend Instant Coffee
₹ 770
View Details
BRITANNIA Nutri Choice 5 Grain High Fibre
₹ 65
View Details
Shunya Herbal Infussion Fizz: 300 ml
₹ 50
View Details
Previous
1
2
3
4
5
6
7
8
9
10
Next
Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options

Our Store

Categories

All Products
Dals & Pulses
Snacks Station
Instant Ready Foods
Beverages Corner
Dairy, Bakery & Frozen Food
Salt, Sugar & Jaggery
Rice & Rice Products
Masala & Spices
Flours & Grains
Cooking Oil & Ghee
Other Essentials

Pay by Cash/Card on Delivery

Ordering for the first time? Worried how the service is going to be? You can place your order now and opt to complete payment by Cash or Card after receiving your order.

Know more

11 Items found

Organic Kabuli Chana
₹ 140
View Details

Tata Sampann Chana Dal
₹ 115
View Details

Tata Sampann Moong Dal
₹ 141
View Details

Tata Sampann Urad Dal
₹ 165
View Details

Satyam Chana Small
₹ 64
View Details

Satyam Moong Whole
₹ 100
View Details

Previous 1 2 Next

Home About us Great Deals Daily Essentials Offers What's Trending Contact FAQs

BAZAAR All category Store Wadilal

Welcome guest!
Sign in | Register

Your Search Results

Categories

All Products
Dals & Pulses
Snacks Station
Instant Ready Foods
Beverages Corner
Dairy, Bakery & Frozen Food
Salt, Sugar & Jaggery
Rice & Rice Products
Masala & Spices
Flours & Grains
Cooking Oil & Ghee
Other Essentials

0 Items found

Sorry! No matching results found

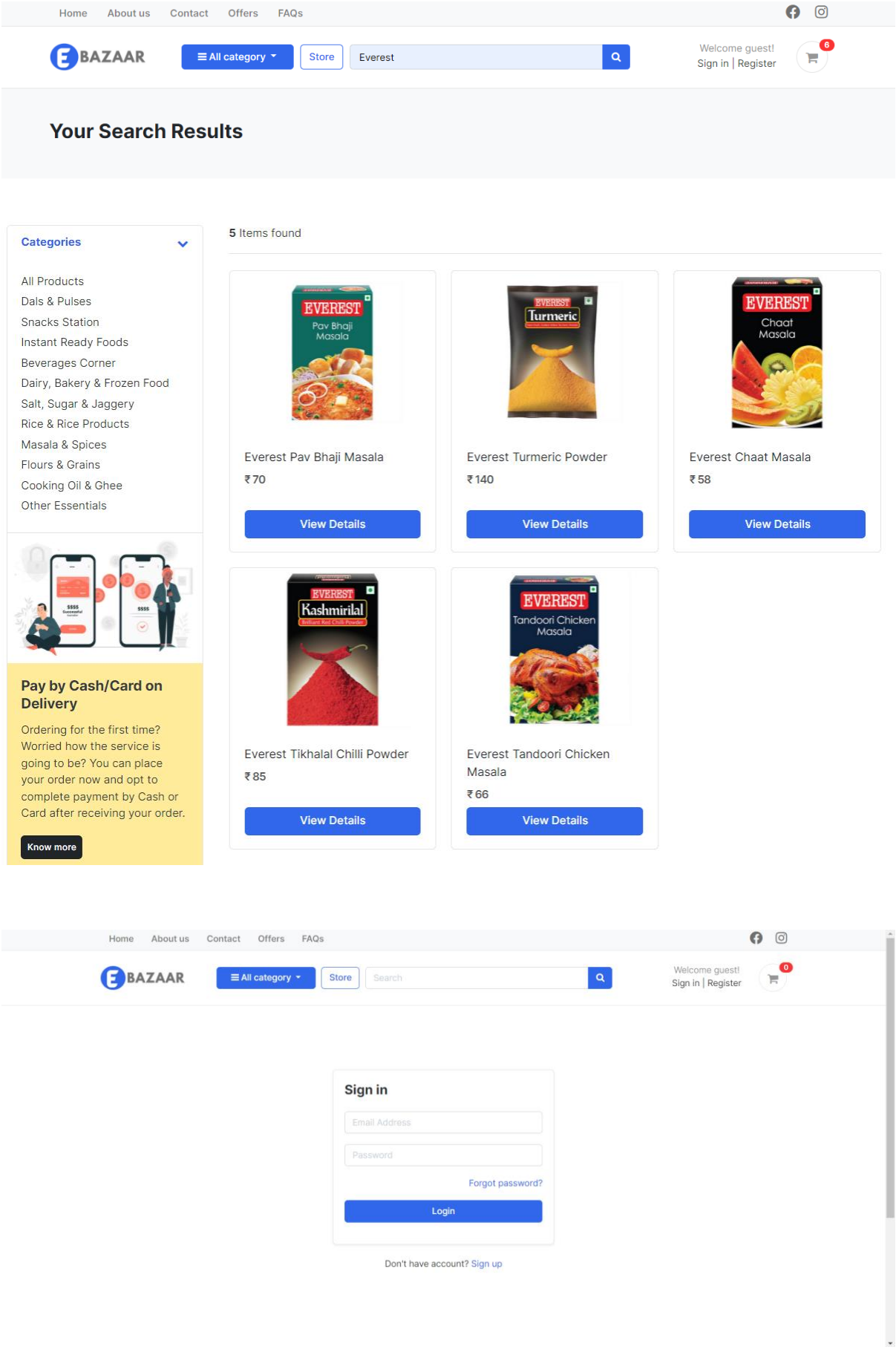
Try a different keyword maybe?

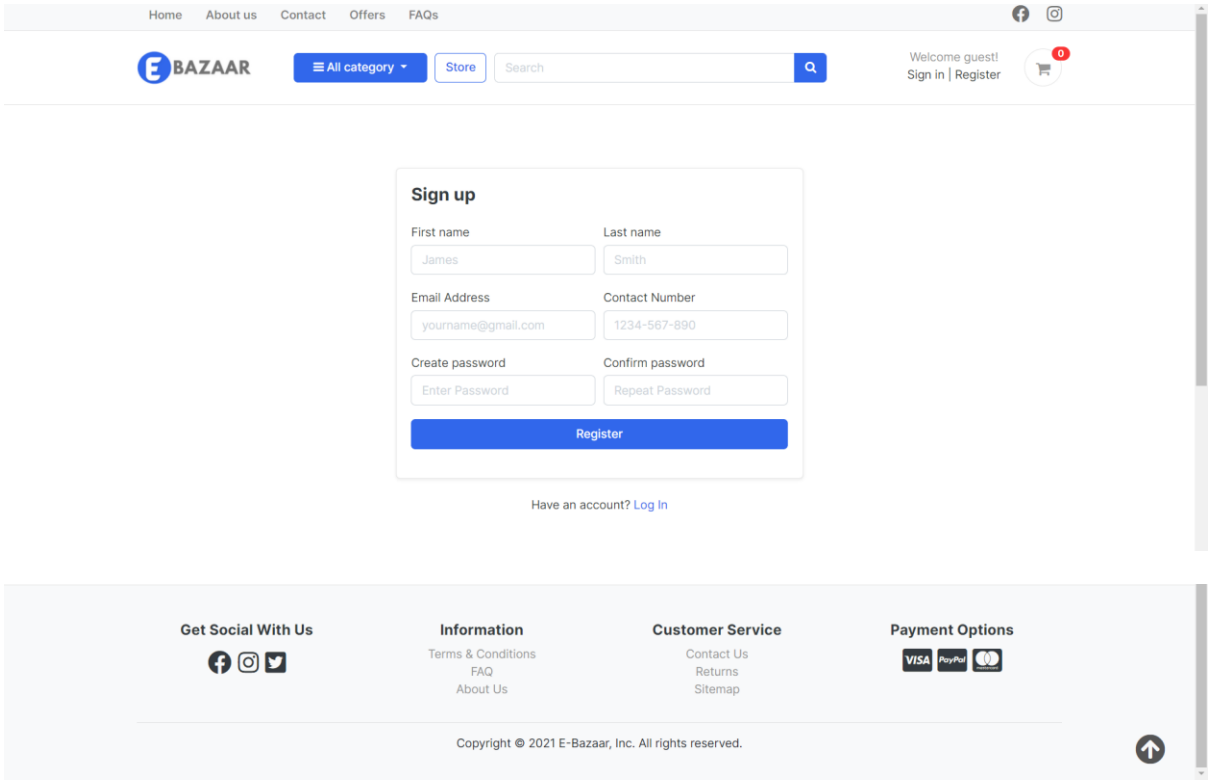
Get Social With Us

Information
Terms & Conditions
FAQ
About Us

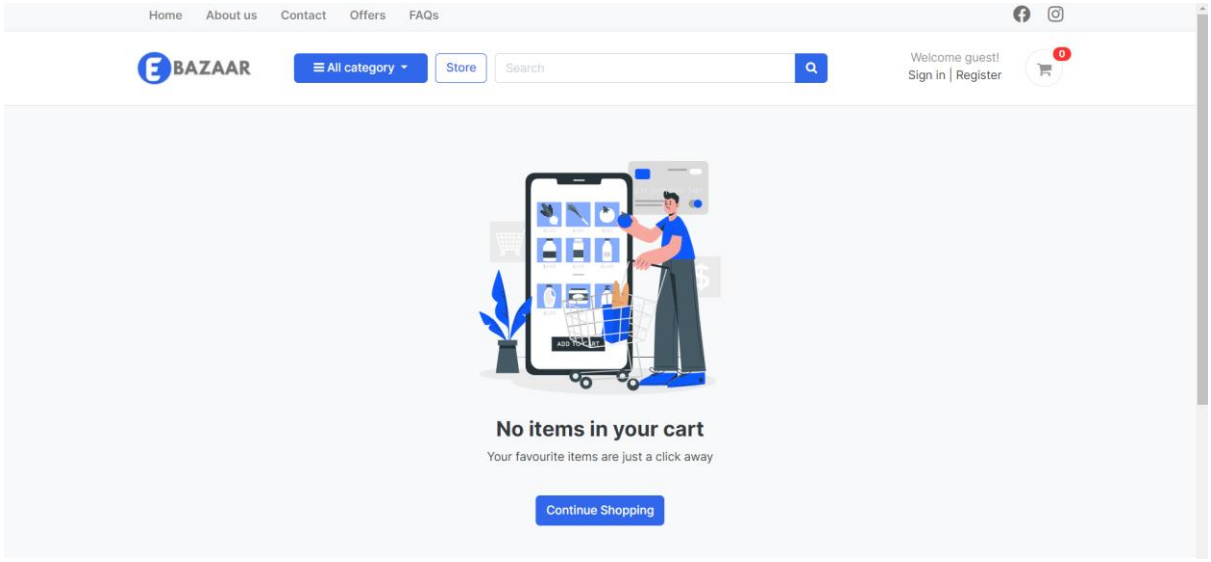
Customer Service
Contact Us
Returns
Sitemap

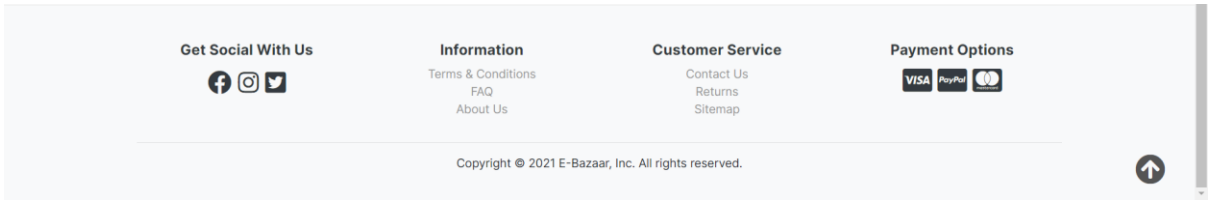
Payment Options

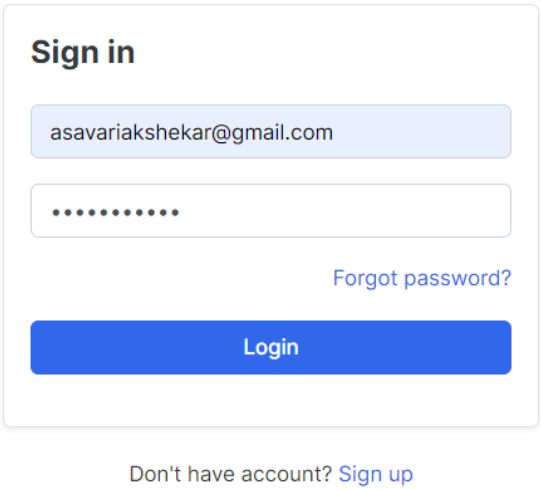

Home
About us
Contact
Offers
FAQs
BAZAAR
All category
Store
Everest
Welcome guest!
Sign in | Register
6
Your Search Results
Categories
All Products
Dals & Pulses
Snacks Station
Instant Ready Foods
Beverages Corner
Dairy, Bakery & Frozen Food
Salt, Sugar & Jaggery
Rice & Rice Products
Masala & Spices
Flours & Grains
Cooking Oil & Ghee
Other Essentials
Pay by Cash/Card on Delivery
Ordering for the first time? Worried how the service is going to be? You can place your order now and opt to complete payment by Cash or Card after receiving your order.
Know more
5 Items found
Everest Pav Bhaji Masala
₹ 70
View Details
Everest Turmeric Powder
₹ 140
View Details
Everest Chaat Masala
₹ 58
View Details
Everest Tikhalal Chilli Powder
₹ 85
View Details
Everest Tandoori Chicken Masala
₹ 66
View Details
Home
About us
Contact
Offers
FAQs
BAZAAR
All category
Store
Search
Welcome guest!
Sign in | Register
0
Sign in
Email Address
Password
Forgot password?
Login
Don't have account? Sign up

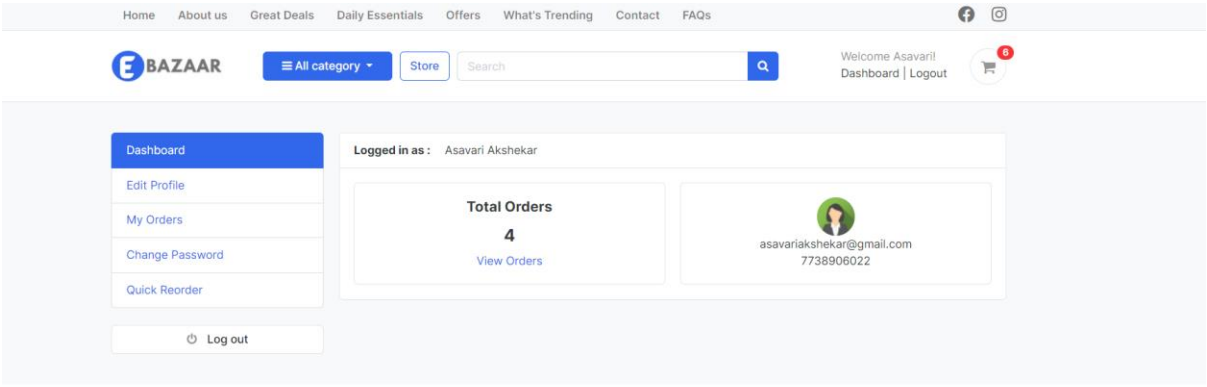

Home
About us
Contact
Offers
FAQs
BAZAAR
All category
Store
Search
Welcome guest!
Sign in | Register
Sign up
First name
James
Last name
Smith
Email Address
yourname@gmail.com
Contact Number
1234-567-890
Create password
Enter Password
Confirm password
Repeat Password
Register
Have an account? Log In

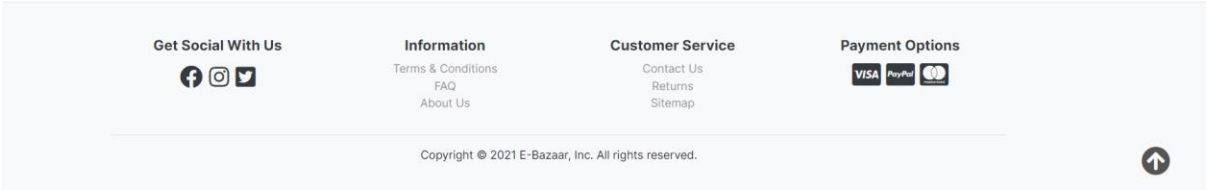
Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.

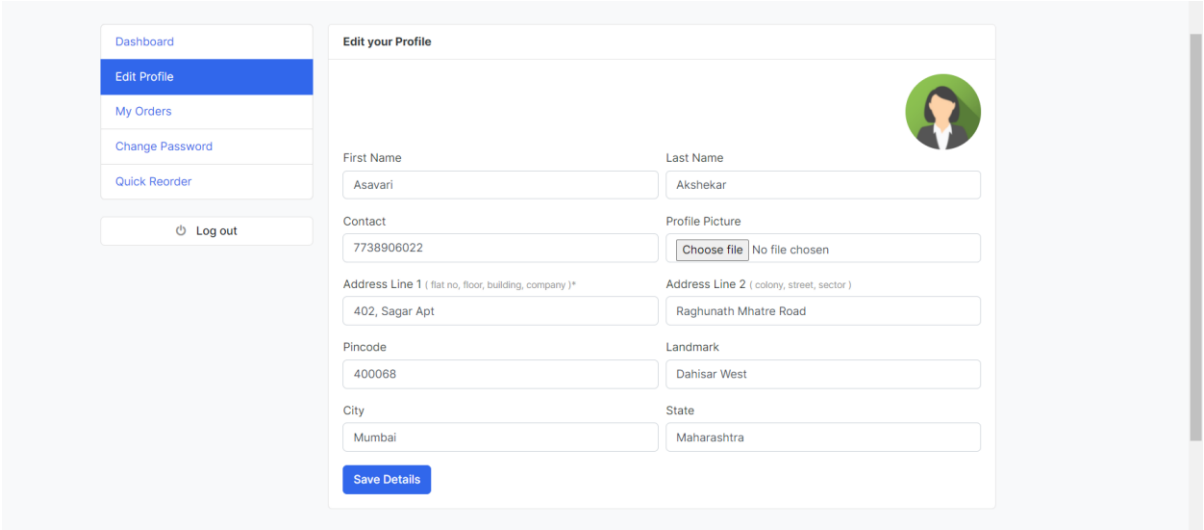

Home
About us
Contact
Offers
FAQs
BAZAAR
All category
Store
Search
Welcome guest!
Sign in | Register
No items in your cart
Your favourite items are just a click away
Continue Shopping

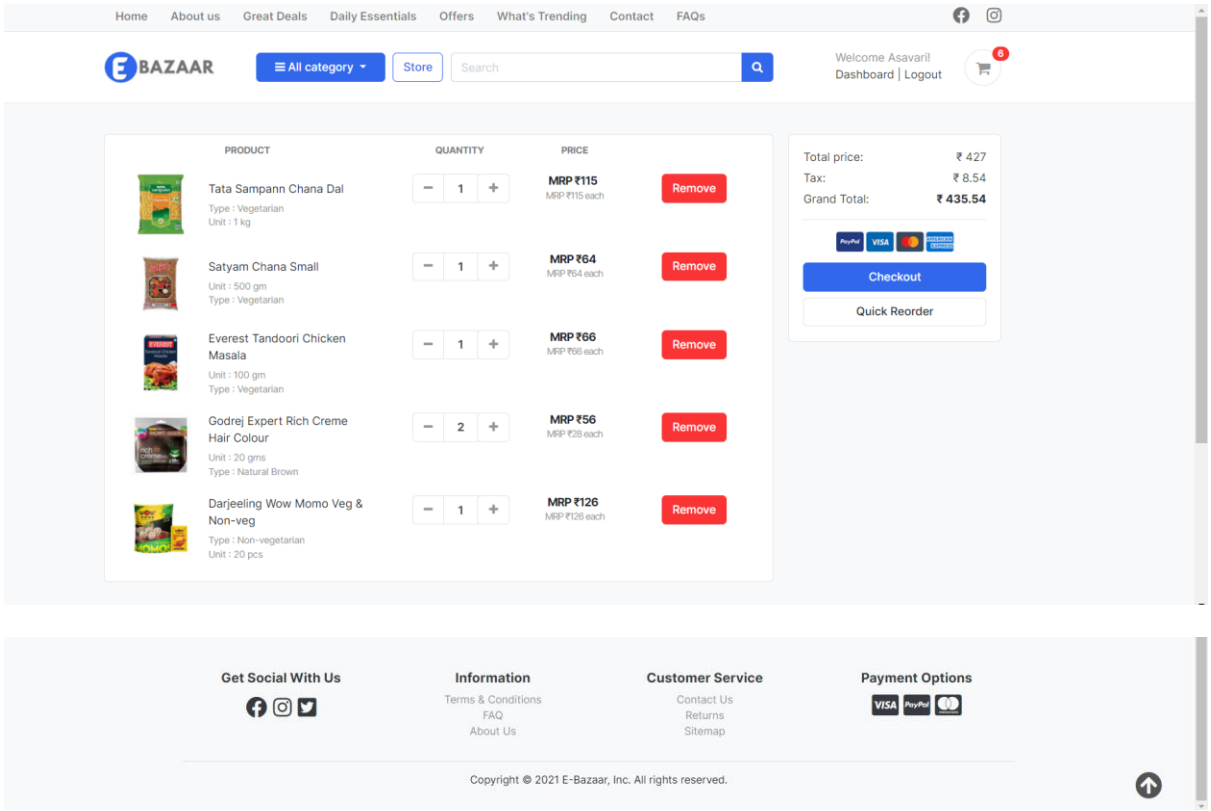

Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.


Sign in
asavariakshekar@gmail.com
Forgot password?
Login
Don't have account? Sign up

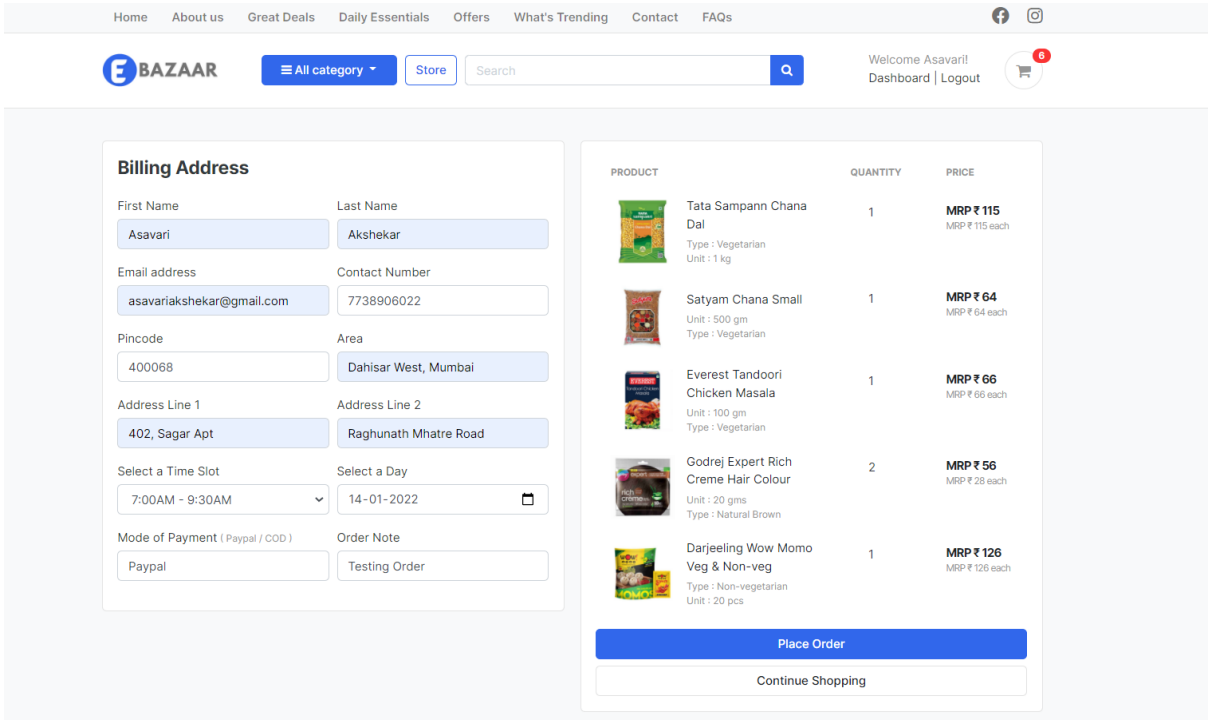

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Dashboard
Edit Profile
My Orders
Change Password
Quick Reorder
Log out
Logged in as : Asavari Akshekar
Total Orders
4
View Orders
asavariakshekar@gmail.com
7738906022

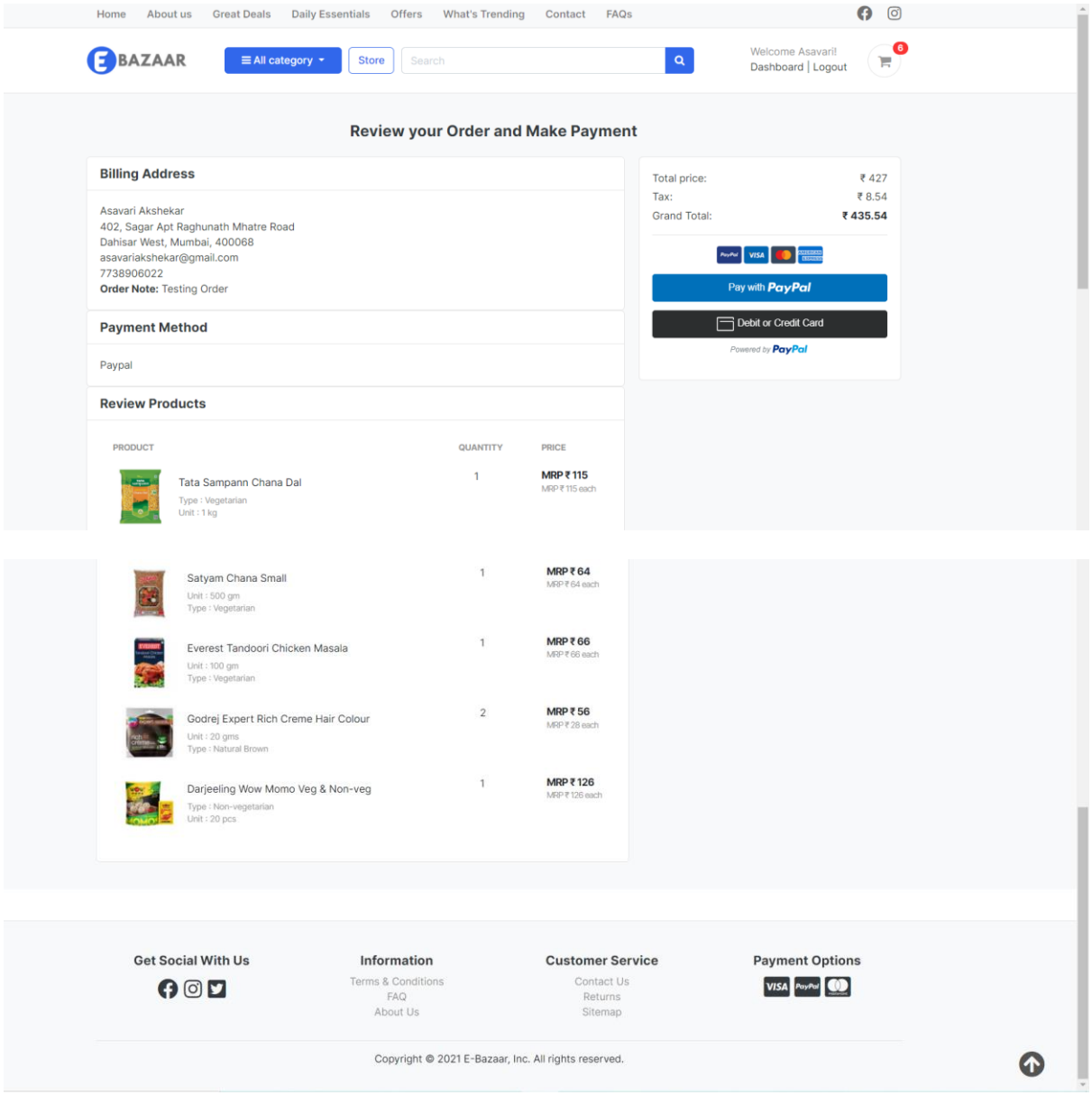

Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.

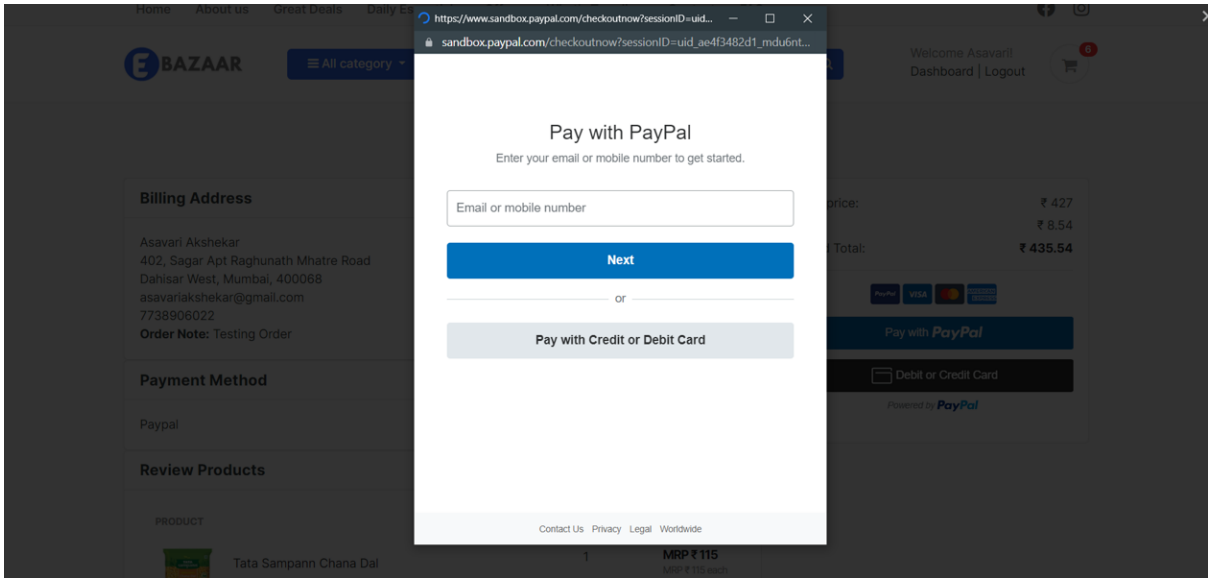

Dashboard
Edit Profile
My Orders
Change Password
Quick Reorder
Log out
Edit your Profile
First Name
Asavari
Last Name
Akshekar
Contact
7738906022
Profile Picture
Choose file
No file chosen
Address Line 1 (flat no, floor, building, company)*
402, Sagar Apt
Address Line 2 (colony, street, sector)
Raghunath Mhatre Road
Pincode
400068
Landmark
Dahisar West
City
Mumbai
State
Maharashtra
Save Details

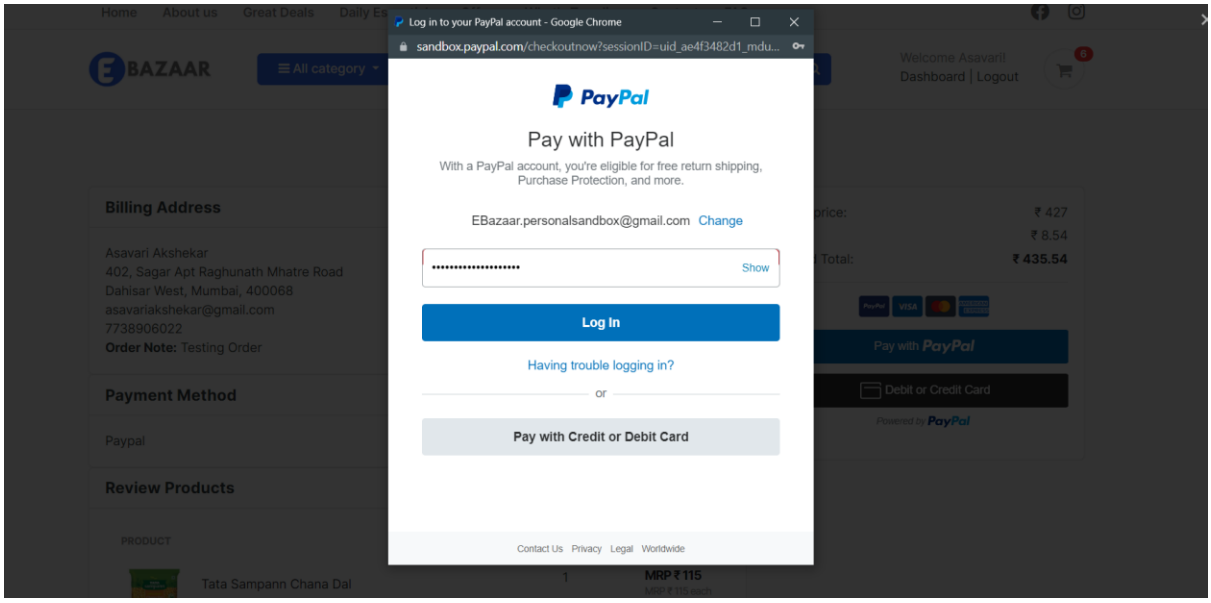

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
PRODUCT
QUANTITY
PRICE
Tata Sampann Chana Dal
Type : Vegetarian
Unit : 1 kg
MRP ₹115
MRP ₹115 each
Remove
Satyam Chana Small
Unit : 500 gm
Type : Vegetarian
MRP ₹64
MRP ₹64 each
Remove
Everest Tandoori Chicken Masala
Unit : 100 gm
Type : Vegetarian
MRP ₹66
MRP ₹66 each
Remove
Godrej Expert Rich Creme Hair Colour
Unit : 20 gms
Type : Natural Brown
MRP ₹56
MRP ₹28 each
Remove
Darjeeling Wow Momo Veg & Non-veg
Type : Non-vegetarian
Unit : 20 pcs
MRP ₹126
MRP ₹126 each
Remove
Total price: ₹ 427
Tax: ₹ 8.54
Grand Total: ₹ 435.54
Checkout
Quick Reorder

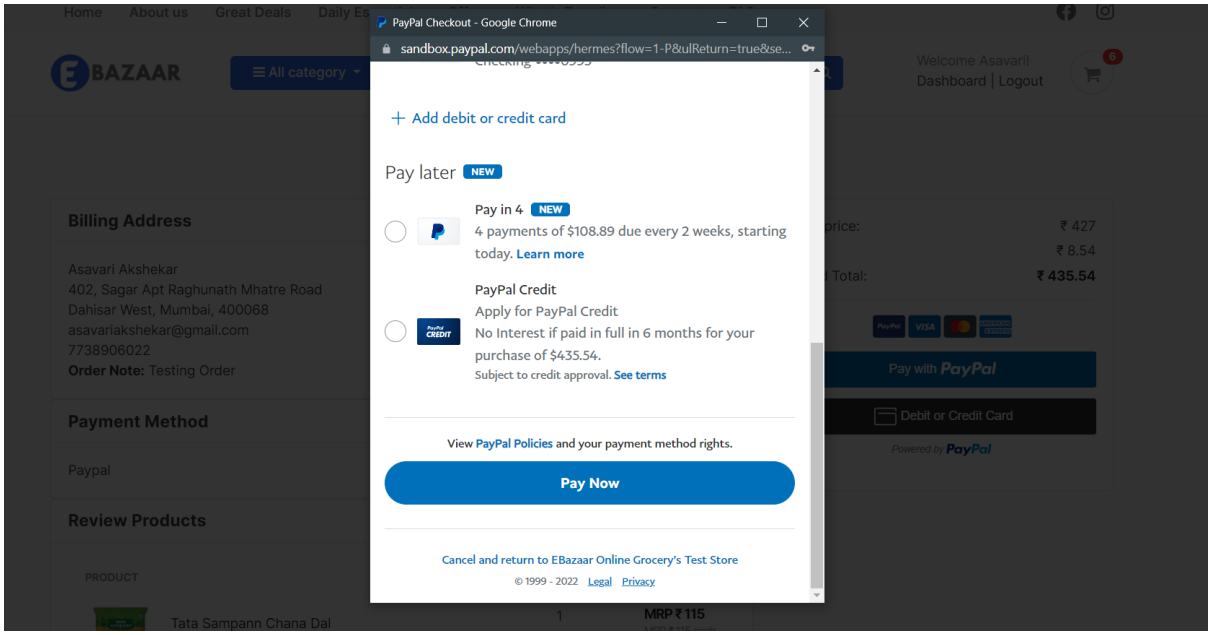
Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.

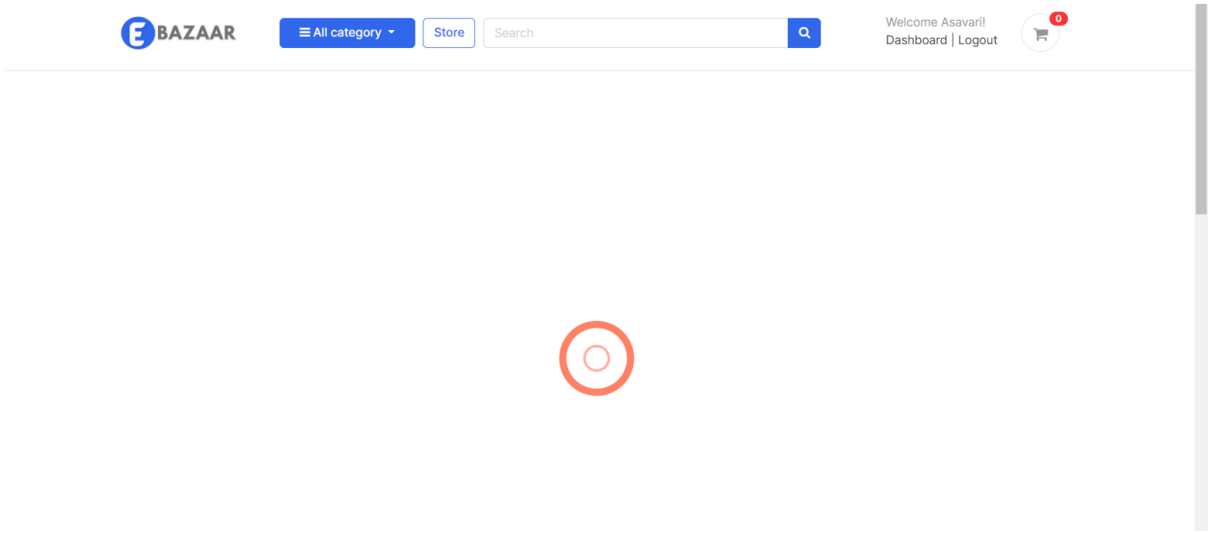

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Billing Address
First Name
Asavari
Last Name
Akshekar
Email address
asavariakshekar@gmail.com
Contact Number
7738906022
Pincode
400068
Area
Dahisar West, Mumbai
Address Line 1
402, Sagar Apt
Address Line 2
Raghunath Mhatre Road
Select a Time Slot
7:00AM - 9:30AM
Select a Day
14-01-2022
Mode of Payment (Paypal / COD)
Paypal
Order Note
Testing Order
PRODUCT
QUANTITY
PRICE
Tata Sampann Chana Dal
Type : Vegetarian
Unit : 1 kg
1
MRP ₹ 115
MRP ₹ 115 each
Satyam Chana Small
Unit : 500 gm
Type : Vegetarian
1
MRP ₹ 64
MRP ₹ 64 each
Everest Tandoori Chicken Masala
Unit : 100 gm
Type : Vegetarian
1
MRP ₹ 66
MRP ₹ 66 each
Godrej Expert Rich Creme Hair Colour
Unit : 20 gms
Type : Natural Brown
2
MRP ₹ 56
MRP ₹ 28 each
Darjeeling Wow Momo Veg & Non-veg
Type : Non-vegetarian
Unit : 20 pcs
1
MRP ₹ 126
MRP ₹ 126 each
Place Order
Continue Shopping

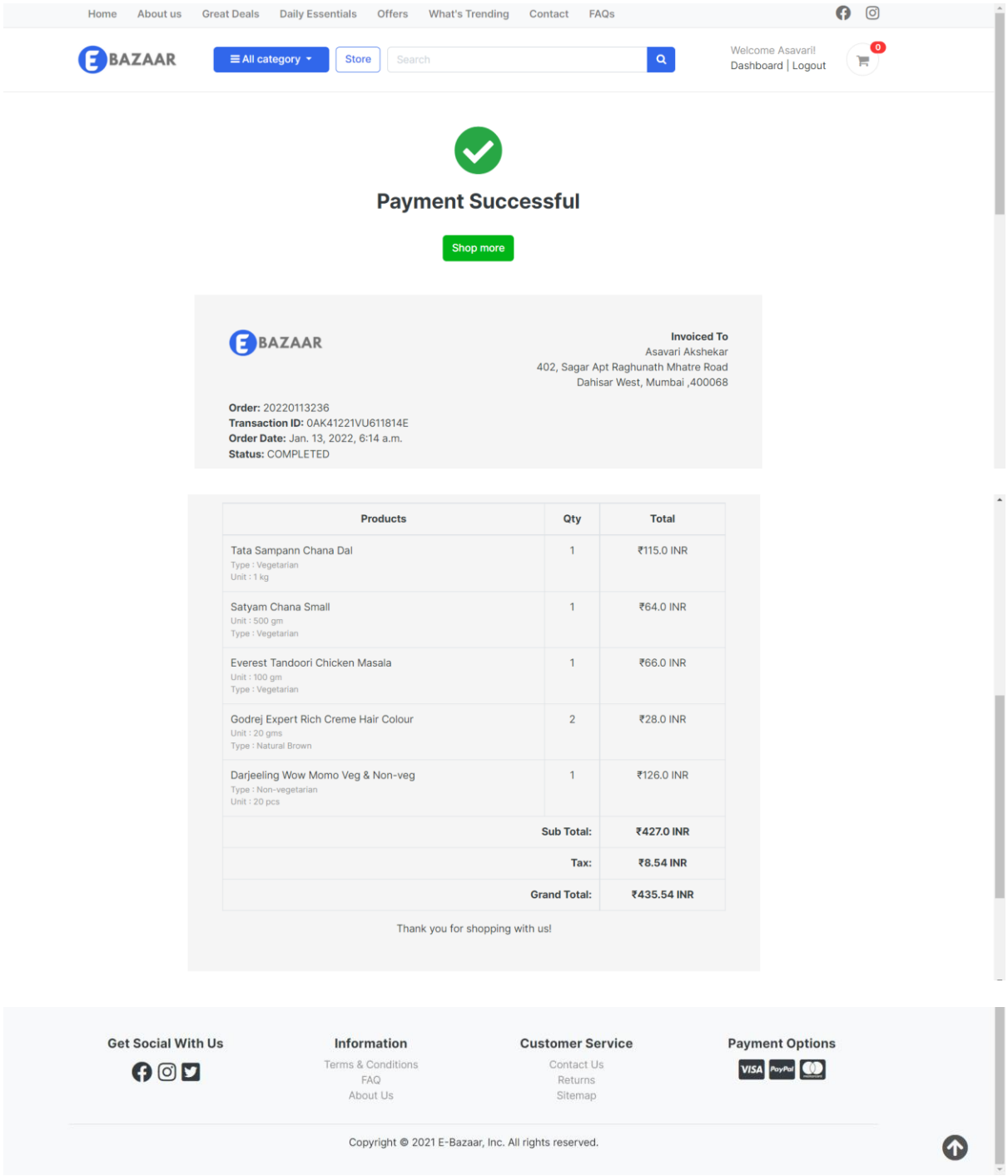

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Review your Order and Make Payment
Billing Address
Asavari Akshekar
402, Sagar Apt Raghunath Mhatre Road
Dahisar West, Mumbai, 400068
asavariakshekar@gmail.com
7738906022
Order Note: Testing Order
Payment Method
Paypal
Review Products
PRODUCT
QUANTITY
PRICE
Tata Sampann Chana Dal
Type : Vegetarian
Unit : 1 kg
1
MRP ₹ 115
MRP ₹ 115 each
Total price: ₹ 427
Tax: ₹ 8.54
Grand Total: ₹ 435.54
Pay with PayPal
Debit or Credit Card
Powered by PayPal
Satyam Chana Small
Unit : 500 gm
Type : Vegetarian
1
MRP ₹ 64
MRP ₹ 64 each
Everest Tandoori Chicken Masala
Unit : 100 gm
Type : Vegetarian
1
MRP ₹ 66
MRP ₹ 66 each
Godrej Expert Rich Creme Hair Colour
Unit : 20 gms
Type : Natural Brown
2
MRP ₹ 56
MRP ₹ 28 each
Darjeeling Wow Momo Veg & Non-veg
Type : Non-vegetarian
Unit : 20 pcs
1
MRP ₹ 126
MRP ₹ 126 each
Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.

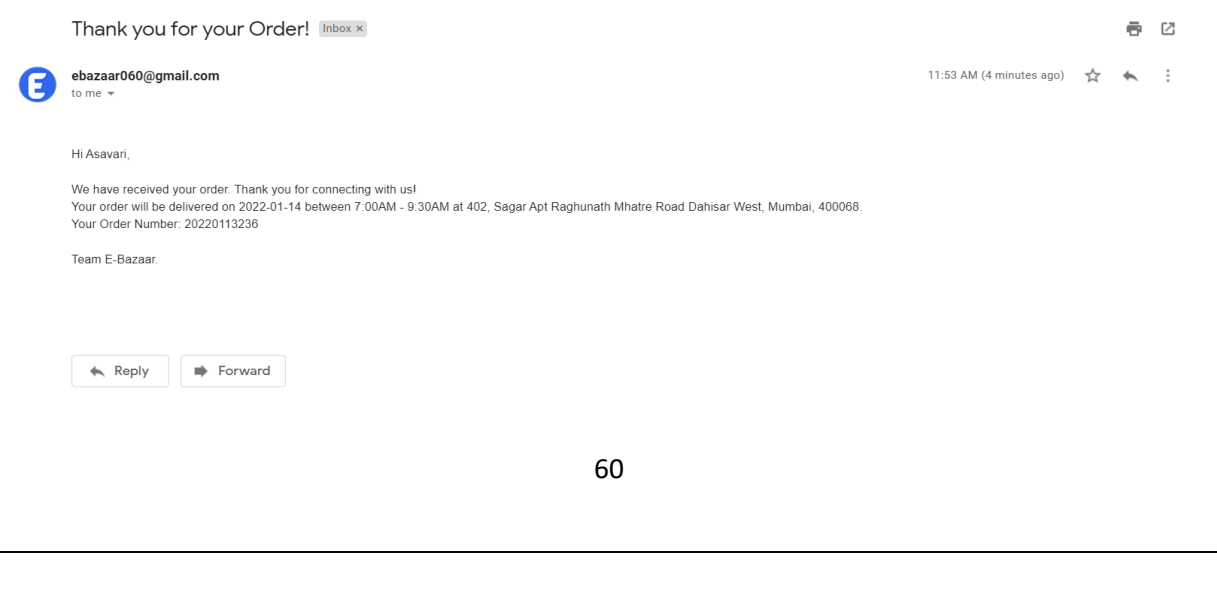

https://www.sandbox.paypal.com/checkoutnow?sessionID=uid...
sandbox.paypal.com/checkoutnow?sessionID=uid_ae4f3482d1_mdu6nt...
Pay with PayPal
Enter your email or mobile number to get started.
Email or mobile number
Next
or
Pay with Credit or Debit Card
Contact Us
Privacy
Legal
Worldwide

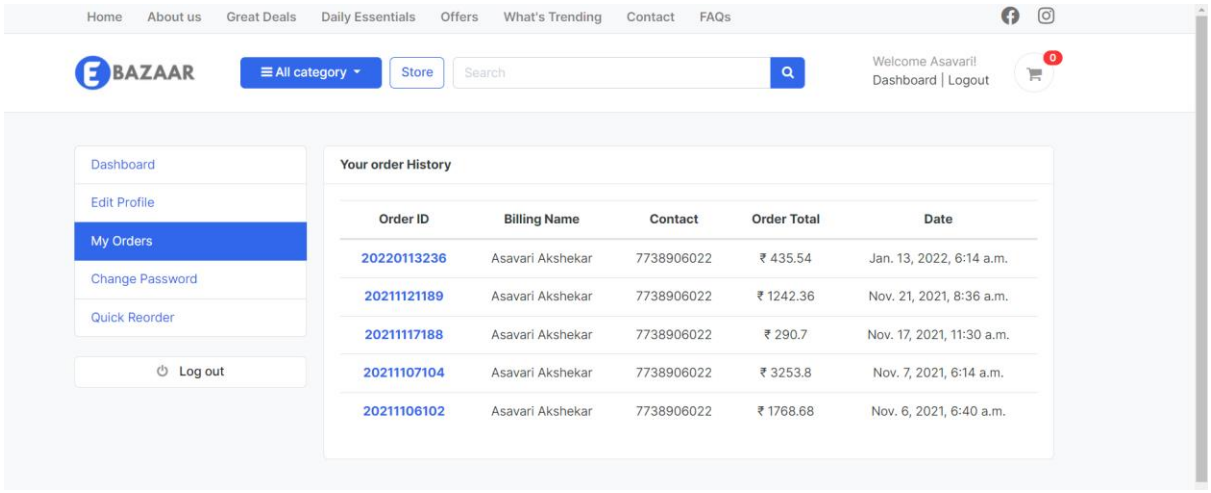

Log in to your PayPal account - Google Chrome
sandbox.paypal.com/checkoutnow?sessionID=uid_ae4f3482d1_mdu...
PayPal
Pay with PayPal
With a PayPal account, you're eligible for free return shipping, Purchase Protection, and more.
EBazaar.personalsandbox@gmail.com Change
Show
Log In
Having trouble logging in?
or
Pay with Credit or Debit Card
Contact Us Privacy Legal Worldwide
Billing Address
Payment Method
Review Products

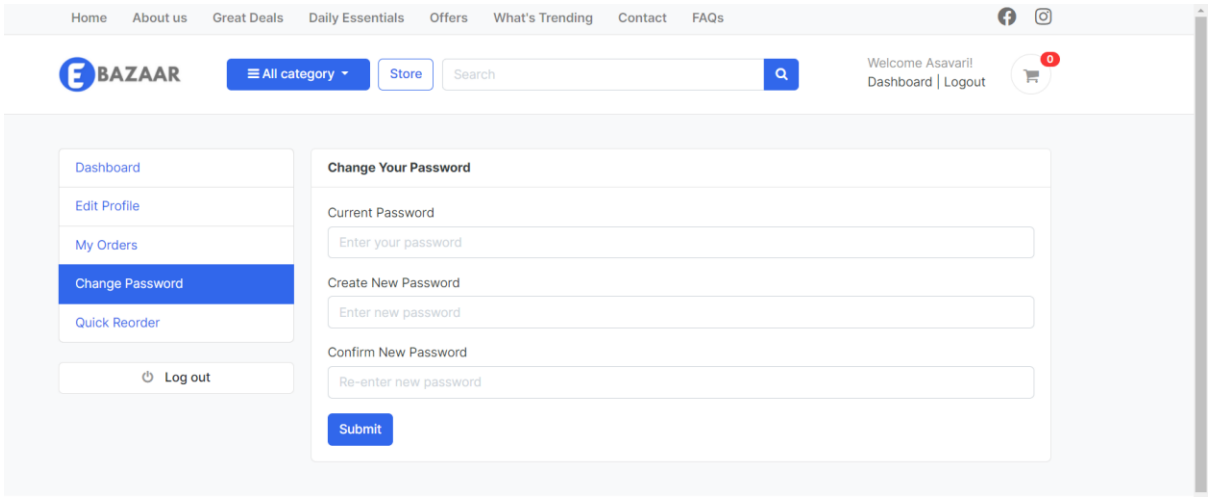

PayPal Checkout - Google Chrome
sandbox.paypal.com/webapps/hermes?flow=1-P&ulReturn=true&se...
Add debit or credit card
Pay later NEW
Pay in 4 NEW
4 payments of $108.89 due every 2 weeks, starting today. Learn more
PayPal Credit
Apply for PayPal Credit
No Interest if paid in full in 6 months for your purchase of $435.54.
Subject to credit approval. See terms
View PayPal Policies and your payment method rights.
Pay Now
Cancel and return to EBazaar Online Grocery's Test Store
© 1999 - 2022 Legal Privacy
Billing Address
Payment Method
Review Products

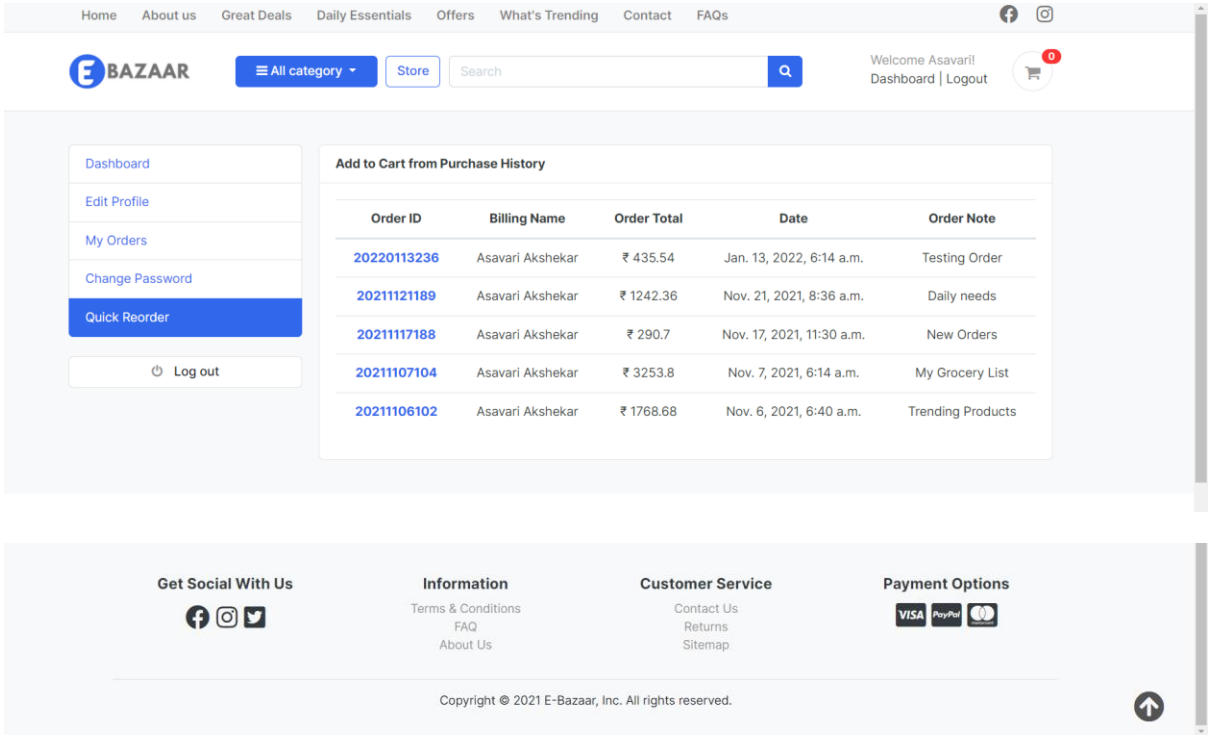

BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout


Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Payment Successful
Shop more
BAZAAR
Invoiced To
Asavari Akshekar
402, Sagar Apt Raghunath Mhatre Road
Dahisar West, Mumbai ,400068
Order: 20220113236
Transaction ID: 0AK41221VU611814E
Order Date: Jan. 13, 2022, 6:14 a.m.
Status: COMPLETED
Products
Qty
Total
Tata Sampann Chana Dal
Type : Vegetarian
Unit : 1 kg
1
₹115.0 INR
Satyam Chana Small
Unit : 500 gm
Type : Vegetarian
1
₹64.0 INR
Everest Tandoori Chicken Masala
Unit : 100 gm
Type : Vegetarian
1
₹66.0 INR
Godrej Expert Rich Creme Hair Colour
Unit : 20 gms
Type : Natural Brown
2
₹28.0 INR
Darjeeling Wow Momo Veg & Non-veg
Type : Non-vegetarian
Unit : 20 pcs
1
₹126.0 INR
Sub Total:
₹427.0 INR
Tax:
₹8.54 INR
Grand Total:
₹435.54 INR
Thank you for shopping with us!
Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.

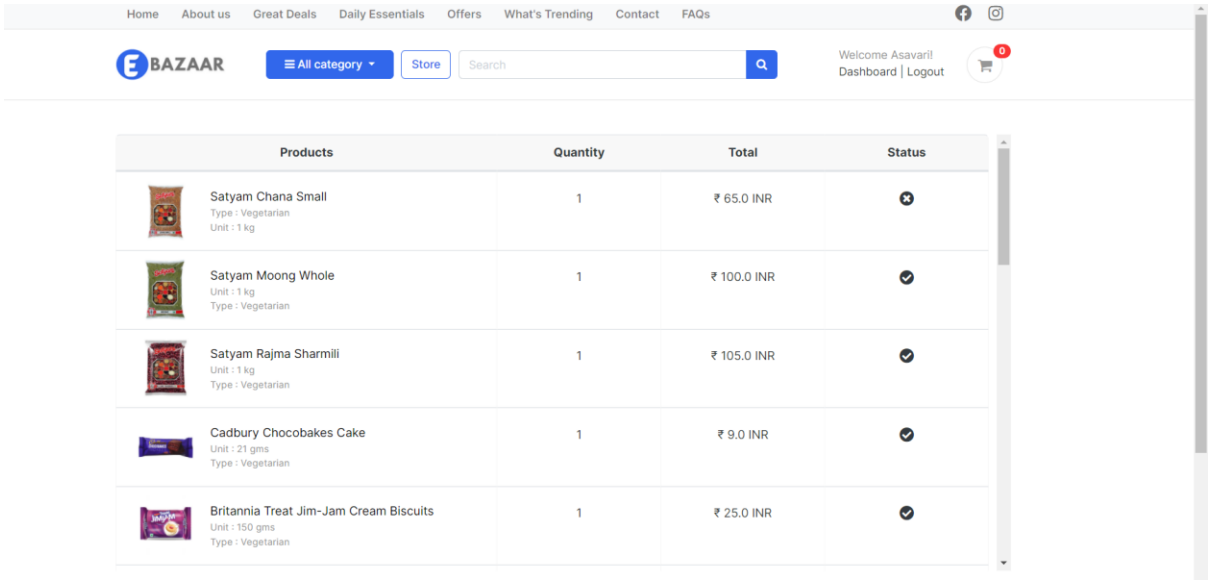

Thank you for your Order!
Inbox ×
ebazaar060@gmail.com
to me
11:53 AM (4 minutes ago)
Hi Asavari,
We have received your order. Thank you for connecting with us!
Your order will be delivered on 2022-01-14 between 7:00AM - 9:30AM at 402, Sagar Apt Raghunath Mhatre Road Dahisar West, Mumbai, 400068.
Your Order Number: 20220113236
Team E-Bazaar.
Reply
Forward

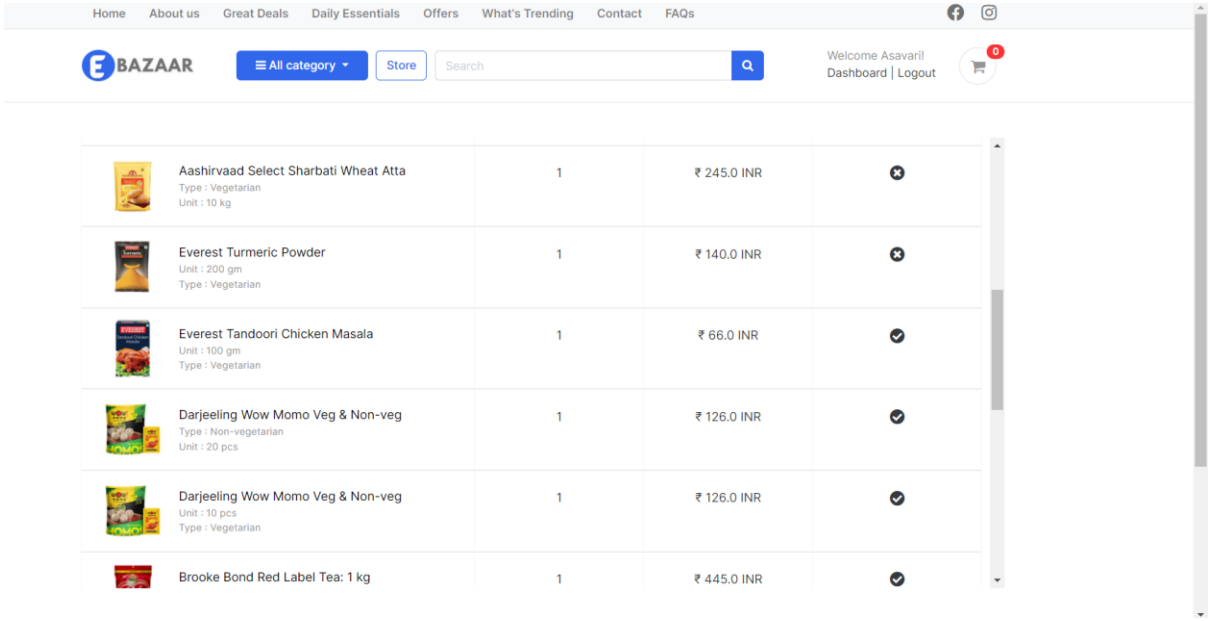

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Dashboard
Edit Profile
My Orders
Change Password
Quick Reorder
Log out
Your order History
Order ID
Billing Name
Contact
Order Total
Date
20220113236
Asavari Akshekar
7738906022
₹ 435.54
Jan. 13, 2022, 6:14 a.m.
20211121189
Asavari Akshekar
7738906022
₹ 1242.36
Nov. 21, 2021, 8:36 a.m.
20211117188
Asavari Akshekar
7738906022
₹ 290.7
Nov. 17, 2021, 11:30 a.m.
20211107104
Asavari Akshekar
7738906022
₹ 3253.8
Nov. 7, 2021, 6:14 a.m.
20211106102
Asavari Akshekar
7738906022
₹ 1768.68
Nov. 6, 2021, 6:40 a.m.

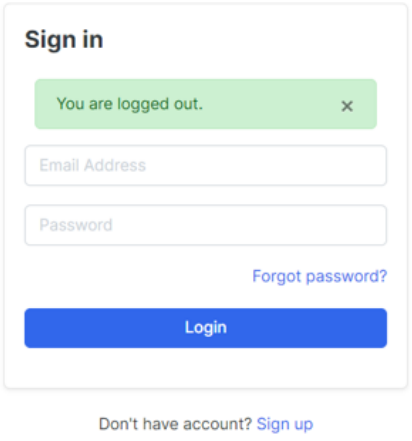

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Dashboard
Edit Profile
My Orders
Change Password
Quick Reorder
Log out
Change Your Password
Current Password
Enter your password
Create New Password
Enter new password
Confirm New Password
Re-enter new password
Submit

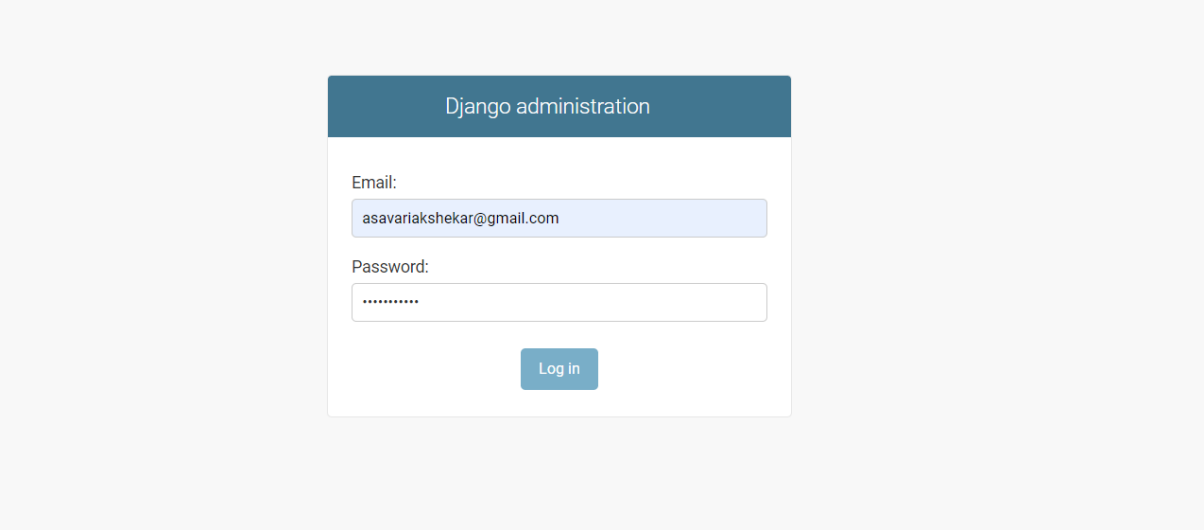

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Dashboard
Edit Profile
My Orders
Change Password
Quick Reorder
Log out
Add to Cart from Purchase History
Order ID
Billing Name
Order Total
Date
Order Note
20220113236
Asavari Akshekar
₹ 435.54
Jan. 13, 2022, 6:14 a.m.
Testing Order
20211121189
Asavari Akshekar
₹ 1242.36
Nov. 21, 2021, 8:36 a.m.
Daily needs
20211117188
Asavari Akshekar
₹ 290.7
Nov. 17, 2021, 11:30 a.m.
New Orders
20211107104
Asavari Akshekar
₹ 3253.8
Nov. 7, 2021, 6:14 a.m.
My Grocery List
20211106102
Asavari Akshekar
₹ 1768.68
Nov. 6, 2021, 6:40 a.m.
Trending Products

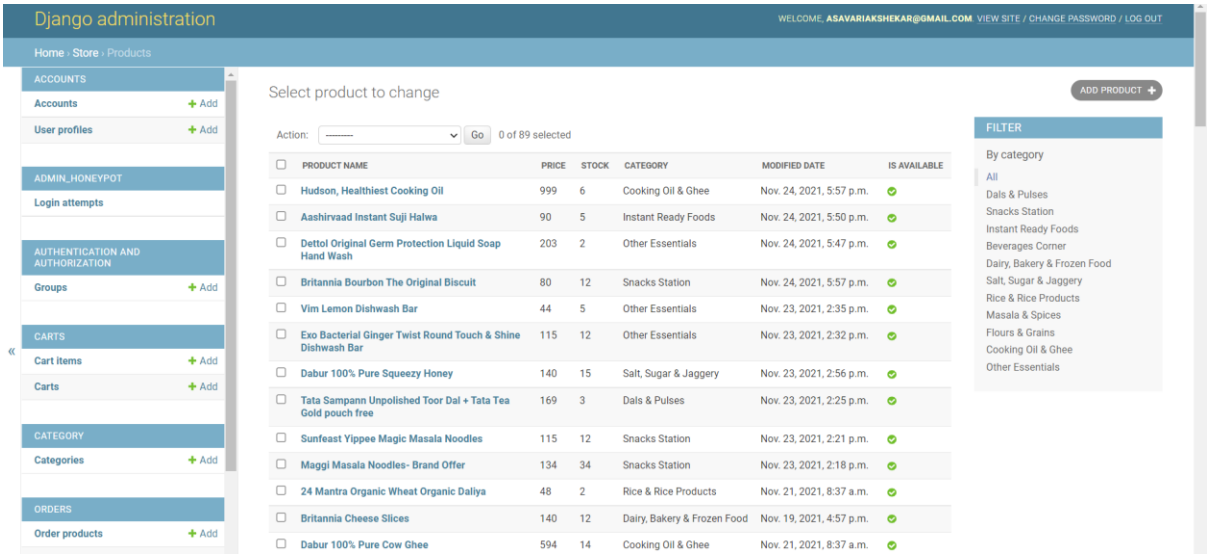
Get Social With Us
Information
Terms & Conditions
FAQ
About Us
Customer Service
Contact Us
Returns
Sitemap
Payment Options
Copyright © 2021 E-Bazaar, Inc. All rights reserved.

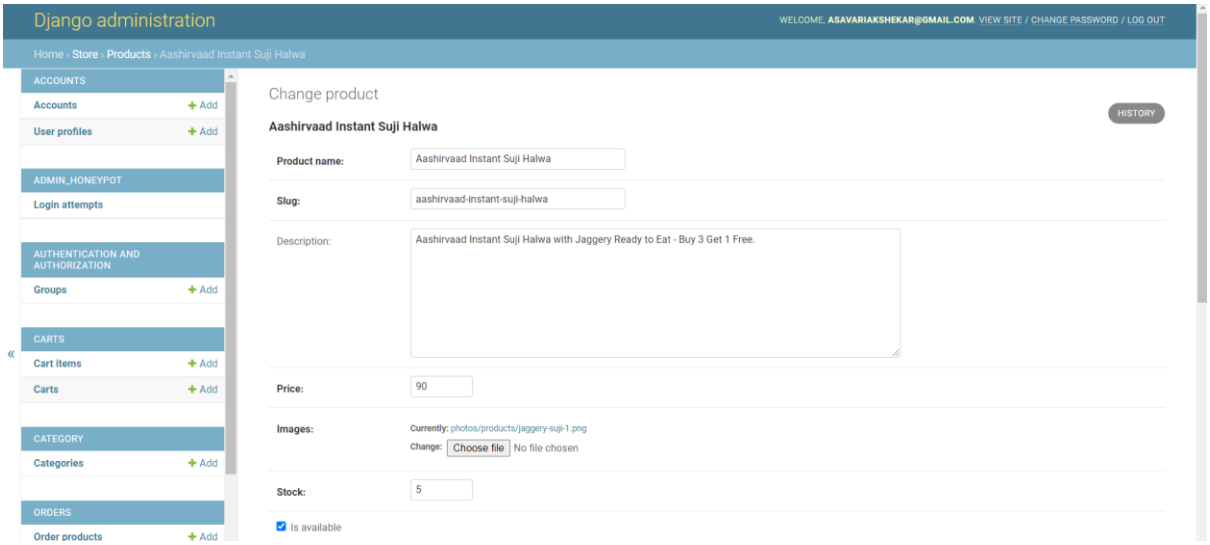

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Products
Quantity
Total
Status
Satyam Chana Small
Type : Vegetarian
Unit : 1 kg
1
₹ 65.0 INR
Satyam Moong Whole
Unit : 1 kg
Type : Vegetarian
1
₹ 100.0 INR
Satyam Rajma Sharmili
Unit : 1 kg
Type : Vegetarian
1
₹ 105.0 INR
Cadbury Chocobakes Cake
Unit : 21 gms
Type : Vegetarian
1
₹ 9.0 INR
Britannia Treat Jim-Jam Cream Biscuits
Unit : 150 gms
Type : Vegetarian
1
₹ 25.0 INR

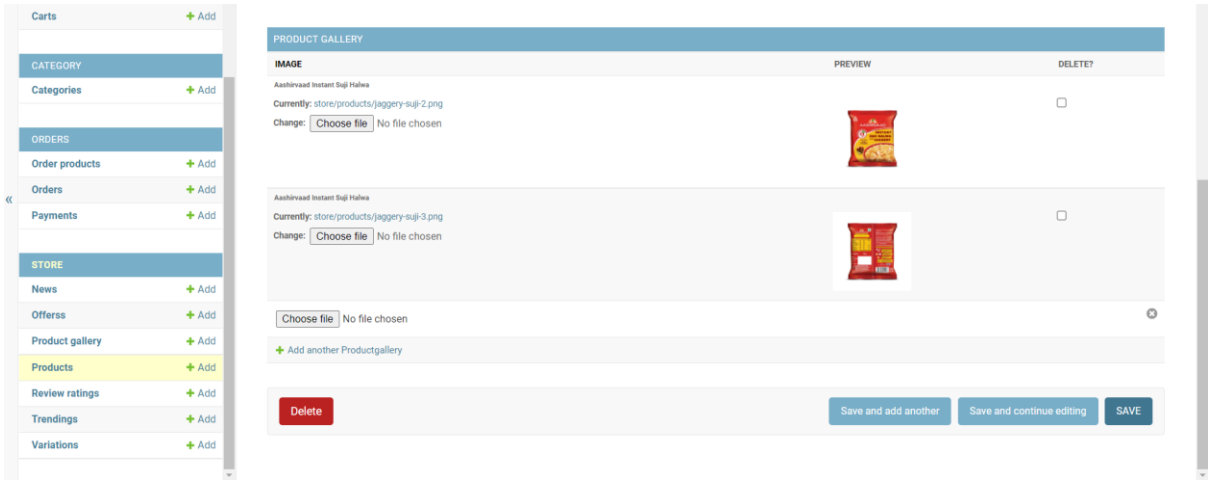

Home
About us
Great Deals
Daily Essentials
Offers
What's Trending
Contact
FAQs
BAZAAR
All category
Store
Search
Welcome Asavari!
Dashboard | Logout
Aashirvaad Select Sharbati Wheat Atta
Type : Vegetarian
Unit : 10 kg
1
₹ 245.0 INR
Everest Turmeric Powder
Unit : 200 gm
Type : Vegetarian
1
₹ 140.0 INR
Everest Tandoori Chicken Masala
Unit : 100 gm
Type : Vegetarian
1
₹ 66.0 INR
Darjeeling Wow Momo Veg & Non-veg
Type : Non-vegetarian
Unit : 20 pcs
1
₹ 126.0 INR
Darjeeling Wow Momo Veg & Non-veg
Unit : 10 pcs
Type : Vegetarian
1
₹ 126.0 INR
Brooke Bond Red Label Tea: 1 kg
1
₹ 445.0 INR

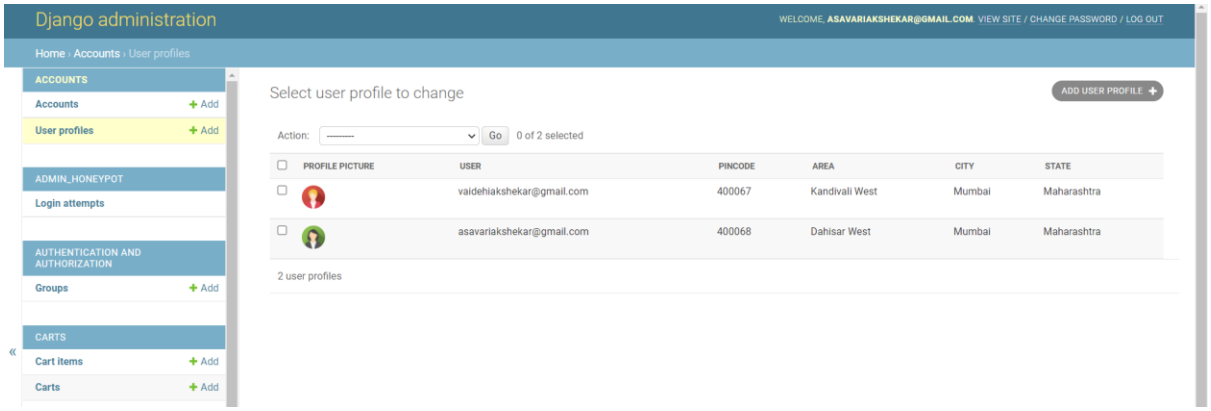

Sign in
You are logged out.
Email Address
Password
Forgot password?
Login
Don't have account? Sign up

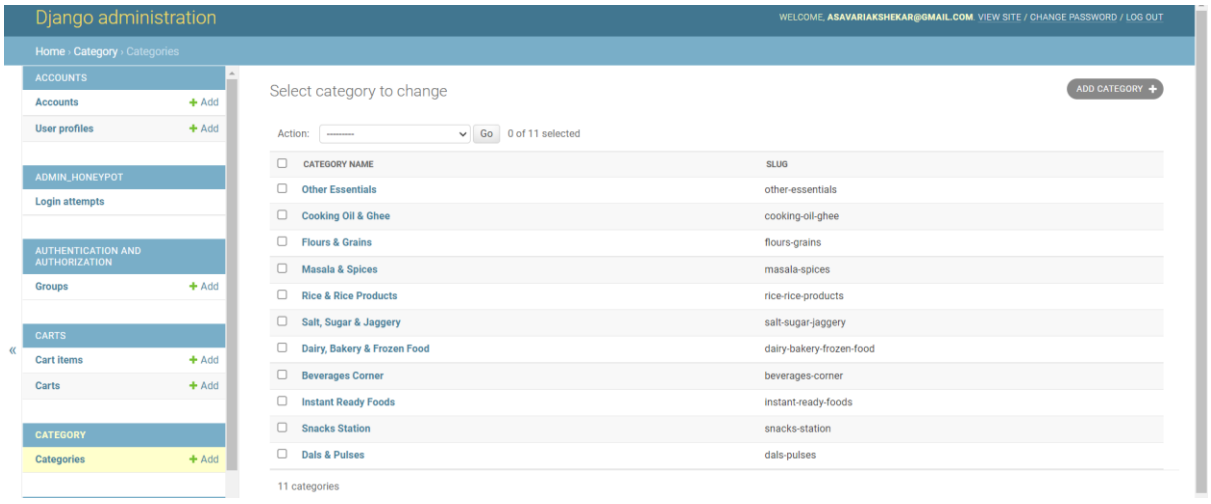

Django administration
Email:
asavariakshekar@gmail.com
Password:
Log in

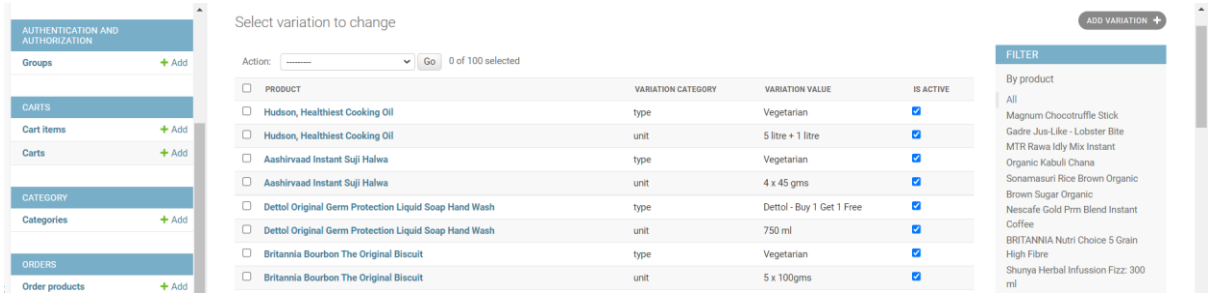

Django administration
WELCOME, ASAVARIAKSHEKAR@GMAIL.COM VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Store › Products
ACCOUNTS
Accounts + Add
User profiles + Add
ADMIN_HONEYPOT
Login attempts
AUTHENTICATION AND AUTHORIZATION
Groups + Add
CARTS
Cart items + Add
Carts + Add
CATEGORY
Categories + Add
ORDERS
Order products + Add
Select product to change
ADD PRODUCT +
Action: ---------- Go 0 of 89 selected
PRODUCT NAME | PRICE | STOCK | CATEGORY | MODIFIED DATE | IS AVAILABLE
Hudson, Healthiest Cooking Oil | 999 | 6 | Cooking Oil & Ghee | Nov. 24, 2021, 5:57 p.m.
Aashirvaad Instant Suji Halwa | 90 | 5 | Instant Ready Foods | Nov. 24, 2021, 5:50 p.m.
Dettol Original Germ Protection Liquid Soap Hand Wash | 203 | 2 | Other Essentials | Nov. 24, 2021, 5:47 p.m.
Britannia Bourbon The Original Biscuit | 80 | 12 | Snacks Station | Nov. 24, 2021, 5:57 p.m.
Vim Lemon Dishwash Bar | 44 | 5 | Other Essentials | Nov. 23, 2021, 2:35 p.m.
Exo Bacterial Ginger Twist Round Touch & Shine Dishwash Bar | 115 | 12 | Other Essentials | Nov. 23, 2021, 2:32 p.m.
Dabur 100% Pure Squeezy Honey | 140 | 15 | Salt, Sugar & Jaggery | Nov. 23, 2021, 2:56 p.m.
Tata Sampann Unpolished Toor Dal + Tata Tea Gold pouch free | 169 | 3 | Dals & Pulses | Nov. 23, 2021, 2:25 p.m.
Sunfeast Yippee Magic Masala Noodles | 115 | 12 | Snacks Station | Nov. 23, 2021, 2:21 p.m.
Maggi Masala Noodles- Brand Offer | 134 | 34 | Snacks Station | Nov. 23, 2021, 2:18 p.m.
24 Mantra Organic Wheat Organic Daliya | 48 | 2 | Rice & Rice Products | Nov. 21, 2021, 8:37 a.m.
Britannia Cheese Slices | 140 | 12 | Dairy, Bakery & Frozen Food | Nov. 19, 2021, 4:57 p.m.
Dabur 100% Pure Cow Ghee | 594 | 14 | Cooking Oil & Ghee | Nov. 21, 2021, 8:37 a.m.
FILTER
By category
All
Dals & Pulses
Snacks Station
Instant Ready Foods
Beverages Corner
Dairy, Bakery & Frozen Food
Salt, Sugar & Jaggery
Rice & Rice Products
Masala & Spices
Flours & Grains
Cooking Oil & Ghee
Other Essentials


Django administration
WELCOME, ASAVARIAKSHEKAR@GMAIL.COM VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Store › Products › Aashirvaad Instant Suji Halwa
ACCOUNTS
Accounts + Add
User profiles + Add
ADMIN_HONEYPOT
Login attempts
AUTHENTICATION AND AUTHORIZATION
Groups + Add
CARTS
Cart items + Add
Carts + Add
CATEGORY
Categories + Add
ORDERS
Order products + Add
Change product
HISTORY
Aashirvaad Instant Suji Halwa
Product name: Aashirvaad Instant Suji Halwa
Slug: aashirvaad-instant-suji-halwa
Description: Aashirvaad Instant Suji Halwa with Jaggery Ready to Eat - Buy 3 Get 1 Free.
Price: 90
Images: Currently: photos/products/jaggery-suji-1.png
Change: Choose file No file chosen
Stock: 5
Is available


Carts + Add
CATEGORY
Categories + Add
ORDERS
Order products + Add
Orders + Add
Payments + Add
STORE
News + Add
Offerss + Add
Product gallery + Add
Products + Add
Review ratings + Add
Trendings + Add
Variations + Add
PRODUCT GALLERY
IMAGE
PREVIEW
DELETE?
Aashirvaad Instant Suji Halwa
Currently: store/products/jaggery-suji-2.png
Change: Choose file No file chosen
Aashirvaad Instant Suji Halwa
Currently: store/products/jaggery-suji-3.png
Change: Choose file No file chosen
Choose file No file chosen
+ Add another Productgallery
Delete
Save and add another
Save and continue editing
SAVE


Django administration
WELCOME, ASAVARIAKSHEKAR@GMAIL.COM VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Accounts › User profiles
ACCOUNTS
Accounts + Add
User profiles + Add
ADMIN_HONEYPOT
Login attempts
AUTHENTICATION AND AUTHORIZATION
Groups + Add
CARTS
Cart items + Add
Carts + Add
Select user profile to change
ADD USER PROFILE +
Action: Go 0 of 2 selected
PROFILE PICTURE
USER
PINCODE
AREA
CITY
STATE
vaidehiakshekar@gmail.com 400067 Kandivali West Mumbai Maharashtra
asavariakshekar@gmail.com 400068 Dahisar West Mumbai Maharashtra
2 user profiles


Django administration
WELCOME, ASAVARIAKSHEKAR@GMAIL.COM VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Category › Categories
ACCOUNTS
Accounts + Add
User profiles + Add
ADMIN_HONEYPOT
Login attempts
AUTHENTICATION AND AUTHORIZATION
Groups + Add
CARTS
Cart items + Add
Carts + Add
CATEGORY
Categories + Add
Select category to change
ADD CATEGORY +
Action: Go 0 of 11 selected
CATEGORY NAME
SLUG
Other Essentials other-essentials
Cooking Oil & Ghee cooking-oil-ghee
Flours & Grains flours-grains
Masala & Spices masala-spices
Rice & Rice Products rice-rice-products
Salt, Sugar & Jaggery salt-sugar-jaggery
Dairy, Bakery & Frozen Food dairy-bakery-frozen-food
Beverages Corner beverages-corner
Instant Ready Foods instant-ready-foods
Snacks Station snacks-station
Dals & Pulses dals-pulses
11 categories


AUTHENTICATION AND AUTHORIZATION
Groups + Add
CARTS
Cart items + Add
Carts + Add
CATEGORY
Categories + Add
ORDERS
Order products + Add
Select variation to change
ADD VARIATION +
Action: Go 0 of 100 selected
PRODUCT
VARIATION CATEGORY
VARIATION VALUE
IS ACTIVE
Hudson, Healthiest Cooking Oil type Vegetarian
Hudson, Healthiest Cooking Oil unit 5 litre + 1 litre
Aashirvaad Instant Suji Halwa type Vegetarian
Aashirvaad Instant Suji Halwa unit 4 x 45 gms
Dettol Original Germ Protection Liquid Soap Hand Wash type Dettol - Buy 1 Get 1 Free
Dettol Original Germ Protection Liquid Soap Hand Wash unit 750 ml
Britannia Bourbon The Original Biscuit type Vegetarian
Britannia Bourbon The Original Biscuit unit 5 x 100gms
FILTER
By product
All
Magnum Chocotruffle Stick
Gadre Jus-Like - Lobster Bite
MTR Rawa Idly Mix Instant
Organic Kabuli Chana
Sonamasuri Rice Brown Organic
Brown Sugar Organic
Nescafe Gold Prm Blend Instant Coffee
BRITANNIA Nutri Choice 5 Grain High Fibre
Shunya Herbal Infussion Fizz: 300 ml

Coding

Frontend: templates

base.html

```
{% load static %}

<!DOCTYPE HTML>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="pragma" content="no-cache" />

<meta http-equiv="cache-control" content="max-age=604800" />

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<title>E-Bazaar | Online Grocery Store</title>

<link href="{% static 'images/icon.png' %}" rel="shortcut icon" type="image/x-icon">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.min.css" integrity="sha512-tS3S5qG0BlhnQROyJXvNjeEM4UpMXHrQfTGmbQ1gKmelCxlSEBUaxhRBj/EFTzpbP4RVSrpEikbmdJobCvhE3g==" crossorigin="anonymous" referrerpolicy="no-referrer" />

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.theme.default.min.css" integrity="sha512-sMXtMNL1zRzolHYKEujM2AqCLUR9F2C4/05cdbxjjLSRvMQIciEPCQZo++nk7go3BtSuK9kfa/s+a4f4i5pLkw==" crossorigin="anonymous" referrerpolicy="no-referrer" />
```

```
<!-- jQuery -->

<script src="{% static 'js/jquery-2.0.0.min.js' %}"
type="text/javascript"></script>

<!-- Bootstrap4 files-->

<script src="{% static 'js/bootstrap.bundle.min.js' %}"
type="text/javascript"></script>

<link href="{% static 'css/bootstrap.css' %}" rel="stylesheet" type="text/css"/>

<!-- Font awesome 5 -->

<link href="{% static 'fonts/fontawesome/css/all.min.css' %}" type="text/css"
rel="stylesheet">

<!-- custom style -->

<link href="{% static 'css/ui.css' %}" rel="stylesheet" type="text/css"/>

<link href="{% static 'css/responsive.css' %}" rel="stylesheet" media="only
screen and (max-width: 1200px)" />

<link href="{% static 'css/index.css' %}" rel="stylesheet" type="text/css"/>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

<!-- custom javascript -->

<script src="{% static 'js/script.js' %}" type="text/javascript"></script>

<!-- Paypal Script -->

<script src="https://www.paypal.com/sdk/js?client-
id=AdN4F0YYEA6v_dOaxxa__bx-ZC-eqy5h7BT2NQjOmhKlQD65mm4-
uFbICnw0P5eDL9cHM2xRf2Sn50vN&currency=USD"></script>

</head>

<body>

<!-- navbar -->

   {% include 'includes/navbar.html' %}
```

```
    {% block content %}
    <!-- content -->
    {% endblock %}
<!-- footer -->
    {% include 'includes/footer.html' %}
```

home.html

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<!-- ========================= SECTION MAIN
========================= -->
<script
src="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/owl.carousel.
min.js" integrity="sha512-
bPs7Ae6pVvhOSiIcyUClR7/q2OAsRiovw4vAkX+zJbw3ShAeeqezq50RIIcIURq7Oa
20rW2n2q+fyXBNcU9lrw==" crossorigin="anonymous" referrerpolicy="no-
referrer"></script>
<section class="section-intro padding-y-sm">
    <div class="container">
        <div class="intro-banner-wrap">
        <img src="{% static 'images/banners/cover.png' %}" class="img-fluid
rounded">
        </div>
    </div> <!-- container //  -->
</section>
<!-- ========================= SECTION MAIN END//
========================= -->
<section class="section-content ">
```

```
<div class="container">
<div class="row">
  <div class="col-md-4">
    <!-- ============================ COMPONENT ITEM BG
================================= -->
    <div class="shadow-sm card-banner"
style="background:#ffeb99;height:170px;">
      <div class="p-4 mt-3" style="width:75%;">
        <h5 class="card-title">Reasonable prices</h5>
        <p>Benefit from cool offers and discounts. So, when you buy more, you
also save more.</p>
      </div>
      <!-- <img src="../images/banners/banner-item1.jpg" height="150"
class="img-bg"> -->
          <span class="icon-lg rounded-circle bg-dark mt-5 mr-4">
                <i class="fas fa-wallet white"></i>
          </span>
    </div>
    <!-- ============================ COMPONENT ITEM BG  END .//
============================ -->
  </div> <!-- col.// -->
  <div class="col-md-4">
    <!-- ============================ COMPONENT ITEM BG
================================= -->
    <div class="shadow-sm card-banner"
style="background:#ccffcc;height:170px;">
      <div class="p-4 mt-3" style="width:75%">
        <h5 class="card-title">Quality</h5>
```

```
      <p>We strive to deliver quality and fresh products to our consumers.</p>
    </div>
        <span class="icon-lg rounded-circle bg-dark mt-5 mr-4">
                <i class="fas fa-medal white"></i>
        </span>
   </div>
  </div> <!-- col.// -->
  <div class="col-md-4">
   <div class="shadow-sm card-banner"
style="background:#b3ecff;height:170px;">
    <div class="p-4 mt-3" style="width:75%">
      <h5 class="card-title">Free delivery</h5>
      <p>Get quick & free home delivery on all your orders billed above INR
500.</p>
    </div>
        <span class="icon-lg rounded-circle bg-dark mt-5 mr-4">
                <i class="fa fa-truck white"></i>
        </span>
   </div>
  </div> <!-- col.// -->
</div> <!-- row.// -->
</div>
</section>
<section class="section-name py-3">
<div class="container">
<header class="section-heading">
```

```
<a href="{% url 'store' %}" class="btn btn-outline-primary float-right">See
all</a>

<h3 class="section-title">Popular Products</h3>

</header><!-- sect-heading -->

<div class="row">

{% for product in products %}

    <div class="col-md-3">

      <div class="card card-product-grid" style="height:94%;">

        <a href="{{ product.get_url }}" class="img-wrap"> <img src="{{
product.images.url }}"> </a>

        <figcaption class="info-wrap">

          <a href="{{ product.get_url }}" class="title" style="font-
size:16px;"><b>{{ product.product_name }}</b></a>

          {% if product.id == 1 %}

            <span class="badge badge-warning mt-1" style="font-size:13px;">
Limited Edition </span>

          {% elif product.id == 7 or product.id == 14 %}

            <span class="badge badge-danger mt-1" style="font-
size:12px;">New</span>

          {% endif %}

          <div class="price mt-1 text-dark">₹ {{ product.price }}</div> <!--
price-wrap.// -->

        </figcaption>

      </div>

    </div> <!-- col.// -->

{% endfor %}

</div> <!-- row.// -->

<div class="intro-banner-wrap py-2">
```

```
   <img src="{% static 'images/banners/deal.png' %}" class="img-fluid
rounded">

</div>

<header class="section-heading py-1" id="Every-Day">

<a href="{% url 'store' %}" class="btn btn-outline-primary float-right">See
all</a>

<h3 class="section-title">Everyday Essentials</h3>

</header><!-- sect-heading -->

<div class="row">

{% for product in all_products %}

      {% if product.id == 87 or product.id == 78 or product.id == 74 or product.id
== 22 or product.id == 49 or product.id == 85 or product.id == 30 or product.id
== 76 %}

      <div class="col-md-3">

         <div class="card card-product-grid" style="height:92%;">

            <a href="{{ product.get_url }}" class="img-wrap"> <img src="{{
product.images.url }}"> </a>

            <figcaption class="info-wrap">

               <a href="{{ product.get_url }}" class="title" style="font-
size:15.5px;"><b>{{ product.product_name }}</b></a>

               {% if product.id == 85 %}

                  <span class="badge badge-success mt-1" style="font-size:12px;">
12% Off </span>

               {% elif product.id == 49 %}

                  <span class="badge badge-success mt-1" style="font-size:12px;">
10% Off </span>

               {% elif product.id == 22 %}

                  <span class="badge badge-success mt-1" style="font-size:12px;">
20% Off </span>
```

```
                {% else %}
                  <span></span>
                {% endif %}
                <div class="rating-star" style="font-size:17px;">
                  <span>
                    <i class="fa fa-star{% if product.averageReview < 0.5 %}-o{% elif product.averageReview >= 0.5 and product.averageReview < 1 %}-half-o {% endif %}" aria-hidden="true"></i>
                    <i class="fa fa-star{% if product.averageReview < 1.5 %}-o{% elif product.averageReview >= 1.5 and product.averageReview < 2 %}-half-o {% endif %}" aria-hidden="true"></i>
                    <i class="fa fa-star{% if product.averageReview < 2.5 %}-o{% elif product.averageReview >= 2.5 and product.averageReview < 3 %}-half-o {% endif %}" aria-hidden="true"></i>
                    <i class="fa fa-star{% if product.averageReview < 3.5 %}-o{% elif product.averageReview >= 3.5 and product.averageReview < 4 %}-half-o {% endif %}" aria-hidden="true"></i>
                    <i class="fa fa-star{% if product.averageReview < 4.5 %}-o{% elif product.averageReview >= 4.5 and product.averageReview < 5 %}-half-o {% endif %}" aria-hidden="true"></i>
                  </span>
                </div>
                <div class="price mt-1 text-dark">₹ {{ product.price }}</div> <!-- price-wrap.// -->
              </figcaption>
            </div>
          </div> <!-- col.// -->
        {% endif %}
{% endfor %}
</div> <!-- row.// -->
```

```
<h3 class="section-title" id="Great-Deals">Great Deals on Products</h3>
<div class="slider-items-owl owl-carousel owl-theme py-4">
{% for d in deals %}
{% if forloop.counter <= 10 %}
<div class="item-slide">
<figure class="card card-product-grid" style="height:310px;">
   <div class="img-wrap">
      <a href="{{ d.get_url }}"><img src="{{ d.images.url }}"></a>
   </div>
   <figcaption class="info-wrap ">
      <h6 class="title"><a href="{{ d.get_url }}" class="text-secondary">{{
d.product_name }}</a></h6>
      {% if d.id == 94 %}
         <span class="badge badge-success ml-1" style="font-size:12px;">10%
Off</span>
      {% elif d.id == 93 %}
         <span class="badge badge-success ml-1" style="font-size:12px;">16%
Off</span>
      {% elif d.id == 92 %}
         <span class="badge badge-success ml-1" style="font-size:12px;">24%
Off</span>
      {% elif d.id == 90 %}
         <span class="badge badge-success ml-1" style="font-size:12px;">20%
Off</span>
      {% elif d.id == 89 %}
         <span class="badge badge-success ml-1" style="font-size:12px;">6%
Off</span>
      {% elif d.id == 98 %}
```

```
        <span class="badge badge-success ml-1" style="font-size:12px;">16% Off</span>
      {% elif d.id == 97 %}
        <span class="badge badge-success ml-1" style="font-size:12px;">25% Off</span>
      {% endif %}
   </figcaption>
</figure> <!-- card // -->
</div>
{% endif %}
{% endfor %}
</div>
<div class="intro-banner-wrap py-0">
<img src="{% static 'images/banners/diwali.png' %}" class="img-fluid rounded">
</div>
</div><!-- container // -->
</section>
<section class="section-name py-3">
  <div class="container" id="Offers">
        <h3 class="section-title">Best Offers on Products</h3>
       <div class="row g-4 py-3"> <!--Row with gap 4-->
              {% for product in offers %}
          <div class="col-md-6 col-lg-3">
            <div class="card">
              <div class="card-body" style="background-color:#FAF2E9;">
```

```
                    <img src="{{ product.images.url }}" class="mb-3"
style="width:100%;" >
                                    <figcaption class="info-wrap">
                                        <a href="{{ product.get_url }}" class="title text-
dark" id="unique" style="font-weight:bold;">{{ product.product_name }}</a>
                                        <div class="price mt-1">₹ {{ product.final_price
}}</div> <!-- price-wrap.// -->
                                        MRP <del  class="text-secondary">₹ {{
product.initial_price }}</del>
                                    </figcaption>


                                    <a href="{{ product.get_url }}" class="btn btn-
block btn-success mt-3">Visit Product</a>
                </div>
              </div>
            </div>
                {% endfor %}
        </div>
         <div class="py-3">
            <div id="carouselExampleControls" class="carousel slide img-fluid
rounded" data-ride="carousel" style="width:95%;margin-left:5%;">
                <div class="carousel-inner">
                    <div class="carousel-item active">
                        <a href="{% url 'store' %}"><img class="d-block w-100"
src="{% static 'images/carousel/s1.png' %}" alt="First slide"></a>
                    </div>
                    <div class="carousel-item">
                        <a href="{% url 'store' %}"><img class="d-block w-100"
src="{% static 'images/carousel/s4.png' %}" alt="Second slide"></a>
```

```
                </div>
                <div class="carousel-item">
                  <a href="{% url 'store' %}"><img class="d-block w-100"
src="{% static 'images/carousel/s3.png' %}" alt="Third slide"></a>
                </div>
              </div>
              <a class="carousel-control-prev"
href="#carouselExampleControls" role="button" data-slide="prev">
                <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
                <span class="sr-only">Previous</span>
              </a>
              <a class="carousel-control-next"
href="#carouselExampleControls" role="button" data-slide="next">
                <span class="carousel-control-next-icon" aria-
hidden="true"></span>
                <span class="sr-only">Next</span>
              </a>
            </div>
          </div>
          <section class="section-name py-0" id="Trending">
            <header class="section-heading">
            <h3 class="section-title">What's Trending</h3>
            </header><!-- sect-heading -->
            <div class="slider-items-owl owl-carousel owl-theme py-1">
              {% for n in new %}
              <div class="item-slide">
              <figure class="card card-product-grid">
```

```
                <div class="img-wrap">
                  {% if forloop.counter == 2 or forloop.counter == 3 or
forloop.counter == 6 %}
                  <span class="badge badge-danger" style="font-size:12px;">
New </span>
                  <a href="{{ n.get_url }}"><img src="{{ n.images.url }}"></a>
                  {% else %}
                  <a href="{{ n.get_url }}"><img src="{{ n.images.url }}"></a>
                  {% endif %}
                </div>
                <figcaption class="info-wrap ">
                  <h6 class="title text-center"><a href="{{ n.get_url }}"
class="text-secondary text-lg"  style="font-size:16px;">{{ n.product_name
}}</a></h6>
                </figcaption>
              </figure> <!-- card // -->
              </div>
              {% endfor %}
            </div>
          </section>
          <section class="section-name mt-4">
                <div class="container" id="About" style="border:1px solid
grey;border-radius: 5px; padding:15px 18px 0 18px;">
                  <h3>E-Bazaar -Online Grocery Store</h3>
 <p>Did you ever imagine that the freshest of fruits and vegetables, top quality
pulses and food grains, dairy products and hundreds of branded items could be
handpicked and delivered to your home, all at the click of a button?
<br>
```

```
 <a class="btn btn-primary btn-sm mt-3" id="hide-button" data-bs-toggle="collapse" href="#collapseExample" role="button" aria-expanded="false" aria-controls="collapseExample">
 Read More...
 </a>
 </p>
 <div class="collapse mb-3" id="collapseExample">
 <p> Best quality products for our quality-conscious customers.</p>
 <p>E-Bazaar.com believes in providing the highest level of customer service and is continuously innovating to meet customer expectations. </p>
                    </div>
                </div>
            </section>
        </div>
</section>
<script type="text/javascript">
   if ($('.slider-items-owl').length > 0) { // check if element exists
                $('.slider-items-owl').owlCarousel({
                    loop:true,
                    margin:10,
                    nav:true,
                    navText: ["<i class='fa fa-chevron-left'></i>", "<i class='fa fa-chevron-right'></i>"],
                    responsive:{
                        0:{
                            items:1
                        },
```

```
                640:{
                    items:3
                },
                1024:{
                    items:4
                }
            }
        })
    }
</script>
</body>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-kQtW33rZJAHjgefvhyyzcGF3C5TFyBQBA13V1RKPf4uH+bwyzQxZ6CmMZHmNBEfJ" crossorigin="anonymous"></script>
</html>
{% endblock %}
```

navbar.html

```
{% load static %}
<header class="section-header ">
<nav class="navbar p-md-0 navbar-expand-sm navbar-light border-bottom " style="background-color: #f8f9fa;">
<div class="container">
    <div class="nav" >
        <ul class="nav mobile-hide">
```

```
            <li><a href="{% url 'home' %}" class="nav-link font-weight-
bold">Home</a></li>

            <li><a href="#About" class="nav-link font-weight-bold">About
us</a></li>

            <li><a href="#Great-Deals" class="nav-link font-weight-bold">Great
Deals</a></li>

            <li><a href="#Every-Day" class="nav-link font-weight-bold">Daily
Essentials</a></li>

            <li><a href="#Offers" class="nav-link font-weight-
bold">Offers</a></li>

            <li><a href="#Trending" class="nav-link font-weight-bold">What's
Trending</a></li>

            <li><a href="#" class="nav-link font-weight-bold">Contact</a></li>

            <li><a href="{% url 'faq' %}" class="nav-link font-weight-
bold">FAQs</a></li>

          </ul> <!-- list-inline //  -->

    </div> <!-- container //  -->

    <div class="nav">

          <ul class="nav">

            <li><a href="#"><i class="fab fa-facebook " style="font-
size:22px;color:#707070;"></i></a></li>

            <li><a href="#"><i class="fab fa-instagram ml-4" style="font-
size:22px;color:#707070;"></i></a></li>

          </ul> <!-- list-inline //  -->

    </div> <!-- container //  -->

</nav>

<section class="header-main border-bottom py-4">

  <div class="container">

<div class="row align-items-center">
```

```
 <div class="col-lg-2 col-md-3 col-6">
  <a href="{% url 'home' %}" class="brand-wrap">
   <img class="logo" src="{% static './images/fi-logo.png' %}">
  </a> <!-- brand-wrap.// -->
 </div>
 <div class="col-lg col-sm col-md col-6 flex-grow-0">
  <div class="category-wrap dropdown d-inline-block float-right">
   <button type="button" class="btn btn-primary dropdown-toggle" data-
toggle="dropdown" style="width:170px;">
    <i class="fa fa-bars"></i>  All category
   </button>
   <div class="dropdown-menu">
    <a class="dropdown-item" href="{% url 'store' %}">All Products</a>
    {% for category in links %}
     <a class="dropdown-item" href="{{category.get_url}}">{{
category.category_name }} </a>
    {% endfor %}
   </div>
  </div>  <!-- category-wrap.// -->
 </div> <!-- col.// -->
 <a href="{% url 'store' %}" class="btn btn-outline-primary">Store</a>
  <div class="col-lg  col-md-6 col-sm-12 col">
   <form action="{% url 'search' %}" class="search" method="GET">
    <div class="input-group w-100">
       <input type="text" class="form-control" placeholder="Search"
name="keyword" style="width:80%;">
```

```
          <div class="input-group-append">
            <button class="btn btn-primary" type="submit">
              <i class="fa fa-search"></i>
            </button>
          </div>
        </div>
    </form> <!-- search-wrap .end// -->
  </div> <!-- col.// -->
 <div class="col-lg-3 col-sm-6 col-8 order-2 order-lg-3">
      <div class="d-flex justify-content-end mb-3 mb-lg-0">
        {% if user.id is None %}
          <div class="widget-header">
            <span class="title text-muted text-center">Welcome guest!</span>
            <div>
              <a href="{% url 'login' %}">Sign in</a> <span class="dark-trans"> |
</span>
              <a href="{% url 'register' %}"> Register</a>
            </div>
          </div>
        {% else %}
          <div class="widget-header">
            <span class="title  text-muted">Welcome {{ user.first_name }}!</span>
            <div>
              <a href="{% url 'dashboard' %}">Dashboard</a> <span class="dark-
transp"> | </span>
              <a href="{% url 'logout' %}"> Logout</a>
            </div>
```

```
          </div>
        {% endif %}
          <a href="{% url 'cart' %}" class="widget-header pl-3 ml-3">
           <div class="icon icon-sm rounded-circle border"><i class="fa fa-shopping-cart" ></i></div>
           <span class="badge badge-pill badge-danger notify">{{ cart_count }}</span>
          </a>
      </div> <!-- widgets-wrap.// -->
    </div> <!-- col.// -->
</div> <!-- row.// -->
</div> <!-- container.// -->
</section> <!-- header-main .// -->
</header> <!-- section-header.// -->
```

footer.html

```
<footer class="mt-5 p-5  border-top  text-center position-relative" style="background-color:#f8f9fa;height:250px;">
       <div class="container">
          <div class="row">
             <div class="col-12 col-md" style="font-size:30px;">
               <h5>Get Social With Us</h5>
               <i class="fab fa-facebook"></i>
               <i class="fab fa-instagram"></i>
               <i class="fab fa-twitter-square"></i>
             </div>
             <div class="col-6 col-md">
```

```
        <h5>Information</h5>
        <ul class="list-unstyled text-small">
          <li><a class="text-muted" href="#">Terms & Conditions</a></li>
          <li><a class="text-muted" href="{% url 'faq' %}">FAQ</a></li>
          <li><a class="text-muted" href="#About">About Us</a></li>
        </ul>
      </div>
      <div class="col-6 col-md">
        <h5>Customer Service</h5>
        <ul class="list-unstyled text-small">
          <li><a class="text-muted" href="#">Contact Us</a></li>
          <li><a class="text-muted" href="#">Returns</a></li>
          <li><a class="text-muted" href="#">Sitemap</a></li>
        </ul>
      </div>
      <div class="col-6 col-md" style="font-size:25px;">
        <h5>Payment Options</h5>
        <i class="fab fa-lg fa-cc-visa"></i>
        <i class="fab fa-lg fa-cc-paypal"></i>
        <i class="fab fa-lg fa-cc-mastercard"></i>
      </div>
    </div>
    <hr>
    <div class="text-center">
```

```
<a href="#" class="position-absolute bottom-0 end-0 text-secondary"
style="margin-left:45%;">
<i class="fas fa-arrow-circle-up" style="font-size:38px;"></i>
            </a>
            <p class="text-dark">Copyright &copy; 2021 E-Bazaar, Inc. All rights
reserved.</p>
          <div>
       </div>
 </footer>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
<script>
   $(document).ready(function(){
            $('.thumb a').click(function(e){
                  e.preventDefault();
                  $('.mainImage img').attr('src', $(this).attr("href"));
            })
   })
   $(document).ready(function(){
         $("#hide-button").click(function(){
            $("#hide-button").hide();
      });
   })
</script>
</body>
</html>
```

Backend: greatstore (app)

settings.py

```
from pathlib import Path
from decouple import config
BASE_DIR = Path(__file__).resolve(strict=True).parent.parent
SECRET_KEY = config('SECRET_KEY')
DEBUG = config('DEBUG', default=True, cast=bool)
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'category',
    'accounts',
    'store',
    'carts',
    'orders',
    'admin_honeypot',
]
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',
    'django_session_timeout.middleware.SessionTimeoutMiddleware',
]
SESSION_EXPIRE_SECONDS = 3600  # 1 hour = 3600
SESSION_EXPIRE_AFTER_LAST_ACTIVITY = True
SESSION_TIMEOUT_REDIRECT = 'accounts/login'
ROOT_URLCONF = 'greatstore.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        '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',
                'category.context_processors.menu_links',
                'carts.context_processors.counter',
            ],
```

```
        },
    },
]
WSGI_APPLICATION = 'greatstore.wsgi.application'
AUTH_USER_MODEL = 'accounts.Account'
#Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
DEFAULT_AUTO_FIELD='django.db.models.AutoField'
# Password validation
# https://docs.djangoproject.com/en/3.1/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/3.1/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/
STATIC_URL = '/static/'
STATIC_ROOT = BASE_DIR /'static'
STATICFILES_DIRS = [
    'greatstore/static',
]
#media files configuration
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR /'media'
from django.contrib.messages import constants as messages
MESSAGE_TAGS = {
```

```
    messages.ERROR: 'danger',
}
#SMTP configuration
EMAIL_HOST = config('EMAIL_HOST')
EMAIL_PORT = config('EMAIL_PORT', cast=int)
EMAIL_HOST_USER = config('EMAIL_HOST_USER')
EMAIL_HOST_PASSWORD = config('EMAIL_HOST_PASSWORD')
EMAIL_USE_TLS = config('EMAIL_USE_TLS', cast=bool)
```

urls.py

```
from django.contrib import admin
from django.urls import path, include
from .import views
from django.conf.urls.static import static
from django.conf import settings
urlpatterns = [  path('admin/', include('admin_honeypot.urls',
namespace='admin_honeypot')),
    path('securelogin/', admin.site.urls),
    path('', views.home, name='home'),
    path('faq/', views.faq, name='faq'),
    path('store/', include('store.urls')),
    path('cart/', include('carts.urls')),
    path('accounts/', include('accounts.urls')),
    #orders
    path('orders/', include('orders.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py

```
from django.shortcuts import render

from store.models import Offers

from store.models import Trending, ReviewRating, New, Product

def home(request):

   products = Trending.objects.all().filter(is_available=True).order_by('-
modified_date')

   offers   = Offers.objects.all().filter(is_available=True)

   new = New.objects.all().filter(is_available=True).order_by('-created_date')

   all_products = Product.objects.all().filter(is_available=True).order_by('-
created_date')

   deals = Product.objects.all().filter(is_available=True).order_by('-
created_date')

   reviews = None

   for product in products:

      reviews = ReviewRating.objects.filter(product_id=product.id, status=True)

   context = {

      'reviews'  : reviews,

      'products' : products,

      'offers'   : offers,

      'new'      : new,

      'all_products' : all_products,

      'deals' : deals,

   }

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

def faq(request):

return render(request, 'faq.html')
```

accounts (app):

models.py

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager
# Create your models here.
class MyAccountManager(BaseUserManager):
   def create_user(self, first_name, last_name, username, email, password=None):
      if not email:
         raise ValueError('User must have an email address')
      if not username:
         raise ValueError('User must have an username')
      user = self.model(
         email = self.normalize_email(email),
         username = username,
         first_name = first_name,
         last_name = last_name,
      )
      user.set_password(password)
      user.save(using=self._db)
      return user
   def create_superuser(self, first_name, last_name, email, username, password):
      user = self.create_user(
         email = self.normalize_email(email),
         username = username,
         password = password,
```

```
            first_name = first_name,
            last_name = last_name,
        )
        user.is_admin = True
        user.is_active = True
        user.is_staff = True
        user.is_superadmin = True
        user.save(using=self._db)
        return user
class Account(AbstractBaseUser):
    first_name      = models.CharField(max_length=50)
    last_name       = models.CharField(max_length=50)
    username        = models.CharField(max_length=50, unique=True)
    email           = models.EmailField(max_length=100, unique=True)
    phone_number    = models.CharField(max_length=50)
    # required
    date_joined     = models.DateTimeField(auto_now_add=True)
    last_login      = models.DateTimeField(auto_now_add=True)
    is_admin        = models.BooleanField(default=False)
    is_staff        = models.BooleanField(default=False)
    is_active       = models.BooleanField(default=False)
    is_superadmin       = models.BooleanField(default=False)
    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['username', 'first_name', 'last_name']
    objects = MyAccountManager()
```

```
 def full_name(self):
      return f'{self.first_name} {self.last_name}'
   def __str__(self):
      return self.email
   def has_perm(self, perm, obj=None):
      return self.is_admin
   def has_module_perms(self, add_label):
      return True
class UserProfile(models.Model):
   user = models.OneToOneField(Account, on_delete=models.CASCADE)
   address_line_1 = models.CharField(blank=True, max_length=100)
   address_line_2 = models.CharField(blank=True, max_length=100)
   profile_picture = models.ImageField(blank=True, upload_to='images/users/')
   pincode = models.CharField(blank=True, max_length=20)
   area = models.CharField(blank=True, max_length=20)
   city = models.CharField(blank=True,  max_length=20)
   state = models.CharField(blank=True, max_length=20)
   phone_number = models.CharField(max_length=50)
   def __str__(self):
      return self.user.first_name
   def full_address(self):
      return f'{self.address_line_1} {self.address_line_2} {self.area} {self.city}
{self.pincode}'
```

views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django import forms
from .forms import RegistrationForm, UserForm, UserProfileForm
from .models import Account, UserProfile
from orders.models import Order, OrderProduct
from django.contrib import messages, auth
from django.contrib.auth import authenticate, login
from django.contrib.auth.decorators import login_required
from django.http import HttpResponse, HttpResponseRedirect
from store.models import Product
from django.core.exceptions import ValidationError
import re
#Verification Email
from django.contrib.sites.shortcuts import get_current_site
from django.template.loader import render_to_string
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
from django.utils.encoding import force_bytes
from django.contrib.auth.tokens import default_token_generator
from django.core.mail import EmailMessage
from carts.views import _cart_id
from carts.models import Cart, CartItem
import requests
# Create your views here.
def register(request):                #creating a user
    if request.method == 'POST':
```

```
        form = RegistrationForm(request.POST)
        if form.is_valid():
            first_name =  form.cleaned_data['first_name'] #cleaned_data used to fetch data
            last_name =  form.cleaned_data['last_name']
            phone_number =  form.cleaned_data['phone_number']
            email =  form.cleaned_data['email']
            password =  form.cleaned_data['password']
            username = email.split("@")[0]
            user = Account.objects.create_user(first_name = first_name, last_name = last_name, email = email, username = username, password = password)
            user.phone_number = phone_number
            user.save()
             # Create a user profile
            profile = UserProfile()
            profile.user_id = user.id
            profile.profile_picture = 'default/default-user.png'
            profile.save()
            #USER ACTIVATION
            current_site = get_current_site(request)
            mail_subject = 'Please activate your account'
            message = render_to_string('accounts/account_verification_email.html', {
                'user': user,
                'domain': current_site,
                'uid': urlsafe_base64_encode(force_bytes(user.pk)),
                'token': default_token_generator.make_token(user),
```

```
            })
            to_email = email
            send_email = EmailMessage(mail_subject, message, to=[to_email])
            send_email.send()
            # messages.success(request, 'Thank you for registering with us. We have sent ypu a verifivation mail to your email address. Please verify it. ')
            return redirect('/accounts/login/?command=verification&email='+email)
    else:
        form = RegistrationForm()
    context = {
        'form' : form,
    }
    return render(request, 'accounts/register.html', context)
def login(request):
    if request.method == 'POST':
        email = request.POST['email']
        password = request.POST['password']
        user = auth.authenticate(email=email, password=password)
        if user is not None:
            try:
                cart = Cart.objects.get(cart_id=_cart_id(request))
                is_cart_item_exists = CartItem.objects.filter(cart=cart).exists()
                if is_cart_item_exists:
                    cart_item = CartItem.objects.filter(cart=cart)
                    #Getting the product variation by cart_id
                    product_variation = []
```

```
for item in cart_item:
    variation = item.variations.all()
    product_variation.append(list(variation))
#Get the cart items from the user to access his product_variation
cart_item = CartItem.objects.filter(user=user)
ex_var_list = []
id = []
for item in cart_item:
    existing_variation = item.variations.all()
    ex_var_list.append(list(existing_variation))
    id.append(item.id)
# product_variation = [1, 2, 3, 4, 6]
# ex_var_list = [4, 6, 3, 5]
for pr in product_variation:
    if pr in ex_var_list:
        index = ex_var_list.index(pr)
        item_id = id[index]
        item = CartItem.objects.get(id=item_id)
        item.quantity += 1
        item.user = user
        item.save()
    else:
        cart_item = CartItem.objects.filter(cart=cart)
        for item in cart_item:
            item.user = user
            item.save()
```

```
            except:
                print('entering inside except block')
                pass
            auth.login(request, user)
            messages.success(request, 'You are now logged in.')
            url = request.META.get('HTTP_REFERER')
            try:
                query = requests.utils.urlparse(url).query
                # next = /cart/checkout/
                params = dict(x.split('=') for x in query.split('&'))
                if 'next' in params:
                    nextPage = params['next']
                    return redirect(nextPage)
            except:
                return redirect('dashboard')
        else:
            messages.error(request, 'Invalid login credentials')
            return redirect('login')
    return render(request, 'accounts/login.html')
@login_required(login_url = 'login')
def logout(request):
    auth.logout(request)
    messages.success(request, 'You are logged out.')
    return redirect('login')
def activate(request, uidb64, token): #decode the token
    try:
```

```
        uid = urlsafe_base64_decode(uidb64).decode()
        user = Account._default_manager.get(pk=uid)
    except(TypeError, ValueError, OverflowError, Account.DoesNotExist):
        user = None
    if user is not None and default_token_generator.check_token(user, token):
        user.is_active = True
        user.save()
        messages.success(request, 'Congratulations! Your account is activated.')
        return redirect('login')
    else:
        messages.error(request, 'Invalid activation link')
        return redirect('register')
@login_required(login_url = 'login')
def dashboard(request):
    orders = Order.objects.order_by('-created_at').filter(user_id=request.user.id,
is_ordered=True)
    orders_count = orders.count()
    userprofile = UserProfile.objects.get(user_id=request.user.id)
    context = {
        'orders_count': orders_count,
        'userprofile': userprofile,
    }
    return render(request, 'accounts/dashboard.html', context)
def forgotPassword(request):
    if request.method == 'POST':
        email = request.POST['email']
        if Account.objects.filter(email=email).exists():
```

```
            user = Account.objects.get(email__exact=email)


            # Reset password email
            current_site = get_current_site(request)
            mail_subject = 'Reset Your Password'
            message = render_to_string('accounts/reset_password_email.html', {
                'user': user,
                'domain': current_site,
                'uid': urlsafe_base64_encode(force_bytes(user.pk)),
                'token': default_token_generator.make_token(user),
            })
            to_email = email
            send_email = EmailMessage(mail_subject, message, to=[to_email])
            send_email.send()
            messages.success(request, 'Password reset email has been sent to your email address.')
            return redirect('login')
        else:
            messages.error(request, 'Account does not exist!')
            return redirect('forgotPassword')
    return render(request, 'accounts/forgotPassword.html')
def resetpassword_validate(request, uidb64, token):
    try:
        uid = urlsafe_base64_decode(uidb64).decode()
        user = Account._default_manager.get(pk=uid)
    except(TypeError, ValueError, OverflowError, Account.DoesNotExist):
        user = None
```

```
    if user is not None and default_token_generator.check_token(user, token):
        request.session['uid'] = uid
        messages.success(request, 'Please reset your password')
        return redirect('resetPassword')
    else:
        messages.error(request, 'This link has been expired!')
        return redirect('login')
def resetPassword(request):
    if request.method == 'POST':
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']
        if password == confirm_password:
            reg = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-
z\d@$!#%*?&]{8,15}$"
            # compiling regex
            pat = re.compile(reg)
            # searching regex
            mat = re.search(pat, password)
            if not mat:
                messages.error(request, 'The password does not meet the password
policy requirements.')
                return redirect('resetPassword')
            else:
                uid = request.session.get('uid')
                user = Account.objects.get(pk=uid)
                user.set_password(password)
```

```
            user.save()
            messages.success(request, 'Password reset successfully')
            return redirect('login')
        else:
            messages.error(request, 'Password does not match!')
            return redirect('resetPassword')
    else:
        return render(request, 'accounts/resetPassword.html')
@login_required(login_url='login')
def my_orders(request):
    orders = Order.objects.filter(user=request.user,
is_ordered=True).order_by('-created_at')
    context = {
        'orders' : orders,
    }
    return render(request, 'accounts/my_orders.html', context)
@login_required(login_url='login')
def edit_profile(request):
    userprofile = get_object_or_404(UserProfile, user=request.user)
    if request.method == 'POST':
        user_form = UserForm(request.POST, instance=request.user)
        profile_form = UserProfileForm(request.POST, request.FILES,
instance=userprofile)
        if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
            messages.success(request, 'Your profile has been updated.')
```

```
            return redirect('edit_profile')
        else:
            user_form = UserForm(instance=request.user)
            profile_form = UserProfileForm(instance=userprofile)
        context = {
            'user_form': user_form,
            'profile_form': profile_form,
            'userprofile': userprofile,
        }
        return render(request, 'accounts/edit_profile.html', context)
@login_required(login_url='login')
def change_password(request):
    if request.method == 'POST':
        current_password = request.POST['current_password']
        new_password = request.POST['new_password']
        confirm_password = request.POST['confirm_password']
        user = Account.objects.get(username__exact=request.user.username)
        if new_password == confirm_password:
            success = user.check_password(current_password)
            if success:
                reg = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-
z\d@$!#%*?&]{8,15}$"
                # compiling regex
                pat = re.compile(reg)
                # searching regex
                mat = re.search(pat, new_password)
```

```
            if not mat:
                messages.error(request, 'The password does not meet the
password policy requirements.')
            else:
                user.set_password(new_password)
                user.save()
                # auth.logout(request)
                messages.success(request, 'Password updated successfully.')
                return redirect('change_password')
        else:
            messages.error(request, 'Please enter valid current password')
            return redirect('change_password')
    else:
        messages.error(request, 'Password does not match!')
        return redirect('change_password')
  return render(request, 'accounts/change_password.html')
@login_required(login_url='login')
def order_detail(request, order_id):
  order_detail = OrderProduct.objects.filter(order__order_number=order_id)
  order = Order.objects.get(order_number=order_id)
  subtotal = 0
  for i in order_detail:
    subtotal += i.product_price * i.quantity
  context = {
    'order_detail': order_detail,
    'order': order,
    'subtotal': subtotal,
```

```
    }
    return render(request, 'accounts/order_detail.html', context)
@login_required(login_url='login')
def my_list(request):
    orders = Order.objects.filter(user=request.user,
is_ordered=True).order_by('-created_at')
    context = {
        'orders' : orders,
    }
    return render(request, 'accounts/my_list.html', context)
@login_required(login_url='login')
def my_list_detail(request, order_id):
    order_detail = OrderProduct.objects.filter(order__order_number=order_id)
    order = Order.objects.get(order_number=order_id)
    context = {
        'order_detail': order_detail,
        'order': order,
    }
    return render(request, 'accounts/my_list_detail.html', context)
```

admin.py

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .models import Account, UserProfile
from django.utils.html import format_html
# Register your models here.
```

```
class AccountAdmin(UserAdmin):
   list_display = ('email', 'first_name', 'last_name', 'username', 'last_login',
'date_joined', 'is_active')
   list_display_links = ('email', 'first_name', 'last_name')
   readonly_fields = ('last_login', 'date_joined')
   ordering = ('-date_joined',)
   filter_horizontal = ()
   list_filter = ()
   fieldsets = ()
class UserProfileAdmin(admin.ModelAdmin):
   def thumbnail(self, object):
      return format_html('<img src="{}" width="30" style="border-
radius:50%;">'.format(object.profile_picture.url))
   thumbnail.short_description = 'Profile Picture'
   list_display = ('thumbnail','user','pincode','area','city','state')


admin.site.register(Account, AccountAdmin)
admin.site.register(UserProfile, UserProfileAdmin)
```

store (app):

models.py

```
from django.db import models
from django.conf import settings
from category.models import Category
from django.urls import reverse
from accounts.models import Account
from django.db.models import Avg, Count
```

```
# Create your models here.


class Product(models.Model):                    #Store Page
    product_name    = models.CharField(max_length=200, unique=True)
    slug            = models.SlugField(max_length=200, unique=True)
    description     = models.TextField(max_length=500, blank=True)
    price           = models.IntegerField()
    images          = models.ImageField(upload_to='photos/products')
    stock           = models.IntegerField()
    is_available    = models.BooleanField(default=True)
    category        = models.ForeignKey(Category, on_delete=models.CASCADE)
#delete the products when the respective category is deleted
    created_date    = models.DateTimeField(auto_now_add=True)
    modified_date   = models.DateTimeField(auto_now=True)
    def get_url(self):
        return reverse('product_detail', args=[self.category.slug, self.slug])
    def __str__(self):          #string representation of the model
        return self.product_name
    def averageReview(self):
        reviews = ReviewRating.objects.filter(product=self,
status=True).aggregate(average=Avg('rating'))
        avg = 0
        if reviews['average'] is not None:
            avg = float(reviews['average'])
        return avg
    def countReview(self):
```

```
        reviews = ReviewRating.objects.filter(product=self,
status=True).aggregate(count=Count('id'))
        count = 0
        if reviews['count'] is not None:
            count = int(reviews['count'])
        return count
class Trending(models.Model):                                #Home Page
    product_name    = models.CharField(max_length=200, unique=True)
    slug            = models.SlugField(max_length=200, unique=True)
    description     = models.TextField(max_length=500, blank=True)
    price           = models.IntegerField()
    images          = models.ImageField(upload_to='photos/products')
    stock           = models.IntegerField()
    is_available    = models.BooleanField(default=True)
    category        = models.ForeignKey(Category, on_delete=models.CASCADE)
#delete the products when the respective category is deleted
    created_date    = models.DateTimeField(auto_now_add=True)
    modified_date   = models.DateTimeField(auto_now=True)
    def get_url(self):
        return reverse('product_detail', args=[self.category.slug, self.slug])
    def __str__(self):          #string representation of the model
        return self.product_name
class VariationManager(models.Manager):
    def units(self):
        return super(VariationManager, self).filter(variation_category='unit',
is_active=True)
    def types(self):
```

```
        return super(VariationManager, self).filter(variation_category='type',
is_active=True)
variation_category_choice = (
    ('type', 'type'),
    ('unit', 'unit'),
)
class Variation(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    variation_category = models.CharField(max_length=100,
choices=variation_category_choice)
    variation_value = models.CharField(max_length=100)
    is_active= models.BooleanField(default=True)
    created_date = models.DateTimeField(auto_now=True)
    objects = VariationManager()
    def __str__(self):
        return self.variation_value
class Offers(models.Model):                    #Home Page
        product_name    = models.CharField(max_length=200, unique=True)
        slug            = models.SlugField(max_length=200, unique=True)
        description     = models.TextField(max_length=500, blank=True)
        initial_price   = models.IntegerField()
        final_price     = models.IntegerField()
        images          = models.ImageField(upload_to='photos/products')
        stock           = models.IntegerField()
        is_available    = models.BooleanField(default=True)
        category        = models.ForeignKey(Category, on_delete=models.CASCADE)
#delete the products when the respective category is deleted
```

```
    created_date    = models.DateTimeField(auto_now_add=True)
    modified_date   = models.DateTimeField(auto_now=True)
    def get_url(self):
        return reverse('product_detail', args=[self.category.slug, self.slug])
    def __str__(self):          #string representation of the model
        return self.product_name
class New(models.Model):                    #Home Page
    product_name    = models.CharField(max_length=200, unique=True)
    slug            = models.SlugField(max_length=200, unique=True)
    description     = models.TextField(max_length=500, blank=True)
    final_price     = models.IntegerField()
    images          = models.ImageField(upload_to='photos/products')
    stock           = models.IntegerField()
    is_available    = models.BooleanField(default=True)
    category        = models.ForeignKey(Category, on_delete=models.CASCADE)
#delete the products when the respective category is deleted
    created_date    = models.DateTimeField(auto_now_add=True)
    modified_date   = models.DateTimeField(auto_now=True)
    def get_url(self):
        return reverse('product_detail', args=[self.category.slug, self.slug])
    def __str__(self):          #string representation of the model
        return self.product_name
class ProductGallery(models.Model):
  product = models.ForeignKey(Product, default=None,
on_delete=models.CASCADE)
  image = models.ImageField(upload_to = 'store/products', max_length=255)
```

```
    def __str__(self):
        return self.product.product_name
    class Meta:
        verbose_name = 'productgallery'
        verbose_name_plural = 'product gallery'
class ReviewRating(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    user = models.ForeignKey(Account, on_delete=models.CASCADE)
    subject = models.CharField(max_length=100, blank=True)
    review = models.TextField(max_length=500, blank=True)
    rating = models.FloatField()
    ip = models.CharField(max_length=20, blank=True)
    status = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.subject
```

views.py

```
from django.shortcuts import render, get_object_or_404, redirect
from .models import Product, ProductGallery, ReviewRating
from category.models import Category
from carts.views import _cart_id
from carts.models import CartItem
from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator
from django.http import HttpResponse
```

```
from django.db.models import Q
from .forms import ReviewForm
from django.contrib import messages
from orders.models import OrderProduct
# Create your views here.
def store(request, category_slug=None):
    categories = None
    products = None
    if category_slug != None:
        categories = get_object_or_404(Category,slug = category_slug)
        products = Product.objects.filter(category=categories, is_available=True)
        paginator = Paginator(products,6)
        page = request.GET.get('page')
        paged_products = paginator.get_page(page)
        product_count = products.count()
    else:
        products = Product.objects.all().filter(is_available=True).order_by('id')
        paginator = Paginator(products,9)
        page = request.GET.get('page')
        paged_products = paginator.get_page(page)
        product_count = products.count()
    context = {
        'products' : paged_products,
        'product_count' : product_count,
    }
    return render(request, 'store/store.html', context)
```

```
def product_detail(request, category_slug, product_slug):
    try:
        single_product = Product.objects.get(category__slug=category_slug, slug=product_slug)
        in_cart = CartItem.objects.filter(cart__cart_id = _cart_id(request), product = single_product).exists() #returns True
    except Exception as e:
        raise e
    if request.user.is_authenticated:
        try:
            orderproduct = OrderProduct.objects.filter(user=request.user, product_id= single_product.id).exists()
        except OrderProduct.DoesNotExist:
            orderproduct = None
    else:
        orderproduct = None
    # Get the reviews
    reviews = ReviewRating.objects.filter(product_id=single_product.id, status=True)
    #Get the product gallery
    product_gallery = ProductGallery.objects.filter(product_id = single_product.id)
    context = {
        'single_product' : single_product,
        'in_cart'       : in_cart,
        'product_gallery': product_gallery,
        'orderproduct'   : orderproduct,
        'reviews': reviews,
```

```
    }
    return render(request, 'store/product_detail.html', context)
def search(request):
        products=0
        product_count = 0
        if 'keyword' in request.GET:
         keyword = request.GET['keyword']
         if keyword:
            products = Product.objects.order_by('-
created_date').filter(Q(description__icontains=keyword) |
Q(product_name__icontains=keyword))
            product_count = products.count()
        context = {
            'products': products,
            'product_count' : product_count,
        }
        return render(request, 'store/store.html', context)
def submit_review(request, product_id):
    url = request.META.get('HTTP_REFERER')
    if request.method == 'POST':
        try:
            reviews = ReviewRating.objects.get(user__id=request.user.id,
product__id=product_id)
            form = ReviewForm(request.POST, instance=reviews)
            form.save()
            messages.success(request, 'Thank you! Your review has been updated.')
            return redirect(url)
```

```
        except ReviewRating.DoesNotExist:
            form = ReviewForm(request.POST)
            if form.is_valid():
                data = ReviewRating()
                data.subject = form.cleaned_data['subject']
                data.rating = form.cleaned_data['rating']
                data.review = form.cleaned_data['review']
                data.ip = request.META.get('REMOTE_ADDR')
                data.product_id = product_id
                data.user_id = request.user.id
                data.save()
                messages.success(request, 'Thank you! Your review has been submitted.')
                return redirect(url)
```

urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.store, name='store'),
    path('category/<slug:category_slug>/', views.store, name='products_by_category'),
    path('category/<slug:category_slug>/<slug:product_slug>/', views.product_detail, name='product_detail'),
    path('search/', views.search, name='search'),
    path('submit_review/<int:product_id>/', views.submit_review, name="submit_review"),]
```

forms.py

```
from django import forms
from .models import ReviewRating
class ReviewForm(forms.ModelForm):
    class Meta:
        model = ReviewRating
        fields = ['subject', 'review', 'rating']
```

orders(app):

models.py

```
from django.db import models
from accounts.models import Account
from store.models import Product, Variation
# Create your models here.
class Payment(models.Model):
    user = models.ForeignKey(Account, on_delete=models.CASCADE)
    payment_id = models.CharField(max_length=100)
    payment_method = models.CharField(max_length=100)
    amount_paid = models.CharField(max_length=100) # this is the total amount
paid
    status = models.CharField(max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.payment_id
```

```
class Order(models.Model):
STATUS = (
        ('New', 'New'),
        ('Accepted', 'Accepted'),
        ('Completed', 'Completed'),
        ('Cancelled', 'Cancelled'),
    )
    user = models.ForeignKey(Account, on_delete=models.SET_NULL, null=True)
    payment = models.ForeignKey(Payment, on_delete=models.SET_NULL,
blank=True, null=True)
    order_number = models.CharField(max_length=20)
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    email = models.EmailField(max_length=50)
    phone = models.CharField(max_length=15)
    pincode = models.CharField(max_length=50)
    area = models.CharField(max_length=50)
    address_line_1 = models.CharField(max_length=50)
    address_line_2 = models.CharField(max_length=50, blank=True)
    order_note = models.CharField(max_length=100, blank=True)
    payment_mode = models.CharField(max_length=100, blank=True)
    time = models.CharField(max_length=100, blank=True)
    date = models.CharField(max_length=100, blank=True)
    order_total = models.FloatField()
    tax = models.FloatField()
    status = models.CharField(max_length=10, choices=STATUS, default='New')
    ip = models.CharField(blank=True, max_length=20)
```

```
    is_ordered = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    def full_name(self):
        return f'{self.first_name} {self.last_name}'
    def full_address(self):
        return f'{self.address_line_1} {self.address_line_2}'
    def __str__(self):
        return self.first_name
class OrderProduct(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    payment = models.ForeignKey(Payment, on_delete=models.SET_NULL,
blank=True, null=True)
    user = models.ForeignKey(Account, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    variations = models.ManyToManyField(Variation, blank=True)
    quantity = models.IntegerField()
    product_price = models.FloatField()
    ordered = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    def __str__(self):
        return self.product.product_name
```

views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, JsonResponse
from carts.models import CartItem
from .forms import OrderForm
import datetime
from .models import Order, Payment, OrderProduct
import json
from store.models import Product
from django.core.mail import EmailMessage
from django.template.loader import render_to_string
# Create your views here.
def payments(request):
    body = json.loads(request.body)
    order = Order.objects.get(user=request.user, is_ordered=False,
order_number=body['orderID'])
    #store transaction details inside payment model
    payment = Payment(
        user = request.user,
        payment_id = body['transID'],
        payment_method = body['payment_method'],
        amount_paid = order.order_total,
        status = body['status'],
    )
    payment.save()
    order.payment = payment
```

```
order.is_ordered = True
order.save()
# Move the cart items to Order Product table
cart_items =  CartItem.objects.filter(user = request.user)
for item in cart_items:
   orderproduct = OrderProduct()  #creating an object of order product
   orderproduct.order_id = order.id
   orderproduct.payment = payment
   orderproduct.user_id = request.user.id
   orderproduct.product_id = item.product_id
   orderproduct.quantity = item.quantity
   orderproduct.product_price = item.product.price
   orderproduct.ordered = True
   orderproduct.save()
   # Save the product and then assign variations
   cart_item = CartItem.objects.get(id=item.id)
   product_variation = cart_item.variations.all()
   orderproduct = OrderProduct.objects.get(id=orderproduct.id)
   orderproduct.variations.set(product_variation)
   orderproduct.save()
   # Reduce the quantity of the sold Products
   product = Product.objects.get(id=item.product_id)
   product.stock -= item.quantity
   product.save()
```

```
 # Clear cart
    CartItem.objects.filter(user=request.user).delete()
    # Send order recieved email to customer
    mail_subject = 'Thank you for your Order!'
    message = render_to_string('orders/order_recieved_email.html', {
        'user': request.user,
        'order': order,
    })
    to_email = request.user.email
    send_email = EmailMessage(mail_subject, message, to=[to_email])
    send_email.send()
    # Send order number and transaction id back to sendData method via
JsonResponse
    data = {
        'order_number': order.order_number,
        'transID': payment.payment_id,
    }
    return JsonResponse(data)
def place_order(request, total=0, quantity=0,):
    current_user = request.user
    #if the cart count <= 0, then redirect back to store
    cart_items = CartItem.objects.filter(user=current_user)
    cart_count = cart_items.count()
    if cart_count <= 0:
        return redirect('store')
    grand_total = 0
    tax = 0
```

```
for cart_item in cart_items:
    total += (cart_item.product.price * cart_item.quantity)
    quantity += cart_item.quantity
tax = (2 * total)/100
grand_total = total + tax
if request.method == 'POST':
    form = OrderForm(request.POST)
    if form.is_valid():
        data = Order()
        data.user = current_user
        data.first_name = form.cleaned_data['first_name']
        data.last_name = form.cleaned_data['last_name']
        data.email = form.cleaned_data['email']
        data.phone = form.cleaned_data['phone']
        data.pincode = form.cleaned_data['pincode']
        data.area = form.cleaned_data['area']
        data.address_line_1 = form.cleaned_data['address_line_1']
        data.address_line_2 = form.cleaned_data['address_line_2']
        data.payment_mode = form.cleaned_data['payment_mode']
        data.time = form.cleaned_data['time']
        data.date = form.cleaned_data['date']
        data.order_note = form.cleaned_data['order_note']
        data.order_total = grand_total
        data.tax = tax
        data.ip = request.META.get('REMOTE_ADDR')
        data.save()
```

```
        # Generate order number
        yr = int(datetime.date.today().strftime('%Y'))
        dt = int(datetime.date.today().strftime('%d'))
        mt = int(datetime.date.today().strftime('%m'))
        d = datetime.date(yr,mt,dt)
        current_date = d.strftime("%Y%m%d") #20210305
        order_number = current_date + str(data.id)
        data.order_number = order_number
        data.save()
        order = Order.objects.get(user = current_user, is_ordered = False,
order_number = order_number)
        context = {
            'order' : order,
            'cart_items' : cart_items,
            'total' : total,
            'tax' : tax,
            'grand_total' : grand_total,
        }
        return render(request, 'orders/payments.html', context)
    else:
        return redirect('checkout')
def order_complete(request):
    order_number = request.GET.get('order_number')
    transID = request.GET.get('payment_id')
    try:
        order = Order.objects.get(order_number=order_number,
is_ordered=True)
```

```
        ordered_products = OrderProduct.objects.filter(order_id=order.id)
        subtotal = 0
        for i in ordered_products:
            subtotal += i.product_price * i.quantity
        payment = Payment.objects.get(payment_id=transID)
        context = {
            'order': order,
            'ordered_products': ordered_products,
            'order_number': order.order_number,
            'transID': payment.payment_id,
            'payment': payment,
            'subtotal': subtotal,
        }
        return render(request, 'orders/order_complete.html', context)
    except (Payment.DoesNotExist, Order.DoesNotExist):
        return redirect('home')
```

carts (app):

models.py

```
from django.db import models
from store.models import Product, Variation
from accounts.models import Account
# Create your models here.
class Cart(models.Model):
    cart_id = models.CharField(max_length=125, blank=True)
    date_added = models.DateField(auto_now_add=True)
```

```
    def __str__(self):
        return self.cart_id
class CartItem(models.Model):
    user = models.ForeignKey(Account, on_delete = models.CASCADE, null=True)
    variations = models.ManyToManyField(Variation, blank=True)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    cart    = models.ForeignKey(Cart, on_delete=models.CASCADE, null=True)
    quantity = models.IntegerField()
    is_active = models.BooleanField(default=True)
    def sub_total(self):
        return self.product.price * self.quantity
    def __unicode__(self):
        return self.product
```

views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from store.models import Product, Variation
from .models import Cart, CartItem
from django.core.exceptions import ObjectDoesNotExist
from django.contrib.auth.decorators import login_required
from django.http  import HttpResponse
# Create your views here.
def _cart_id(request):
    cart = request.session.session_key
    if not cart:
        cart = request.session.create()
```

```
    return cart #returns cart id
def add_cart(request, product_id):
    current_user = request.user
    product = Product.objects.get(id=product_id) #get the product
    #if the user is authenticated
    if current_user.is_authenticated:
        product_variation = []
        if request.method == 'POST':
            for item in request.POST:
                key = item
                value = request.POST[key]
                try:
                    variation = Variation.objects.get(product=product,
variation_category__iexact=key, variation_value__iexact=value)
                    product_variation.append(variation)
                except:
                    pass
        is_cart_item_exists = CartItem.objects.filter(product=product,
user=current_user).exists()
        if is_cart_item_exists:
            cart_item = CartItem.objects.filter(product=product, user=current_user)
            ex_var_list = []
            id = []
            for item in cart_item:
                existing_variation = item.variations.all()
                ex_var_list.append(list(existing_variation))
                id.append(item.id)
```

```
            if product_variation in ex_var_list:
                #increase the cart item quantity
                index = ex_var_list.index(product_variation)
                item_id = id[index]
                item = CartItem.objects.get(product=product, id=item_id)
                item.quantity +=1
                item.save()
            else:
                item = CartItem.objects.create(product=product, quantity=1,
user=current_user)
                if len(product_variation) > 0:
                    item.variations.clear()
                    item.variations.add(*product_variation)
                item.save()
        else:
            cart_item = CartItem.objects.create(
                product = product,
                quantity = 1,
                user = current_user,
            )
            if len(product_variation) > 0:
                cart_item.variations.clear()
                cart_item.variations.add(*product_variation)
            cart_item.save()
        return redirect('cart')
    # is the user is not authenticated
    else:
```

```
    product_variation = []
    if request.method == 'POST':
        for item in request.POST:
            key = item
            value = request.POST[key]
            try:
                variation = Variation.objects.get(product=product,
variation_category__iexact=key, variation_value__iexact=value)
                product_variation.append(variation)
            except:
                pass
    try:
        cart = Cart.objects.get(cart_id=_cart_id(request)) #get the cart using the
cart_id prsent in the session
    except Cart.DoesNotExist:
        cart = Cart.objects.create(
            cart_id = _cart_id(request)
        )
    cart.save()
    is_cart_item_exists = CartItem.objects.filter(product=product,
cart=cart).exists()
    if is_cart_item_exists:
        cart_item = CartItem.objects.filter(product=product, cart=cart)
        #existing variations  ->database
        #current variation ->product_variation
        #item_id ->database
        ex_var_list = []
```

```
        id = []
        for item in cart_item:
            existing_variation = item.variations.all()
            ex_var_list.append(list(existing_variation))
            id.append(item.id)
        print(ex_var_list)
        if product_variation in ex_var_list:
            #increase the cart item quantity
            index = ex_var_list.index(product_variation)
            item_id = id[index]
            item = CartItem.objects.get(product=product, id=item_id)
            item.quantity +=1
            item.save()
        else:
            item = CartItem.objects.create(product=product, quantity=1,
cart=cart)
            if len(product_variation) > 0:
                item.variations.clear()
                item.variations.add(*product_variation)
            item.save()
    else:
        cart_item = CartItem.objects.create(
            product = product,
            quantity = 1,
            cart = cart,
        )
        if len(product_variation) > 0:
```

```
            cart_item.variations.clear()
            cart_item.variations.add(*product_variation)
        cart_item.save()
    return redirect('cart')
def remove_cart(request, product_id, cart_item_id):
    product = get_object_or_404(Product, id=product_id)
    try:
        if request.user.is_authenticated:
            cart_item = CartItem.objects.get(product=product, user=request.user,
id=cart_item_id)
        else:
            cart =  Cart.objects.get(cart_id=_cart_id(request))
            cart_item = CartItem.objects.get(product=product, cart=cart,
id=cart_item_id)
        if cart_item.quantity > 1:
            cart_item.quantity -= 1
            cart_item.save()
        else:
            cart_item.delete()
    except:
        pass
    return redirect('cart')
def remove_cart_item(request, product_id, cart_item_id):
    product = get_object_or_404(Product, id=product_id)
    if request.user.is_authenticated:
            cart_item = CartItem.objects.get(product=product, user=request.user,
id=cart_item_id)
```

```
    else:
        cart = Cart.objects.get(cart_id=_cart_id(request))
        cart_item = CartItem.objects.get(product=product, cart=cart,
id=cart_item_id)
    cart_item.delete()
    return redirect('cart')
def cart(request, total=0, quantity=0, cart_items=None):
    try:
        tax = 0
        grand_total = 0
        if request.user.is_authenticated:
            cart_items = CartItem.objects.filter(user=request.user, is_active = True)
        else:
            cart = Cart.objects.get(cart_id = _cart_id(request))
            cart_items = CartItem.objects.filter(cart=cart, is_active = True)
        for cart_item in cart_items:
            total += (cart_item.product.price * cart_item.quantity)
            quantity += cart_item.quantity
        tax = (2 * total)/100
        grand_total = total + tax
    except ObjectDoesNotExist:
        pass
    context = {
        'total' : total,
        'quantity' : quantity,
        'cart_items' : cart_items,
        'tax' : tax,
```

```
        'grand_total' : grand_total,
    }
    return render(request, 'store/cart.html', context)
@login_required(login_url='login')
def checkout(request, total=0, quantity=0, cart_items=None):
    try:
        tax = 0
        grand_total = 0
        if request.user.is_authenticated:
            cart_items = CartItem.objects.filter(user=request.user, is_active = True)
        else:
            cart = Cart.objects.get(cart_id = _cart_id(request))
            cart_items = CartItem.objects.filter(cart=cart, is_active = True)
        for cart_item in cart_items:
            total += (cart_item.product.price * cart_item.quantity)
            quantity += cart_item.quantity
        tax = (2 * total)/100
        grand_total = total + tax
    except ObjectDoesNotExist:
         pass
    context = {
        'total' : total,
        'quantity' : quantity,
        'cart_items' : cart_items,
        'tax' : tax,
        'grand_total' : grand_total, } return render(request, 'store/checkout.html',
context)
```

urls.py

```
from django.urls import path

from . import views

urlpatterns = [

   path('', views.cart, name='cart'),

   path('add_cart/<int:product_id>/', views.add_cart, name='add_cart'),

   path('remove_cart/<int:product_id>/<int:cart_item_id>/',
views.remove_cart, name='remove_cart'),

   path('remove_cart_item/<int:product_id>/<int:cart_item_id>/',
views.remove_cart_item, name='remove_cart_item'),

   path('checkout/', views.checkout, name='checkout'),

]
```

Secure Coding Practices

- Secure Coding Practices
- Prevention from SQL Injection
- Authentication

Future Enhancements

1. Flash Sale Notification

The idea is to keep bringing back users at the right time so they do not miss the sale. With the help of push notifications, we can schedule multiple notifications and send them at a later time. This way we will not miss updating users about the sale.

2. Develop a Chat Option for Customer Service

With live chat, you give customers a way to reach you in the exact moment that they have questions or problems they can’t solve. This feels much better than sending an email to a support team; with email, it’s hard to know when you’ll get a response back.

3. Launching an App

References And Bibliography

Official documentation from:

https://docs.djangoproject.com/en/4.0/

https://getbootstrap.com/docs/4.6/getting-started/introduction/