JNAN VIKAS MANDAL'S

JVM'S DEGREE COLLEGE OF INFORMATION TECHNOLOGY,

AIROLI, NAVI MUMBAI – 400708

DEPARTMENT OF INFORMATION TECHNOLOGY

NAAC RE-ACCREDIATED GRADE 'A' (CGPA-3.33)

PRACTICAL WORK SUBMITTED IN PARTIAL

FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

BY

DILIP GAIKWAD

ROLL NUMBER: 621

UNDER THE ESTEEMED GUIDANCE OF

DR. SUNITHA JOSHI

ASST. PROF. SAROJINI BIRADAR

ASST. PROF. SANJIVANI NALKAR

E-Kart

E-Kart online shopping

A Project Report

Submitted in partial fulfilment of the

Requirement for the award of the Degree

of

M.Sc. (INFORMATION TECHNOLOGY)

Mr. Dilip Gaikwad

& Mr. Omkar Dhumal

Under the esteemed guidance of

Dr. Sunita Joshi

M.Sc. Information Technology Incharge

DEPARTMENT OF INFORMATION TECHNOLOGY

JVM Mehta College of Arts, Science and Commerce

(Affiliated to University of Mumbai)

NAAC Reaccredited Grade 'A'(CGPA-

3.33)AIROLI, NAVI MUMBAI – 400

708

MAHARASHTRA

2022-2023

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information . Incomplete proforma of approval in any respect be summarily rejected.)

1. Name of the Students:

   Dilip Gaikwad & Omkar Dhumal

2. Title of the Project

   E-Kart online shopping

3. Name of the Guide

   Prof. Mustufa Nullwala

4. Teaching experience of the guide

5. Is this your first submission?

Signature of the student: **Dilip Gaikwad & Omkar Dhumal**

Signature of the Guide: .....

Signature of the Coordinator: .....

Date: 28/05/2023

JNAN VIKAS MANDAL'S

PADMASHREE DR. R.T.DOSHI DEGREE COLLEGE OF INFORMATION TECHNOLOGY

MOHANLAL RAICHAND MEHTA COLLEGE OF COMMERCE

DIWALIMAA DEGREE COLLEGE OF SCIENCE

AMRATLAL RAICHAND MEHTA COLLEGE OF ARTS

JVM'S DEGREE COLLEGE OF INFORMATION TECHNOLOGY

Affiliated to University of Mumbai (Permanently Unaided College)

NAAC Reaccredited Grade 'A'(CGPA-3.33)

CERTIFICATE

This is certifying that the project entitled, "E-Kart online shopping", is bonafied work of **Mr. Dilip Gaikwad & Mr. Omkar Dhumal** bearing **Roll. No :(621 & 622)** Submitted in partial fulfillment of the requirements for the award of degree of MSc. INFORMATION TECHNOLOGY from University of Mumbai.

Internal Guide Coordinator External Examiner

External Examiner:

Date: College Seal

ABSTRACT

With the evolution of technology and the wave of digitalization, more and more businesses are adapting to tech evolutions. The ultimate purpose behind this evolution is to make shopping a hassle-free experience for everyone. Electronic Commerce is process of doing business through computer networks. A person sitting on his chair in front of a computer can access all the facilities of the Internet to buy or sell the products.
Unlike traditional commerce that is carried out physically with effort of a person to go & get products, ecommerce has made it easier for human to reduce physical work and to save time.

Speaking of obstacles, there are a lot of them that need to be uprooted before e-commerce can compete with traditional commerce. The biggest obstacle in the course of advancement of e-commerce is that the consumer's senses are limited to seeing and hearing the product. The second largest problem that e - commerce has been facing over the past few years is that of security. Traditional buyers and sellers are still paranoid about conducting business online.

Finally, in order to make the online shopping experience even more better, there are a lot of new technologies like WordPress + WooCommerce , Shopify, Magento and BigCommerce that have emerged over the last couple of years.

Using such technologies we are building an e-commerce website project for our curriculum, in which a user can login can view the varieties of products, add them to wishlist, buy them, check the reviews given by previous buyers manage his account credentials and a user can does a secure payment using paypal payment service.

Implementing a strong admin system for the website where admin can control all the data of customer, product, category, groups, users, accounts, permissions and many more.

ACKNOWLEDGEMENT

It is indeed a matter of great pleasure and proud to be able to present this project on "**E-Kart online shopping project** ".

The completion of the project work is a milestone in student life and its execution is inevitable in the hands of guide.

I would like to tender our sincere thanks to the **Prof. Mrs. Sunitha Joshi** and all teachers for their co-operation. I would also like to express our deep regards and gratitude to the Principal **Dr. Leena Sarkar** I will wish to thank the non-teaching staff and my friends who have helped me all time in one way or the other.

Really it is highly impossible to repay the department of all the people who have directly or indirectly helped me for performing the project

DECLARATION

I hereby declare that the project work entitled "**E-Kart online shopping** " done at place where by the project is done , has not been in any case duplicated to submit to any other university for the award of any other degree. To the best of my knowledge other than me, no other has submitted to any other University. The project is done in partial fulfillment of the requirement for the award of degree in **M.Sc. (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

- Dilip Gaikwad
- Omkar Dhumal

TABLE OF CONTENT

Chapter No.	TITLE	Page No
➢	**Abstract**	
➢	**Acknowledgement**	
➢	**DECLARATION**	
CHAPTER 1	**PROJECT DOMAIN**	8
1.1	Background	
1.2	Objective	10
1.3	Purpose and Scope	11
1.4	Availability	
1.5	Literature Review	12
CHAPTER 2	**SYSTEM ANALYSIS**	
2.1	Existing System	
2.2	Proposed System	15
2.3	Requirement Analysis	16
2.3.1	Hardware Requirements	18
2.3.2	Software Requirements	18
2.4	Justification of selection of Technology	19
CHAPTER 3	**SYSTEM DESIGN**	
3.1	Module Division	21
3.2	ER Diagrams	25
3.3	DFD/UML Diagrams	26
CHAPTER 4	IMPLEMENTATION AND TESTING	
	Implementation	
	Testing	
CHAPTER 5	**SOURCE CODE & RESULTS**	
	Application and development	
	Result	
CHAPTER 6	**CONCLUSION AND FUTURE ENHANCEMENT**	35
CHAPTER 7	**REFERENCES**	36

PROJECT DOMAIN

- **Automation:** It is the creation and application of technologies to produce and deliver goods and services with minimal human intervention. The implementation of automation technologies, techniques and processes improve the efficiency, reliability, and/or speed of many tasks that were previously performed by humans

- **Data Science**: Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data,and apply knowledge and actionable insights from data across a broad range of application domains. Data science is related to data mining, machine learning and big data.

- **Reporting & analysis:** Power BI bases a report on a single dataset. Report designers create the visuals in a report to represent nuggets of information. The visuals aren't static. They update as the underlying data changes. You can interact with the visuals and filters as you dig into the data to discover insights and look for answers. Like a dashboard, but more so, a report is highly interactive and highly customizable. The extent of what you can do with a report will depend on the role and permissions assigned by the report designer.

Background

The background of e-commerce dates back to the early days of the internet, with the first known online transaction taking place in the early 1970s. This first transaction was a sale of marijuana via the ARPANET, which was one of the first operational packet switching networks, and the predecessor of the global internet we know today.

The background of e-commerce can be traced back to the early days of the internet, with early examples of online transactions taking place in the 1970s. However, the modern concept of e-commerce as we know it today began to take shape in the 1990s with the widespread adoption of the World Wide Web and the development of web browsers. This made it possible for businesses to create and maintain websites, and for consumers to easily purchase goods and services online.

However, the modern concept of e-commerce as we know it today began to take shape in the 1990s with the widespread adoption of the World Wide Web and the development of web browsers. This made it possible for businesses to create and maintain websites, and for consumers to easily purchase goods and services online. The growth of e-commerce was also supported by the development of secure payment technologies such as SSL (Secure Socket Layer) and SET (Secure Electronic Transactions), which made online transactions more secure and trustworthy.

The first online retail website, called Internet Shopping Network, was launched in 1992. But it was the launch of Amazon.com in 1995 that truly popularized e - commerce, with the company becoming one of the world's largest retailers. The late 1990s and early 2000s saw the emergence of many other e-commerce companies, such as eBay and PayPal, and online marketplaces like Alibaba and Amazon.

Over the years, e-commerce has grown to become a major force in the global economy, with online sales expected to reach $4 trillion by 2020.

Objective

Easy to use Improve security well managed easy to understand Search the GPS location of the user. If a user wants to know his or her location, he or she will command the application, which will then search for the user's location using GPS and prompt the user with the location. Using this helps people a lot. We can help people live normal lives by developing this application.

1) **Manage Online Selling Costs In A Strategic Way:**

This module helps the user voice-command the application; the application will catch the voice command and convert it into digital data and take appropriate action.

2) **Establish Deeper Business Relationships**:

In this case, the application will be given permission to read the messages, as it's important to allow application to read the messages so that the user can read or send messages by voice command.

3) **Provide a Unique Customer Experience**

In this case, the application will be given permission to read the contacts, as it is important to allow application to read the contacts so that the user can call any person by voice command.

4) **Improve Customer Loyalty**

The user will instruct the application to send or read messages in this module. will either read the message and send a message or read the message for the user.

5) **Make your eCommerce Website Mobile Responsive**

If the user wants to know his existing location, then the user will give a command, and then the application will search the user's location and prompt the user.

Purpose, Scope and Availability

Purpose

The purpose of e-commerce is to facilitate the buying and selling of goods and services online. It allows businesses to reach a global audience and to expand their customer base. It also provides customers with more convenience and choice, as they can shop from the comfort of their own homes and have access to a wider range of products and services.

The main benefits of e-commerce include:

- Increased accessibility: E-commerce makes it possible for customers to shop 24/7, regardless of their location.
- Greater convenience: Customers can shop from anywhere, at any time, and make purchases without having to leave their homes.
- Wider selection: Online retailers can offer a wider range of products and services than traditional brick-and-mortar stores.
- Cost savings: E-commerce can reduce the costs associated with running a physical store, such as rent and staffing costs.
- Personalization: E-commerce websites can use technologies like machine learning and big data analytics to personalize the customer experience and make recommendations based on browsing and buying history.
- Increased competition: E-commerce helps to level the playing field for small and large businesses, allowing them to compete on a more equal footing.
- Increased revenue: By reaching a wider customer base and reducing costs, e-commerce can help businesses increase their revenue.

E-commerce also enables businesses to expand their global reach, increase their revenue and profits and make their products and services more accessible to customers worldwide.

Scope

The scope of ecommerce includes the buying and selling of goods and services through the internet. This includes online retail stores, online marketplaces, and businesses that sell through social media platforms or their own websites. Ecommerce also includes the use of electronic payment methods and the management of electronic customer data. Ecommerce has grown rapidly in recent years, with many businesses now conducting a significant portion of their sales online. The scope of ecommerce is constantly evolving as new technologies and platforms become available.

Availability

E-commerce is available worldwide through the internet, making it accessible to anyone with an internet connection. Online retail stores, marketplaces, and other e-commerce platforms can be accessed from anywhere, allowing customers to shop from the comfort of their own homes or on the go using mobile devices. Many e-commerce businesses also ship products internationally, expanding their reach even further.

However, the availability of e-commerce can vary depending on a number of factors, such as the level of internet penetration in a given region, the infrastructure for online payments, and the regulations in place for online transactions. In some countries, e-commerce is still in the early stages of development and may not be as widely available or developed as in more mature e-commerce markets.

Literature Review

The Electronic Commerce, or E-commerce, industry is one of the most enlightened sectors of the economy. The industry is growing very rapidly, so data collection and estimation are particularly difficult. Therefore, one has to rely largely on research by both government and private organization.

According to the U.S. Survey Department, manufacturing sector is the largest supplier to e-commerce sales which has 47.4% of their total shipments, followed by vendors which is having 28.6% of their total sales. These two sectors make the business-to-business groups. Electronic commerce is generally considered to be the sales feature of e-business. It also consists of the exchange of data to facilitate the financing and payment aspects of business transactions. This is an active and resourceful way of communicating within an organization and one of the most operative and useful ways of leading business.

E-commerce today gained so much popularity because its essential technologies are worked out at huge steps. We are even offered to feel the product to better understand its shape, size and quality. In these benefits why to go out somewhere else when all you have to do is make an order, choose the delivery method, put up your feet and wait till the order is supplied right to your door-step.

The literature review may also examine the impact of e-commerce on businesses, including the ways in which e-commerce has changed the competitive landscape, and the challenges and opportunities that e-commerce presents for businesses. Additionally, it may examine the impact of e-commerce on consumers, including the ways in which e-commerce has changed the way people shop, and the ways in which e-commerce has affected consumer privacy and security.

Another important aspect that can be covered in a literature review is the impact of e-commerce on the economy, and how it affects employment, income, and productivity. It can also cover the impact of e-commerce on retail, wholesale and distribution channels, and the emergence of new business models such as the sharing economy.

Finally, the literature review may also examine the role of mobile devices and social media in e-commerce, including the ways in which these technologies are used to engage customers, and the ways in which they are used to facilitate online transactions.

Chapter 2

System Analysis

2.1 Existing System

Traditional commerce refers to the commercial transactions or exchange of buying or selling product/goods from business to person without use of internet which is a older method of business style and comes under traditional business. Now a days people are not preferring this as it is time taking and needs physical way of doing business.

Disadvantages of Existing System:

- Physically availability
- Ordering goods primarily via telex.
- Unmanageable
- Tough to understand the requirement.
- No availability of varieties
- Time consuming
- Tired full job

Proposed System

The proposed system is interactive and highly user-friendly here we will implement E-Kart web application to overcome everyday problems the traditional method used. Visually impaired person is a walking stick to the market to buy the daily need goods but these methods are ineffective because they are not reliable with the rapid advancement in modern technology like this application has been developed, it helps people to view products purchase them add to wish list, create an manage his/her account for examp le if the user wants to buy a product he simply need to login in the application and search the product and the application will show varieties of products to him, the user can mark the product to wish list and also purchase the product.

The application allows the user to manage his/her account and perform various activities like change password, checks varities of products compare the cost and many more.

This application serves a exciting user experience to the user, easy to use easy to navigate the application, manage the orders and account.

Planning and Scheduling

Gantt chart in Project Management is basically a visual view of the tasks that are scheduled over time. They are needed for planning of all projects of all sizes and they are quite a useful way of displaying which work is scheduled to be executed on a specific day. Gantt charts help you in viewing the start and end dates of the project in a single simple view. Many people in today's world are quite used to creating lists of tasks in Excel or different spreadsheet tools. They might also have created a simple Gantt Chart. It works fine whenever you are creating a list of things for anyone to view. However, when you add more people to that activity, at that time it becomes very much easier to create the Gantt Chart in an online manner. The benefits of the Gantt Chart are many. We have listed the most important benefits down below:

- Planning & Scheduling of Projects
- The Planning & Scheduling of tasks
- Planning & Scheduling of tasks across multiple projects
- Viewing tasks Over Time
- Planning in Sprints
- Team Collaboration
- Scheduling Team's work
- Determining Planned Timelines Vs Actual Timelines on Project

GANTT CHART

A Gantt chart is a graphical depiction of a project schedule. It's is a type of bar chart that shows the start and finish dates of several elements of a project that include resources, milestones, tasks, and dependencies. Henry Gantt, an American mechanical engineer, designed the Gantt chart.
The Gantt chart is the most widely used chart in project management. These charts are useful in planning a project and defining the sequence of tasks that require completion. In most instances, the chart is displayed as a horizontal b ar chart.

Horizontal bars of different lengths represent the project timeline, which can include task sequences, duration, and the start and end dates for each task. The horizontal bar also shows how much of a task requires completion.

A Gantt chart helps in scheduling, managing, and monitoring specific tasks and resources in a project. The chart shows the project timeline, which includes scheduled and completed work over a period. The Gantt chart aids project managers in communicating project status or plans and also helps ensure the project remains on track.

The chart identifies tasks that may be executed in parallel and those that cannot be started or finished until other tasks are complete. The Gantt chart can help detect potential bottlenecks and identify tasks that may have been excluded from the project timeline.

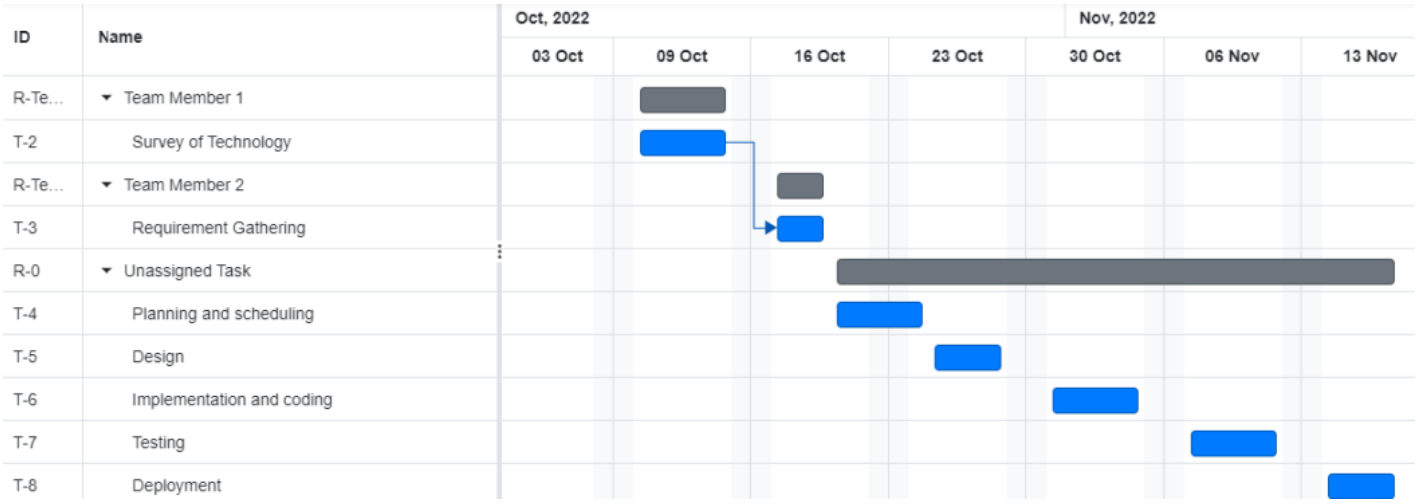
The chart depicts task slack time or additional time for completion of a task that should not delay the project, noncritical activities that may be delayed, and critical activities that must be executed on time.

Gantt charts can be used in managing projects of all sizes and types. These charts are utilized in several industries and for a range of projects. These may include building infrastructure like dams, bridges, and highways. They may also include software development and other technologies. Project management tools, such as Microsoft Visio, Project, SharePoint, and Excel, or specialized

software, such as Gantto or Matchware, can help in designing

Ganttcharts

Activities	Start Date	End Date	Days to Complete
Synopsis	07-10-2022	10-10-2022	4
Survey of Technology	11-10-2022	14-10-2022	4
Requirement Gathering	17-10-2022	19-10-2022	3
Planning and scheduling	20-10-2022	24-10-2022	5
Design	25-10-2022	28-10-2022	4
Implementation and coding	31-10-2022	04-11-2022	5
Testing	07-11-2022	11-11-2022	5
deployment	14-11-2022	17-11-2022	4



Requirement Analysis

Requirements analysis is the activity of elaborating basic requirements established during the inception, elicitation and negotiation tasks. Requirements analysis results in the software specification detailing the operational characteristics, interface with other system elements, and constraints that the software must meet

Based on their functions, requirements can be classified into: (a) Functional requirements, which describe system functionalities or services; (b) Non-functional requirements, which define system properties and cons traints (e.g. reliability, response time and storage requirements). The functional and non-functional requirements can further be classified into few types and have relationship as shown in Figure 1 [18]. As this research mainly concerns with business requirements that will be used in designing user requirements and the business rules that will be included in the use-case document, brief descriptions are provided for the three as follows:

(a) Business requirements: Represent high-level objectives of the organization or customer who requests the system. They describe why the organization is implementing the system (the objectives the organization hopes to achieve).

(b) User requirements: Describe user goals or tasks that the users must be able to perform with the product.

(c) Business rules: The rules in the organization that affect the system, which include corporate policies, government regulations, industry standards, accounting practices, and computational algorithms.

Hardware Requirements

Processor	Pentium 4
RAM	4 GB or more
Hard disk	16 GB or more
Input Device	Keyboard, Mouse
Output Device	Android

Software Requirements

Operating System	Windows OS
Front End:	HTML, CSS, Javascript, Ajax, JSON
Back End:	Python
Framework:	Django

Chapter 3
System Design

3.1 Module Division

3.1.1 System Features

In the life of the software development, problem analysis provides a base for design and development phase. The problem is analyzed so that sufficient matter is provided to design a new system. Large problems are sub-divided into smaller once to make them understandable and easy for finding solutions. Same in this project all the task are sub -divided and categorized.

System Modules:

- **Admin**

  - Controls Users
  - Control Products:
  - Control Product Categories
  - Add update delete Product
  - Users management
  - Product stock management

- **Use cases**

  - Create user account
  - List Products
  - Add Products from cart
  - Process payment

Website or online platform: This is the online storefront where customers can browse and purchase products or services.

Shopping cart and checkout: These are the features that allow customers to add items to their cart and complete the purchase process.

Payment processing: This component enables the acceptance of electronic payments, such as credit card or online banking payments.

Inventory management: This component keeps track of the products or services available for sale and ensures that inventory levels are accurate.

Order fulfilment: This includes the process of receiving, processing, and shipping orders to customers.

Customer service: This includes providing support and assistance to customers before, during, and after a purchase.

Marketing and promotion: This includes advertising and other efforts to attract customers to the e-commerce site.

Analytics and reporting: This includes tracking website traffic and sales data to help the business make informed decisions.

Content management: This includes creating, editing, and publishing content on the e-commerce site, such as product descriptions and images.

Security: This includes measures to protect customer data and prevent fraud, such as encryption and secure socket layer (SSL) certificates.

ER Diagram:

An Entity-Relationship (ER) diagram for an e-commerce website would typically include the following entities and their relationships:

Customer: This entity represents a customer who has registered on the e - commerce website and can place orders.

Product: This entity represents the products or services that are available for sale on the e-commerce website.

Order: This entity represents an order placed by a customer, and it includes details such as the products purchased, the quantity, and the total cost.

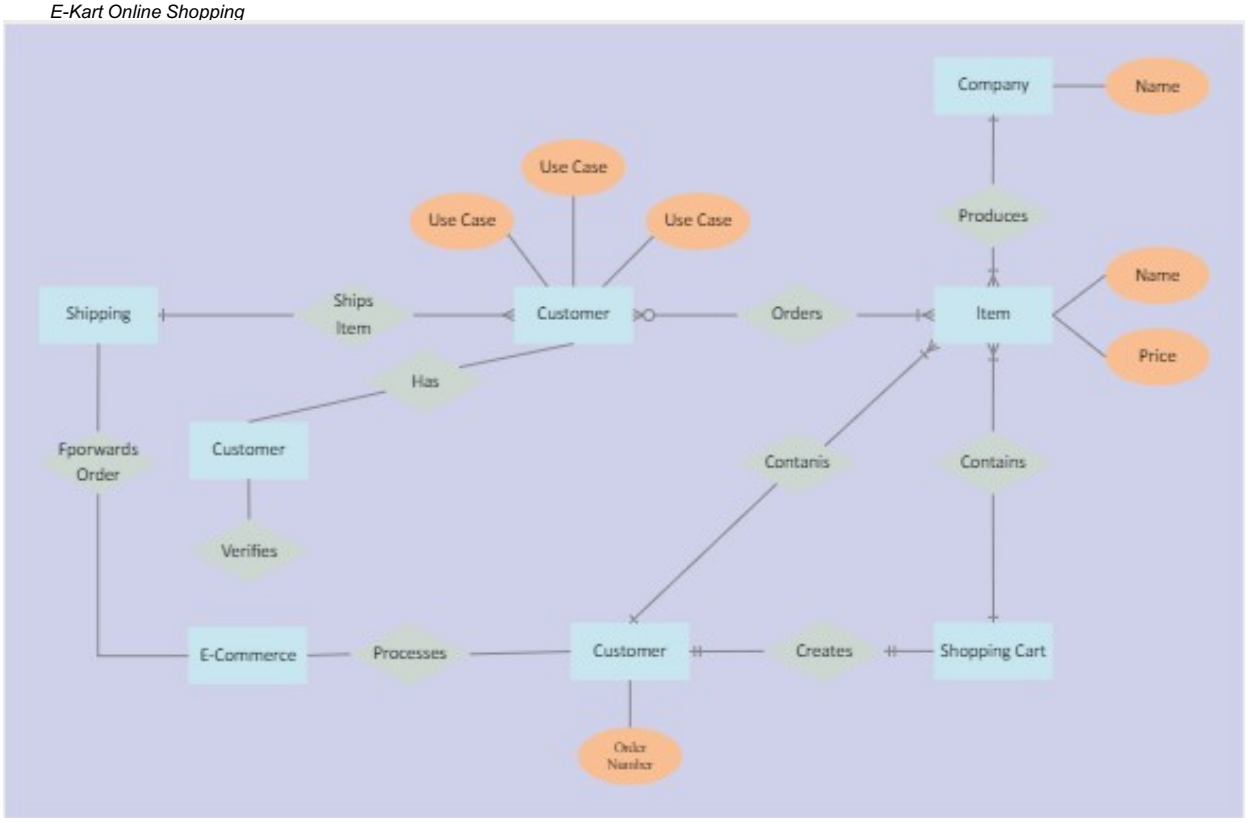
Order Item: This entity represents an individual product or service that is included in an order, and it includes details such as the product ID, quantity, and price.

Payment: This entity represents a payment made by a customer for an order, and it includes details such as the payment method and transaction ID.

Shipping: This entity represents the shipping details for an order, and it includes details such as the shipping address and shipping method.

Review: This entity represents a review left by a customer on a product, and it includes details such as the rating and the text of the review.

Category: This entity represents the category of a product, and it includes details such as the name and the parent category.


Company
Name
Use Case
Use Case
Use Case
Produces
Name
Shipping
Ships Item
Customer
Orders
Item
Price
Has
Fporwards Order
Customer
Contanis
Contains
Verifies
E-Commerce
Processes
Customer
Creates
Shopping Cart
Order Number

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers.

● Entity An entity is an object or component of data. An entity is represented as a rectangle in an ERdiagram. For example: Student and College and these two entities have many to one relationship as many student studies in a single college. An entity that cannot be uniquely identified by its own attributes and relies on the relationship with another entity is called a weak entity. The weak entity is represented by a double rectangle.

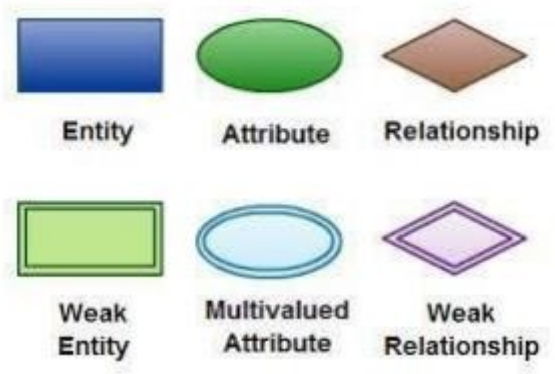
● Attribute An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1. Key attribute
2. Composite attribute
3. Multivalued attribute
4. Derived attribute

● Relationship A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

● ER Diagram Symbols



Data Flow Diagram

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system, and it can be used to represent the flow of data in an e-commerce system. A DFD for an e-commerce website would typically include the following components:

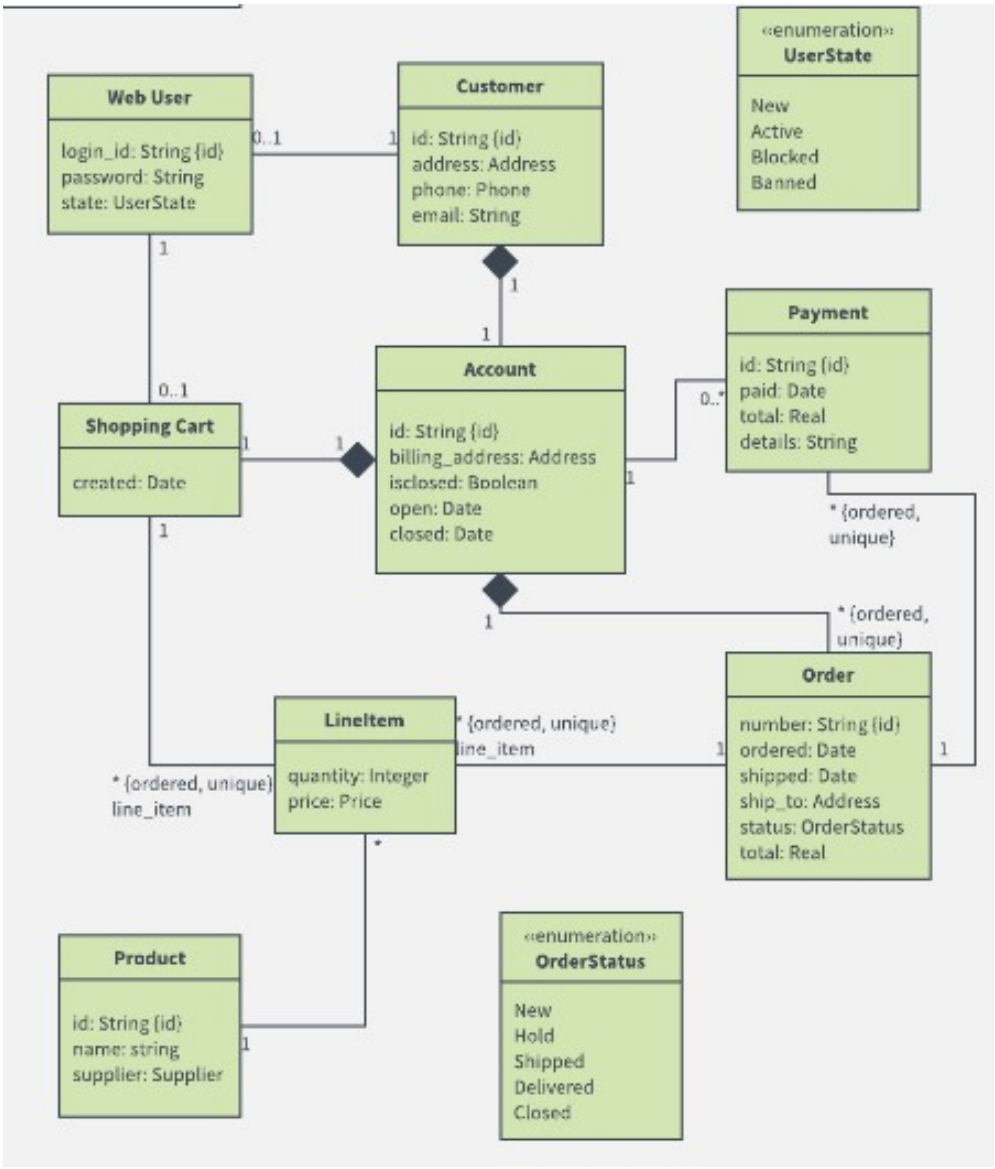
1. **External entities**: These are entities outside the e-commerce system that interact with it, such as customers and suppliers.

2. **Processes:** These are the steps that are performed within the e-commerce system, such as receiving an order, processing a payment, or updating inventory levels.

3. **Data stores**: These are the locations where data is stored within the e-commerce system, such as a database of customer information or a warehouse inventory.

4. **Data flows**: These are the paths that data takes as it moves through the e-commerce system, such as an order being placed by a customer and then being processed by the system.

Class Diagram



Activity Diagram:

An activity diagram for an e-commerce system would typically include a number of different flowchart-style shapes connected by arrows. The shapes would represent various activities or actions, such as adding items to a shopping cart, checking out, and making a payment. The arrows would indicate the flow of control between the different activities, and may be labeled with conditions or triggers that determine when the flow moves from one activity to another. Some common elements of an activity diagram for an e-commerce system might include:

- A "start" or "entry" point, which represents the beginning of the process.
- Activities related to browsing and searching for products, such as filtering by category or using a search bar.
- Activities related to adding items to a shopping cart and viewing the contents of the cart
- Activities related to creating or logging into an account, such as entering personal information or selecting a shipping address
- Activities related to making a payment and finalizing an order
- An "end" or "exit" point, which represents the completion of the process

USE CASE DIAGRAM:

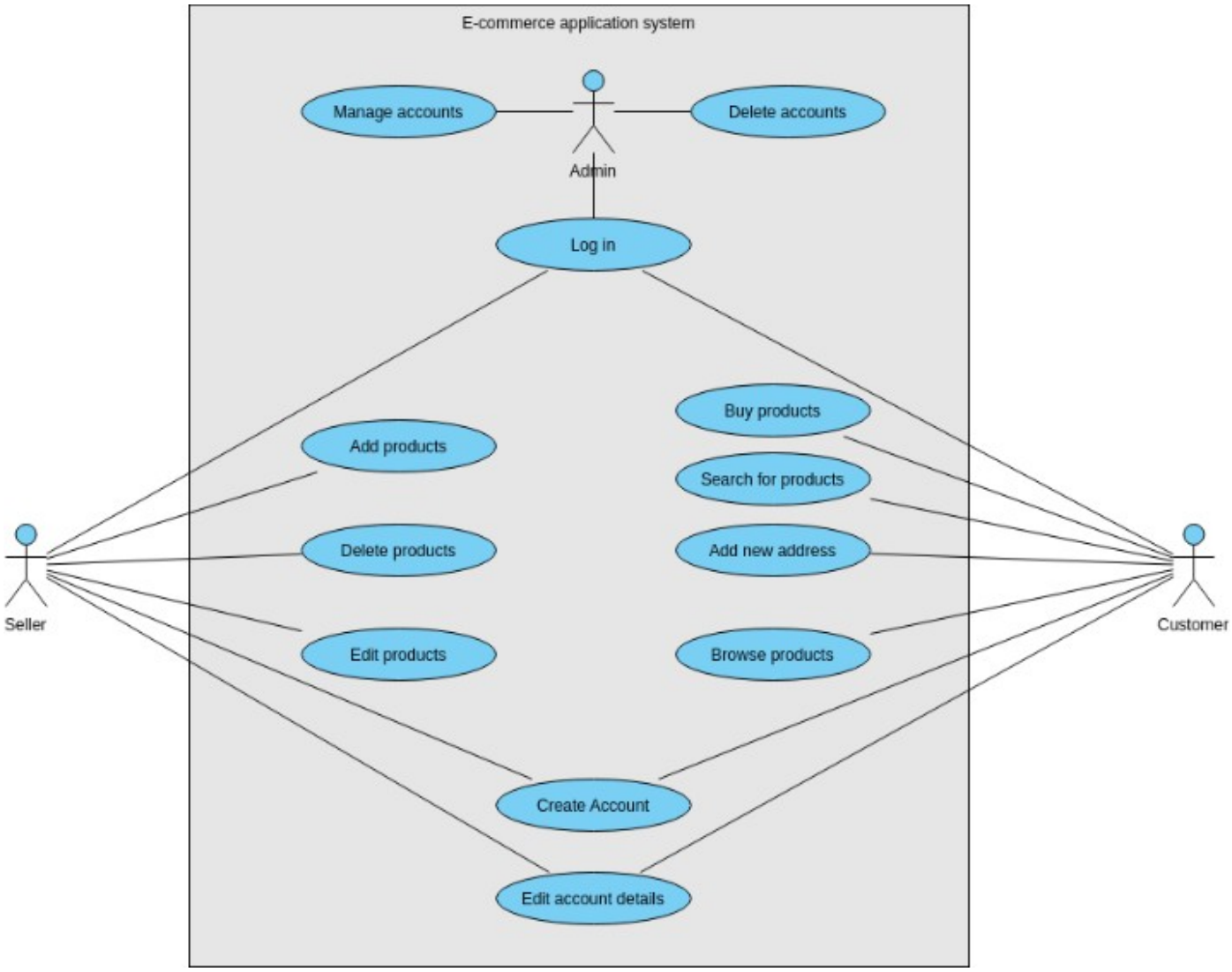
Use cases provide a high-level view of the system. They capture to a certain extent system structure. Use case describe sequences of actions a system performs that yield an observable result of value to a particular actor. Sequence of actions:

- Set of functions, algorithmic procedures, internal processes, etc.
- System performs: system functionalities
- An observable result of value to a user
- A particular actor: individual or device Use Cases modelling is an effective means of communicating with users and other stakeholders about the system and what is intended to do. Use Cases support a relationship with scenarios and relevant activities (e.g., testing).

Support requirements engineering activities and the requirement process

- Capture what a system is supposed to do, i.e., systems functional requirements
- Describe sequences of actions a system performs that yield an observable result of value to a particular actor
- Model actions of the system at its external interface
- Capture how the system coordinates human actions

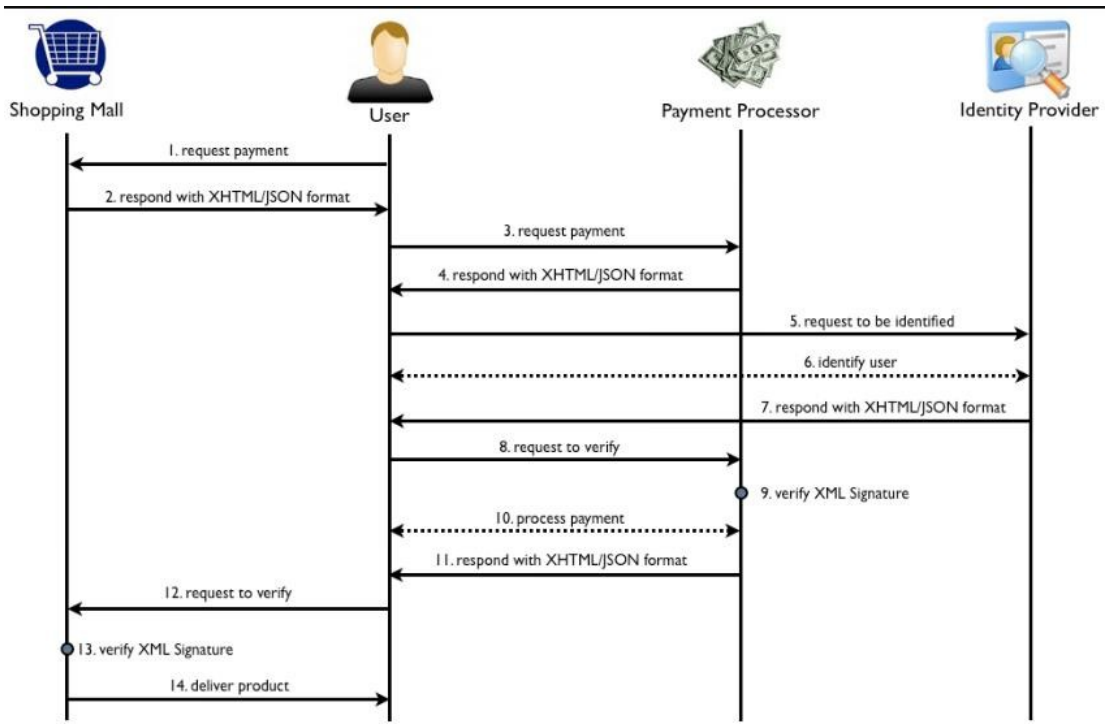
Actors - something with a behaviour or role, e.g., a person, another system, organization.


E-commerce application system
Manage accounts
Delete accounts
Admin
Log in
Buy products
Add products
Search for products
Delete products
Add new address
Seller
Customer
Edit products
Browse products
Create Account
Edit account details

Sequence Diagram

A sequence diagram in the context of an e-commerce system would show the interactions between various objects, such as a customer, a shopping cart, a payment gateway, and a database, as the customer interacts with the system to purchase a product.

- The customer opens the e-commerce website and browses the catalog.
- The customer selects a product and adds it to the shopping cart.
- The customer proceeds to checkout and enters shipping and billing information.
- The system sends the customer's information to the payment gateway for verification and processing.
- The payment gateway sends a response to the system indicating whether the payment was successful or not.
- If the payment is successful, the system sends the customer's order details to the database for storage and sends an order confirmation to the customer.
- If the payment is not successful, the system sends an error message to the customer and prompts them to try again.



Conclusion

The conclusion of an e-commerce project would summarize the key findings and accomplishments of the project. It would also discuss any challenges or limitations encountered during the project, and suggest areas for future improvement. Some possible points that could be included in a conclusion for an e-commerce project are:

- A description of the e-commerce platform developed and its key features
- An overview of the user testing and feedback received
- A discussion of any technical challenges encountered and how they were overcome
- A summary of key metrics such as website traffic, conversion rates, and sales
- An analysis of the business impact of the project, including increasedrevenue, customer satisfaction, and market share
- Suggestions for future enhancements such as new features, integrations,or marketing strategies

Scope For Future Development

There are many areas for future development in e-commerce, here are a fewexamples:

- **Personalization**: With the rise of artificial intelligence, machine learning, and big data, e-commerce companies can personalize the customer experience by recommending products based on the customer's browsing and purchase history.

- **Mobile optimization**: As more and more customers use their smartphones to shop online, it's important for e-commerce companies to optimize their websites and apps for mobile devices.

- **Augmented reality**: Augmented reality can be used to enhance the online shopping experience by allowing customers to virtually try on clothes or see how furniture would look in their home.

- **Same-day and on-demand delivery:** Customers are becoming increasingly demanding when it comes to delivery times, and e -commerce companies will need to adapt to meet these demands by offering same - day and on-demand delivery options.

- **Social commerce**: Social media platforms are becoming increasingly important in e-commerce, as more and more customers use them to discover and purchase products.

- **Blockchain technology**: Blockchain technology can be used to improve trust and security in e-commerce transactions.

- **Voice commerce**: With the popularity of voice assistants such as Amazon's Alexa, Google Home, and Apple's Siri, e-commerce companies need to explore opportunities for voice-enabled commerce.

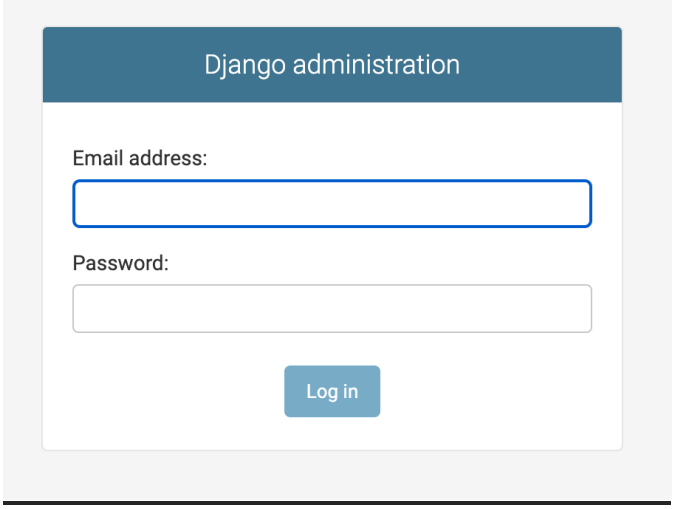
- **Subscription-based models**: Companies will need to explore new ways to generate recurring revenue through subscription-based models.

- **Omnichannel**: Omnichannel is the integration of all channels, to make the customer experience seamless across all touchpoints.

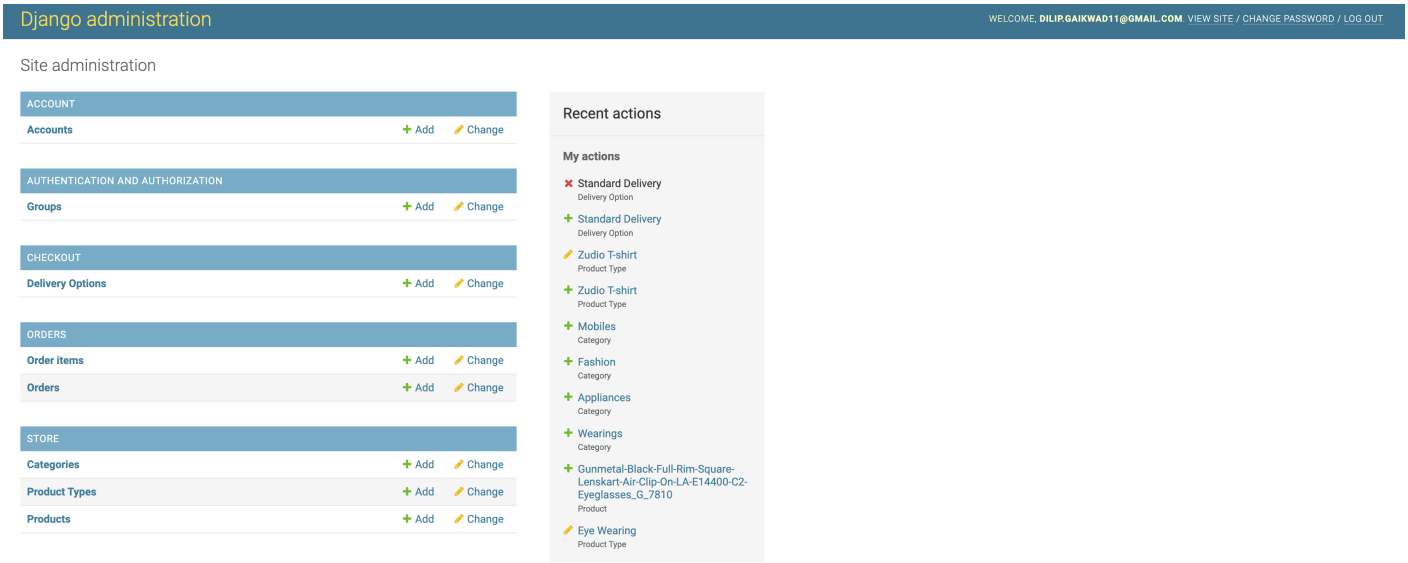
CHAPTER -4

UI DESIGN

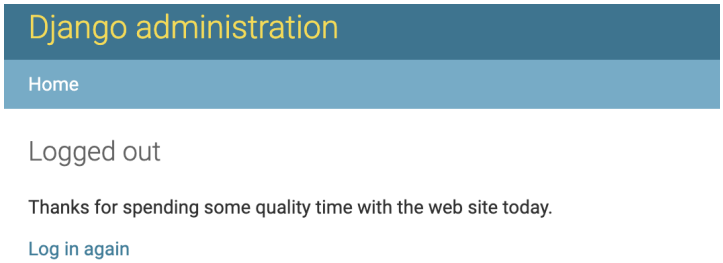
Admin Login



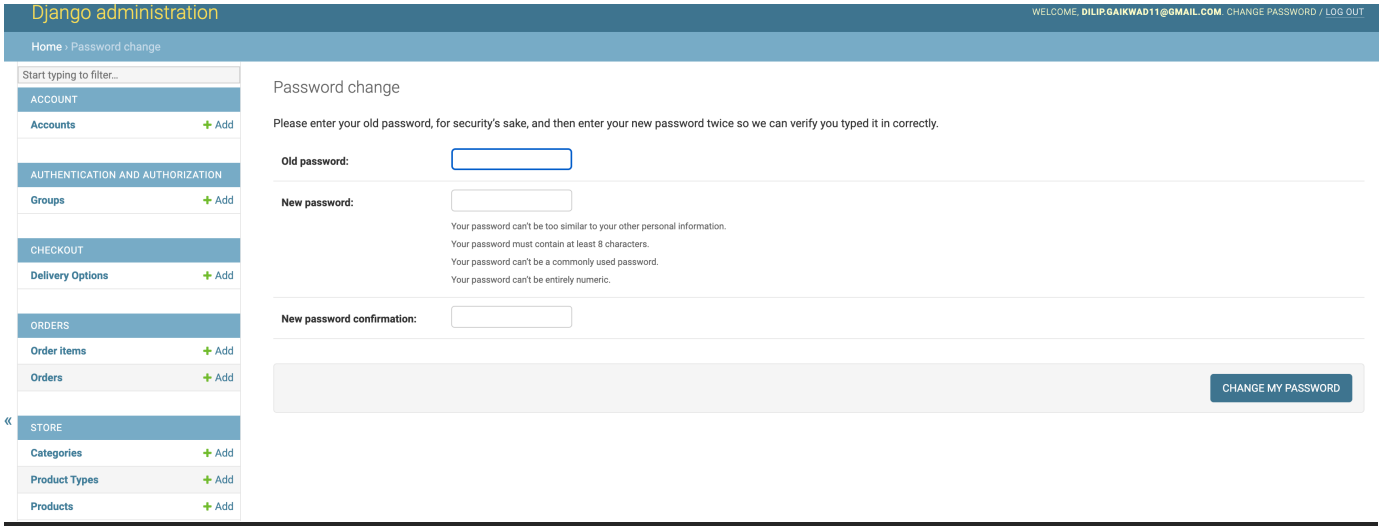
Admin Homepage



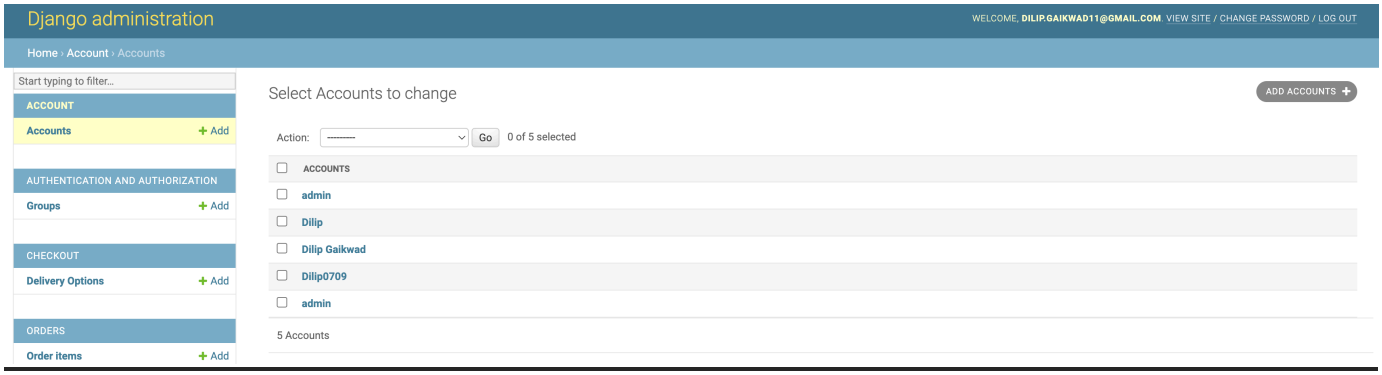
Logout



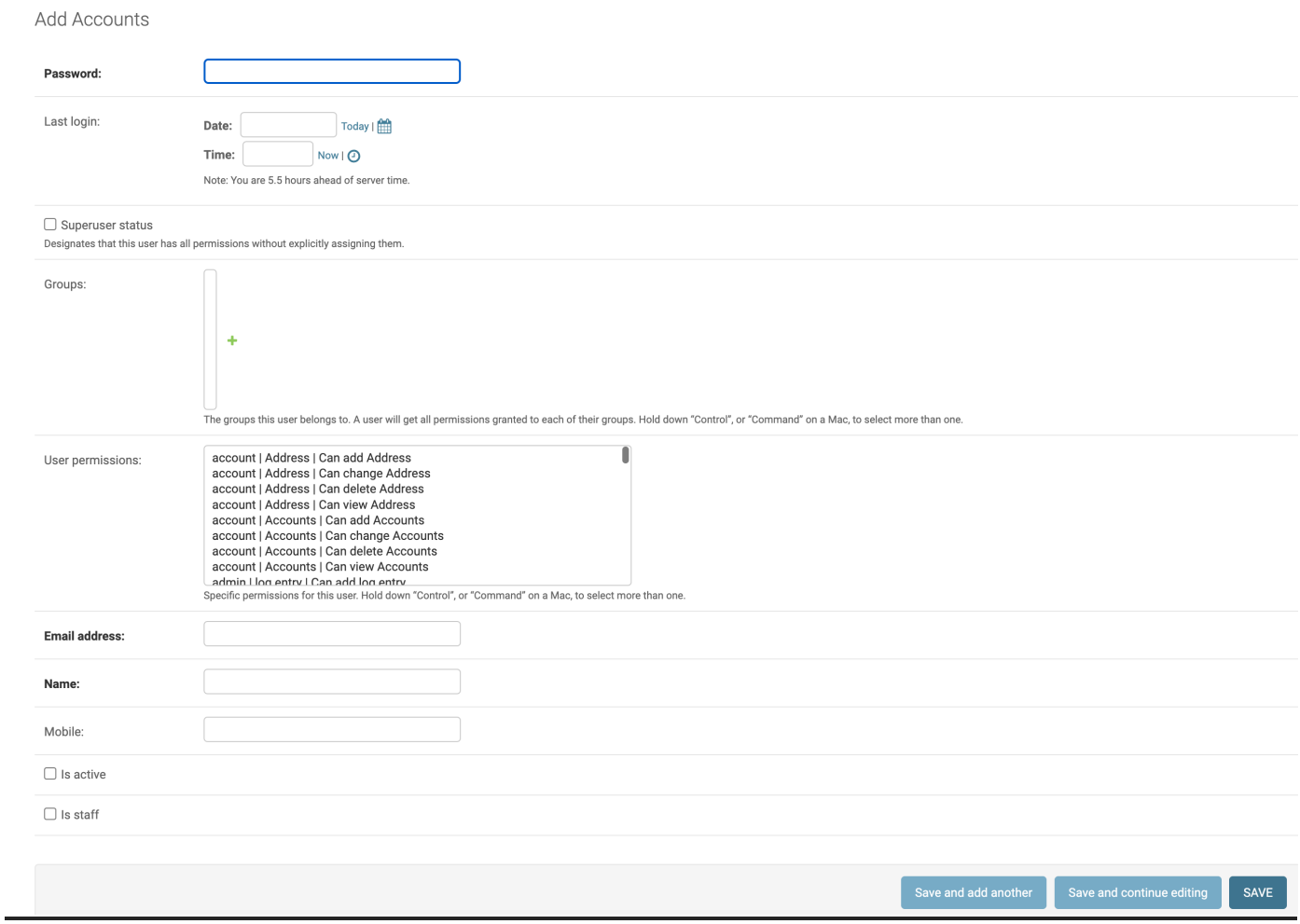
Password Change



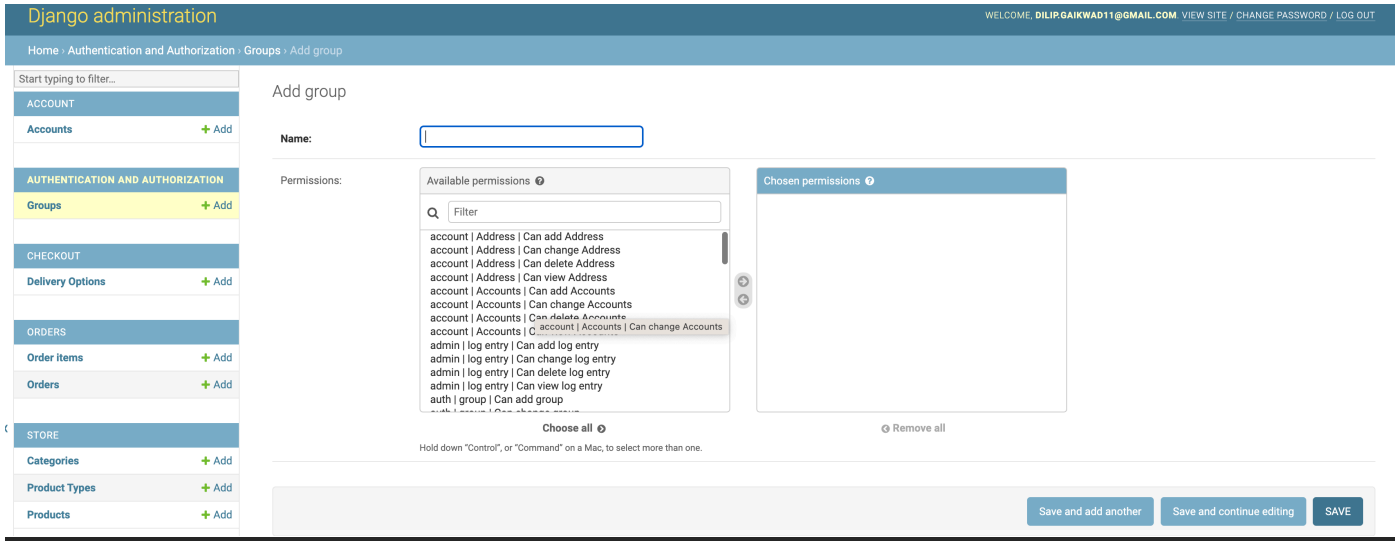
Account Page



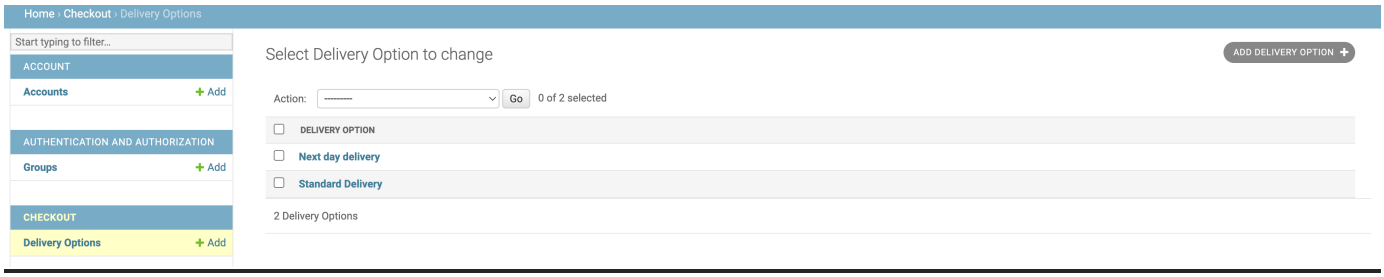
Add Account Page



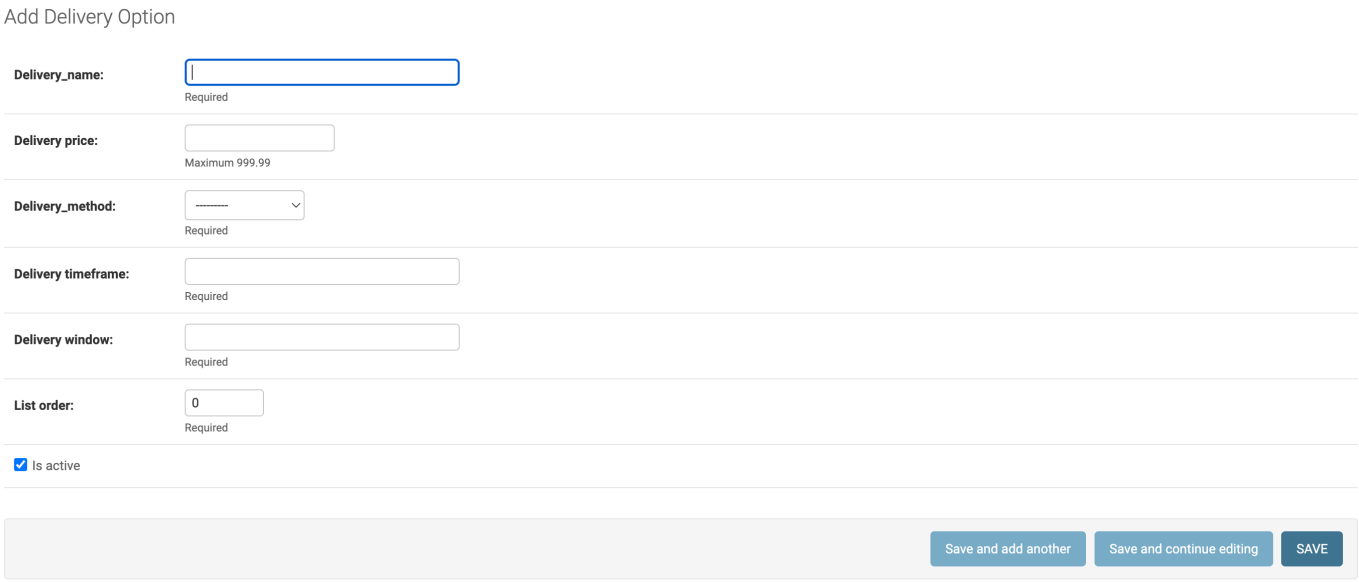
Add Group Page



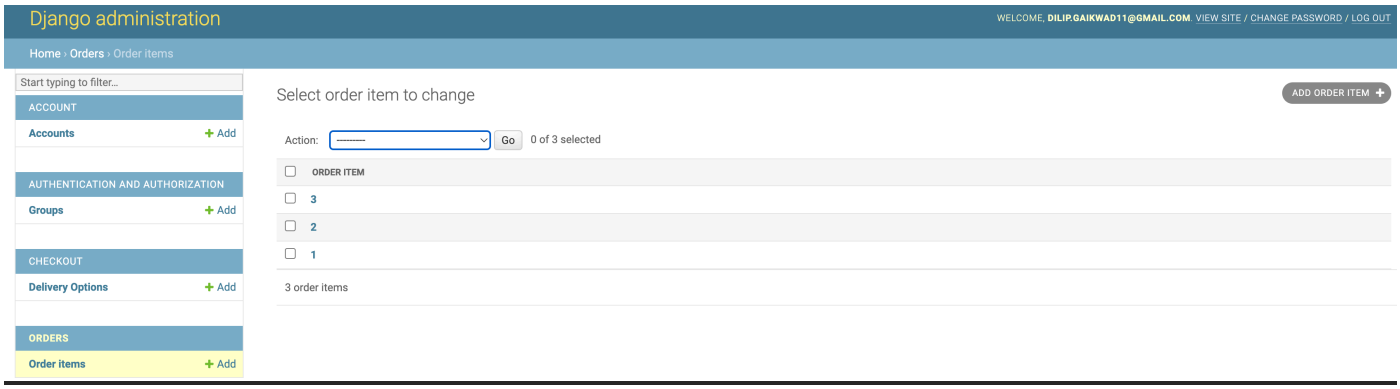
Delivery Option Management



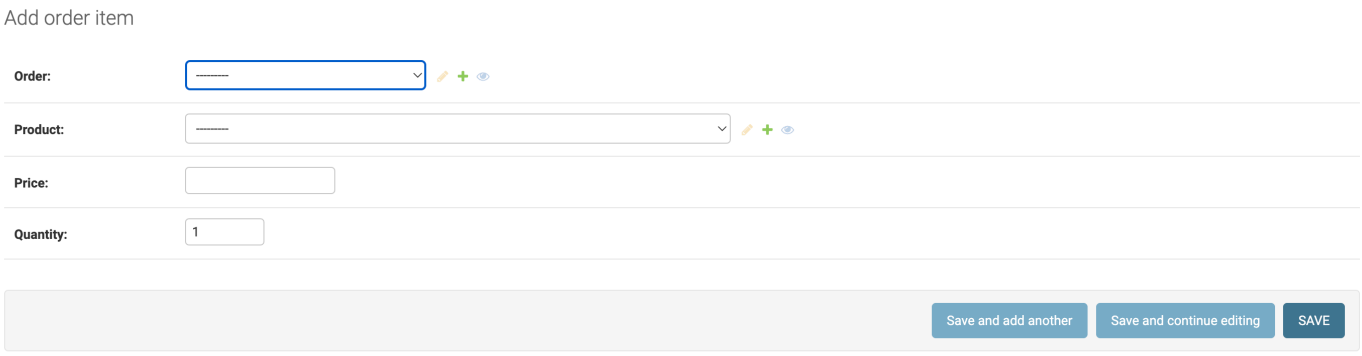
Add Delivery Option



Order Management Page

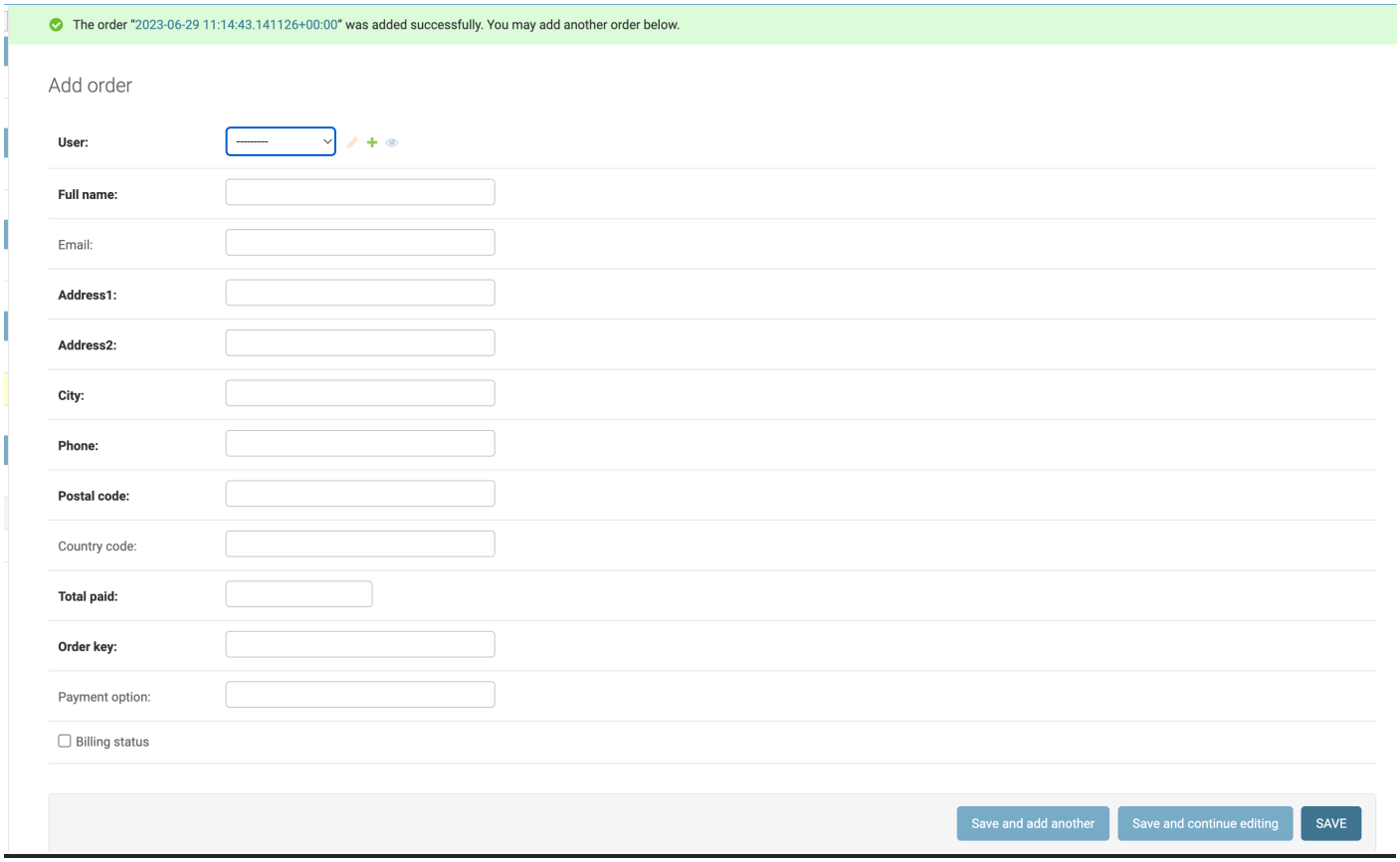


Add Order Item



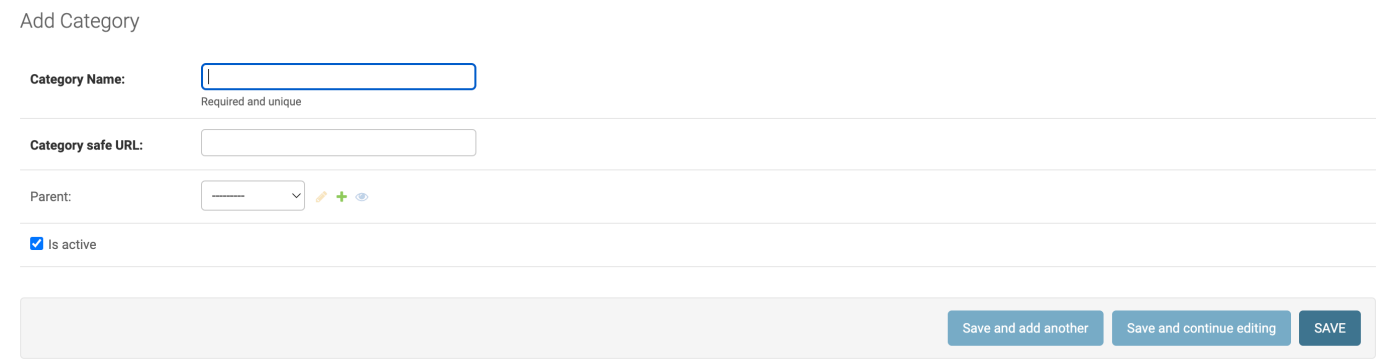
Change Order Page

Add Order Page



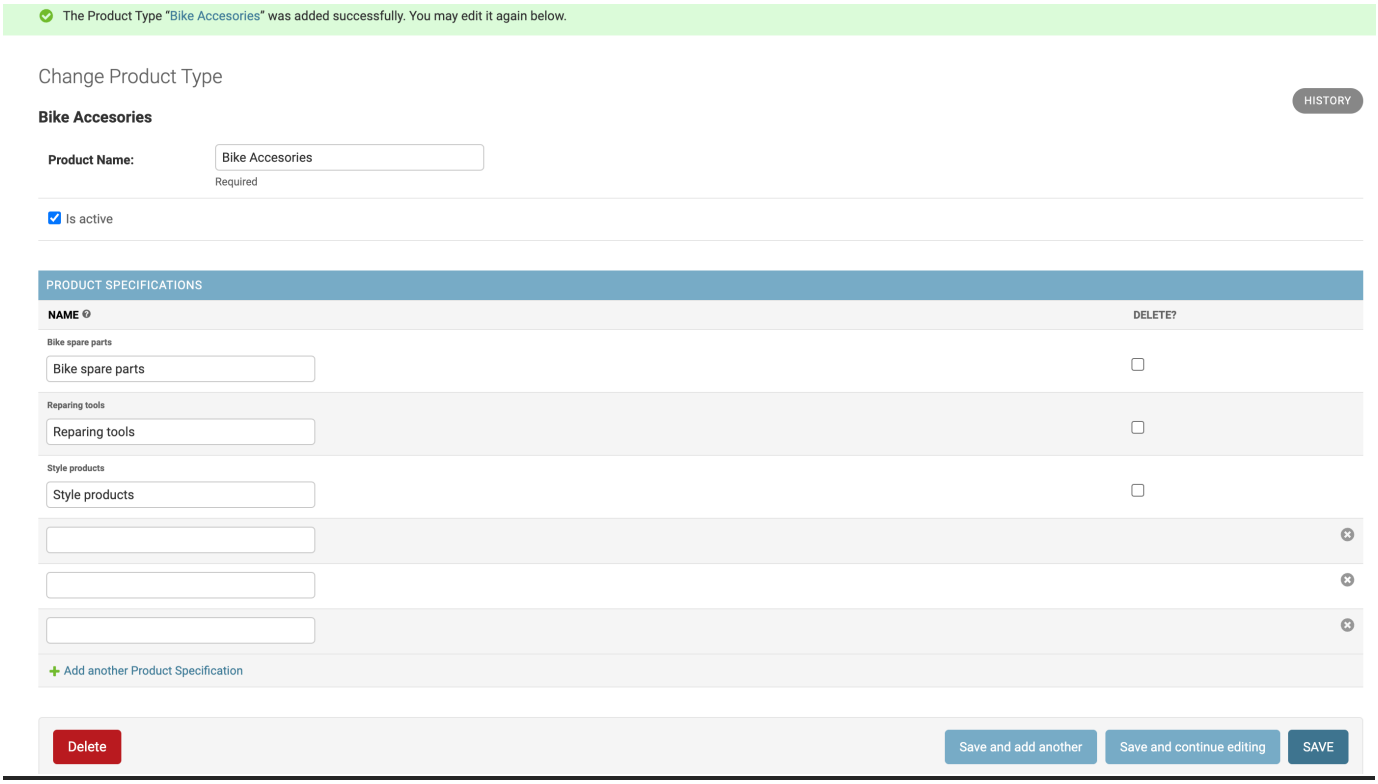
Category Management Page

Add Category Page

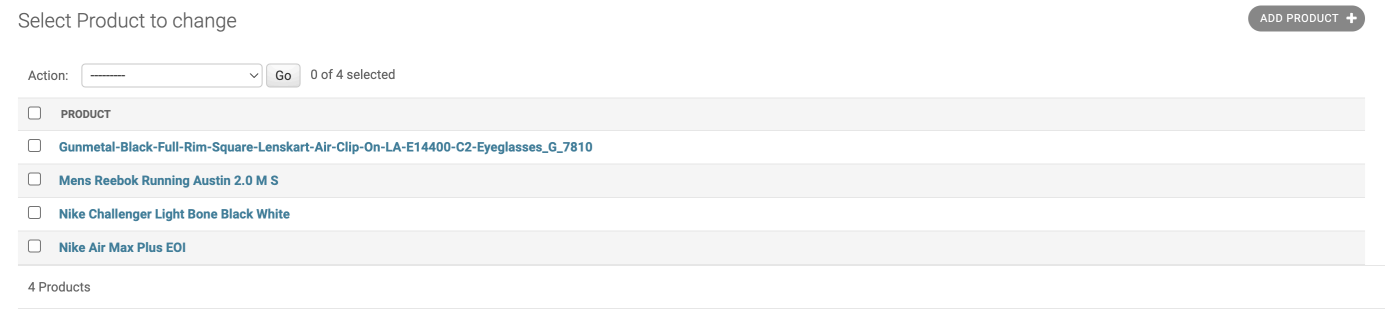


Product Type Management

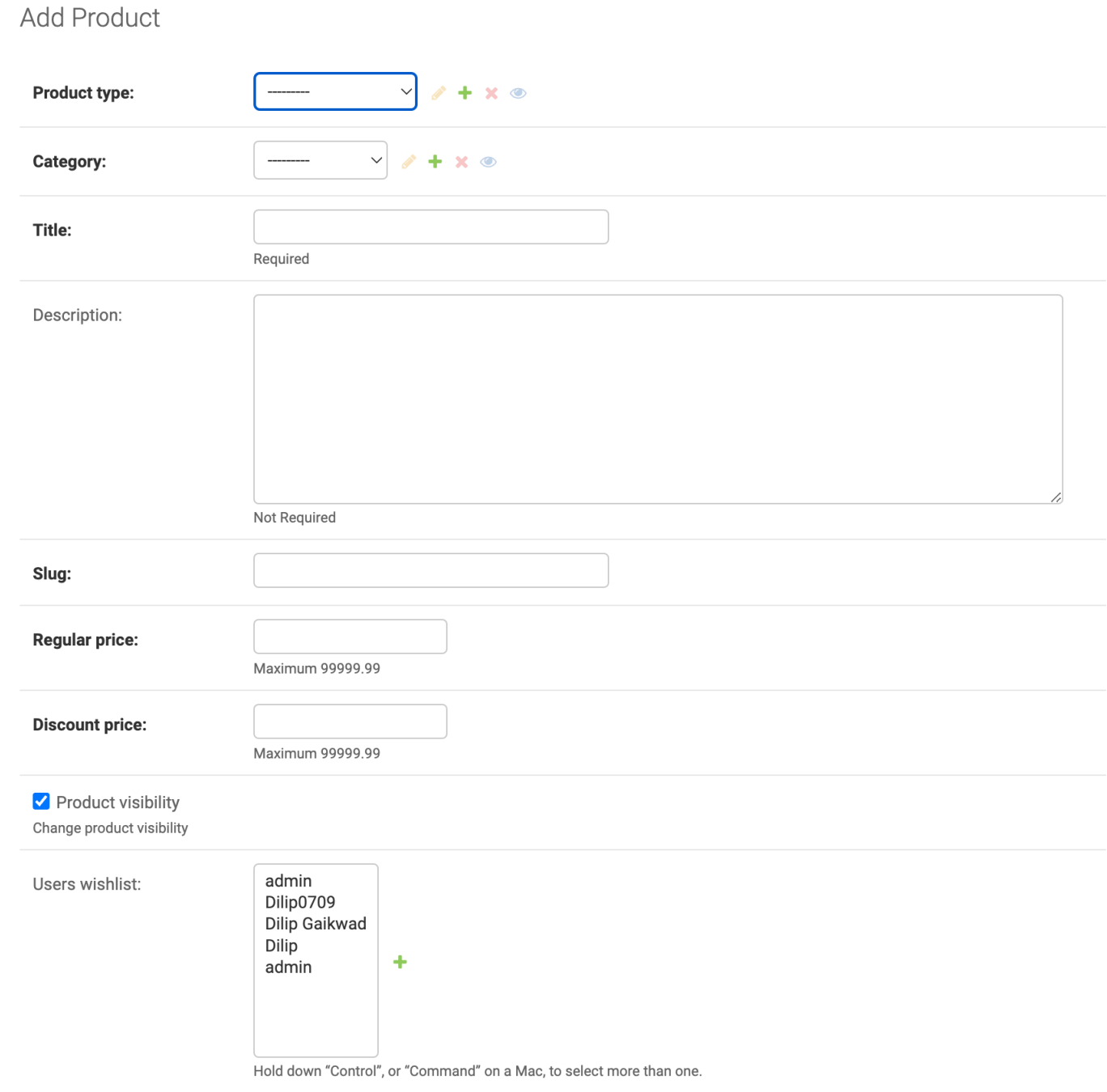
Add Product Type



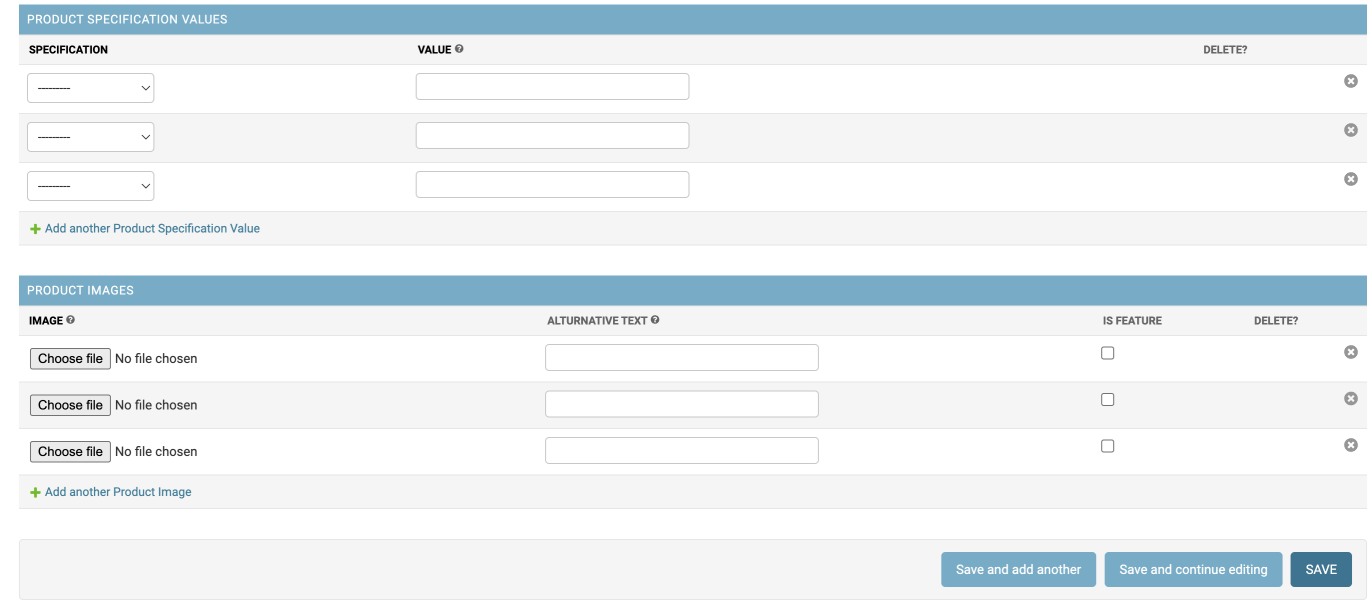
Product Management page



Add Product Page

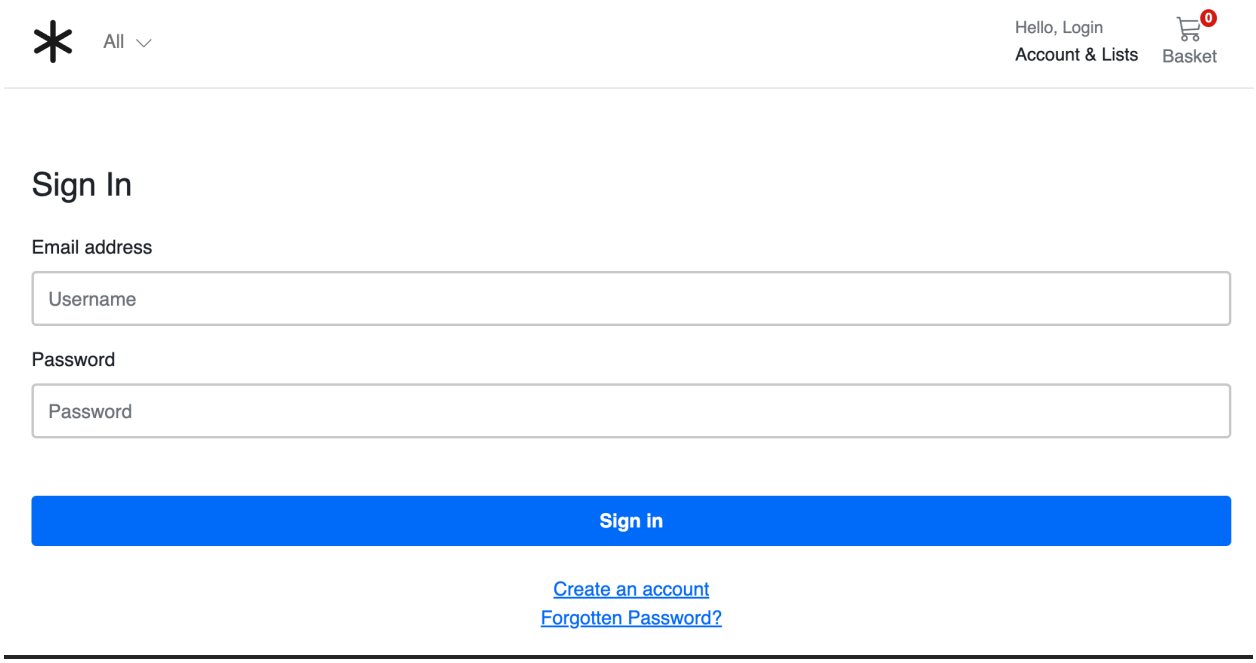


Product Specification page

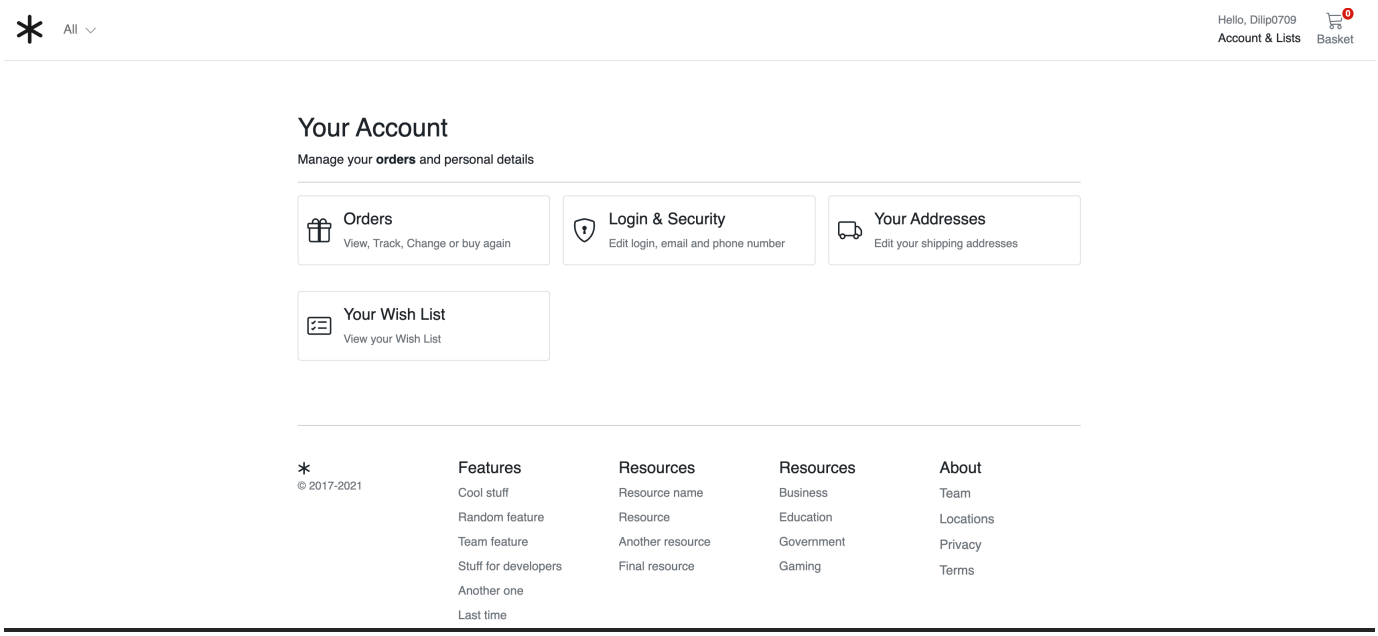


Frontend UI

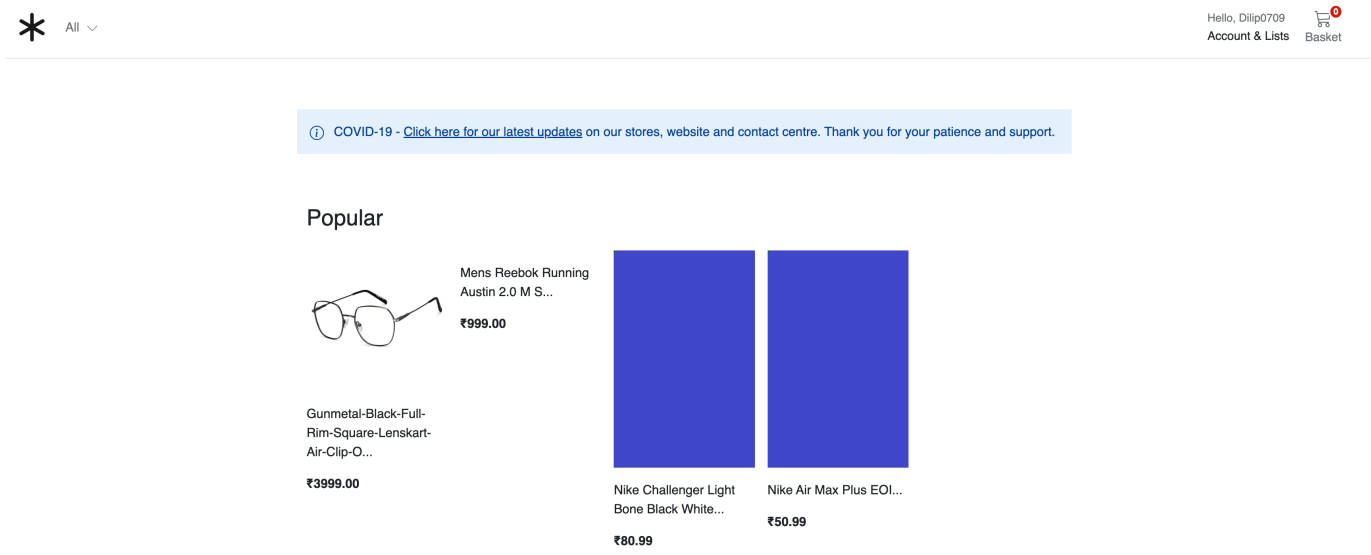
Sign In Page



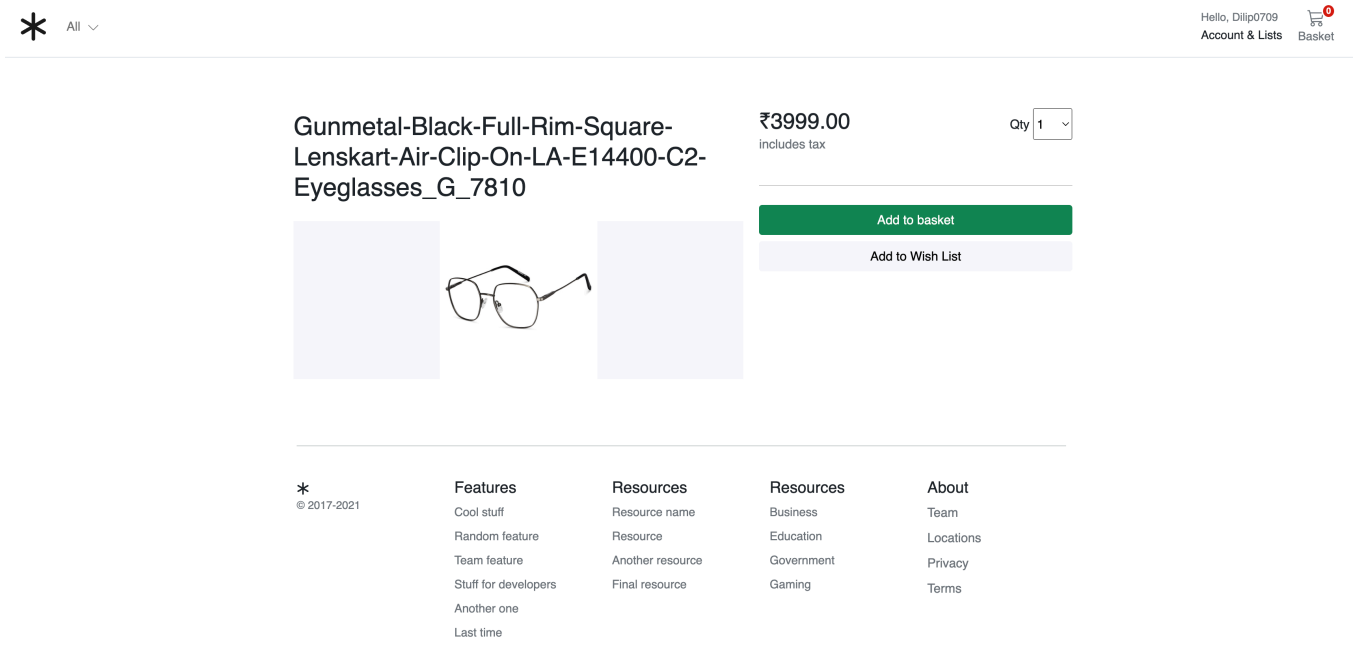
Account Page



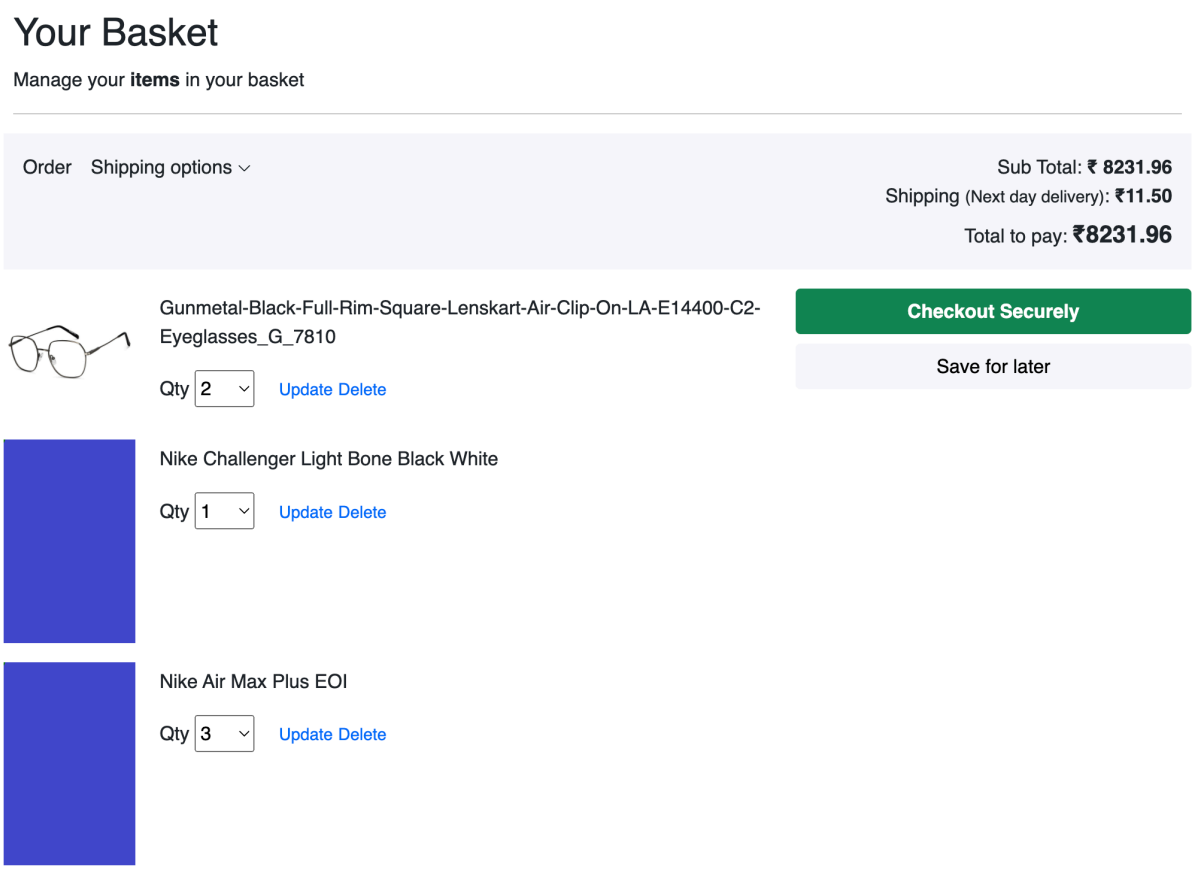
Homepage



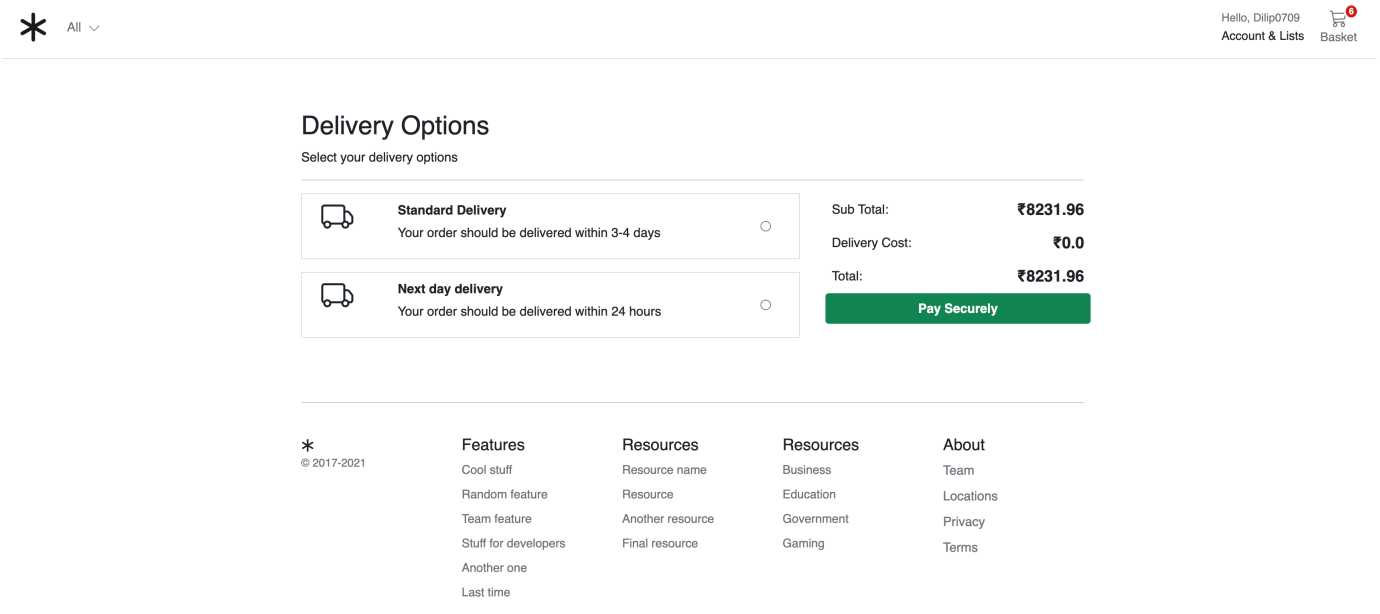
Purchase Page



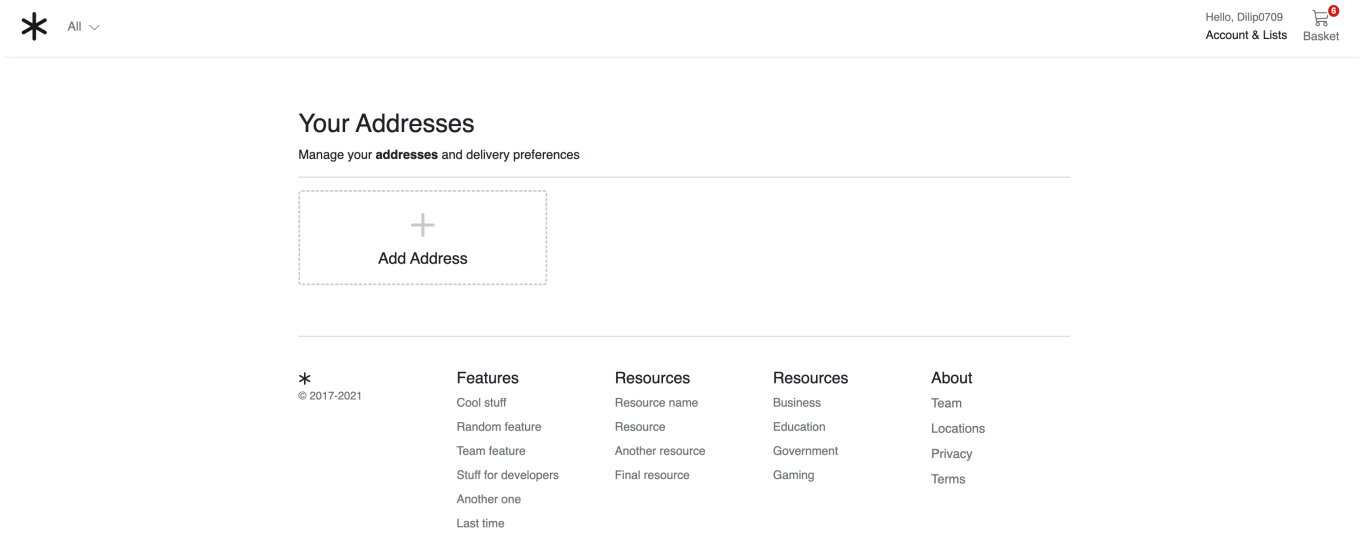
Basket Page



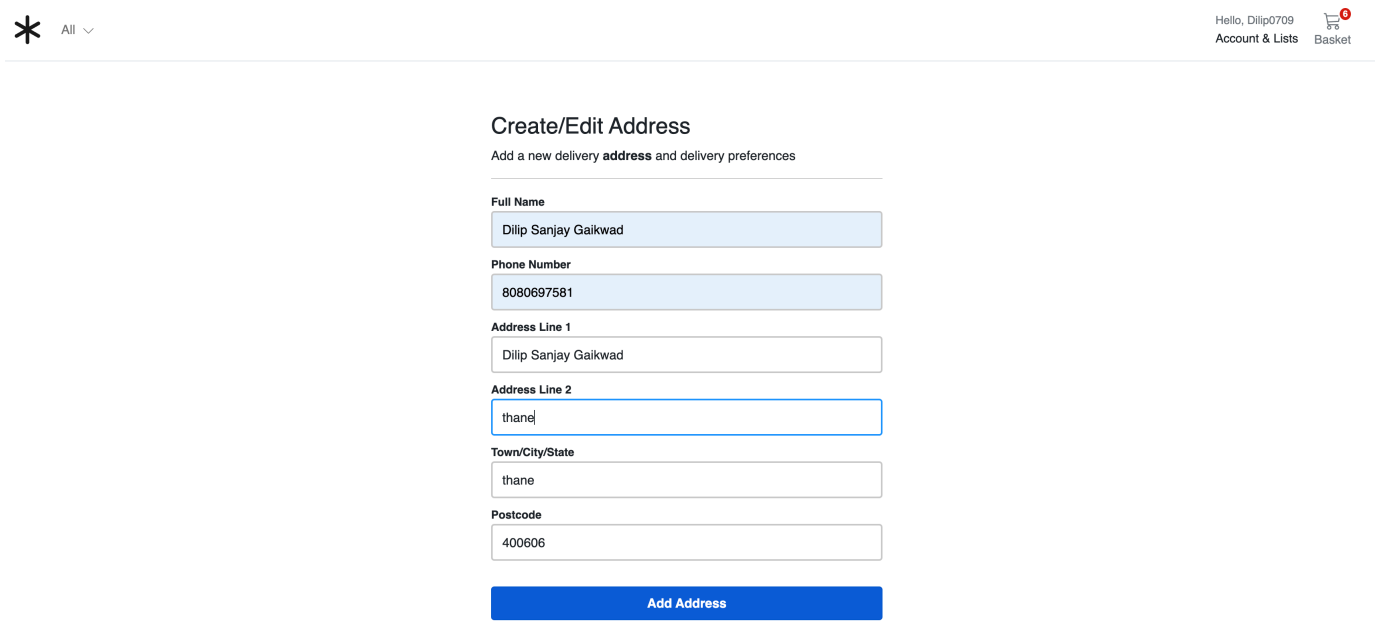
Delivery Option Page



Address Page



Add Address Page



Add Wishlist

All

Hello, Dilip0709
Account & Lists

Basket 0

My Wishlist

Manage your Wishlist

Gunmetal-Black-Full-Rim-Square-Lenskart-Air-Clip-On-LA-E14400-C2-Eyeglasses_G_7810

Gunmetal-Black-Full-Rim-Square-Lenskart-Air-Clip-On-LA-E14400-C2-Eyeglasses_G_7810

3999.00

Remove from Wishlist

Nike Challenger Light Bone Black White

Nike Challenger Light Bone Black White

80.99

Remove from Wishlist

© 2017-2021

Features	Resources	Resources	About
Cool stuff	Resource name	Business	Team
Random feature	Resource	Education	Locations
Team feature	Another resource	Government	Privacy
Stuff for developers	Final resource	Gaming	Terms

Create Account Page

All

Hello, Login
Account & Lists

Basket 0

Create an account

It's free and only takes a minute.

Enter Username (Required)

Username

Email (Required)

E-mail

Password At least 8 characters and 1 digit

Password

Please fill in this field.

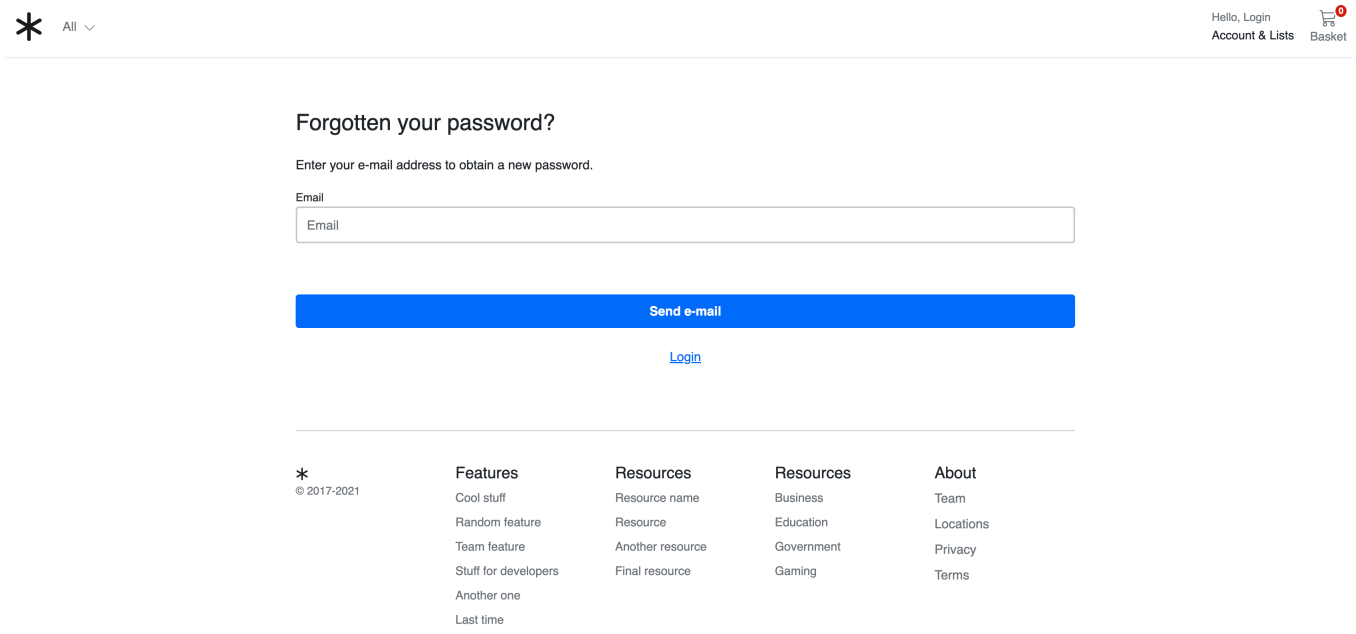
Repeat password (Required)

Repeat Password

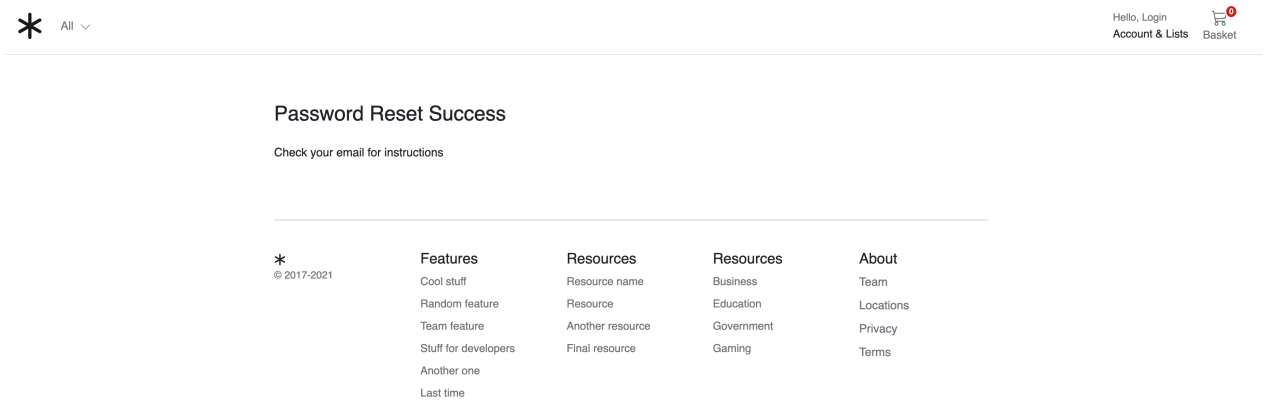
Register

Already have an account?

Forgot Password Page



Password reset




```
MIME-Version: 1.0
Content-Transfer-Encoding: 8bit
Subject: Password reset on 127.0.0.1:8000
From: webmaster@localhost
To: dilip.gaikwad11@gmail.com
Date: Thu, 29 Jun 2023 13:24:11 -0000
Message-ID: <168804505113.84465.4417364042482404478@Dilips-MacBook-Pro.local>


You're receiving this email because you requested a password reset for your user account at 127.0.0.1:8000.

Please go to the following page and choose a new password:

http://127.0.0.1:8000/account/password_reset_confirm/Mg/bqk58b-b2573cec3f8c3615964d0a271bc37441

Your username, in case you've forgotten: dilip.gaikwad11@gmail.com

Thanks for using our site!

The 127.0.0.1:8000 team
```

Account

Code: *models.py*

```
import uuid

from django.contrib.auth.models import (AbstractBaseUser,
BaseUserManager,
                                        PermissionsMixin)
from django.core.mail import send_mail
from django.db import models
from django.utils.translation import gettext_lazy as _


class CustomAccountManager(BaseUserManager):

    def create_superuser(self, email, name, password, **other_fields):

        other_fields.setdefault('is_staff', True)
        other_fields.setdefault('is_superuser', True)
        other_fields.setdefault('is_active', True)

        if other_fields.get('is_staff') is not True:
            raise ValueError(
                'Superuser must be assigned to is_staff=True.')
        if other_fields.get('is_superuser') is not True:
            raise ValueError(
                'Superuser must be assigned to is_superuser=True.')

        return self.create_user(email, name, password, **other_fields)

    def create_user(self, email, name, password, **other_fields):

        if not email:
            raise ValueError(_('You must provide an email address'))

        email = self.normalize_email(email)
        user = self.model(email=email, name=name,
                          **other_fields)
        user.set_password(password)
        user.save()
        return user
```

```
class Customer(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(_('email address'), unique=True)
    name = models.CharField(max_length=150)
    mobile = models.CharField(max_length=20, blank=True)
    is_active = models.BooleanField(default=False)
    is_staff = models.BooleanField(default=False)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    objects = CustomAccountManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['name']

    class Meta:
        verbose_name = "Accounts"
        verbose_name_plural = "Accounts"

    def email_user(self, subject, message):
        send_mail(
            subject,
            message,
            'l@1.com',
            [self.email],
            fail_silently=False,
        )

    def __str__(self):
        return self.name

class Address(models.Model):
    """
    Address
    """

    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
editable=False)
    customer = models.ForeignKey(Customer, verbose_name=_("Customer"),
on_delete=models.CASCADE)
    full_name = models.CharField(_("Full Name"), max_length=150)
    phone = models.CharField(_("Phone Number"), max_length=50)
    postcode = models.CharField(_("Postcode"), max_length=50)
    address_line = models.CharField(_("Address Line 1"),
max_length=255)
    address_line2 = models.CharField(_("Address Line 2"),
max_length=255)
    town_city = models.CharField(_("Town/City/State"), max_length=150)
```

```
    delivery_instructions = models.CharField(_("Delivery
Instructions"), max_length=255)
    created_at = models.DateTimeField(_("Created at"),
auto_now_add=True)
    updated_at = models.DateTimeField(_("Updated at"), auto_now=True)
    default = models.BooleanField(_("Default"), default=False)

    class Meta:
        verbose_name = "Address"
        verbose_name_plural = "Addresses"

    def __str__(self):
        return "Address"
```

Views.py

```
import django
from django.utils.encoding import force_str
django.utils.encoding.force_text = force_str
from django.contrib import messages
from django.contrib.auth import login, logout
from django.contrib.auth.decorators import login_required
from django.contrib.sites.shortcuts import get_current_site
from django.http import HttpResponse, HttpResponseRedirect
from django.shortcuts import get_object_or_404, redirect, render
from django.template.loader import render_to_string
from django.urls import reverse
from django.utils.encoding import force_bytes, force_text
from django.utils.http import urlsafe_base64_decode,
urlsafe_base64_encode
from orders.models import Order
from orders.views import user_orders
from store.models import Product

from .forms import RegistrationForm, UserAddressForm, UserEditForm
from .models import Address, Customer
from .tokens import account_activation_token


@login_required
def wishlist(request):
    products = Product.objects.filter(users_wishlist=request.user)
    return render(request, "account/dashboard/user_wish_list.html",
{"wishlist": products})


@login_required
def add_to_wishlist(request, id):
    product = get_object_or_404(Product, id=id)
    if product.users_wishlist.filter(id=request.user.id).exists():
        product.users_wishlist.remove(request.user)
        messages.success(request, product.title + " has been removed
from your WishList")
    else:
        product.users_wishlist.add(request.user)
        messages.success(request, "Added " + product.title + " to your
WishList")
    return HttpResponseRedirect(request.META["HTTP_REFERER"])


@login_required
```

```
def dashboard(request):
    orders = user_orders(request)
    return render(request, "account/dashboard/dashboard.html",
{"section": "profile", "orders": orders})


@login_required
def edit_details(request):
    if request.method == "POST":
        user_form = UserEditForm(instance=request.user,
data=request.POST)

        if user_form.is_valid():
            user_form.save()
    else:
        user_form = UserEditForm(instance=request.user)

    return render(request, "account/dashboard/edit_details.html",
{"user_form": user_form})


@login_required
def delete_user(request):
    user = Customer.objects.get(user_name=request.user)
    user.is_active = False
    user.save()
    logout(request)
    return redirect("account:delete_confirmation")


def account_register(request):

    if request.user.is_authenticated:
        return redirect("account:dashboard")

    if request.method == "POST":
        registerForm = RegistrationForm(request.POST)
        if registerForm.is_valid():
            user = registerForm.save(commit=False)
            user.email = registerForm.cleaned_data["email"]
            user.set_password(registerForm.cleaned_data["password"])
            user.is_active = False
            user.save()
            current_site = get_current_site(request)
            subject = "Activate your Account"
            message = render_to_string(
                "account/registration/account_activation_email.html",
```

```
                {
                    "user": user,
                    "domain": current_site.domain,
                    "uid": urlsafe_base64_encode(force_bytes(user.pk)),
                    "token": account_activation_token.make_token(user),
                },
            )
            user.email_user(subject=subject, message=message)
            return render(request,
"account/registration/register_email_confirm.html", {"form":
registerForm})
    else:
        registerForm = RegistrationForm()
    return render(request, "account/registration/register.html",
{"form": registerForm})


def account_activate(request, uidb64, token):
    try:
        uid = force_text(urlsafe_base64_decode(uidb64))
        user = Customer.objects.get(pk=uid)
    except (TypeError, ValueError, OverflowError, user.DoesNotExist):
        user = None
    if user is not None and account_activation_token.check_token(user,
token):
        user.is_active = True
        user.save()
        login(request, user)
        return redirect("account:dashboard")
    else:
        return render(request,
"account/registration/activation_invalid.html")


# Addresses


@login_required
def view_address(request):
    addresses = Address.objects.filter(customer=request.user)
    return render(request, "account/dashboard/addresses.html",
{"addresses": addresses})


@login_required
def add_address(request):
    if request.method == "POST":
```

```
        address_form = UserAddressForm(data=request.POST)
        if address_form.is_valid():
            address_form = address_form.save(commit=False)
            address_form.customer = request.user
            address_form.save()
            return HttpResponseRedirect(reverse("account:addresses"))
    else:
        address_form = UserAddressForm()
    return render(request, "account/dashboard/edit_addresses.html",
{"form": address_form})


@login_required
def edit_address(request, id):
    if request.method == "POST":
        address = Address.objects.get(pk=id, customer=request.user)
        address_form = UserAddressForm(instance=address,
data=request.POST)
        if address_form.is_valid():
            address_form.save()
            return HttpResponseRedirect(reverse("account:addresses"))
    else:
        address = Address.objects.get(pk=id, customer=request.user)
        address_form = UserAddressForm(instance=address)
    return render(request, "account/dashboard/edit_addresses.html",
{"form": address_form})


@login_required
def delete_address(request, id):
    address = Address.objects.filter(pk=id,
customer=request.user).delete()
    return redirect("account:addresses")


@login_required
def set_default(request, id):
    Address.objects.filter(customer=request.user,
default=True).update(default=False)
    Address.objects.filter(pk=id,
customer=request.user).update(default=True)

    previous_url = request.META.get("HTTP_REFERER")

    if "delivery_address" in previous_url:
        return redirect("checkout:delivery_address")
```

```
    return redirect("account:addresses")


@login_required
def user_orders(request):
    user_id = request.user.id
    orders =
Order.objects.filter(user_id=user_id).filter(billing_status=True)
    return render(request, "account/dashboard/user_orders.html",
{"orders": orders})
```

urls.py

```
from django.contrib.auth import views as auth_views
from django.urls import path
from django.views.generic import TemplateView

from . import views
from .forms import PwdResetConfirmForm, PwdResetForm, UserLoginForm

# https://docs.djangoproject.com/en/3.1/topics/auth/default/
#
https://ccbv.co.uk/projects/Django/3.0/django.contrib.auth.views/PasswordResetConfirmView/

app_name = "account"

urlpatterns = [
    path(
        "login/",

auth_views.LoginView.as_view(template_name="account/login.html",
form_class=UserLoginForm),
        name="login",
    ),
    path("logout/",
auth_views.LogoutView.as_view(next_page="/account/login/"),
name="logout"),
    path("register/", views.account_register, name="register"),
    path("activate/<slug:uidb64>/<slug:token>)/",
views.account_activate, name="activate"),
    # Reset password
    path(
        "password_reset/",
        auth_views.PasswordResetView.as_view(

template_name="account/password_reset/password_reset_form.html",
            success_url="password_reset_email_confirm",

email_template_name="account/password_reset/password_reset_email.html",
            form_class=PwdResetForm,
        ),
        name="pwdreset",
    ),
    path(
        "password_reset_confirm/<uidb64>/<token>",
        auth_views.PasswordResetConfirmView.as_view(
```

```
template_name="account/password_reset/password_reset_confirm.html",
            success_url="password_reset_complete/",
            form_class=PwdResetConfirmForm,
        ),
        name="password_reset_confirm",
    ),
    path(
        "password_reset/password_reset_email_confirm/",

TemplateView.as_view(template_name="account/password_reset/reset_status
.html"),
        name="password_reset_done",
    ),
    path(
        "password_reset_confirm/Mg/password_reset_complete/",

TemplateView.as_view(template_name="account/password_reset/reset_status
.html"),
        name="password_reset_complete",
    ),
    # User dashboard
    path("dashboard/", views.dashboard, name="dashboard"),
    path("profile/edit/", views.edit_details, name="edit_details"),
    path("profile/delete_user/", views.delete_user,
name="delete_user"),
    path(
        "profile/delete_confirm/",

TemplateView.as_view(template_name="account/dashboard/delete_confirm.ht
ml"),
        name="delete_confirmation",
    ),
    path("addresses/", views.view_address, name="addresses"),
    path("add_address/", views.add_address, name="add_address"),
    path("addresses/edit/<slug:id>/", views.edit_address,
name="edit_address"),
    path("addresses/delete/<slug:id>/", views.delete_address,
name="delete_address"),
    path("addresses/set_default/<slug:id>/", views.set_default,
name="set_default"),
    path("user_orders/", views.user_orders, name="user_orders"),
    # Wish List
    path("wishlist", views.wishlist, name="wishlist"),
    path("wishlist/add_to_wishlist/<int:id>", views.add_to_wishlist,
name="user_wishlist"),
]
```

Forms.py

```
from django import forms
from django.contrib.auth.forms import (AuthenticationForm,
PasswordResetForm,
                                       SetPasswordForm)

from .models import Customer, Address

class UserAddressForm(forms.ModelForm):
    class Meta:
        model = Address
        fields = ["full_name", "phone", "address_line", "address_line2",
"town_city", "postcode"]

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields["full_name"].widget.attrs.update(
            {"class": "form-control mb-2 account-form", "placeholder":
"Full Name"}
        )
        self.fields["phone"].widget.attrs.update({"class": "form-control
mb-2 account-form", "placeholder": "Phone"})
        self.fields["address_line"].widget.attrs.update(
            {"class": "form-control mb-2 account-form", "placeholder":
"Full Name"}
        )
        self.fields["address_line2"].widget.attrs.update(
            {"class": "form-control mb-2 account-form", "placeholder":
"Full Name"}
        )
        self.fields["town_city"].widget.attrs.update(
            {"class": "form-control mb-2 account-form", "placeholder":
"Full Name"}
        )
        self.fields["postcode"].widget.attrs.update(
            {"class": "form-control mb-2 account-form", "placeholder":
"Full Name"}
        )



class UserLoginForm(AuthenticationForm):

    username = forms.CharField(widget=forms.TextInput(
        attrs={'class': 'form-control mb-3', 'placeholder': 'Username',
'id': 'login-username'}))
```

```
    password = forms.CharField(widget=forms.PasswordInput(
        attrs={
            'class': 'form-control',
            'placeholder': 'Password',
            'id': 'login-pwd',
        }
    ))


class RegistrationForm(forms.ModelForm):

    user_name = forms.CharField(label='Enter Username', min_length=4,
max_length=50, help_text='Required')
    email = forms.EmailField(max_length=100, help_text='Required',
error_messages={'required': 'Sorry, you will need an email'})
    password = forms.CharField(label='Password',
widget=forms.PasswordInput)
    password2 = forms.CharField(
        label='Repeat password', widget=forms.PasswordInput)

    class Meta:
        model = Customer
        fields = ('user_name', 'email',)

    def clean_username(self):
        user_name = self.cleaned_data['user_name'].lower()
        r = Customer.objects.filter(user_name=user_name)
        if r.count():
            raise forms.ValidationError("Username already exists")
        return user_name

    def clean_password2(self):
        cd = self.cleaned_data
        if cd['password'] != cd['password2']:
            raise forms.ValidationError('Passwords do not match.')
        return cd['password2']

    def clean_email(self):
        email = self.cleaned_data['email']
        if Customer.objects.filter(email=email).exists():
            raise forms.ValidationError(
                'Please use another Email, that is already taken')
        return email

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['user_name'].widget.attrs.update(
```

```
            {'class': 'form-control mb-3', 'placeholder': 'Username'})
        self.fields['email'].widget.attrs.update(
            {'class': 'form-control mb-3', 'placeholder': 'E-mail',
'name': 'email', 'id': 'id_email'})
        self.fields['password'].widget.attrs.update(
            {'class': 'form-control mb-3', 'placeholder': 'Password'})
        self.fields['password2'].widget.attrs.update(
            {'class': 'form-control', 'placeholder': 'Repeat Password'})


class PwdResetForm(PasswordResetForm):

    email = forms.EmailField(max_length=254, widget=forms.TextInput(
        attrs={'class': 'form-control mb-3', 'placeholder': 'Email',
'id': 'form-email'}))

    def clean_email(self):
        email = self.cleaned_data['email']
        u = Customer.objects.filter(email=email)
        if not u:
            raise forms.ValidationError(
                'Unfortunatley we can not find that email address')
        return email


class PwdResetConfirmForm(SetPasswordForm):
    new_password1 = forms.CharField(
        label='New password', widget=forms.PasswordInput(
            attrs={'class': 'form-control mb-3', 'placeholder': 'New
Password', 'id': 'form-newpass'}))
    new_password2 = forms.CharField(
        label='Repeat password', widget=forms.PasswordInput(
            attrs={'class': 'form-control mb-3', 'placeholder': 'New
Password', 'id': 'form-new-pass2'}))


class UserEditForm(forms.ModelForm):

    email = forms.EmailField(
        label='Account email (can not be changed)', max_length=200,
widget=forms.TextInput(
            attrs={'class': 'form-control mb-3', 'placeholder': 'email',
'id': 'form-email', 'readonly': 'readonly'}))

    user_name = forms.CharField(
        label='Firstname', min_length=4, max_length=50,
widget=forms.TextInput(
```

```
            attrs={'class': 'form-control mb-3', 'placeholder':
'Username', 'id': 'form-firstname', 'readonly': 'readonly'}))

    first_name = forms.CharField(
        label='Username', min_length=4, max_length=50,
widget=forms.TextInput(
            attrs={'class': 'form-control mb-3', 'placeholder':
'Firstname', 'id': 'form-lastname'}))

    class Meta:
        model = Customer
        fields = ('email', 'user_name', 'first_name',)

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['user_name'].required = True
        self.fields['email'].required = True
```

Basket Page

Basket.py

```
from decimal import Decimal

from checkout.models import DeliveryOptions
from django.conf import settings
from store.models import Product


class Basket:
    """
    A base Basket class, providing some default behaviors that
    can be inherited or overrided, as necessary.
    """

    def __init__(self, request):
        self.session = request.session
        basket = self.session.get(settings.BASKET_SESSION_ID)
        if settings.BASKET_SESSION_ID not in request.session:
            basket = self.session[settings.BASKET_SESSION_ID] = {}
        self.basket = basket

    def add(self, product, qty):
        """
        Adding and updating the users basket session data
        """
        product_id = str(product.id)

        if product_id in self.basket:
            self.basket[product_id]["qty"] = qty
        else:
            self.basket[product_id] = {"price":
str(product.regular_price), "qty": qty}

        self.save()

    def __iter__(self):
        """
        Collect the product_id in the session data to query the database
        and return products
        """
        product_ids = self.basket.keys()
        products = Product.objects.filter(id__in=product_ids)
        basket = self.basket.copy()
```

```
        for product in products:
            basket[str(product.id)]["product"] = product

        for item in basket.values():
            item["price"] = Decimal(item["price"])
            item["total_price"] = item["price"] * item["qty"]
            yield item

    def __len__(self):
        """
        Get the basket data and count the qty of items
        """
        return sum(item["qty"] for item in self.basket.values())

    def update(self, product, qty):
        """
        Update values in session data
        """
        product_id = str(product)
        if product_id in self.basket:
            self.basket[product_id]["qty"] = qty
        self.save()

    def get_subtotal_price(self):
        return sum(Decimal(item["price"]) * item["qty"] for item in
self.basket.values())

    def get_delivery_price(self):
        newprice = 0.00

        if "purchase" in self.session:
            newprice =
DeliveryOptions.objects.get(id=self.session["purchase"]["delivery_id"]).d
elivery_price

        return newprice

    def get_total_price(self):
        newprice = 0.00
        subtotal = sum(Decimal(item["price"]) * item["qty"] for item in
self.basket.values())

        if "purchase" in self.session:
            newprice =
DeliveryOptions.objects.get(id=self.session["purchase"]["delivery_id"]).d
elivery_price
```

```
        total = subtotal + Decimal(newprice)
        return total

    def basket_update_delivery(self, deliveryprice=0):
        subtotal = sum(Decimal(item["price"]) * item["qty"] for item in
self.basket.values())
        total = subtotal + Decimal(deliveryprice)
        return total

    def delete(self, product):
        """
        Delete item from session data
        """
        product_id = str(product)

        if product_id in self.basket:
            del self.basket[product_id]
            self.save()

    def clear(self):
        # Remove basket from session
        del self.session[settings.BASKET_SESSION_ID]
        del self.session["address"]
        del self.session["purchase"]
        self.save()

    def save(self):
        self.session.modified = True
```

Urls.py

```
from django.urls import path

from . import views

app_name = 'basket'

urlpatterns = [
    path('', views.basket_summary, name='basket_summary'),
    path('add/', views.basket_add, name='basket_add'),
    path('delete/', views.basket_delete, name='basket_delete'),
    path('update/', views.basket_update, name='basket_update'),
]
```

Views.py

```
  from django.http import JsonResponse
  from django.shortcuts import get_object_or_404, render

  from store.models import Product

  from .basket import Basket


  def basket_summary(request):
      basket = Basket(request)
      return render(request, 'basket/summary.html', {'basket': basket})


  def basket_add(request):
      basket = Basket(request)
      if request.POST.get('action') == 'post':
          product_id = int(request.POST.get('productid'))
          product_qty = int(request.POST.get('productqty'))
          product = get_object_or_404(Product, id=product_id)
          basket.add(product=product, qty=product_qty)

          basketqty = basket.__len__()
          response = JsonResponse({'qty': basketqty})
          return response


  def basket_delete(request):
```

```
    basket = Basket(request)
    if request.POST.get('action') == 'post':
        product_id = int(request.POST.get('productid'))
        basket.delete(product=product_id)

        basketqty = basket.__len__()
        baskettotal = basket.get_total_price()
        response = JsonResponse({'qty': basketqty, 'subtotal':
baskettotal})
        return response


def basket_update(request):
    basket = Basket(request)
    if request.POST.get('action') == 'post':
        product_id = int(request.POST.get('productid'))
        product_qty = int(request.POST.get('productqty'))
        basket.update(product=product_id, qty=product_qty)

        basketqty = basket.__len__()
        basketsubtotal = basket.get_subtotal_price()
        response = JsonResponse({'qty': basketqty, 'subtotal':
basketsubtotal})
        return response
```

Checkout

models.py

```
from django.db import models
from django.utils.translation import gettext_lazy as _


class DeliveryOptions(models.Model):
    """
    The Delivery methods table contining all delivery
    """

    DELIVERY_CHOICES = [
        ("IS", "In Store"),
        ("HD", "Home Delivery"),
        ("DD", "Digital Delivery"),
    ]

    delivery_name = models.CharField(
        verbose_name=_("delivery_name"),
        help_text=_("Required"),
        max_length=255,
    )
    delivery_price = models.DecimalField(
        verbose_name=_("delivery price"),
        help_text=_("Maximum 999.99"),
        error_messages={
            "name": {
                "max_length": _("The price must be between 0 and
999.99."),
            },
        },
        max_digits=5,
        decimal_places=2,
    )
    delivery_method = models.CharField(
        choices=DELIVERY_CHOICES,
        verbose_name=_("delivery_method"),
        help_text=_("Required"),
        max_length=255,
    )
    delivery_timeframe = models.CharField(
        verbose_name=_("delivery timeframe"),
        help_text=_("Required"),
        max_length=255,
```

```
    )
    delivery_window = models.CharField(
        verbose_name=_("delivery window"),
        help_text=_("Required"),
        max_length=255,
    )
    order = models.IntegerField(verbose_name=_("list order"),
help_text=_("Required"), default=0)
    is_active = models.BooleanField(default=True)

    class Meta:
        verbose_name = _("Delivery Option")
        verbose_name_plural = _("Delivery Options")

    def __str__(self):
        return self.delivery_name


class PaymentSelections(models.Model):
    """
    Store payment options
    """

    name = models.CharField(
        verbose_name=_("name"),
        help_text=_("Required"),
        max_length=255,
    )

    is_active = models.BooleanField(default=True)

    class Meta:
        verbose_name = _("Payment Selection")
        verbose_name_plural = _("Payment Selections")

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

Paypal.py

```
import sys

from paypalcheckoutsdk.core import PayPalHttpClient,
SandboxEnvironment


class PayPalClient:
    def __init__(self):
        self.client_id =
"ATJ5e0B5fnkU4UnQMLGHZCIOu82OB3YXnvUp8JmcVtZVqczhpzGlpaZ-
IXgGfXNvSb7cwiKQuWl0MPMV"
        self.client_secret =
"ELjPKWkB9hGPgtpHvqrkoZHU_lYl_PMK9dmZAptbYvudx3f_8KmpqxAFdqRuhCkO2jaWN
KNz-aZdA06B"
        self.environment =
SandboxEnvironment(client_id=self.client_id,
client_secret=self.client_secret)
        self.client = PayPalHttpClient(self.environment)
```

urls.py

```
from django.urls import include, path

from . import views

app_name = "checkout"

urlpatterns = [
  path("deliverychoices", views.deliverychoices, name="deliverychoices"),
  path("basket_update_delivery/", views.basket_update_delivery,
name="basket_update_delivery"),
  path("delivery_address/", views.delivery_address, name="delivery_address"),
  path("payment_selection/", views.payment_selection, name="payment_selection"),
  path("payment_complete/", views.payment_complete, name="payment_complete"),
  path("payment_successful/", views.payment_successful, name="payment_successful"),
]
```

views.py

```
import json

from account.models import Address
from basket.basket import Basket
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.http import HttpResponseRedirect, JsonResponse
from django.shortcuts import render
from orders.models import Order, OrderItem

from .models import DeliveryOptions


@login_required
def deliverychoices(request):
    deliveryoptions = DeliveryOptions.objects.filter(is_active=True)
    return render(request, "checkout/delivery_choices.html",
{"deliveryoptions": deliveryoptions})


@login_required
def basket_update_delivery(request):
    basket = Basket(request)
    if request.POST.get("action") == "post":
        delivery_option = int(request.POST.get("deliveryoption"))
        delivery_type =
DeliveryOptions.objects.get(id=delivery_option)
        updated_total_price =
basket.basket_update_delivery(delivery_type.delivery_price)

        session = request.session
        if "purchase" not in request.session:
            session["purchase"] = {
                "delivery_id": delivery_type.id,
            }
        else:
            session["purchase"]["delivery_id"] = delivery_type.id
            session.modified = True

        response = JsonResponse({"total": updated_total_price,
"delivery_price": delivery_type.delivery_price})
        return response


@login_required
```

```
def delivery_address(request):
    session = request.session
    if "purchase" not in request.session:
        messages.success(request, "Please select a delivery option")
        return HttpResponseRedirect(request.META.get("HTTP_REFERER"))

    addresses =
Address.objects.filter(customer=request.user).order_by("-default")

    if not addresses:
        messages.warning(request, "No addresses found for the user.")
        return HttpResponseRedirect(request.META.get("HTTP_REFERER"))

    if "address" not in request.session:
        session["address"] = {"address_id": str(addresses[0].id)}
    else:
        session["address"]["address_id"] = str(addresses[0].id)
        session.modified = True

    return render(request, "checkout/delivery_address.html",
{"addresses": addresses})

# def delivery_address(request):

#     session = request.session
#     if "purchase" not in request.session:
#         messages.success(request, "Please select delivery option")
#         return HttpResponseRedirect(request.META["HTTP_REFERER"])

#     addresses =
Address.objects.filter(customer=request.user).order_by("-default")

#     if "address" not in request.session:
#         session["address"] = {"address_id": str(addresses[0].id)}
#     else:
#         session["address"]["address_id"] = str(addresses[0].id)
#         session.modified = True

#     return render(request, "checkout/delivery_address.html",
{"addresses": addresses})


@login_required
def payment_selection(request):

    session = request.session
    if "address" not in request.session:
```

```
        messages.success(request, "Please select address option")
        return HttpResponseRedirect(request.META["HTTP_REFERER"])

    return render(request, "checkout/payment_selection.html", {})


####
# PayPay
###
from paypalcheckoutsdk.orders import OrdersGetRequest

from .paypal import PayPalClient


@login_required
def payment_complete(request):
    PPClient = PayPalClient()

    body = json.loads(request.body)
    data = body["orderID"]
    user_id = request.user.id

    requestorder = OrdersGetRequest(data)
    response = PPClient.client.execute(requestorder)

    total_paid = response.result.purchase_units[0].amount.value

    basket = Basket(request)
    order = Order.objects.create(
        user_id=user_id,

full_name=response.result.purchase_units[0].shipping.name.full_name,
        email=response.result.payer.email_address,

address1=response.result.purchase_units[0].shipping.address.address_li
ne_1,

address2=response.result.purchase_units[0].shipping.address.admin_area
_2,

postal_code=response.result.purchase_units[0].shipping.address.postal_
code,

country_code=response.result.purchase_units[0].shipping.address.countr
y_code,
        total_paid=response.result.purchase_units[0].amount.value,
        order_key=response.result.id,
```

```
            payment_option="paypal",
            billing_status=True,
        )
        order_id = order.pk

        for item in basket:
            OrderItem.objects.create(order_id=order_id,
    product=item["product"], price=item["price"], quantity=item["qty"])

        return JsonResponse("Payment completed!", safe=False)


    @login_required
    def payment_successful(request):
        basket = Basket(request)
        basket.clear()
        return render(request, "checkout/payment_successful.html", {})
```

Store

admin.py

```
from django import forms
from django.contrib import admin
from mptt.admin import MPTTModelAdmin

from .models import (
    Category,
    Product,
    ProductImage,
    ProductSpecification,
    ProductSpecificationValue,
    ProductType,
)

admin.site.register(Category, MPTTModelAdmin)


class ProductSpecificationInline(admin.TabularInline):
    model = ProductSpecification
```

```
@admin.register(ProductType)
class ProductTypeAdmin(admin.ModelAdmin):
    inlines = [
        ProductSpecificationInline,
    ]


class ProductImageInline(admin.TabularInline):
    model = ProductImage


class ProductSpecificationValueInline(admin.TabularInline):
    model = ProductSpecificationValue


@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    inlines = [
        ProductSpecificationValueInline,
        ProductImageInline,
    ]
```

models.py

```
from django.conf import settings
from django.db import models
from django.urls import reverse
from django.utils.translation import gettext_lazy as _
from mptt.models import MPTTModel, TreeForeignKey


class Category(MPTTModel):
    """
    Category Table implimented with MPTT.
    """

    name = models.CharField(
        verbose_name=_("Category Name"),
        help_text=_("Required and unique"),
        max_length=255,
        unique=True,
    )
    slug = models.SlugField(verbose_name=_("Category safe URL"), max_length=255, unique=True)
    parent = TreeForeignKey("self", on_delete=models.CASCADE, null=True, blank=True,
related_name="children")
    is_active = models.BooleanField(default=True)

    class MPTTMeta:
        order_insertion_by = ["name"]

    class Meta:
        verbose_name = _("Category")
        verbose_name_plural = _("Categories")

    def get_absolute_url(self):
        return reverse("store:category_list", args=[self.slug])

    def __str__(self):
        return self.name


class ProductType(models.Model):
    """
    ProductType Table will provide a list of the different types
    of products that are for sale.
    """
```

```
    name = models.CharField(verbose_name=_("Product Name"), help_text=_("Required"),
max_length=255, unique=True)
    is_active = models.BooleanField(default=True)

    class Meta:
        verbose_name = _("Product Type")
        verbose_name_plural = _("Product Types")

    def __str__(self):
        return self.name


class ProductSpecification(models.Model):
    """
    The Product Specification Table contains product
    specifiction or features for the product types.
    """

    product_type = models.ForeignKey(ProductType, on_delete=models.RESTRICT)
    name = models.CharField(verbose_name=_("Name"), help_text=_("Required"), max_length=255)

    class Meta:
        verbose_name = _("Product Specification")
        verbose_name_plural = _("Product Specifications")

    def __str__(self):
        return self.name


class Product(models.Model):
    """
    The Product table contining all product items.
    """

    product_type = models.ForeignKey(ProductType, on_delete=models.RESTRICT)
    category = models.ForeignKey(Category, on_delete=models.RESTRICT)
    title = models.CharField(
        verbose_name=_("title"),
        help_text=_("Required"),
        max_length=255,
    )
    description = models.TextField(verbose_name=_("description"), help_text=_("Not Required"),
blank=True)
    slug = models.SlugField(max_length=255)
    regular_price = models.DecimalField(
```

```
        verbose_name=_("Regular price"),
        help_text=_("Maximum 99999.99"),
        error_messages={
            "name": {
                "max_length": _("The price must be between 0 and 99999.99."),
            },
        },
        max_digits=7,
        decimal_places=2,
    )
    discount_price = models.DecimalField(
        verbose_name=_("Discount price"),
        help_text=_("Maximum 99999.99"),
        error_messages={
            "name": {
                "max_length": _("The price must be between 0 and 99999.99."),
            },
        },
        max_digits=7,
        decimal_places=2,
    )
    is_active = models.BooleanField(
        verbose_name=_("Product visibility"),
        help_text=_("Change product visibility"),
        default=True,
    )
    created_at = models.DateTimeField(_("Created at"), auto_now_add=True, editable=False)
    updated_at = models.DateTimeField(_("Updated at"), auto_now=True)
    users_wishlist = models.ManyToManyField(settings.AUTH_USER_MODEL,
related_name="user_wishlist", blank=True)

    class Meta:
        ordering = ("-created_at",)
        verbose_name = _("Product")
        verbose_name_plural = _("Products")

    def get_absolute_url(self):
        return reverse("store:product_detail", args=[self.slug])

    def __str__(self):
        return self.title


class ProductSpecificationValue(models.Model):
    """
```

```
    The Product Specification Value table holds each of the
    products individual specification or bespoke features.
    """

    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    specification = models.ForeignKey(ProductSpecification, on_delete=models.RESTRICT)
    value = models.CharField(
        verbose_name=_("value"),
        help_text=_("Product specification value (maximum of 255 words"),
        max_length=255,
    )

    class Meta:
        verbose_name = _("Product Specification Value")
        verbose_name_plural = _("Product Specification Values")

    def __str__(self):
        return self.value


class ProductImage(models.Model):
    """
    The Product Image table.
    """

    product = models.ForeignKey(Product, on_delete=models.CASCADE,
related_name="product_image")
    image = models.ImageField(
        verbose_name=_("image"),
        help_text=_("Upload a product image"),
        upload_to="images/",
        default="images/default.png",
    )
    alt_text = models.CharField(
        verbose_name=_("Alturnative text"),
        help_text=_("Please add alturnative text"),
        max_length=255,
        null=True,
        blank=True,
    )
    is_feature = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True, editable=False)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
```

```
        verbose_name = _("Product Image")
        verbose_name_plural = _("Product Images")
```

urls.py

```
from django.urls import path

from . import views

app_name = 'store'

urlpatterns = [
    path('', views.product_all, name='store_home'),
    path('<slug:slug>', views.product_detail, name='product_detail'),
    path('shop/<slug:category_slug>/', views.category_list,
name='category_list'),
]
```

Views.py

```
from django.shortcuts import get_object_or_404, render

from .models import Category, Product


def product_all(request):
    products =
Product.objects.prefetch_related("product_image").filter(is_active=Tru
e)
    return render(request, "store/index.html", {"products": products})


def category_list(request, category_slug=None):
    category = get_object_or_404(Category, slug=category_slug)
    products = Product.objects.filter(

category__in=Category.objects.get(name=category_slug).get_descendants(
include_self=True)
    )
    return render(request, "store/category.html", {"category":
category, "products": products})


def product_detail(request, slug):
    product = get_object_or_404(Product, slug=slug, is_active=True)
    return render(request, "store/single.html", {"product": product})
```

Template

Base.html

```
{% load static %}
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <title>{% block title %}Store - Low Prices in Books & more{% endblock %}</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"
crossorigin="anonymous">
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"
    integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-
ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW"
crossorigin="anonymous">
  </script>
  <link rel="stylesheet" href="{% static 'core/css/base.css' %}">
  <link rel="stylesheet" href="{% block stylesheet %}{% endblock %}">
</head>

<body>
  <header class="pb-3">
    <nav class="navbar navbar-expand-md navbar-light bg-white border-bottom">
      <div class="container-fluid px-md-4">
        <div class="d-flex w-100 navbar-collapse">
          <a class="navbar-brand d-flex-inline" href="/">
            <svg xmlns="http://www.w3.org/2000/svg" width="36" height="36" fill="currentColor"
class="bi bi-asterisk"
              viewBox="0 0 16 16">
              <path
                d="M8 0a1 1 0 0 1 1 1v5.268l4.562-2.634a1 1 0 1 1 1 1.732L10 8l4.562 2.634a1 1 0 1 1-
1 1.732L9 9.732V15a1 1 0 1 1-2 0V9.732l-4.562 2.634a1 1 0 1 1-1-1.732L6 8 1.438 5.366a1 1 0 0
1 1-1.732L7 6.268V1a1 1 0 0 1 1-1z" />
            </svg>
          </a>
```

```
        <ul class="navbar-nav me-auto mb-2 mb-md-0">
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle d-none d-md-block fw500" href="#"
id="navbarDropdown" role="button"
              data-bs-toggle="dropdown" aria-expanded="false">
              All
              <i class="ps-1"><svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor"
                class="bi bi-chevron-down" viewBox="0 0 16 16">
                <path fill-rule="evenodd"
                  d="M1.646 4.646a.5.5 0 0 1 .708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6
6a.5.5 0 0 1-.708 0l-6-6a.5.5 0 0 1 0-.708z" />
              </svg></i>
            </a>
            <ul class="dropdown-menu rounded-0 border-0" aria-labelledby="navbarDropdown">
              <li><a class="dropdown-item" href="{% url "store:store_home" %}">All</a></li>
              {% for c in categories %}
              <li {% if category.slug == c.slug %}class="selected" {% endif %}>
                <a class="dropdown-item" href="{{ c.get_absolute_url }}">{{ c.name|title }}</a>
              </li>
              {% endfor %}
            </ul>
          </li>
        </ul>
        <button class="navbar-toggler border-0" type="button" data-bs-toggle="collapse"
          data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false"
          aria-label="Toggle navigation">
          <div>
            <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor"
class="bi bi-list"
              viewBox="0 0 16 16">
              <path fill-rule="evenodd"
                d="M2.5 11.5A.5.5 0 0 1 3 11h10a.5.5 0 0 1 0 1H3a.5.5 0 0 1-.5-.5zm0-4A.5.5 0 0 1 3
7h10a.5.5 0 0 1 0 1H3a.5.5 0 0 1-.5-.5zm0-4A.5.5 0 0 1 3 3h10a.5.5 0 0 1 0 1H3a.5.5 0 0 1-.5-.5z"
/>
            </svg>
          </div>
          <span class="fs15 fw500">Shop</span>
        </button>
        <div>
          <div class="collapse navbar-collapse" id="navbarSupportedContent2">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
              <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle text-reset" href="#" id="navbarDropdown"
role="button"
```

```
            data-bs-toggle="dropdown" aria-expanded="false">
            <div class="small text-muted">Hello,
             {% if user.is_authenticated %}
              {{ request.user.name }}
             {% else %}
             Login
             {% endif %}
            </div>
            <span class="fs15 fw500">Account &amp;amp; Lists
            </span>
            </span>
           </a>
           <div class="dropdown-menu rounded-0" aria-labelledby="navbarDropdown">
            <div class="card border-0">
             <div class="card-body">
              <h6 class="card-title">My Account</h6>
              <ul class="list-group list-group-flush dropdown-account-link">
               {% if user.is_authenticated %}
               <a href="{% url "account:dashboard" %}" class="text-reset small list-group-item p-
0 pb-1 border-0 dropdown-account-link">Your Account</a>
                <a href="#" class="text-reset small list-group-item p-0 pb-1 border-0 dropdown-
account-link">Orders</a>
                <a href="{% url "account:logout" %}" class="text-reset small list-group-item p-0
pb-1 border-0 dropdown-account-link">Logout</a>
               {% else %}
               <a href="{% url "account:login" %}" class="text-reset small list-group-item p-0 pb-
1 border-0 dropdown-account-link">Login</a>
               {% endif %}
              </ul>
             </div>
            </div>
           </div>
          </li>
         </ul>
        </div>
       </div>
       <a type="button" role="button" href="{% url "basket:basket_summary" %}"
        class="btn btn-outline-secondary border-0 basket-btn">
        {% with total_qty=basket|length %}
        <div id="basket-qty" class="basket-qty">
         {% if total_qty > 0 %}
         {{ total_qty }}
         {% else %}
         0
         {% endif %}
```

```
        </div>
        {% endwith %}
        <div>
          <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor"
class="bi bi-cart3"
            viewBox="0 0 16 16">
            <path
              d="M0 1.5A.5.5 0 0 1 .5 1H2a.5.5 0 0 1 .485.379L2.89 3H14.5a.5.5 0 0 1 .49.598l-1
5a.5.5 0 0 1-.465.401l-9.397.472L4.415 11H13a.5.5 0 0 1 0 1H4a.5.5 0 0 1-.491-.408L2.01 3.607
1.61 2H.5a.5.5 0 0 1-.5-.5zM3.102 4l.84 4.479 9.144-.459L13.89 4H3.102zM5 12a2 2 0 1 0 0 4 2 2
0 0 0 0-4zm7 0a2 2 0 1 0 0 4 2 2 0 0 0 0-4zm-7 1a1 1 0 1 1 0 2 1 1 0 0 1 0-2zm7 0a1 1 0 1 1 0 2 1 1
0 0 1 0-2z" />
          </svg>
        </div>
        <span class="fs15 fw500">Basket</span>
      </a>
    </div>
    <div class="d-md-none d-lg-none d-xl-none">
      <div class="collapse navbar-collapse" id="navbarCollapse">
        <ul class="navbar-nav me-auto mb-2 mb-md-0">
          <li><a class="dropdown-item" href="{% url "store:store_home" %}">All</a></li>
          {% for c in categories %}
          <li {% if category.slug == c.slug %}class="selected" {% endif %}>
            <a class="dropdown-item" href="{{ c.get_absolute_url }}">{{ c.name|title }}</a>
          </li>
          {% endfor %}
        </ul>
        <h6 class="card-title">My Account</h6>
          <ul class="list-group list-group-flush dropdown-account-link">
            {% if user.is_authenticated %}
            <a href="{% url "account:dashboard" %}" class="dropdown-item">Your Account</a>
            <a href="#" class="dropdown-item">Orders</a>
            <a href="{% url "account:logout" %}" class="dropdown-item">Logout</a>
            {% else %}
            <a href="{% url "account:login" %}" class="dropdown-item">Login</a>
            {% endif %}
          </ul>
      </div>
    </div>
    <form class="d-flex w-100 d-md-none">
      <input class="form-control me-2" type="search" placeholder="Search products or FAQ"
aria-label="Search">
      <button class="btn btn-outline-secondary" type="submit">Search</button>
    </form>
  </div>
```

```
  </nav>
 </header>

 <main class="pt-5">
  <div id="content">{% block content %} {% endblock %}</div>
 </main>

 <footer class="container py-5 footer">
  <hr>
  <div class="row pt-4">
   <div class="col-12 col-md">
    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor"
class="bi bi-asterisk"
     viewBox="0 0 16 16">
     <path
      d="M8 0a1 1 0 0 1 1 1v5.268l4.562-2.634a1 1 0 0 1 1 1.732L10 8l4.562 2.634a1 1 0 0 1 1-1
1.732L9 9.732V15a1 1 0 0 1-2 0V9.732l-4.562 2.634a1 1 0 0 1-1-1.732L6 8 1.438 5.366a1 1 0 0 1
1-1.732L7 6.268V1a1 1 0 0 1 1-1z" />
    </svg>
    <small class="d-block mb-3 text-muted">&copy; 2017-2021</small>
   </div>
   <div class="col-6 col-md fs15">
    <h5>Features</h5>
    <ul class="list-unstyled">
     <li><a class="link-secondary text-decoration-none" href="#">Cool stuff</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Random feature</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Team feature</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Stuff for developers</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Another one</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Last time</a></li>
    </ul>
   </div>
   <div class="col-6 col-md fs15">
    <h5>Resources</h5>
    <ul class="list-unstyled">
     <li><a class="link-secondary text-decoration-none" href="#">Resource name</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Resource</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Another resource</a></li>
     <li><a class="link-secondary text-decoration-none" href="#">Final resource</a></li>
    </ul>
   </div>
   <div class="col-6 col-md fs15">
    <h5>Resources</h5>
    <ul class="list-unstyled">
     <li><a class="link-secondary text-decoration-none" href="#">Business</a></li>
```

```
        <li><a class="link-secondary text-decoration-none" href="#">Education</a></li>
        <li><a class="link-secondary text-decoration-none" href="#">Government</a></li>
        <li><a class="link-secondary text-decoration-none" href="#">Gaming</a></li>
       </ul>
      </div>
      <div class="col-6 col-md">
       <h5>About</h5>
       <ul class="list-unstyled">
        <li><a class="link-secondary text-decoration-none" href="#">Team</a></li>
        <li><a class="link-secondary text-decoration-none" href="#">Locations</a></li>
        <li><a class="link-secondary text-decoration-none" href="#">Privacy</a></li>
        <li><a class="link-secondary text-decoration-none" href="#">Terms</a></li>
       </ul>
      </div>
     </div>
    </footer>
  </body>

  </html>
```

Address.html

```
{% extends "../sub_base.html" %}
{% block title %}Edit Addresses{% endblock %}



{% block sub_content %}
<h1 class="h2">Your Addresses</h1>
<div>Manage your <b>addresses</b> and delivery preferences</div>
<hr />
<div class="container px-0">
 <div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
  <div class="col">
   <a href="{% url "account:add_address" %}" class="text-reset text-decoration-none"
role="button"
    style="max-width: 540px;">
    <div class="card mb-3 h-100" style="border: dashed 2px #ccc;">
     <div class="row g-0 h-100">
      <div class="col-12" style="min-height:100px">
       <div class="card-body text-center position-absolute top-50 start-50 translate-
middle">
        <svg style="color:#ccc;" xmlns="http://www.w3.org/2000/svg" width="60"
height="60" fill="currentColor"
```

```
            class="bi bi-plus" viewBox="0 0 16 16">
            <path
              d="M8 4a.5.5 0 0 1 .5.5v3h3a.5.5 0 0 1 0 1h-3v3a.5.5 0 0 1-1 0v-3h-3a.5.5 0 0 1 0-
1h3v-3A.5.5 0 0 1 8 4z" />
           </svg>
           <h1 class="h5">Add Address</h1>
          </div>
         </div>
        </div>
       </div>
     </a>
   </div>
   {% for address in addresses %}
   <div class="col">
    <div class="card pb-3">
     <div class="card-header bg-white small text-muted">
      {% if address.default %}
       Default
      {% endif %}
       
     </div>
     <div class="card-body small pb-1">
      <p class="card-text m-0 fw-bold">{{address.full_name}}</p>
      <p class="card-text m-0">{{address.address_line}}</p>
      <p class="card-text m-0">{{address.address_line2}}</p>
      <p class="card-text m-0">{{address.town_city}}</p>
      <p class="card-text m-0">{{address.postcode}}</p>
      <p class="card-text m-0">Phone number: {{address.phone}}</p>
      <div class="pt-5">
       <a href="{% url 'account:edit_address' address.id %}" class="text-decoration-
none">Edit</a>
        |
       <a href="{% url 'account:delete_address' address.id %}" class="text-decoration-
none">Delete</a>
       {% if not address.default %}
       | <a href="{% url 'account:set_default' address.id %}" class="text-decoration-
none">Set Default</a>
       {% endif %}
      </div>
     </div>
    </div>
    </a>
   </div>
   {% endfor %}
  </div>
```

```
</div>

{% endblock %}
```

Dashboard.py

```
{% extends "../sub_base.html" %}
{% block title %}Dashboard{% endblock %}

{% block sub_content %}

<div class="col-12">
  <h1 class="h2">Your Account</h1>
</div>
<div class="col-12 d-flex justify-content-between">
  <div>Manage your <b>orders</b> and personal details</div>
  {% comment %} <div><a href="{% url "account:edit_details" %}">Change
Details</a></div> {% endcomment %}
</div>
<hr />

<div class="row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3">
  <div class="col">
    <a href="{% url 'account:user_orders' %}" class="text-reset text-decoration-none"
role="button" style="max-width: 540px;">
      <div class="card mb-3">
        <div class="row g-0">
          <div class="col-2 position-relative">
            <svg class="position-absolute top-50 start-50 translate-middle"
xmlns="http://www.w3.org/2000/svg"
              width="30" height="30" fill="currentColor" class="bi bi-gift" viewBox="0 0 16 16">
              <path
                d="M3 2.5a2.5 2.5 0 0 1 5 0 2.5 2.5 0 0 1 5 0v.006c0 .07 0 .27-.038.494H15a1 1 0 0
1 1 1v2a1 1 0 0 1-1 1v7.5a1.5 1.5 0 0 1-1.5 1.5h-11A1.5 1.5 0 0 1 1 14.5V7a1 1 0 0 1-1-
1V4a1 1 0 0 1 1-1h2.038A2.968 2.968 0 0 1 3 2.506V2.5zm1.068.5H7v-.5a1.5 1.5 0 1 0-3
0c0 .085.002.274.045.43a.522.522 0 0 0 .023.07zM9 3h2.932a.56.56 0 0 0 .023-.07c.043-
.156.045-.345.045-.43a1.5 1.5 0 0 0-3 0V3zM1 4v2h6V4H1zm8 0v2h6V4H9zm5
3H9v8h4.5a.5.5 0 0 0 .5-.5V7zm-7 8V7H2v7.5a.5.5 0 0 0 .5.5H7z" />
            </svg>
          </div>
          <div class="col-10">
            <div class="card-body ps-0 ps-md-1">
              <h1 class="h5">Orders</h1>
              <p class="card-text small text-muted">View, Track, Change or buy again</p>
```

```
        </div>
       </div>
      </div>
     </div>
    </a>
   </div>
   <div class="col">
    <a href="{% url "account:edit_details" %}" class="text-reset text-decoration-none"
role="button"
     style="max-width: 540px;">
     <div class="card mb-3">
      <div class="row g-0">
       <div class="col-2 position-relative">
        <svg class="position-absolute top-50 start-50 translate-middle"
xmlns="http://www.w3.org/2000/svg"
         width="30" height="30" fill="currentColor" class="bi bi-shield-lock" viewBox="0 0
16 16">
          <path
           d="M5.338 1.59a61.44 61.44 0 0 0-2.837.856.481.481 0 0 0-.328.39c-.554
4.157.726 7.19 2.253 9.188a10.725 10.725 0 0 0 2.287
2.233c.346.244.652.42.893.533.12.057.218.095.293.118a.55.55 0 0 0 .101.025.615.615 0 0
0 .1-.025c.076-.023.174-.061.294-.118.24-.113.547-.29.893-.533a10.726 10.726 0 0 0
2.287-2.233c1.527-1.997 2.807-5.031 2.253-9.188a.48.48 0 0 0-.328-.39c-.651-.213-1.75-
.56-2.837-.855C9.552 1.29 8.531 1.067 8 1.067c-.53 0-1.552.223-
2.662.524zM5.072.56C6.157.265 7.31 0 8 0s1.843.265 2.928.56c1.11.3 2.229.655
2.887.87a1.54 1.54 0 0 1 1.044 1.262c.596 4.477-.787 7.795-2.465 9.99a11.775 11.775 0 0
1-2.517 2.453 7.159 7.159 0 0 1-1.048.625c-.28.132-.581.24-.829.24s-.548-.108-.829-
.24a7.158 7.158 0 0 1-1.048-.625 11.777 11.777 0 0 1-2.517-2.453C1.928 10.487.545 7.169
1.141 2.692A1.54 1.54 0 0 1 2.185 1.43 62.456 62.456 0 0 1 5.072.56z" />
          <path
           d="M9.5 6.5a1.5 1.5 0 0 1-1 1.415l.385 1.99a.5.5 0 0 1-.491.595h-.788a.5.5 0 0 1-
.49-.595l.384-1.99a1.5 1.5 0 1 1 2-1.415z" />
         </svg>
        </div>
        <div class="col-10">
         <div class="card-body ps-0 ps-md-1">
          <h1 class="h5">Login & Security</h1>
          <p class="card-text small text-muted">Edit login, email and phone number</p>
         </div>
        </div>
       </div>
      </div>
     </a>
    </div>
    <div class="col">
```

```
   <a href="{% url "account:addresses" %}" class="text-reset text-decoration-none"
role="button"
     style="max-width: 540px;">
     <div class="card mb-3">
      <div class="row g-0">
       <div class="col-2 position-relative">
        <svg class="position-absolute top-50 start-50 translate-middle"
xmlns="http://www.w3.org/2000/svg"
         width="30" height="30" fill="currentColor" class="bi bi-truck" viewBox="0 0 16
16">
          <path
           d="M0 3.5A1.5 1.5 0 0 1 1.5 2h9A1.5 1.5 0 0 1 12 3.5V5h1.02a1.5 1.5 0 0 1
1.17.563l1.481 1.85a1.5 1.5 0 0 1 .329.938V10.5a1.5 1.5 0 0 1-1.5 1.5H14a2 2 0 1 1-4
0H5a2 2 0 1 1-3.998-.085A1.5 1.5 0 0 1 0 10.5v-7zm1.294 7.456A1.999 1.999 0 0 1 4.732
11h5.536a2.01 2.01 0 0 1 .732-.732V3.5a.5.5 0 0 0-.5-.5h-9a.5.5 0 0 0-.5.5v7a.5.5 0 0 0
.294.456zM12 10a2 2 0 0 1 1.732 1h.768a.5.5 0 0 0 .5-.5V8.35a.5.5 0 0 0-.11-.312l-1.48-
1.85A.5.5 0 0 0 13.02 6H12v4zm-9 1a1 1 0 1 0 0 2 1 1 0 0 0-2zm9 0a1 1 0 1 0 0 2 1 1 0 0 0
0-2z" />
        </svg>
       </div>
       <div class="col-10">
        <div class="card-body ps-0 ps-md-1">
         <h1 class="h5">Your Addresses</h1>
         <p class="card-text small text-muted">Edit your shipping addresses</p>
        </div>
       </div>
      </div>
     </div>
   </a>
  </div>
  <div class="col">
   <a href="{% url "account:wishlist" %}" class="text-reset text-decoration-none"
role="button"
     style="max-width: 540px;">
     <div class="card mb-3">
      <div class="row g-0">
       <div class="col-2 position-relative">
        <svg class="position-absolute top-50 start-50 translate-middle"
xmlns="http://www.w3.org/2000/svg"
         width="30" height="30" fill="currentColor" class="bi bi-card-checklist" viewBox="0
0 16 16">
          <path
           d="M14.5 3a.5.5 0 0 1 .5.5v9a.5.5 0 0 1-.5.5h-13a.5.5 0 0 1-.5-.5v-9a.5.5 0 0 1 .5-
.5h13zm-13-1A1.5 1.5 0 0 0 0 3.5v9A1.5 1.5 0 0 0 1.5 14h13a1.5 1.5 0 0 0 1.5-1.5v-9A1.5
1.5 0 0 0 14.5 2h-13z" />
```

```
          <path
            d="M7 5.5a.5.5 0 0 1 .5-.5h5a.5.5 0 0 1 0 1h-5a.5.5 0 0 1-.5-.5zm-1.496-.854a.5.5
0 0 1 0 .708l-1.5 1.5a.5.5 0 0 1-.708 0l-.5-.5a.5.5 0 1 1 .708-.708l.146.147 1.146-1.147a.5.5
0 0 1 .708 0zM7 9.5a.5.5 0 0 1 .5-.5h5a.5.5 0 0 1 0 1h-5a.5.5 0 0 1-.5-.5zm-1.496-.854a.5.5
0 0 1 0 .708l-1.5 1.5a.5.5 0 0 1-.708 0l-.5-.5a.5.5 0 0 1 .708-.708l.146.147 1.146-1.147a.5.5
0 0 1 .708 0z" />
        </svg>
       </div>
       <div class="col-10">
        <div class="card-body ps-0 ps-md-1">
         <h1 class="h5">Your Wish List</h1>
         <p class="card-text small text-muted">View your Wish List</p>
        </div>
       </div>
      </div>
     </div>
    </a>
   </div>

</div>

{% endblock %}
```

Edit_address.html

```
{% extends "../sub_base.html" %}
{% block title %}Edit Addresses{% endblock %}

{% block sub_content %}
<div class="col-6 mx-auto">
  <h1 class="h3">Create/Edit Address</h1>
  <div>Add a new delivery <b>address</b> and delivery preferences</div>
  <hr />
  <form name="address_form" class="account-form" method="post"
enctype="multipart/form-data">
    {% if form.errors %}
    <div class="alert alert-primary" role="alert">
      Error: Please try again!
    </div>
    {% endif %}
    {% csrf_token %}
    <label class="small fw-bold">{{ form.full_name.label}}</label>
    {{ form.full_name }}
    <label class="small fw-bold">{{ form.phone.label}}</label>
    {{ form.phone }}
    <label class="small fw-bold">{{ form.address_line.label}}</label>
    {{ form.address_line }}
    <label class="small fw-bold">{{ form.address_line2.label}}</label>
    {{ form.address_line2 }}
    <label class="small fw-bold">{{ form.town_city.label}}</label>
    {{ form.town_city }}
    <label class="small fw-bold">{{ form.postcode.label}}</label>
    {{ form.postcode }}
    <button class="btn btn-primary btn-block py-2 mb-4 mt-4 fw-bold w-100" type="button"
value="Submit" onclick="submitForm()">
    Add Address
    </button>
  </form>

</div>

<script>
function submitForm() {
  var form = document.getElementsByName('address_form')[0];
  form.submit(); // Submit the form
  form.reset();  // Reset all form data
  return false; // Prevent page refresh
}
```

```
</script>
{% endblock %}
```

Edit_details.html

```
{% extends "../sub_base.html" %}
{% block title %}Edit Profile{% endblock %}
{% block sub_content %}

<div class="col-6 mx-auto">
<form class="account-form" method="post" enctype="multipart/form-
data">
    {% if user_form.is_valid and profile_form.is_valid  %}
    <div class="alert alert-primary" role="alert">
        Details successfully updated!
    </div>
    <p class="small text-center pt-0">
        <a href="{% url "account:dashboard" %}">Back to Dashboard</a>
    </p>
    {%else%}
    <h3>Change your details</h3>
    <p>You can edit your account using the following form:</p>
    {% endif %}
    {% if form.errors %}
    <div class="alert alert-primary" role="alert">
        Error: Please try again!
    </div>
    {% endif %}
    {% csrf_token %}
    <label class="small font-weight-bold">{{
user_form.email.label}}</label>
    {{ user_form.email }}
    <label class="small font-weight-bold">{{
user_form.first_name.label}}</label>
    {{ user_form.first_name }}
    <button class="btn btn-primary btn-block py-2 mb-4 mt-5 fw-bold
w-100" type="submit" value="Save changes">Save
        Changes</button>
    <hr class="mb-3">
</form>
```

```
  <form class="account-form" action="{% url "account:delete_user" %}"
  method="post">
      {% csrf_token %}
      <p class=" h3 pt-4 font-weight-bold">Delete Account</p>
      <p>Are you sure you want to delete your account?</p>
      <button type="submit" role="button" class="btn btn-danger btn-
  block py-2 mb-4 mt-5 fw-bold w-100">Delete</button>
  </form>
  </div>

  {% endblock %}
```

User_order_.html

```
{% extends "../sub_base.html" %}
{% block title %}User Orders{% endblock %}

{% block sub_content %}

<div class="col-12">
  <h1 class="h2">User Orders</h1>
</div>
<div class="col-12 d-flex justify-content-between">
  <div>Manage your <b>orders</b> and personal details</div>
</div>
<hr />
 <div class="container">
    {% for order in orders %}
    <div class="row g-3">
      <div class="col-12 bg-light p-3 d-flex justify-content-
between">
        <div class="d-flex d-flex-inline">
          <div class="pe-3">{{ order.created }}</div>
          <div class="dropdown">
          <a class="text-reset text-decoration-none dropdown-toggle"
href="#" role="link" id="dropdownLink" data-bs-toggle="dropdown"
aria-expanded="false">
          Dispacted to
          <svg xmlns="http://www.w3.org/2000/svg" width="12"
height="12" fill="currentColor" class="bi bi-chevron-down"
viewBox="0 0 16 16">
```

```
            <path fill-rule="evenodd" d="M1.646 4.646a.5.5 0 0 1
.708 0L8 10.293l5.646-5.647a.5.5 0 0 1 .708.708l-6 6a.5.5 0 0 1-.708
0l-6-6a.5.5 0 0 1 0-.708z"/>
          </svg>
          </a>
          <ul class="dropdown-menu" aria-labelledby="dropdownLink">
            <li class="item small">{{order.full_name}}</li>
            <li class="item small">{{order.address1}}</li>
            <li class="item small">{{order.address2}}</li>
            <li class="item small">{{order.post_code}}</li>
          </ul>
          </div>
        </div>
        <div class="text-end">
          Total paid: <span class="fw-bold">₹{{ order.total_paid
}}</span>
        </div>
      </div>
      <div class="col-md-5 col-lg-4 order-md-last p-0 order-3">
        <div class="d-grid gap-2 ">
          <button class="btn btn-warning" type="button">Problem with
order</button>
          <button class="btn btn-light" type="button">Leave a
review</button>
        </div>
      </div>
      <div class="col-md-7 col-lg-8 p-0">
        {% for item in order.items.all %}
        <div class="card mb-3 border-0">
          <div class="row g-0">
            <div class="col-md-2 d-none d-md-block">
            {% for image in item.product.product_image.all %}
            {% if image.is_feature %}
            <img class="img-fluid" alt="Responsive image" src="{{
image.image.url }}" alt="{{ image.image.alt_text }}">
            {% endif %}
            {% endfor %}
            </div>
            <div class="col-md-10">
              <div class="card-body p-3">
                <a class="text-decoration-none" href="{{
item.product.get_absolute_url }}">
                  <p class="card-text
small">{{item.product|title}}</p>
                </a>
              </div>
            </div>
```

```
          </div>
        </div>
        {% endfor %}
      </div>
    </div>
    {% endfor %}
  </div>


{% endblock %}
```