# SEAM

LOPSANG LAMA

TEXAS COLLEGE OF MANAGEMENT & IT

DEPARTMENT OF INFORMATION TECHNOLOGY

LINCOLN UNIVERSITY COLLEGE

SEPTEMBER 2023

## Declaration

I hereby declare that the project work entitled “SEAM” submitted to the Faculty of Science, Lincoln University College, Kathmandu is an original piece of work under the supervision of Mr. Samir Bhandari and is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Technology (BIT). This project work report has not been submitted to any other university or institution for the award of any degree.

Signature:

Name of Student: Lopsang Lama

Date: 2023-19-27

## Supervisor’s Recommendation

The project work report entitled SEAM submitted by Lopsang of Texas College of Management and IT, is prepared under my supervision as per the procedure and format requirements laid by the Faculty of Science, Lincoln University College, as partial fulfillment of the requirements for the degree of Bachelor of Information Technology (BIT). I, therefore, recommend the project work report for evaluation.

Signature:

Name of Supervisor: Samir Singh Bhandari

Date:

## Abstract

The rapid evolution of the internet, originally conceived for communication, has revolutionized countless facets of modern life. In today's digital era, nearly all aspects of society, from interpersonal connections to intricate business transactions, have been seamlessly integrated into the online realm. One of the most significant manifestations of this transformation is the advent and growth of electronic commerce, commonly referred to as e-commerce.

E-commerce encompasses the practice of buying and selling goods and services through digital platforms and the internet. This dynamic field, which initially centered around business-to-business (B2B) interactions, has expanded to encompass business-to-consumer (B2C) and consumer-to-consumer (C2C) transactions. In response to this evolution, we present "SEAM," a user-friendly web-based application designed to facilitate the exchange of electronic applications within the B2C and C2C sectors.

This documentation offers comprehensive insights into SEAM, outlining its development process and key features. It highlights the strategic utilization of Python (Django) for the backend and React.js for the frontend, ensuring an seamless and efficient user experience. By navigating through this documentation, readers will gain a deeper understanding of SEAM's role in the ever-evolving landscape of e-commerce and the broader digital economy.

## Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the completion of this project.

First and foremost, I am deeply thankful to my supervisor, Mr. Samir Bhandari, for his guidance and patience throughout the research process. His insightful feedback has played a big role in this work.

I am grateful to my friends and classmates who have offered their assistance and moral support during this journey.

This project would not have been possible without the collective effort of all those mentioned above. Thank you for being a part of this Journey.

Table of Contents

## List of Figure

# CHAPTER 1:

## INRODUCTION:

### 1. INTRODUCTION:

In the modern era, the demand for electronic applications is increasing as technology is advancing making it available and able to do day to day operations. To address the demand for electronic applications, a need for e-commerce platforms is seen.

SEAM provides a web application-based e-commerce platform for electronic applications as well as secondhand applications. SEAM is an e-commerce web application that connects the buyer and seller through internet or online. People can view the available products without registering but people must register as users to buy the products. There are similar types of websites in Nepal like Daraz, Hamrobazar, etc.,

### 2. PROBLEM STATEMENT:

- Limited range of products available in traditional market, Time-consuming, and Not knowing seller Credibility.
- There are many times when buying a secondhand product, many people get scammed as the product quality will not be same as mentioned previously.

### 3. OBJECTIVE:

The objective of this project is: -

- To develop a user-friendly web application that provides a platform for buying and selling second-hand electronic applications.

- To address the problems of traditional method of buying and selling electronic devices such as time-consuming, limited reach, Credibility of Seller etc.,

4. SCOPE AND LIMITATION:

The Scope of this project are: -

I. User Registration and Profiles:
   - User account creation and management for both buyers and sellers.
   - User profiles that store personal information.

II. Shopping Cart and Order:
   - Virtual shopping cart for customers to accumulate items for purchase.
   - Secure and streamlined Order process that includes order id, shipping details, and order review.

The Limitation of this project are: -

a) Mobile Responsive:
While every website is created mobile-friendly and responsive, achieving seamless mobile responsiveness will still be challenging.

b) Integration Challenges:
Integrating with third-party services like payment gateways, shipping providers, and inventory management systems can sometimes be complex, leading to potential compatibility issues and additional development costs.

c) Search and Filtering:
Robust search functionality to help users quickly find the products they're looking for.

d) Reviews and Ratings:
Customer reviews and ratings for products and sellers to build trust and help customers make informed decisions.

e) Mobile Applications:

Its mobile app is also a limitation in this project, as I don't have sufficient knowledge in app development. It will be created in the future if I continue this project further.

5. DEVELOPMENT AND METHODOLOGY:

The project will follow an agile development methodology, emphasizing iterative development and continuous feedback. The initial phase will involve requirement gathering and system design. Subsequently, the development team will implement the core functionalities, conduct thorough testing, and refine the application based on user feedback. The project will culminate with the deployment of the e-commerce web application.

Agile Methodology Advantages: -

- It helps to break down the project into small and manageable increments called iterations which helps me to develop/deliver the function pieces of software faster.
- It is flexible. And the test plan is reviewed after each sprint.

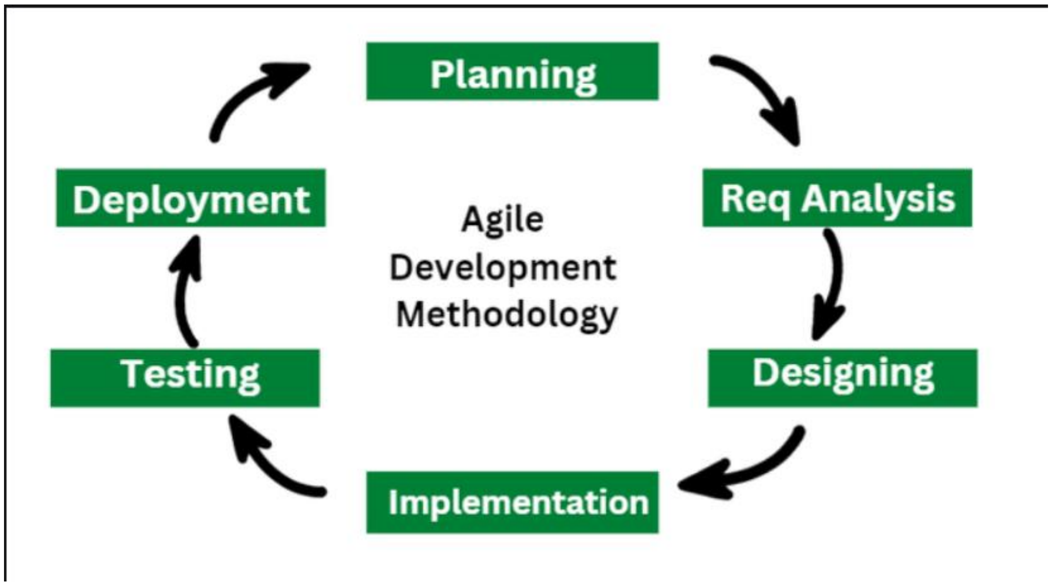


**Fig 1: Agile Development Method**

# CHAPTER 2:

## BACKGROUND STUDY AND LITERATURE REVIEW:

1. BACKGROUND STUDY:

   In the early stage of e-commerce, E-commerce was mainly focused on B2B, B2C providing the product and services of their organizations or companies. As the importance of e-commerce platforms started to grow, many large, medium, and small sized businesses or organizations also started to create their own e-commerce platform to reach their customers and have a wide range of audiences.
   However, in today's time e-commerce has evolved into C2C also.

2. LITERATURE REVIEW:

   Literature review 1:

   Electronic commerce has affected the ways in which enterprises and countries produce, trade, and compete in fundamental and qualitative fashions. Even in countries with poor infrastructure and access to information technology, evidence exists that dynamic enterprises and governments have taken advantages of the possibilities offered by e-commerce. Countries with poor communication and internet infrastructures should therefore act now in order to develop a strong e-commerce market to prevent landing on the wrong side of the digital divide. This article introduces the history of internet and the current state of e-commerce in Nepal. It aims to identify projects that can possibly facilitate the growth of e-commerce ventures in underdeveloped countries such as Nepal. Finally, the article explains what ought to be done to establish a profitable e-marketplace in Nepal.

   Reference: https://www.researchgate.net/publication/264818866_E-commerce_in_Nepal_a_case_study_of_an_underdeveloped_country

Literature Review 2:

The implementation of digitalization in the industrial sector surely impacts several sectors, particularly in business. The development of information technology and rapid economic globalization have initiated the role of e-commerce in economic trading activities over the globe. It takes a lot of research to initiate online new business. Therefore, it is important to reveal the existence of gap among previous research. Systematic literature review method is applied to analyze the role of e-commerce in trading activity as well as to provide the improvement for future research. As many as 28 e-commerce related literatures are analyzed comprehensively and systematically based on protocol review. The result of the research confirms the opportunity for future research on e-commerce systems that enables the integration on business processes. The supporting variables of e-commerce include business branding, social and economic development, efficient system e-commerce platform, framework application.

Reference: https://www.researchgate.net/publication/361911355_The_Role_of_e-Commerce_A_Systematic_Literature_Review

# CHAPTER 3:

## SYSTEM ANALYSIS AND DESIGN:

1) SYSTEM ANALYSIS:
   a) Requirement Analysis:

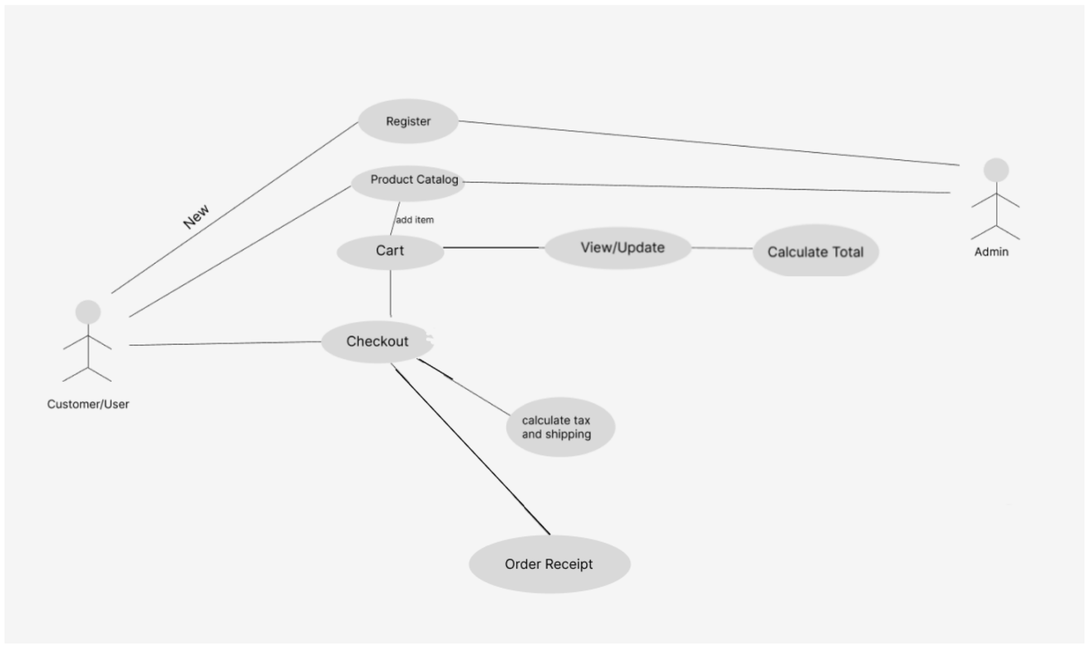


**Fig 2: Use case Diagram for SEAM**

i. Functional Requirements:

The use case description are as follows:

| S.N. | Functional requirement | description |
|---|---|---|
| 1 | Register | The User must register themselves before placing order |
| 2 | Sign-In | The user and administrator should login to the system to gain access. User should compulsory login to place the order. |
| 3 | View Product Catalog | Every person (registered User or not and admin) should be able to see the product catalog in home page. |
| 4 | Cart | The product should be added to Cart when a user clicks on the particular button (add to Cart button) of a product card. |
| 5 | Checkout | When a user clicks on the checkout button, the user should be directed to the payment sector. |
| 6 | Calculate total | Total cost should be calculated with according to the item present in Cart including the tax and shipping cost. |
| 7 | Payment | |
| 8 | Order report | After the payment is done, A order report should be generated alerting the user/customer about the successful order placement |

ii. Non-Functional Requirements:

a. Usability:
The Application should provide a user-friendly interface and environment.

b. Less Response Time:
The application must exhibit fast response times.

c. Security:
The application must secure/maintain the access permission according to the roles and responsibilities (i.e., Only system Administrators have the authority to modify the user as well as product data, whereas user should only be able to modify the product data that they are selling)

d. Availability
The Application should be available for 24 hours a day except for the maintenance time.

b) Feasibility Analysis:

a. Operational Feasibility:
Operational feasibility is a crucial aspect of project evaluation that assesses whether a proposed system or project can be effectively operated and maintained once it is implemented. Customer support provides a channel for users to seek assistance, ask questions, and get help with any issues they encounter while using the website. This assistance is vital for enhancing the user experience and resolving user concerns.

b. Schedule Feasibility:

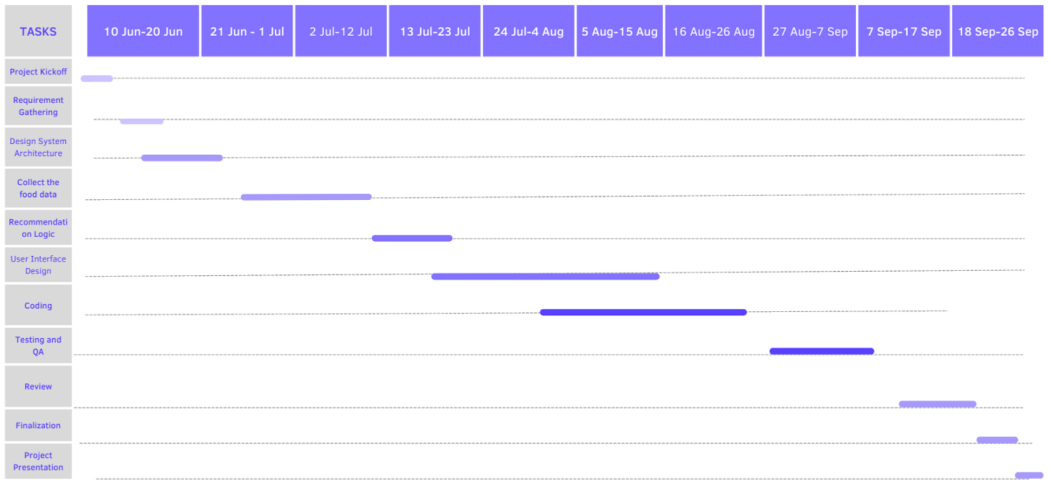


**Fig 3: Gantt Chart**

c. Technical Feasibility:
While developing this project, I have used Python (Django) as a backend and React.js as a frontend. I have specified the version of the important software/tools that can affect the project in the future when using this project.

i. Backend requirements:
   a. Django = 4.2.4,
   b. Django-cors-headers = 4.2.0,
   c. Djangorestframework = 3.14.0,

d. Twilio = 8.5.0,
e. Pillow = 10.0.0

ii. Frontend requirements:
  a. @fontawesome/fontawesome-svg-core: "^6.4.2,
  b. @fontawesome/free-regular-svg-icons: "^6.4.2,
  c. @fontawesome/free-solid-svg-icons: "^6.4.2,
  d. @fontawesome/react-fontawesome: ^0.2.0,
  e. bootstrap5.3.1,
  f. react: ^18.2.0,
  g. react-bootstrap: ^2.8.0,
  h. react-dom:^18.2.0,
  i. react-router-dom: ^6.15.0

iii. Database requirements:
SQLite.

2) System Design
  a) Data Modeling:

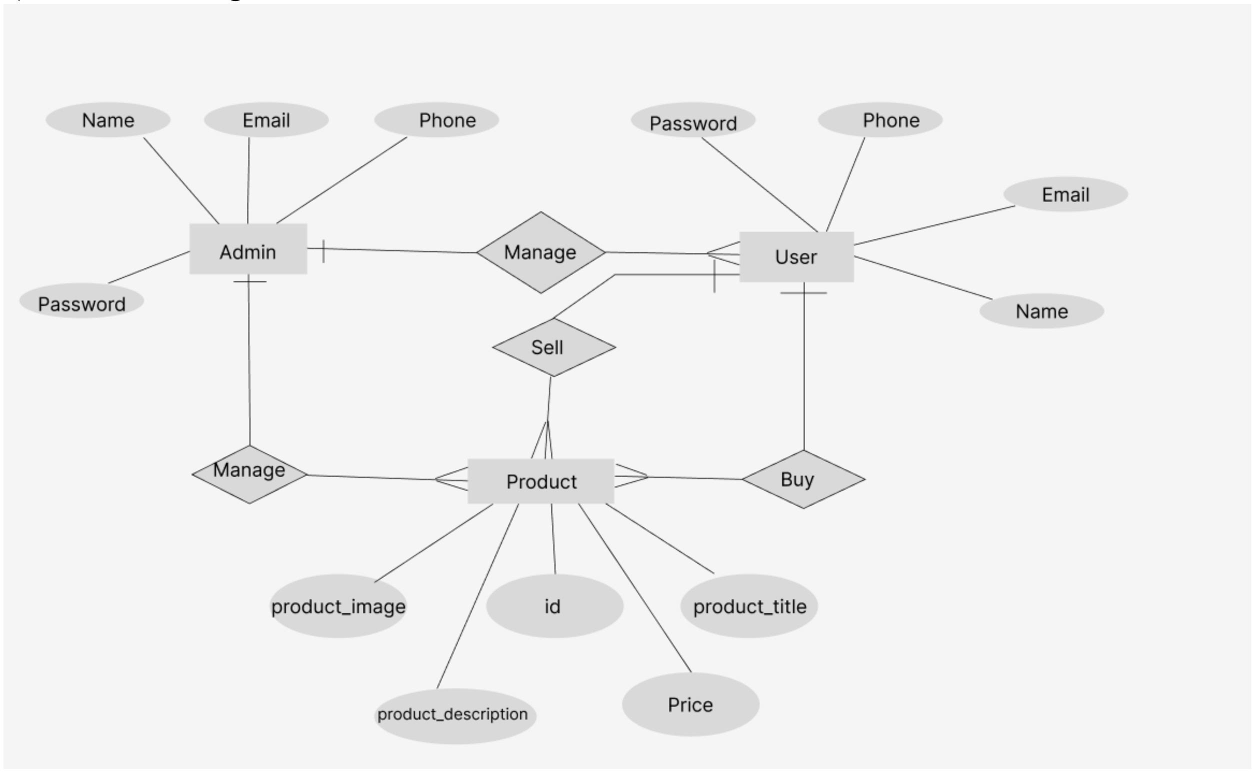


**Fig 4: ER Diagram**

b) Process Modeling:

Context Diagram:

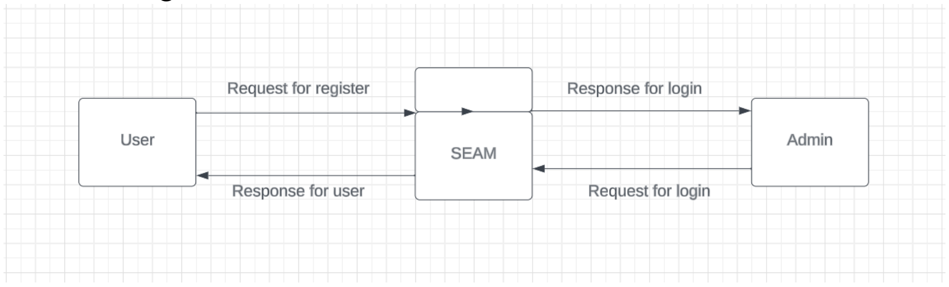


Fig 5: Context Diagram

DFD Level 1:

User Part:

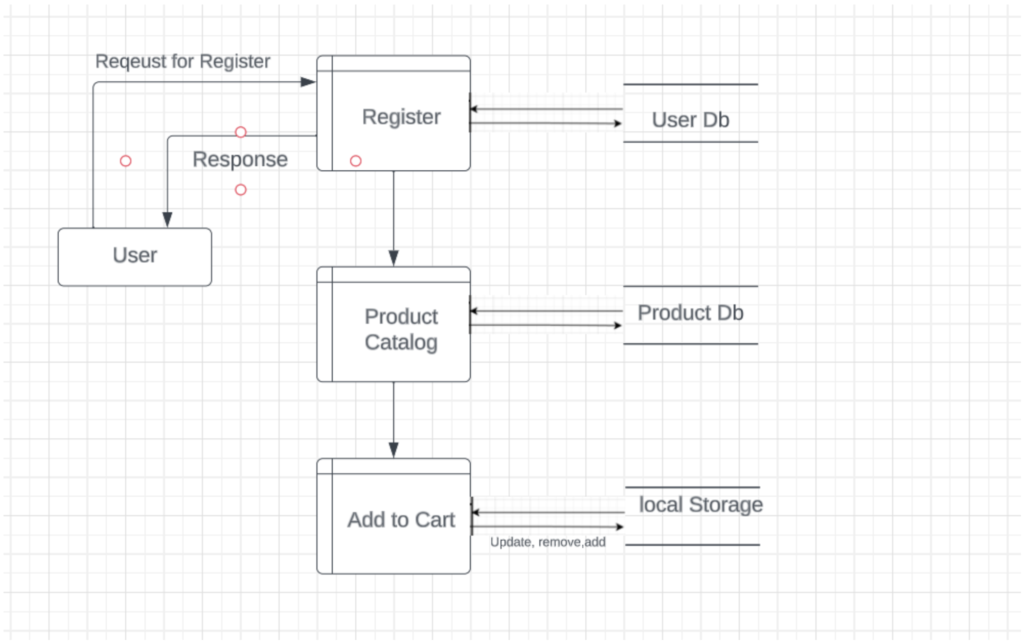


Fig 6: DFD Level 1 of User

Admin Part:

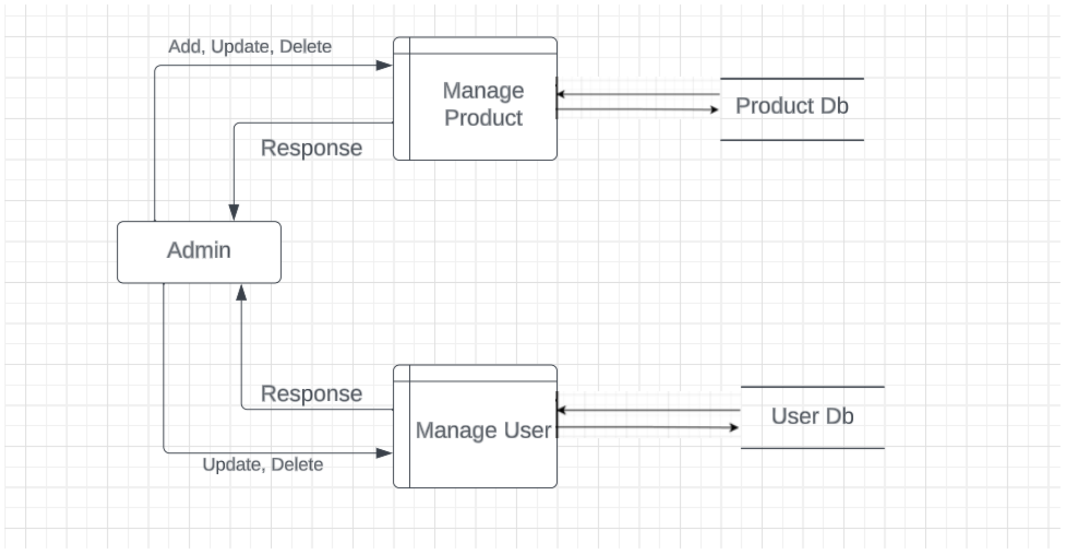


Fig 7: DFD Level 1: (Admin)

c) Database Design:

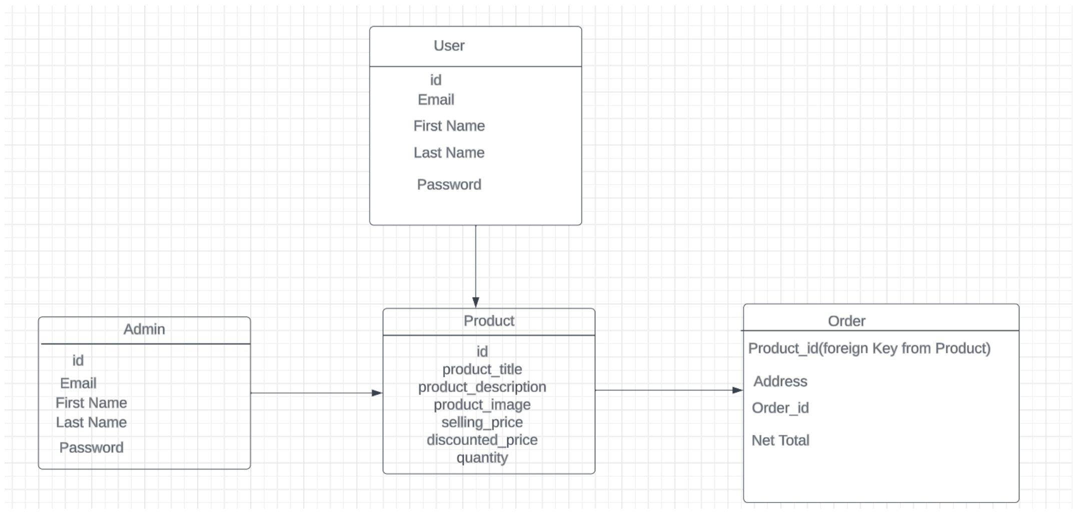


Fig 8: Database Design

# Chapter 4: Implementation and Testing

## 1. Implementation:

### 1.1. Tools Used:

There are many ways to develop a website with the use of different languages and tools. The tools used in making this project were selected due to my knowledge in those languages.
React developer tool in google extension for better view/operation in web.
Frontend: React.js
Backend: Django
Database: db. SQLite

### 1.2. Testing:

To ensure the quality and functionality of the web, Several Testing were done, for several functions and component of project.

Test Case Scenario for User Registration:

| SN | Test Description | Test Data | Expected Result |
|---|---|---|---|
| 1 | Register with all the field empty | Empty | Bad Request Redirected to register Page |
| 2 | Register with all the field data except first Name field | First Name:<br>Last Name: Lama<br>Email: Lopsang@example.com<br>Password: Password<br>Confirm Password: Password | Bad Request Redirected to register Page |
| 3 | Register with all the field data correct but not giving valid email | First Name: Lopsang<br>Last Name: Lama<br>Email: Lopsanglafanva<br>Password: Password<br>Confirm Password: Password | Bad Request Redirected to register Page |
| 4 | Registered with Not matching Password | First Name: Lopsang<br>Last Name: Lama<br>Email: Lopsang@example.com<br>Password: password<br>Confirm Password: LopsamgLama123 | Alert the user about not matching password just below the Confirm Password input field |
| 5 | Registered with all valid data | First Name: Lopsang<br>Last Name: Lama | Success Alert indicating the successful register |

|  |  |  |  |
|---|---|---|---|
|  |  | Email: Lopsang@example.com<br>Password: Password<br>Confirm Password: Password | and redirected to homepage |

Test Case Scenario for Login:

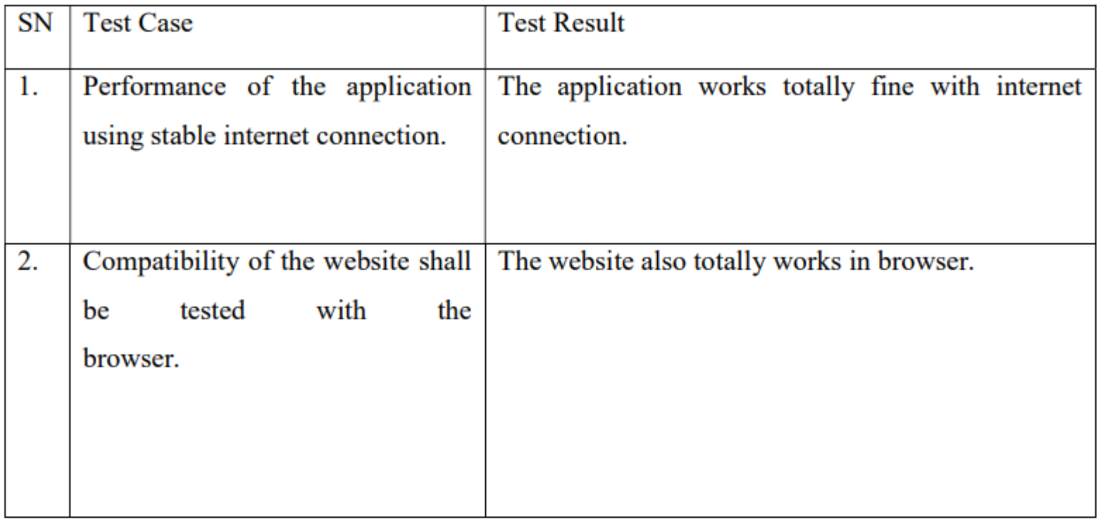
| SN | Test Description | Test Data | Expected Result |
|---|---|---|---|
| 1 | Login with incorrect email but with correct password | Email: lopsang23@example.com<br>Password: Password | Alert the user indicating email or password is incorrect.<br><br>Redirect to login page |
| 2 | Login with correct email but with incorrect password | Email: Lopsang@example.com<br>Password: P@ssW0rD | Alert the user indicating email or password is incorrect.<br><br>Redirect to login page |
| 3 | Login with both correct credentials | Email: Lopsang@example.com<br>Password: Password | Alert the user indication successful login.<br><br>Redirect the user to homepage |

Test Case Scenario for Order:

| SN | Test Description | Test Data | Expected Result |
|---|---|---|---|
| 1 | Address not filled when ordering the product | Address: | Alert the user showing that the address is required in order to place the order successfully. |

|  |  |  |  |
|---|---|---|---|
| 2 | Address given when ordering the product | Address: Jorpati, Kathmandu | Alert the user of successfully order placement, and provide the user the order receipt |

Test Case Scenario for System:



| SN | Test Case | Test Result |
|---|---|---|
| 1. | Performance of the application using stable internet connection. | The application works totally fine with internet connection. |
| 2. | Compatibility of the website shall be tested with the browser. | The website also totally works in browser. |

# Chapter 5: Conclusion and Future Recommendations

- **Conclusion:**

  “SEAM” is an online web-based application that provides a platform for users to buy or sell products without third-party interference. Seam mainly focusses on the electronic second-hand application, which also helps in a way to reduce the e-waste by recycling and re-using.

  In conclusion, "SEAM" holds the potential to provide users with a convenient and secure platform for buying and selling second-hand electronic applications. By addressing the limitations of traditional methods and incorporating modern web technologies, it aims to contribute to a more sustainable and efficient e-commerce ecosystem.

- **Future Recommendations:**
  The future expansion are: -
  - Mobile Application Development:
    Expanding the platform to include mobile applications can significantly enhance user accessibility and engagement.

  - User Reviews and Ratings:
    Implementing a user review and rating system can boost trust and confidence among users, making it easier for them to make informed decisions.

  - Integration with Payment Gateways:
    Integrating with various payment gateways to provide users with multiple payment options and enhance transaction security.

  - Enhanced Search and Filtering:
    Implementing advanced search and filtering features can help users quickly find the products they're looking for, improving the overall user experience.

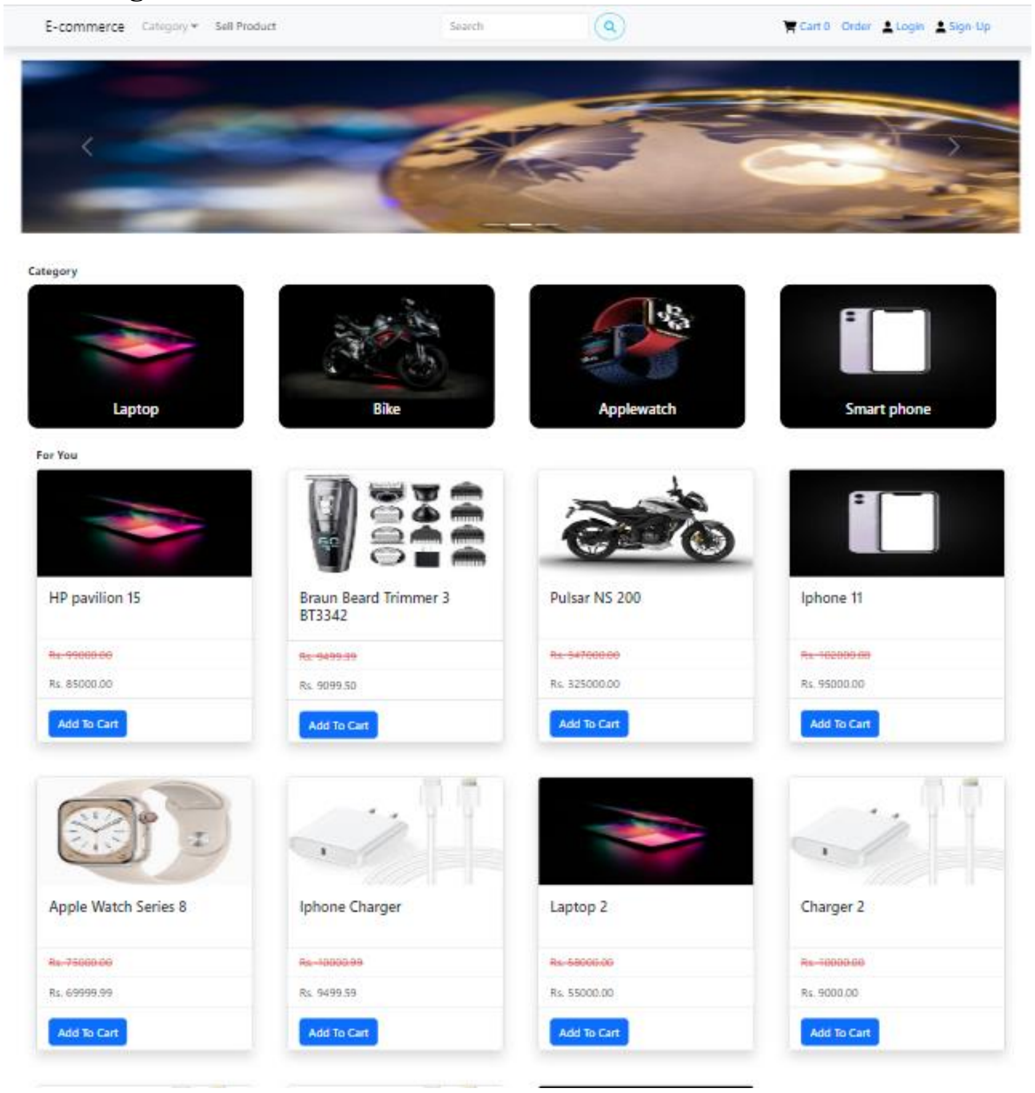
## Reference

- E-commerce in Nepal: A case study of underdeveloped country. Author Penjor Ngudup, Jason C.H. Chen, Binshan Lin (https://www.researchgate.net/publication/264818866_E-commerce_in_Nepal_a_case_study_of_an_underdeveloped_country)

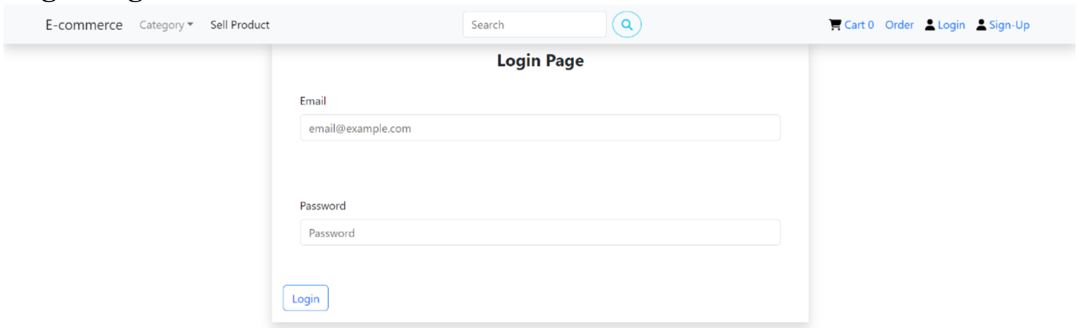
- The role of E-commerce (https://www.researchgate.net/publication/361911355_The_Role_of_e-Commerce_A_Systematic_Literature_Review)

## Annex

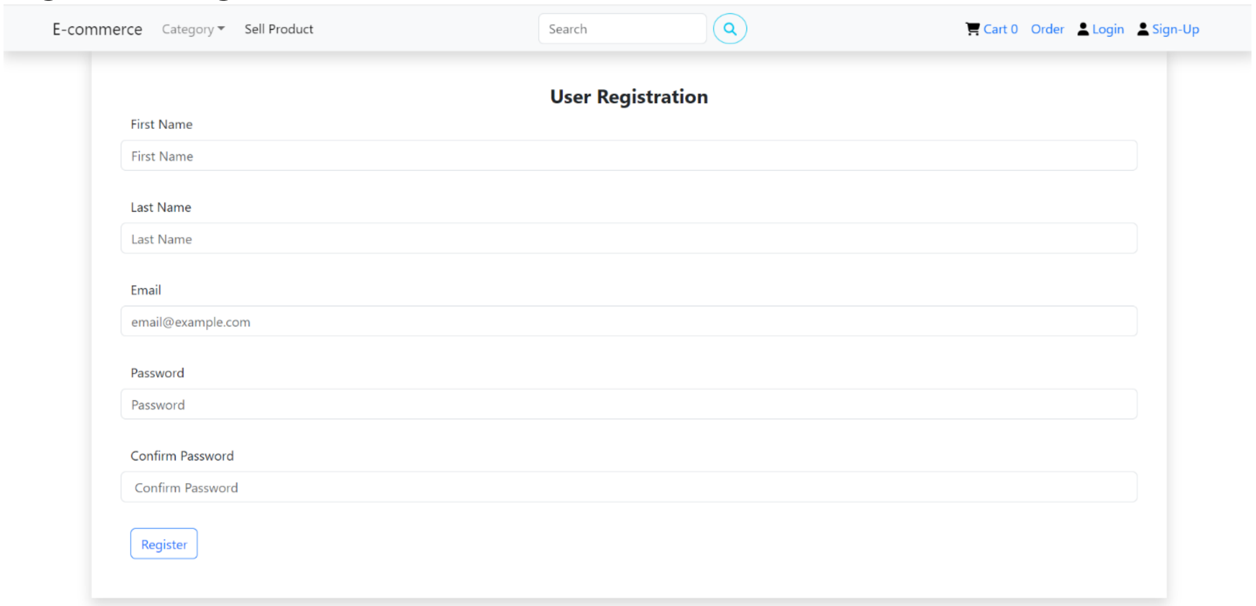
### Home Page View:



### Login Page View:



## Registration Page:



User Registration

First Name: First Name

Last Name: Last Name

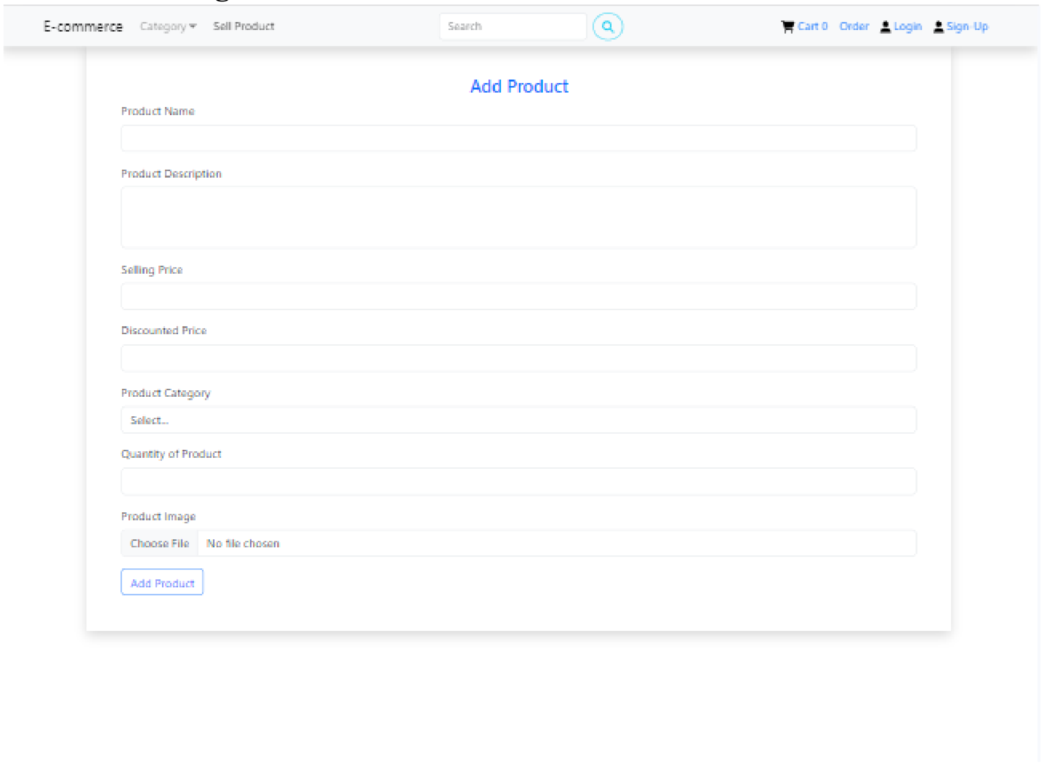
Email: email@example.com

Password: Password

Confirm Password: Confirm Password

Register

## Sell Product Page View:



Add Product

Product Name

Product Description

Selling Price

Discounted Price

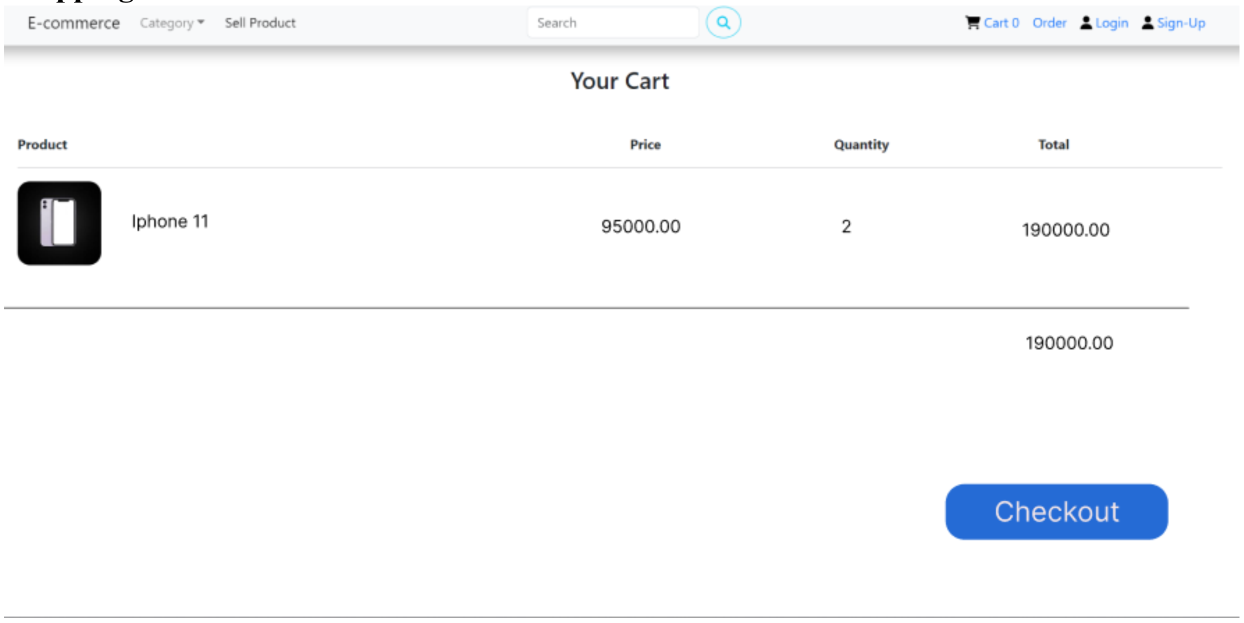
Product Category: Select...

Quantity of Product

Product Image: Choose File No file chosen

Add Product

**Shopping Cart view:**



Your Cart

| Product | Price | Quantity | Total |
|---|---|---|---|
| Iphone 11 | 95000.00 | 2 | 190000.00 |
| | | | 190000.00 |

**Source Code:**

**Backend(Django):**

**Models.py:**

```
      from django.db import models

from django.contrib.auth.models import BaseUserManager, AbstractBaseUser


class UserManager(BaseUserManager):
    def create_user(self, email, firstName,lastName,
password=None,password2=None):
        """
        Creates and saves a User with the given email,name and password.
        """
        if not email:
            raise ValueError("Users must have an email address")

        user = self.model(
            email=self.normalize_email(email),
            firstName = firstName,
            lastName = lastName
        )

        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, email, firstName,lastName,
password=None,password2=None):
        """
        Creates and saves a User with the given email,name and password.

        """
        user = self.create_user(
            email,
            password=password,
            firstName=firstName,
            lastName=lastName
        )
        user.is_admin = True
        user.save(using=self._db)
        return user


class MyUser(AbstractBaseUser):
    email = models.EmailField(
        verbose_name="email address",
        max_length=255,
        unique=True,
    )
    firstName = models.CharField(max_length=150)
    lastName = models.CharField(max_length=150)
    is_active = models.BooleanField(default=True)
    is_admin = models.BooleanField(default=False)

    objects = UserManager()

    USERNAME_FIELD = "email"
    REQUIRED_FIELDS = ["firstName","lastName"]

    def __str__(self):
        return self.email

    def has_perm(self, perm, obj=None):
        "Does the user have a specific permission?"
```

```python
        # Simplest possible answer: Yes, always
        return self.is_admin

    def has_module_perms(self, app_label):
        "Does the user have permissions to view the app `app_label`?"
        # Simplest possible answer: Yes, always
        return True

    @property
    def is_staff(self):
        "Is the user a member of staff?"
        # Simplest possible answer: All admins are staff
        return self.is_admin



class task(models.Model):
    user = models.ForeignKey(MyUser, on_delete=models.CASCADE)
    title = models.CharField(max_length=255)
    description = models.TextField(max_length=255)
    due_date = models.DateTimeField()
    Completed = models.BooleanField(default=False)

    def __str__ (self):
        return self.title



CATEGORY_CHOICES = (

    ('LP', 'Laptop'),
    ('SM', 'Smartphone'),
    ('BK', 'Bike'),
    ('WH', 'Watch'),
    ('TM', 'Trimmer'),
    ('CH', 'Charger'),

)



class ProductList(models.Model):
    product_title = models.CharField(max_length=150)
    product_description = models.TextField(max_length=255)
    selling_price = models.DecimalField(max_digits=8, decimal_places=2)
    discounted_price = models.DecimalField(max_digits=8, decimal_places=2)
    category = models.CharField(choices=CATEGORY_CHOICES, max_length=2)
    quantity = models.IntegerField(default= 1)
    product_image = models.ImageField(upload_to='product')

    def __str__ (self):
        return self.product_title




class Orders(models.Model):
    id = models.AutoField(primary_key=True)
    products = models.ManyToManyField(ProductList, related_name='orders')
    address = models.CharField(max_length=255)
    GrandTotal = models.DecimalField(max_digits=9,decimal_places=2)
```

**Urls.py**

```
from django.urls import path
from Todo import views

urlpatterns = [
    path('UserData',views.UserProfileView.as_view(),name='registration'),
    path('UserData/<int:id>', views.UserProfileEditView.as_view(),name='edit'),
    path('',views.TaskView.as_view(), name='taskview'),
    path('task/<int:id>',views.TaskDetails.as_view(), name='taskDetails'),
    path('login',views.LoginView.as_view(), name = 'login'),
    path('productlist', views.ProductListView.as_view() , name='products'),
    path('order',views.OrderListView.as_view(),name="order"),

]
```

**Serializers.py**

```
from rest_framework import serializers

from Todo.models import MyUser, task, ProductList



#Custom Serializer
```

```
class UserSerializer(serializers.ModelSerializer):

    password2 =
serializers.CharField(style={'input_type':'password'},write_only=True)
    class Meta:
        model = MyUser
        fields = ["id","email","firstName","lastName","password","password2"]
        extra_kwargs = {
            'password':{'write_only':True}
        }

    def validate(self,attrs):
        password = attrs.get('password')
        password2 = attrs.get('password2')

        if password != password2:
            raise serializers.ValidationError("Password doesn't match ")

        return attrs

    def create(self, validate_data):
        return MyUser.objects.create_user(**validate_data)


#Task Serializer

class TaskSerializer(serializers.ModelSerializer):

    class Meta:
        model = task
        fields = ["id","user","title","description","due_date","Completed"]


class LoginSerializer(serializers.ModelSerializer):
    email = serializers.EmailField(max_length = 255)
    class Meta:
        model = MyUser
        fields = ["email","password"]



class ProductSerializer(serializers.ModelSerializer):

    class Meta:
        model = ProductList
        fields = ["id",
"product_title","product_description","selling_price","discounted_price","categor
y","quantity","product_image"]


class OrdersSerializer(serializers.ModelSerializer):

    class Meta:
        model= Orders
        fields = ["id","products","address","GrandTotal"]
```

**admin.py**

```
from django.contrib import admin
from Todo.models import MyUser,task,ProductList
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin

# Register your models here.
class UserAdmin(BaseUserAdmin):


    # The fields to be used in displaying the User model.
    # These override the definitions on the base UserAdmin
    # that reference specific fields on auth.User.
    list_display = ["id","email", "firstName","lastName", "is_admin"]
    list_filter = ["is_admin"]
    fieldsets = [
        (None, {"fields": ["email", "password"]}),
        ("Personal info", {"fields": ["firstName","lastName"]}),
        ("Permissions", {"fields": ["is_admin"]}),
    ]
    # add_fieldsets is not a standard ModelAdmin attribute. UserAdmin
    # overrides get_fieldsets to use this attribute when creating a user.
    add_fieldsets = [
        (
            None,
            {
                "classes": ["wide"],
                "fields": ["email", "firstName","lastName", "password1",
"password2"],
            },
        ),
    ]
    search_fields = ["email","id"]
    ordering = ["email"]
    filter_horizontal = []


# Now register the new UserAdmin...
admin.site.register(MyUser, UserAdmin)


@admin.register(task)

class TaskModelAdmin(admin.ModelAdmin):
    list_display = ["id","user","title","description","due_date","Completed"]


@admin.register(ProductList)
class ProductModelAdmin(admin.ModelAdmin):
    list_display = ["id", "product_title","product_description",
"selling_price","discounted_price","category","quantity","product_image"]

@admin.register(Orders)
class OrderModelAdmin(admin.ModelAdmin):
    list_display = ["id", "product_titles", "address", "GrandTotal"]

    def product_titles(self, obj):
        # Concatenate product titles from related ProductList objects
        return ", ".join([product.product_title for product in
obj.products.all()])

    product_titles.short_description = "Products"
```

**views.py:**

```
from django.shortcuts import render
from rest_framework.response import Response
from rest_framework.decorators import APIView,api_view
from Todo.serializers import TaskSerializer, UserSerializer,
LoginSerializer,ProductSerializer
from rest_framework import status
from .models import MyUser, task, ProductList
from django.contrib.auth import authenticate
```

```python
# Create your views here.
class UserProfileView(APIView):
    def get(self,request):
        details = MyUser.objects.all()
        serializer = UserSerializer(details, many=True)
        return Response(serializer.data , status=status.HTTP_200_OK)

    def post(self,request):
        serializer = UserSerializer(data= request.data)
        if serializer.is_valid():
            user = serializer.save()
            message = "Registration Successful"
            context = {
                "message":message,
                "data" : serializer.data
            }

            return Response(context,status = status.HTTP_201_CREATED)

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


class UserProfileEditView(APIView):
    def get(self, request, id):
        try:
            detail = MyUser.objects.get(id =id)
        except MyUser.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)


        serializer = UserSerializer(detail)
        return Response(serializer.data, status = status.HTTP_200_OK)

    def put(self,request,id):
        try:
            detail = MyUser.objects.get(id =id)
        except MyUser.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)

        serializer = UserSerializer(detail,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,status=status.HTTP_202_ACCEPTED)
        return Response(serializer.errors , status=status.HTTP_400_BAD_REQUEST)
    def delete(self,request,id):
        try:
            detail = MyUser.objects.get(id =id)
        except MyUser.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)

        # serializer = UserSerializer(detail)
        detail.delete()
        return Response(status = status.HTTP_204_NO_CONTENT)






class TaskView(APIView):
    def get(self,request):
        Task = task.objects.all()
        serializer = TaskSerializer(Task,many=True)
        return Response(serializer.data ,status=status.HTTP_302_FOUND)

    def post(self,request):
        Task = task.objects.all()
        serializer = TaskSerializer(Task,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class TaskDetails(APIView):
    def get(self,request,id):
        try:
            Task = task.objects.get(id=id)
        except task.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)

        serializer = TaskSerializer(Task)
        return Response(serializer.data, status=status.HTTP_302_FOUND)

    def put(self,request,id):
        try:
            Task = task.objects.get(id =id)
        except task.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)
        serializer = TaskSerializer(Task,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,status=status.HTTP_202_ACCEPTED)
        return Response(serializer.errors , status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request, id):
        try:
            Task = task.objects.get(id=id)
        except task.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)

        Task.delete()
        return Response(status = status.HTTP_204_NO_CONTENT)


class LoginView(APIView):
    def post(self,request,format=None):
        serializer = LoginSerializer(data= request.data)
        if serializer.is_valid(raise_exception=True):
            email = serializer.data.get('email')
            password = serializer.data.get('password')
            user = authenticate(email=email, password= password)
            if user is not None:
                message = "login successful"
                return Response(message,status=status.HTTP_200_OK)
            else:
                return Response({'errors':{'non_field_erros':['Email or password
is not valid']}},status=status.HTTP_401_UNAUTHORIZED)
        return Response(serializer.errors,status=status.HTTP_400_BAD_REQUEST)


class ProductListView(APIView):
    def get(self,request):
        products = ProductList.objects.all()
        serializer = ProductSerializer(products, many=True)
        return Response(serializer.data , status=status.HTTP_200_OK)

    def post(self,request):
        serializer = ProductSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status= status.HTTP_400_BAD_REQUEST)
```

```python
class OrderListView(APIView):
    def get(self,request):
        order = Orders.objects.all()
        serializer = OrdersSerializer(order, many=True)
        return Response(serializer.data , status=status.HTTP_200_OK)

    def post(self, request):
        serializer = OrdersSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

**settings.py**

```python
"""
Django settings for TodoApi project.

Generated by 'django-admin startproject' using Django 4.2.4.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-ltcb_j=n5tyi=9_9n2i18$a&3-*rppq6uoi=oct8ce3g*7aedf'

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

ALLOWED_HOSTS = []
```

```
# Application definition

INSTALLED_APPS = [
    'Todo',
    'corsheaders',
    'rest_framework',
    'rest_framework.authtoken',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]


MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'TodoApi.urls'

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

WSGI_APPLICATION = 'TodoApi.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
```

```
USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/



STATIC_URL = 'static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR/'media'

CORS_ALLOWED_ORIGINS = [

    "http://localhost:5173",
    "http://127.0.0.1:5173",
    "http://127.0.0.1:5174",
    "http://localhost:5174",

]

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

AUTH_USER_MODEL = 'Todo.MyUser'
```

**Frontend( React.js)**

**App.jsx**

```
import { useState } from 'react'
import './App.css'
import NavScrollExample from './components/Navbar';
import DarkVariantExample from './components/BannerSlider';
import CardContainer from './components/CategoryContainer/CardContainer';
import {BrowserRouter,Routes,Route } from 'react-router-dom';
import PlaintextExample from './components/SignIn/Login';
import Homepage from './components/Homepage';
import Register from './components/SignIn/SignUp';
import GetProductTest from './components/Products/GetProductTest';
import AddProduct from './components/Products/AddProduct';
import CartItem from './components/Cart/CartItem';


function App() {


  return (
    <>
    <BrowserRouter>
    <NavScrollExample />
    <Routes>

      <Route path="/login" element={<PlaintextExample />} />
      <Route path="/register" element={<Register />} />
      <Route path="/" element={<Homepage />} />
      <Route path="/productlist" element={<GetProductTest />} />
      <Route path="/addProduct" element= {<AddProduct />} />
      <Route path="/Cart" element={<CartItem />}></Route>

    </Routes>
    </BrowserRouter>
    </>
  )
}

export default App
```

**Navbar.jsx**

```
import Button from 'react-bootstrap/Button';
import Container from 'react-bootstrap/Container';
import Form from 'react-bootstrap/Form';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';
import {Link} from 'react-router-dom';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { faCartShopping,faMagnifyingGlass ,faUser} from '@fortawesome/free-solid-
svg-icons'

import {useState, useEffect} from 'react';

function NavScrollExample() {
  const [cartCount, setCartCount] = useState(0);

  useEffect(() => {
    // Load the cart count from local storage when the component mounts
    let productNumbers = localStorage.getItem('cartItems');

    if (productNumbers) {
      setCartCount(parseInt(productNumbers));
    }
  }, []);


  return (
    <Navbar expand="lg" className="bg-body-tertiary fixed-top shadow">
      <Container fluid className="mx-5">
        <Navbar.Brand className="nav-brand" href="/">E-commerce</Navbar.Brand>
        <Navbar.Toggle aria-controls="navbarScroll" />
        <Navbar.Collapse id="navbarScroll">
          <Nav
            className="me-auto my-2 my-lg-0"
            style={{ maxHeight: '100px' }}
            navbarScroll
          >

            <NavDropdown title="Category" id="navbarScrollingDropdown" >
              <NavDropdown.Item href="#action3">Bike</NavDropdown.Item>
              <NavDropdown.Item href="#action4">
                Laptop
              </NavDropdown.Item>
              <NavDropdown.Item href="#action5">
                Smart Phone
              </NavDropdown.Item>
            </NavDropdown>
          <Link to="/addProduct" className="text-bold mx-3 mt-2 text-decoration-
none text-black "> Sell Product </Link>
          </Nav>


          <Form className="d-flex">
            <Form.Control
              type="search"
              placeholder="Search"
              className="me-2"
              aria-label="Search"
            />
            <Button variant="outline-info" className="rounded-circle">
<FontAwesomeIcon icon={faMagnifyingGlass} /></Button>
          </Form>
          <Nav className="ms-auto my-2 my-lg-0">

            <Link to="/" className="navbar-cart text-bold mx-2 text-decoration-
none "><FontAwesomeIcon icon={faCartShopping} color="black"/> Cart
<span>{cartCount}</span> </Link>
            <Link to="/order" className="text-bold mx-2 text-decoration-none ">
Order </Link>
            <Link to="/login" className="text-bold mx-2 text-decoration-none ">
<FontAwesomeIcon icon={faUser} color="black"/> Login </Link>
            <Link to="/register" className="text-bold mx-2 text-decoration-none">
<FontAwesomeIcon icon={faUser} color="black"/> Sign-Up </Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default NavScrollExample;
```

**Homepage.jsx**

```
import React from 'react'
import NavScrollExample from './Navbar'
import DarkVariantExample from './BannerSlider'
import CardContainer from './CategoryContainer/CardContainer'
import GetProductTest from './Products/GetProductTest'
```

```
export default function Homepage() {


  return (
    <>
    <div className="Mainbody" >
    <DarkVariantExample />
    <CardContainer />
    <GetProductTest />
    <br/>
    <br/>
    </div>
    </>
  )
}
```

**BannerSlider.jsx**

```
import Carousel from 'react-bootstrap/Carousel';
import React from 'react';
import Img1 from '../assets/images/banner6.jpg';
import Img2 from '../assets/images/banner8.jpg';
import Img3 from '../assets/images/banner9.jpg';



function DarkVariantExample() {
    const image = {
        height : "270px",
        objectFit:"fill",
        padding: "15px"
    }
    const carousel = {
      marginTop:"60px"
    }
  return (
    <>
    <div className="container-fluid ">
      <Carousel data-bs-theme="light"  style={carousel}>
        <Carousel.Item>
          <img
            className="d-block w-100 "
            style = {image}
            src={Img1}
            alt="First slide"
          />

        </Carousel.Item>
        <Carousel.Item>
          <img
            className="d-block w-100"
            style = {image}
            src={Img2}
            alt="Second slide"
          />

        </Carousel.Item>
        <Carousel.Item>
          <img
            className="d-block w-100"
            style = {image}
            src={Img3}
            alt="Third slide"
          />

        </Carousel.Item>
      </Carousel>
    </div>
    </>
  );
}

export default DarkVariantExample;
```

**Login.jsx**

```
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button';
import React, { useState } from 'react';
import axios from "../AxiosRequests/BaseUrl";
import 'react-toastify/dist/ReactToastify.css';
import { ToastContainer, toast } from 'react-toastify';
import NavScrollExample from '../Navbar';
import { useNavigate } from 'react-router-dom';


export default function loginFunction() {
  const navigate = useNavigate();
  const [errors, SetErrors] = useState({
    email:"",
    password : ""
  })

  const [loginDetails,setLoginDetails] = useState({
    email:"",
    password:""
  })

  const handleChange = (e)=>{
  setLoginDetails((prev)=>{
    return{
      ...prev,
      [e.target.name] : e.target.value
    }
  })
  }
  const handleSubmit = async(e,values)=>{

    if(values.email.length== 0){
    SetErrors((prev)=>{
      return {
        ...prev,
        email:"Email is required"
      }
    })
    }
    else if (values.password.length == 0){
      SetErrors((prev) => {
        return {
          ...prev,
          password : "password is required"
        }
      })
    }
    else{
    axios.post("/login",values).then((res) => {
    console.log(res)
    toast.success("Login successfully")
    setTimeout(()=>{
      navigate("/")
      },1000)
    }).catch((error) => {
      if(error.response.status==401){

        toast.error("email or password incorrect")
      }
      else if(error.response.status==500){

        toast.error("Something went wrong")
      }
    })
    }
  e.preventDefault()
  }
return (
  <>

  <div className="container w-50 mt-5 shadow p-3" >
      <ToastContainer/>
      <h4 className="text-center fw-bold"> Login Page</h4>
      <Form onSubmit={(e)=>handleSubmit(e,loginDetails)} noValidate>
        <Form.Group  className="mb-3 p-4" controlId="formPlaintextEmail">
          <Form.Label >
            Email
          </Form.Label>
          <Form.Control type="email"
placeholder="email@example.com"  name="email" onChange={handleChange}
value={loginDetails.email}/>
          <p className="text-danger">{errors.email} </p>
        </Form.Group>
        <Form.Group  className="mb-3 p-4" controlId="formPlaintextPassword">
          <Form.Label>
            Password
          </Form.Label>

          <Form.Control type="password" placeholder="Password" name="password"
onChange={handleChange} value={loginDetails.password}/>
          <p> {errors.password} </p>
        </Form.Group>
        <Button variant="outline-primary" type='submit'>Login</Button>{' '}
      </Form>
    </div>
    </>
  );
}
```

**SignUp.jsx**

```
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';
import Button from 'react-bootstrap/Button';
import React, {  useState } from 'react';
import axios from '../AxiosRequests/BaseUrl';
import 'react-toastify/dist/ReactToastify.css';

import { ToastContainer ,toast} from 'react-toastify';
import NavScrollExample from '../Navbar';
import { useNavigate } from 'react-router-dom';


function Register() {
  const navigate = useNavigate();
  const [error, SetError] = useState ({
    firstName:"",
    lastName: "",
    email:"",
    password:"",
    password2:""
  })
  const [RegDetails, SetRegDetails] = useState({
    firstName:"",
    lastName: "",
    email:"",
    password:"",
    password2:""
  })

  const handleChange = (e) => {
    SetRegDetails((prev) => {
      return {
        ...prev,
        [e.target.name]: e.target.value
      }
    })
  }

  const handleSubmit = async(e,values) => {
        if (values.password !== values.password2) {
          SetError((prev) =>{
            return {
              ...prev,
              password2: "Passwords doesn't Match"
            }
          })
        }
        else if(values.password === 0){
          SetError((prev) =>{
            return {
              ...prev,
              password: "Password is required"
            }
          })
        }
        else if(values.password2 === 0){
          SetError((prev) =>{
            return {
              ...prev,
              password2: "Password2 is required"
            }
          })
        }
        else if(values.firstName === 0){
          SetError((prev) =>{
            return {
              ...prev,
              firstName: "First Name is required"
            }
          })
        }
        else if(values.lastName === 0){
          SetError((prev) =>{
            return {
              ...prev,
              lastName: "Last Name is required"
            }
          })
        }
        else if(values.email === 0){
          SetError((prev) =>{
            return {
              ...prev,
              email: "Email is required"
            }
          })
        }
        else
        axios.post("/UserData",values).then((res) => {
          console.log(res)
          toast.success("Registration Successful")
          setTimeout(()=>{
            navigate("/")
          },1000)
        }).catch ((error) => {

        console.log(error.message)
        if(error.response.status == 400 ){
          toast.error("Bad Request")
        }
        else if(error.response.status == 500){
          toast.error("Something went wrong")
        }

      })



  e.preventDefault();

  }



  return (
    <>

    <div className="container shadow p-3 mt-5 justify-content-center p-5" >

    <ToastContainer/>
      <h4 className="text-center fw-bold"> User Registration</h4>

      <Form onSubmit={(e) => handleSubmit(e,RegDetails)} >
        <Form.Group as={Row} className="mb-3" controlId="fName">
          <Form.Label >
            First Name
          </Form.Label>
          <Form.Control type="text" placeholder="First Name" name="firstName"
onChange={handleChange} value={RegDetails.firstName} />
          <p className="text-danger">{error.firstName}</p>
        </Form.Group>

        <Form.Group as={Row} className="mb-3 " controlId="lName">
```

```
          <Form.Label >
            Last Name
          </Form.Label>
          <Form.Control type="Text" placeholder="Last Name" name="lastName"
onChange={handleChange} value={RegDetails.lastName} />
          <p className="text-danger">{error.lastName}</p>
        </Form.Group>

        <Form.Group as={Row} className="mb-3" controlId="UserEmail">
          <Form.Label >
            Email
          </Form.Label>
          <Form.Control type="email" placeholder="email@example.com"
name="email"  onChange={handleChange} value={RegDetails.email}/>
          <p className="text-danger">{error.email}</p>
        </Form.Group>

        <Form.Group as={Row} className="mb-3" controlId="Password">
          <Form.Label >
            Password
          </Form.Label>
          <Form.Control type="password" placeholder="Password"
name="password"  onChange={handleChange} value={RegDetails.password}/>
          <p className="text-danger">{error.password}</p>
        </Form.Group>
        <Form.Group as={Row} className="mb-3" controlId="ConfirmPassword">
          <Form.Label >
            Confirm Password
          </Form.Label>
          <Form.Control type="password" placeholder=" Confirm Password"
name="password2" onChange={handleChange}  value={RegDetails.password2}/>
          <p className="text-danger">{error.password2}</p>
        </Form.Group>
        <Button variant="outline-primary" type='submit'>Register</Button>{'  '}
      </Form>
    </div>

    </>
  );
}

export default Register;
```

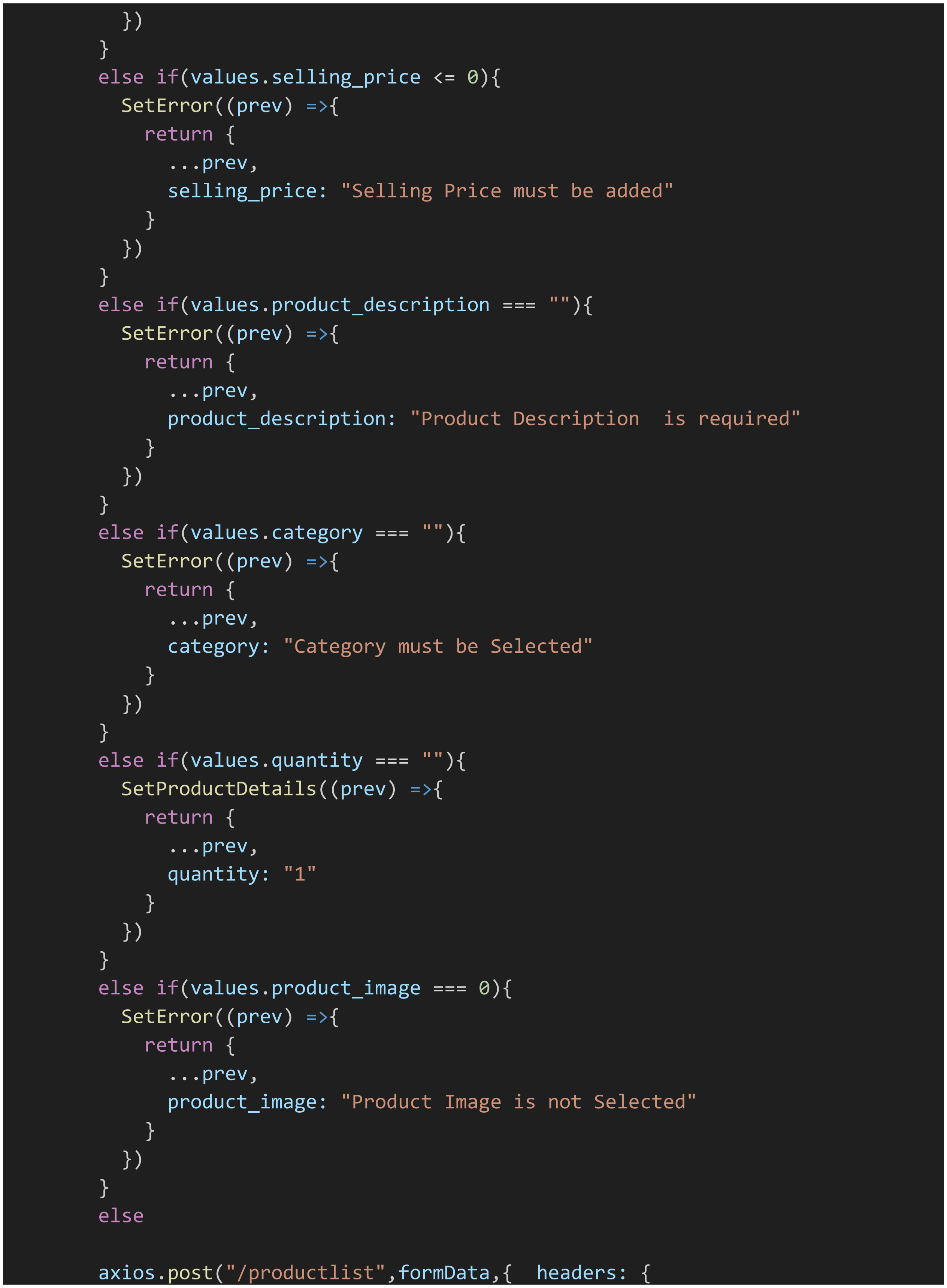
**GetProductTest.jsx**

```
import React,{useState, useEffect} from 'react';
import axios from '../AxiosRequests/BaseUrl';
import Card from 'react-bootstrap/Card';
import ListGroup from 'react-bootstrap/ListGroup';
import Button from 'react-bootstrap/Button';
import CartItem from '../Cart/CartItem';


export const ProductDataContext = React.createContext();

export default function GetProductTest() {


  const [ProductData, setProductData]=useState ([]);
  const [isError, setIsError] = useState("");

  const GetApiData = async() => {
        try {
            const res = await axios.get("/productlist");
            setProductData(res.data);
            console.log(res.data)

        } catch (error) {
            setIsError(error);
            console.log(error)
        }

  }
  const CardStyle = {
      height:"380px",
      width: "300px"
  }
  const ImgStyle = {
      height:"150px",
      objectFit: "",
      // width: "100%"
  }
  const TitleStyle = {
      height: "100px",
  }

  const ButtonStyle = {
      height: "60px"
  }
  let carts = document.querySelectorAll('.add-cart');



  function cartNumbers(productDetails) {
    let productNumbers = localStorage.getItem('cartItems');
    productNumbers = parseInt(productNumbers);

    if (productNumbers) {
      localStorage.setItem('cartItems', productNumbers + 1);
      document.querySelector('.navbar-cart span').textContent = productNumbers +
1;
    } else {
      localStorage.setItem('cartItems', 1);
      document.querySelector('.navbar-cart span').textContent = 1;
    }

    setItem(productDetails);
  }

function setItem(productDetails){


}
const AddCart = (product) => {
  let productDetails = {
    product: {
      title: product.product_title,
      image: product.product_image,
      description: product.product_description,
      selling_price: product.selling_price,
      discounted_price: product.discounted_price,
      quantity: product.quantity,
    }
  }

  // Output the object to the console
  console.log(productDetails);
  localStorage.setItem('productdetails', JSON.stringify(productDetails));
  cartNumbers(productDetails); // Pass the productDetails object to cartNumbers
}
```

```
  // useEffect(() => {
  //  axios
  //   .get("http://127.0.0.1:8000/UserData")
  //   .then((res) => setUserData(res.data))
  //   .catch((error)=> setIsError(error.message));
  // },[])

  useEffect(() => {
    GetApiData();
  },[])



  return (
    <>

    <div className=" Product-List ">
    <span className = "section-title fw-bold mx-5 " > For You </span>
    {isError !== "" && <h2>{isError} </h2>}
    <div className="product-list mt-2 d-flex gap-5 justify-content-space-between
flex-wrap mx-5">
     {ProductData.map((product) => {
      const {id,product_title,product_image,product_description, selling_price,
discounted_price} = product;
      return <div className="product shadow"  key={id}>
        <Card  className="card-main" style={CardStyle}>
          <Card.Img variant="top" src={"http://localhost:8000" + product_image}
style={ImgStyle} />
          <Card.Body style={TitleStyle}>
            <Card.Title >{product_title}</Card.Title>
          </Card.Body>
          <ListGroup className="list-group-flush" >
            <ListGroup.Item className="sell-price text-danger text-decoration-
line-through">Rs. {selling_price}</ListGroup.Item>
            <ListGroup.Item className="discount-price">Rs.
{discounted_price}</ListGroup.Item>
          </ListGroup>
          <Card.Body style={ButtonStyle}>
            <Button variant="primary" type="submit" className="add-cart"
onClick={() => AddCart(product)}>Add To Cart</Button>
          </Card.Body>
        </Card>
      </div> }

     )}
    </div>
    </div>
</>
)
}
```

**AddProduct.jsx**

```
import React,{useState} from 'react';
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button' ;
import axios from '../AxiosRequests/BaseUrl';
import 'react-toastify/dist/ReactToastify.css';
import { toast, ToastContainer} from 'react-toastify';
import NavScrollExample from '../Navbar';
import { useNavigate } from 'react-router-dom';


export default function AddProduct() {

  const navigate = useNavigate();
    const [error, SetError] = useState ({
        product_title:"",
        product_description: "",
        selling_price:"",
        discounted_price:"",
        category:"",
        quantity:"",
        product_image:null,
    })

    const [ProductDetails, SetProductDetails] = useState({
        product_title:"",
        product_description: "",
        selling_price:"",
        discounted_price:"",
        category:"",
        quantity:"",
        product_image:null,

    })
    const handleChange = (e) => {
        if (e.target.type === "file"){
            SetProductDetails((prev) => ({
                ...prev,
                [e.target.name]: e.target.files[0], // Assuming you want to store
the first selected file
              }));
        }
        else{
            SetProductDetails((prev) => ({
                ...prev,
                [e.target.name]: e.target.name === 'selling_price' ||
e.target.name === 'discounted_price'
                  ? parseFloat(e.target.value)
                  : e.target.value,
              }));
            }
    }

    const handleSubmit = async(e,values) => {
        SetError({});
        const formData = new FormData();
        formData.append("product_title", values.product_title);
        formData.append("product_description", values.product_description);
        formData.append("selling_price", values.selling_price);
        formData.append("discounted_price", values.discounted_price);
        formData.append("category", values.category);
        formData.append("quantity", values.quantity);
        formData.append("product_image", values.product_image);
        if (values.product_title === "") {
          SetError((prev) =>{
            return {
              ...prev,
              product_title: "Product Name is Required"
            }
          })
        }
        else if(values.discounted_price <= 0){
          SetError((prev) =>{
            return {
              ...prev,
              discounted_price: "Discounted Price must be added so that When
offer comes , We can show this "
            }
```



```
          })
        }
        else if(values.selling_price <= 0){
          SetError((prev) =>{
            return {
              ...prev,
              selling_price: "Selling Price must be added"
            }
          })
        }
        else if(values.product_description === ""){
          SetError((prev) =>{
            return {
              ...prev,
              product_description: "Product Description  is required"
            }
          })
        }
        else if(values.category === ""){
          SetError((prev) =>{
            return {
              ...prev,
              category: "Category must be Selected"
            }
          })
        }
        else if(values.quantity === ""){
          SetProductDetails((prev) =>{
            return {
              ...prev,
              quantity: "1"
            }
          })
        }
        else if(values.product_image === 0){
          SetError((prev) =>{
            return {
              ...prev,
              product_image: "Product Image is not Selected"
            }
          })
        }
        else

        axios.post("/productlist",formData,{  headers: {
            "Content-Type": "multipart/form-data", // Set the content type to 
multipart/form-data
          },
        })
        .then((res) => {
          console.log(res)
          toast.success("Product Added Successful")
            setTimeout(()=>{
navigate("/")
            },1000)



        }).catch ((error) => {
        console.log(error)
        console.log(error.message)
        if(error.response.status == 400 ){
          toast.error("Bad Request")
        }
        else if(error.response.status == 500){
          toast.error("Something went wrong")
        }

      })



  e.preventDefault();

  }









  return (
    <>
    <div className="container"  >

    <div className="add-product mt-5 mx-5 shadow p-5" >
    <ToastContainer/>
        <h4 className="text-center text-primary" > Add Product</h4>
        <Form  onSubmit={(e) => handleSubmit(e,ProductDetails)}
encType="multipart/form-data">
            <Form.Group controlId="product_title" className="mb-3">
                <Form.Label className="">Product Name</Form.Label>
                <Form.Control type="text"
name="product_title"  onChange={handleChange}
value={ProductDetails.product_title} />
                <p className="text-danger" >{error.product_title}</p>
            </Form.Group>
            <Form.Group controlId="product_description" className="mb-3">
                <Form.Label className="">Product Description</Form.Label>
                <Form.Control type="text"
as="textarea"  name="product_description" rows={3} onChange={handleChange}
value={ProductDetails.product_description} />
                <p className="text-danger" >{error.product_description}</p>
            </Form.Group>
            <Form.Group controlId="selling_price" className="mb-3">
                <Form.Label className="">Selling Price</Form.Label>
                <Form.Control type="number"
name="selling_price"  onChange={handleChange}
value={ProductDetails.selling_price} />
                <p className="text-danger" >{error.selling_price}</p>
            </Form.Group>
            <Form.Group controlId="Discounted_price" className="mb-3">
                <Form.Label className="">Discounted Price</Form.Label>
                <Form.Control type="number"
name="discounted_price"  onChange={handleChange}
value={ProductDetails.discounted_price}/>
                <p className="text-danger" >{error.discounted_price} </p>
            </Form.Group>
            <Form.Group controlId="category" className="mb-3">
                <Form.Label >Product Category</Form.Label>
                <Form.Control
                    as="select"
                    onChange={handleChange}
                    name="category"
                    values={ProductDetails.category}
                >
                    <option value="">Select...</option>
                    <option value="BK">Bike</option>
                    <option value="SM">Smartphone</option>
                    <option value="CH">Charger</option>
                    <option value="WH">Watch</option>
                    <option value="TM">Trimmer</option>
                    <option value="LP">Laptop</option>
                </Form.Control >
                <p className="text-danger" >{error.category}</p>
            </Form.Group>
            <Form.Group controlId="quantity" className="mb-3">
                <Form.Label >Quantity of Product</Form.Label>
                <Form.Control type="number"
name="quantity"  onChange={handleChange} value={ProductDetails.quantity}/>
                <p className="text-danger" >{error.quantity} </p>
            </Form.Group>
            <Form.Group controlId="product_image" className="mb-3">
                <Form.Label >Product Image</Form.Label>
                <Form.Control type="file" name="product_image"
onChange={handleChange} />
                <p className="text-danger" >{error.product_image}</p>
            </Form.Group>

            <Button variant="outline-primary" type="submit" >
                Add Product
            </Button>
        </Form>
    </div>
    </div>
    </>
  )
}
```

**CardItem.jsx**

```
import Card from 'react-bootstrap/Card';
import React from 'react';


function BasicExample({ image, Title }) {

  const CardStyle = {
    height:"200px",
    width:"300px",
    backgroundColor : "black",
    borderRadius:"15px"
  }
  const CardBody ={
    height:"20%",
    textAlign: "center"
  }

  const CardImg = {
    height: "80%",
    borderTopLeftRadius: "15px",
    borderTopRightRadius: "15px"
  }
  return (

      <Card className='text-white mt-2'  style={CardStyle}>
        <Card.Img variant="top" src={image} style={CardImg}/>

          <Card.Title style={CardBody}>{Title}</Card.Title>


      </Card>
  );
}

export default BasicExample;
```

**CardContainer.jsx**

```
import React from 'react'
import BasicExample from './CardItem'
import Img1 from '../../assets/images/banner6.jpg';
import Img2 from '../../assets/images/banner8.jpg';
import Img3 from '../../assets/images/banner9.jpg';
import Img4 from '../../assets/images/laptop.jpg';
import Img5 from '../../assets/images/bike.jpg';
import Img6 from '../../assets/images/applewatch.jpg';
import Img7 from '../../assets/images/iphone 11.jpg';


export default function CardContainer() {

  const BackgroundSection = {
    backgroundColor: "light",
    borderRadius: "20px"
  }
  return (
    <>
        <div className="container-fluid " >
            <div className="section p-4" style={BackgroundSection}>
                <span className = "section-title fw-bold " > Category </span>
                <div className="section-body d-flex gap-5 justify-content-space-
between flex-wrap">
                    <BasicExample image={Img4} Title="Laptop" />
                    <BasicExample image={Img5} Title="Bike" />
                    <BasicExample image={Img6} Title="Applewatch" />
                    <BasicExample image={Img7} Title="Smart phone" />


                </div>
            </div>


        </div>

    </>
  )
}
```

### CartItem.jsx

```
import React , { useContext, useEffect } from 'react'
import NavScrollExample from '../Navbar'
import Card from 'react-bootstrap/Card';
import Img7 from '../../assets/images/iphone 11.jpg';
import Col from 'react-bootstrap/Col';
import Row from 'react-bootstrap/Row';
import { ProductDataContext } from '../Products/GetProductTest';

export default function CartItem() {
  const ProductData = useContext(ProductDataContext);
   const TopMargin = {
    marginTop : "80px"
   }
   const CartContainer = {
    height:"100px",
    width:"100px",
    borderRadius: "15px"
   }
  return (
    <>
    <NavScrollExample />
    <div className="cart" style={TopMargin}>
      <h3 className="text-center"> Your Cart </h3>

      <div className="cart-container mx-5 mt-5" >
        <Row>
          <Col xs={6} className="fw-bold "> Product</Col>
          <Col xs={2} className="fw-bold"> Price</Col>
          <Col xs={2}className="fw-bold">Quantity</Col>
          <Col xs={2}className="fw-bold"> Total</Col>
        </Row>
        <hr/>

        <Card.Img variant="top" src={Img7} style={CartContainer} />

      </div>

    </div>
    </>
  )
}
```

**BaseUrl.jsx**

```
import axios from "axios";


const Api = axios.create({
    baseURL : "http://127.0.0.1:8000" ,
});

export default Api;
```