in collaboration with

Softwarica College of IT & E-commerce | Coventry University

Reflective Report
On
Gaming Accessories House
For
STW205CDE -Developing the Modern Web

Submitted By:

Name: Sabin Dangal

University ID-10174700

YouTube video Link: https://youtu.be/zH0KjypbGWU

Repository Link: https://github.com/softwarica-github/final-assignment-sabin149-1.git

Important notes
Please refer to the Assignment Presentation Requirements for advice on how to set out your assignment. These can be found on the *Softwarica' s Moodle Course Page*.
You are expected to use the CU Harvard referencing format on any written work. For support and advice on how this students can contact Centre for Academic Writing (CAW).
Please notify your registry course support team and module leader for disability support.
Any student requiring an extension should follow the university process as outlined at here
The College cannot take responsibility for any coursework lost or corrupted on disks, laptops or personal computer. Students should therefore regularly back-up any work and are advised to save it on external media or system.
If there are technical or performance issues that prevent students submitting Coursework through the online coursework submission system on the day of a coursework deadline, an appropriate extension to the coursework submission deadline will be agreed. This extension will normally be 24 hours or the next working day if the deadline falls on a Friday or over the weekend period. This will be communicated via email and as a Softwarica' s Moodle announcement.
You must complete the 'Assessment Submission and Declaration Form'. The form is available on *Softwarica' s Moodle Course Page*.
Please make a note of the recommended word count. You could lose marks if you write 10%moreor less than this.
You must submit a paper copy and digital copy (on disk or similarly acceptable medium). Media containing viruses, or media that cannot be run directly, will result in a fail grade being awarded for this assessment.
All electronic media will be checked for plagiarism.

Table of Contents

Chapter 1 Introduction: 3
1.1 Brief Introduction of the System 3
1.2 Features: 3
1.3 Aims: 3
1.4 Objectives: 4
Chapter 2. Frontend 4
2.1 Technologies: 4
2.2 Screenshots: 6
2.3 Importance of responsive design 26
2.4 Responsive Check 27
Chapter 3 Databases 32
3.1 Introduction 32
3.3 Screenshots 33
Chapter 4 Backend 37
4.1 Technologies: 37
4.2 Screenshots 37
Chapter 5 Project Issues 49
5.1 Issues During the Project Development 49
5.2 Limitations of the Project 49
5.3 Future Works 49
Conclusion 50
Reference 51

Chapter 1 Introduction:

1.1 Brief Introduction of the System

Gaming Accessories House is the e-commerce website which sells different gaming accessories such as gaming headsets, mouse, monitor, controller, keyboard, etc. In this web application, customers can see all the gaming accessories products along with product details like product name, price, description, brand, etc. They search for different their desired products which are available on this website. They can add different products to the cart and purchased them later. For this, the customer should first login into the website. Then they can add a shipping address for delivering their products. Similarly, they can pay up for the products using PayPal or credit card.

1.2 Features:

- **Search Functionality:** On this website, customers can search for different products according to the product title, brand, and category.
- **Checkout Function:** The checkout is quite simple and easy. Customers can add multiple shipping addresses and choose one for delivering their orders.
- **Product Description:** On this website, customers see product descriptions in all products with accurate details.
- **Add to Cart:** Customer can add products to the cart. Carts are directly saved into the database in which customers can see cart products after many days.
- **Good Payment Method:** On this website, payment is done using secure payment service PayPal or credit cards.

1.3 Aims:

This website aims to accelerate sustainable growth by extending consumer presence, lowering service costs, and providing consumers with exclusive experiences.

1.4 Objectives:

Some of the objectives are listed below:

- **To provide a unique customer experience:**
  Websites should be made more user-friendly so that people will choose the website which contains more features comparing to the other websites.
- **To reduce cost management:**
  Digital marketing should be done so that it will help to reduce the cost which is spent in calling the customers.
- **To increase the loyal customer:**
  By providing customers the best experience will help to improve the customers in the website.
- **To develop specific goal:**
  While doing the e-commerce business, a certain target should be made to get more success in that business.
- **To make the website responsive.**
  Most people use smart mobile devices for buying anything online so a responsive website should be made.

Chapter 2. Frontend

2.1 Technologies:

Technologies that are used in frontend for creating this website are:

- **HTML:** HTML (Lubbers et al. 2011) which stands for Hypertext Markup Language is the markup language that is used to create web pages. It helps to define the layout of the system.
- **CSS:** Cascading Style Sheets (CSS) (Duckett 2011) helps to style HTML elements that are displayed on an electronic screen.
- **Bootstrap:** Bootstrap **(Spurlock 2013)** is the framework of CSS which is used to create a responsive website for supporting different screen-sized devices.

- **JavaScript:** JavaScript (JS) (Crockford 2008) is a scripting programming language that is used for updating and changing HTML and CSS. In other words, it is used for making web pages more interactive.
- **jQuery:** jQuery is a JavaScript framework that makes JavaScript programming a lot easier in DOM handling and AJAX.

2.2 Screenshots:

2.2.1 Project Structure

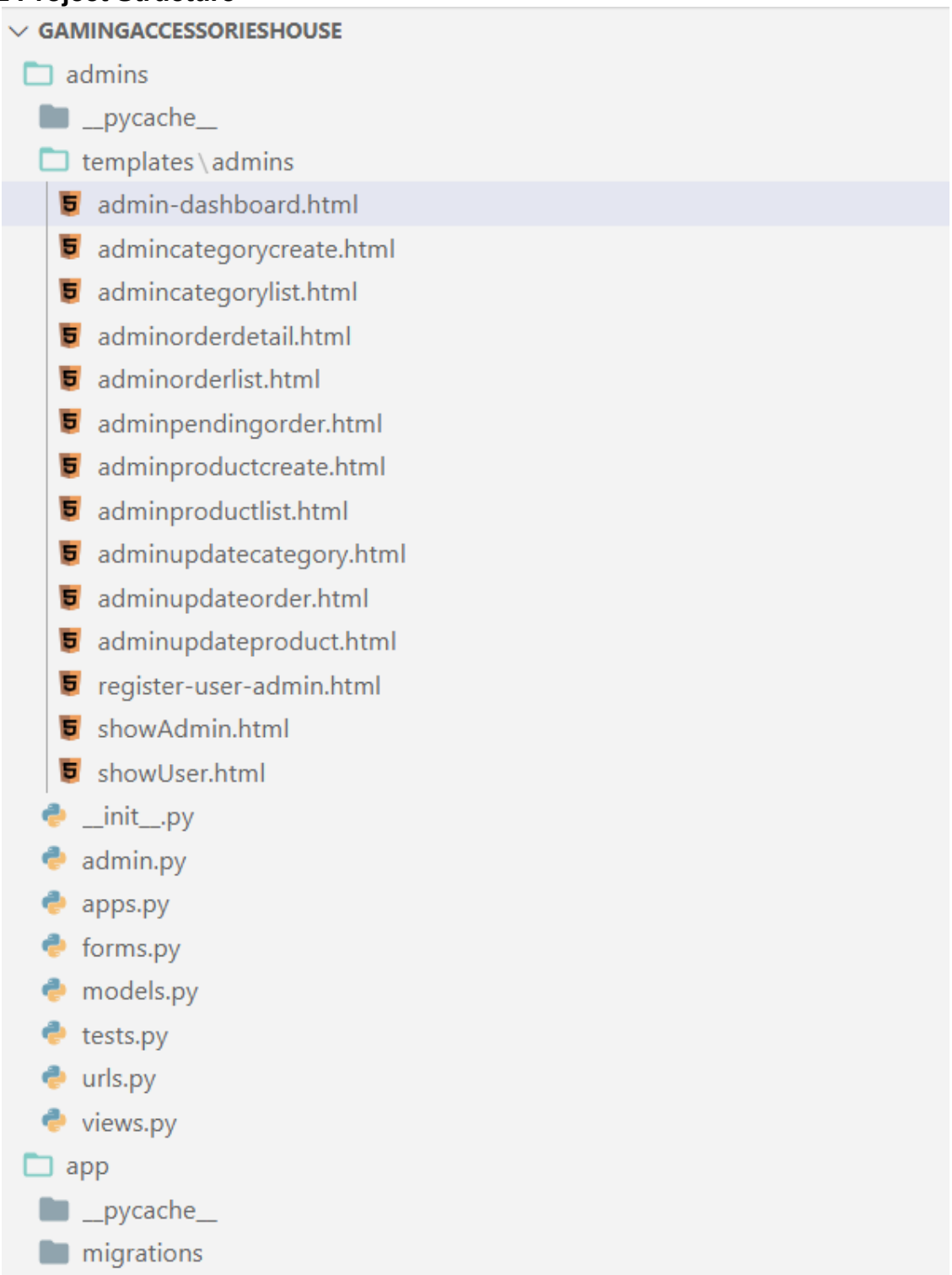

Figure 1 Project Structure

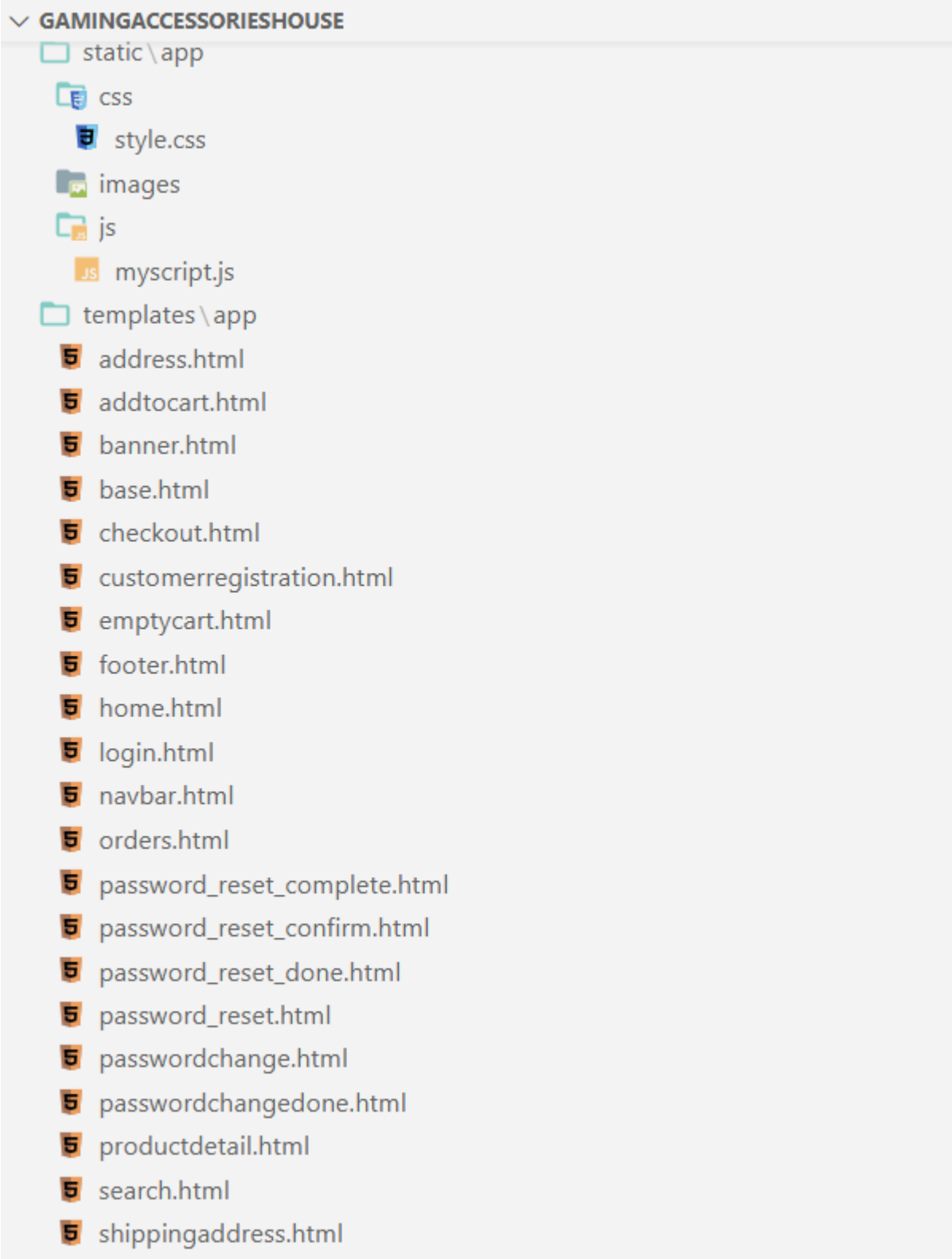

Figure 2 Project Structure

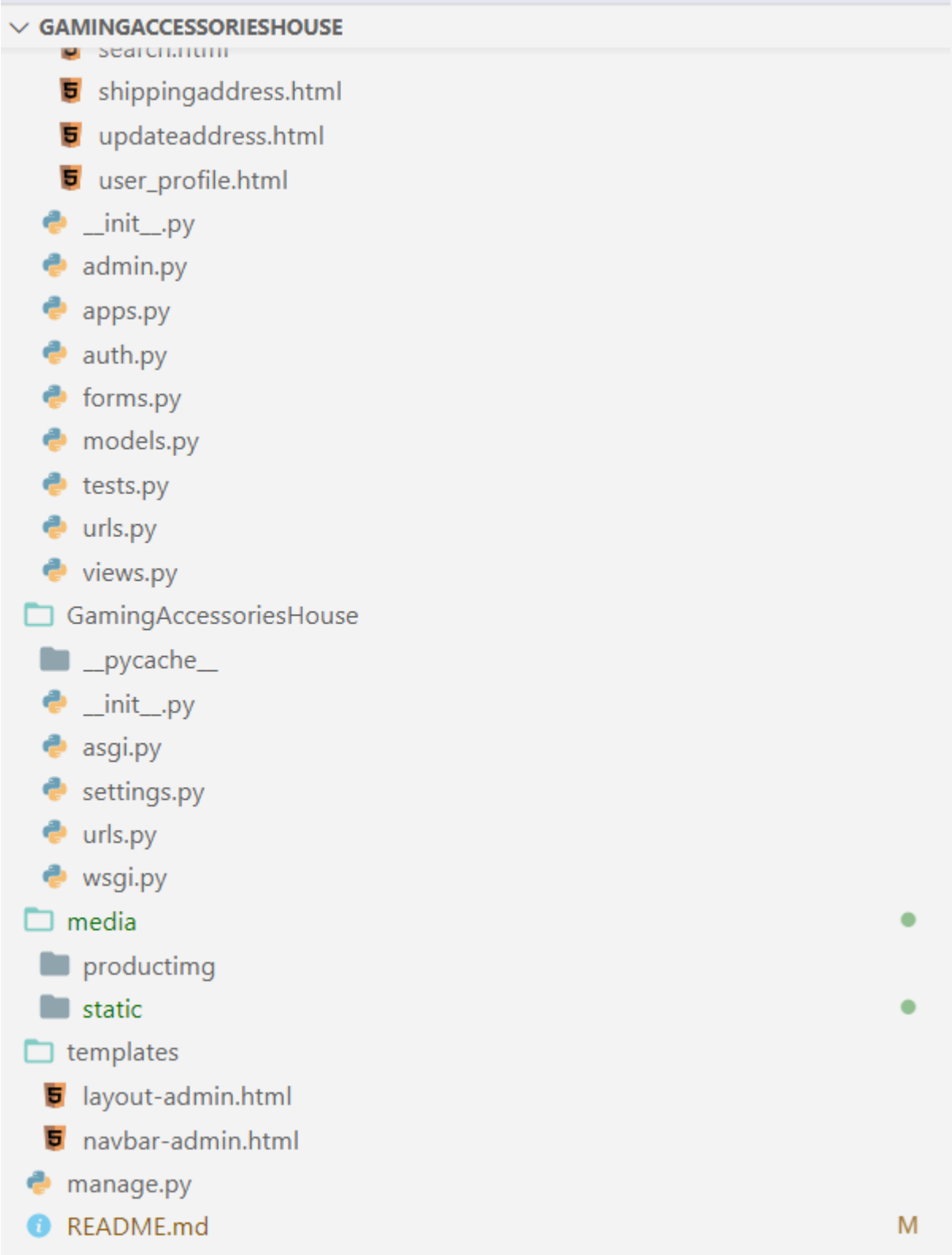

Figure 3 Project Structure

2.2.2 HTML Codes and Outputs

```html
{% extends 'app/base.html' %} {% block title %} All Products {% endblock %} {% block banner %}
{% include 'app/banner.html' %} {% endblock banner %} {% block main-content %}

<div class="container-fluid">
  <div class="all-products">
    <div class="container">
      <br />
      {% for cat in allcategories %}
      <p class="category-title">{{cat.title}}</p>
      <div class="row">
        {% for p in cat.product_set.all %}
        <div class="col-md-3 card mb-2 p-2 product-details">
          <div class="m-2">
            <a href="{% url 'productdetail' p.slug %}" class="btn">
              <img
                src="{{p.product_image.url}}"
                alt=""
                class="product-img card-img-top"
              />

              <div class="details">
                <span class="ptitle">{{p.title}}</span><br />
                <p>
                  <span class="mprice"> Rs. {{p.marked_price}}</span>
                  <span class="sprice">Rs. {{p.selling_price}}</span>
                </p>
              </div>
            </a>
          </div>
        </div>
        {% endfor %}
      </div>
      {% endfor %}
    </div>
  </div>
</div>

{% endblock main-content %}
```

Figure 4 All Products Code

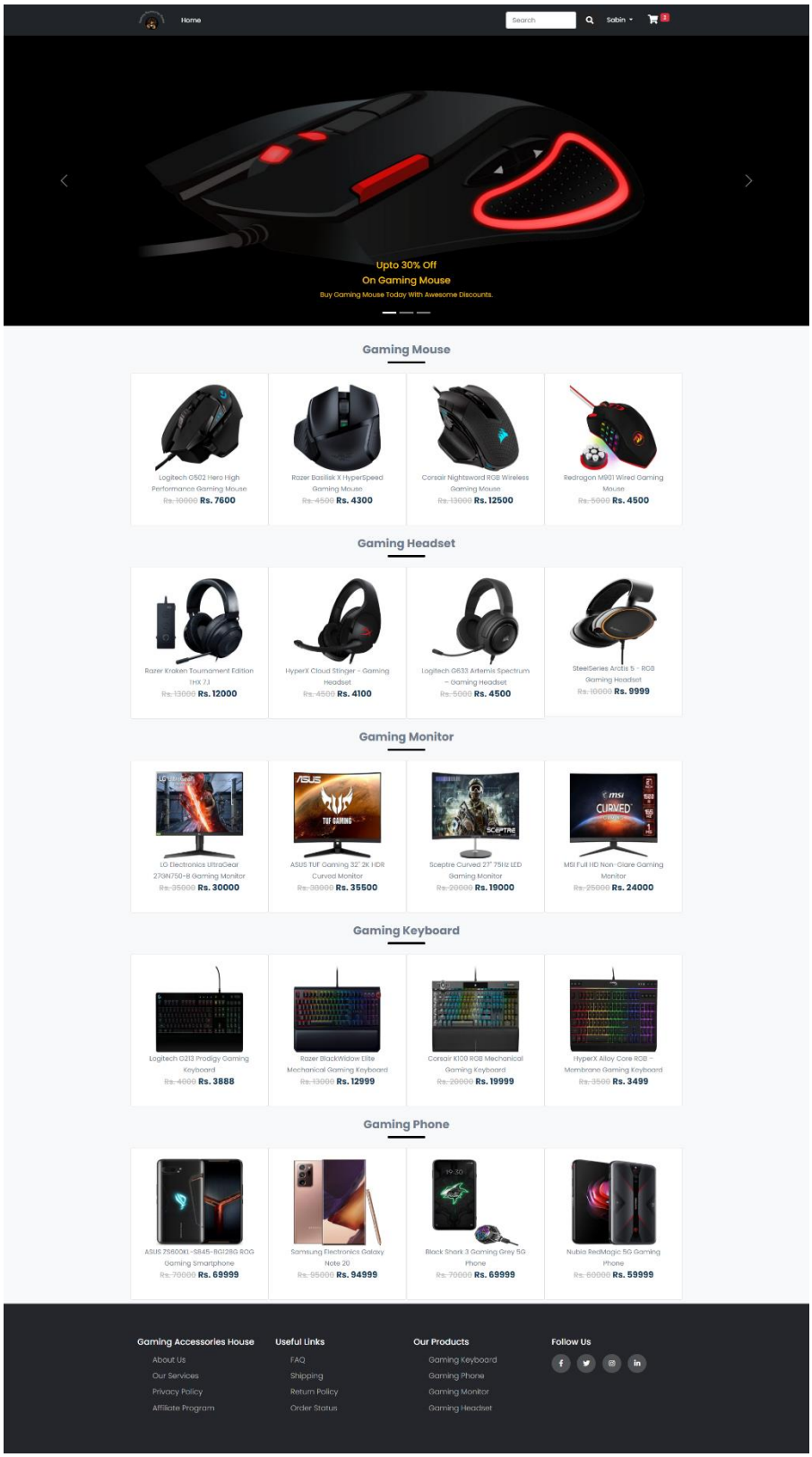

Figure 5 All Products Output

```html
{% extends 'layout-admin.html' %} {% block title %}
<title>Admin-Dashboard</title>
{% endblock %} {% block content %}

<div class="container">
  <h1>Dashboard</h1>
  {% for message in messages %}
  {% if message.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
  <h6 class="success_message py-3 px-1 my-2 text-center">{{message}}</h6>
  {% endif %} {% if message.level == DEFAULT_MESSAGE_LEVELS.ERROR%}
  <h6 class="error_message py-3 px-1 my-2 text-center">{{message}}</h6>
  {% endif %} {% endfor %}
  <div class="row">
    <div class="col">
      <div class="card text-white bg-success mb-3" style="max-width: 35rem">
        <div class="card-header text-center">Total Users</div>
        <div class="card-body c-body">
          <a href="/admin-dashboard/show-user"
            ><h1 class="card-title text-center">{{user_count}}</h1></a
          >
        </div>
      </div>
    </div>
    <div class="col">
      <div class="card text-white bg-danger mb-3" style="max-width: 35rem">
        <div class="card-header text-center">Total Admins</div>
        <div class="card-body c-body">
          <a href="/admin-dashboard/show-admin"
            ><h1 class="card-title text-center">{{admin_count}}</h1></a
          >
        </div>
      </div>
    </div>
  </div>
  <div class="row">
    <div class="col">
      <div class="card text-white bg-info mb-3" style="max-width: 35rem">
        <div class="card-header text-center">Total Products</div>
        <div class="card-body c-body">
          <a href="/admin-dashboard/admin-product/list"
            ><h1 class="card-title text-center">{{product_count}}</h1></a
          >
        </div>
      </div>
    </div>

    <div class="col">
      <div class="card text-white bg-secondary mb-3" style="max-width: 35rem">
        <div class="card-header text-center">Total Category</div>
        <div class="card-body c-body">
          <a href="/admin-dashboard/admin-category/list"
            ><h1 class="card-title text-center">{{category_count}}</h1></a
          >
        </div>
      </div>
    </div>
  </div>

  <div class="row">
    <div class="col">
      <div class="card text-white bg-dark mb-3" style="max-width: 35rem">
        <div class="card-header text-center">Total Pending Order</div>
        <div class="card-body c-body">
          <a href="/admin-dashboard/admin-pendingorder/"
            ><h1 class="card-title text-center">{{order_pending_count}}</h1></a
          >
        </div>
      </div>
    </div>
    <div class="col">
      <div class="card text-white bg-primary mb-3" style="max-width: 35rem">
        <div class="card-header text-center">Total Orders</div>
        <div class="card-body c-body">
          <a href="/admin-dashboard/admin-all-orders/"
            ><h1 class="card-title text-center">{{order_count}}</h1></a
          >
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

Figure 6 Admin Dashboard Code

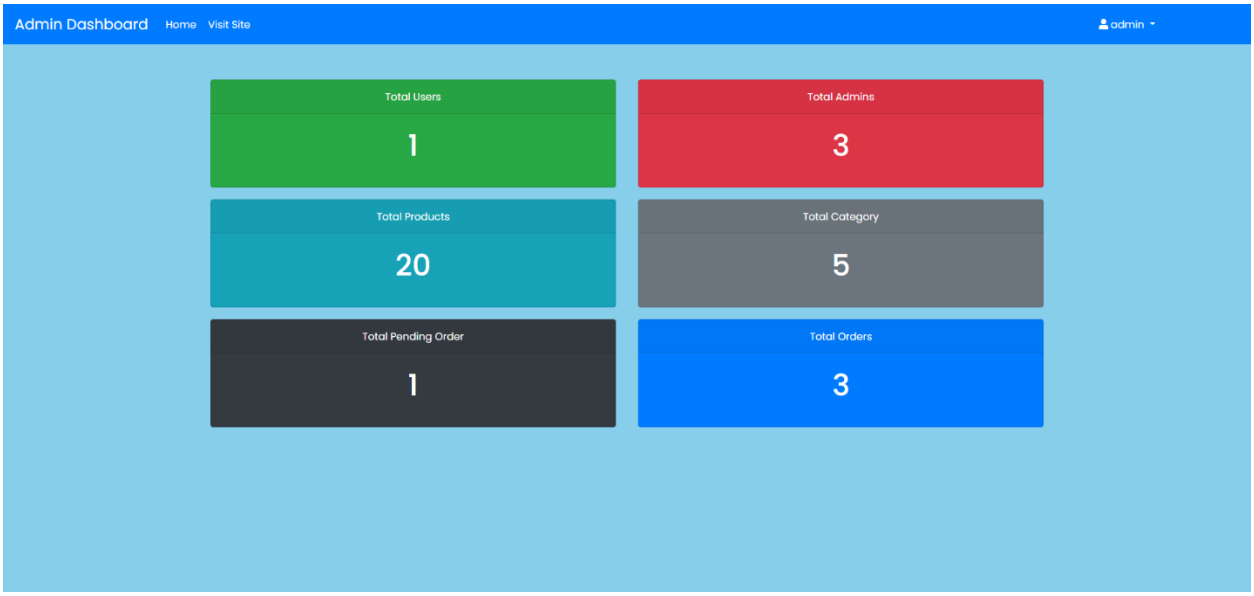

Figure 7 Admin Dashboard Output

```html
{% extends 'app/base.html' %} {% load static %}
{% block title %}
Cart
{% endblock title %} {% block main-content %}
<div class="container my-5">
  <div class="row">
    <h1 class="text-center mb-5">Shopping Cart</h1>
    <div class="col-sm-8">
      <div class="card">
        <div class="card-body">
          <h3>Cart</h3>
          {% for cart in carts %}
<hr>
          <div class="row">
            <div class="col-sm-3 text-center align-self-center">
              <img
                src="{{cart.product.product_image.url}}"
                alt=""
                srcset=""
                class="img-fluid img-thumbnail shadow-sm"
                height="150"
                width="150">
            </div>
            <div class="col-sm-9">
              <div>
                <h5>{{cart.product.title}}</h5>

                <div class="my-3">
                  <label for="quantity">Quantity:</label>
                  <a class="minus-cart btn" pid="{{cart.product.id}}"><i class="fas fa-minus-square fa-lg"></i
                  ></a>
                  <span id="quantity">{{cart.quantity}}</span>
                  <a class="plus-cart btn" pid="{{cart.product.id}}"
                    ><i class="fas fa-plus-square fa-lg"></i
                  ></a>
                </div>
                <div class="d-flex justify-content-between">
                  <a class="btn btn-sm btn-secondary mr-3 remove-cart" pid="{{cart.product.id}}"
                    >Remove item
                  </a>
                  <p class="mb-0">
                    <span><strong>Rs.{{cart.product.selling_price}}</strong></span>
                  </p>
                </div>
              </div>
            </div>
          </div>
          {% endfor %}
        </div>
      </div>
    </div>
    <div class="col-sm-4">
      <div class="card">
        <div class="card-body">
          <h3>The Total Amount of</h3>
          <ul class="list-group">
            <li
              class="list-group-item d-flex justify-content-between align-items-center border-0 px-0 pb-0">
              Amount<span>Rs. <span id="amount">{{amount}}</span></span>
            </li>
            <li
              class="list-group-item d-flex justify-content-between align-items-center border-0 px-0 mb-3">
              <div><strong>Total</strong> <small>(including VAT)</small></div>
              <span><strong>Rs. <strong id="totalamount">{{totalamount}}</strong></strong></span>
            </li>
          </ul>
          <div class="d-grid">
            <a href="{% url 'checkout' %}" class="btn btn-primary">Place Order</a>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock main-content %}
```

Figure 8 Shopping Cart Code

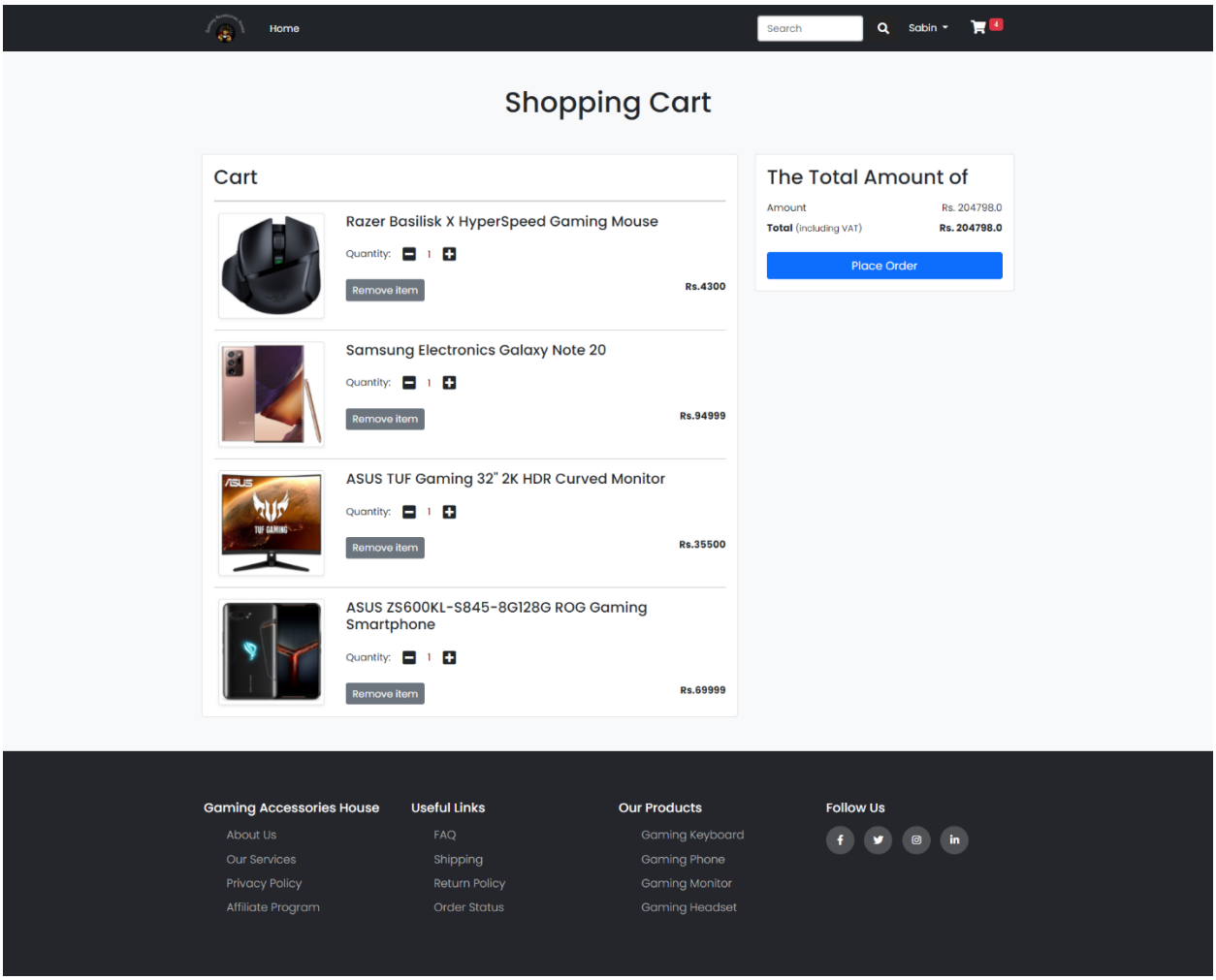

Figure 9 Shopping Cart Output

```
{% extends 'app/base.html' %} {% load static %} {% block title %}Buy Now
{% endblock title %} {% block main-content %}

<div class="container-lg">
  <div class="row mt-5">
    <div class="col-sm-7">
      <div class="card">
        <h4 class="mt-2 text-center">Order Summary</h4>
        <hr />
        <table class="table table-responsive">
          <thead>
            <tr>
              <th>SN</th>
              <th>Product</th>
              <th>Quantity</th>
              <th>Rate</th>
              <th>Total Cost</th>
            </tr>
          </thead>
          <tbody>
            {% for op in cart_items %}
            <tr>
              <td>{{forloop.counter}}</td>
              <td>{{op.product.title}}</td>
              <td>{{op.quantity}}</td>
              <td>Rs. {{op.product.selling_price}}</td>
              <td>Rs. {{op.total_cost}}</td>
            </tr>
            {% endfor %}
            <tr>
              <th colspan="5" class="text-center">
                Total Amount: Rs. {{totalamount}}/-
              </th>
            </tr>
          </tbody>
        </table>
      </div>
    </div>
    <div class="col-sm-4">
      <h4>
        <a href=" {% url 'shippingaddress' %} " class="btn btn-dark"
          >Add Address</a
        >
      </h4>
      {% if add %}
      <h4>Select Shipping Address</h4>
      <hr />
      <form action="/paymentdone" id="myform">
        {% for ad in add %}

        <div class="card">
          <div class="card-body">
            <h5>{{ad.name}}</h5>
            <p>
              {{ad.email}},{{ad.address}}, {{ad.city}}, {{ad.province}} -
              {{ad.zipcode}}
```

Figure 10 Payment Code 1

```
            </p>
          </div>
        </div>
        <div class="form-check mt-2 mb-5">
          <input
            class="form-check-input"
            type="radio"
            name="custid"
            id="custadd{{forloop.counter}}"
            value="{{ad.id}}"
          />

          <label
            class="form-check-label fw-bold"
            for="custadd{{forloop.counter}}"
          >
            Address: {{forloop.counter}}
          </label>
        </div>
        {% endfor %}
        <div class="text-end">
          <div id="paypal-button-container"></div>
        </div>
      </form>
      {% else %}
      <h2>Add Shipping Address</h2>
      {% endif %}
    </div>
  </div>
</div>
{% endblock main-content %} {% block payment-gateway %}
<script src=
"https://www.paypal.com/sdk/js?client-id=AelKoVrPsxbFGj6ea16Xotd79f1qLR7UhkvYRYCbIeq1OyoilgMUzAe1Rb9DDl2fK4trKvSwW
0I21yBJ
&currency=USD"></script>
<script>
  paypal
    .Buttons({
      createOrder: function (data, actions) {
        return actions.order.create({
          purchase_units: [
            {
              amount: {
                value: "{{totalamount}}",
              },
            },
          ],
        });
      },
      onApprove: function (data, actions) {
        return actions.order.capture().then(function (details) {
          alert(
            "Transaction completed by " + details.payer.name.given_name + "!"
          );
          document.getElementById("myform").submit();
        });
      },
    })
    .render("#paypal-button-container");
</script>
{% endblock payment-gateway %}
```

Figure 11 Payment Code 2

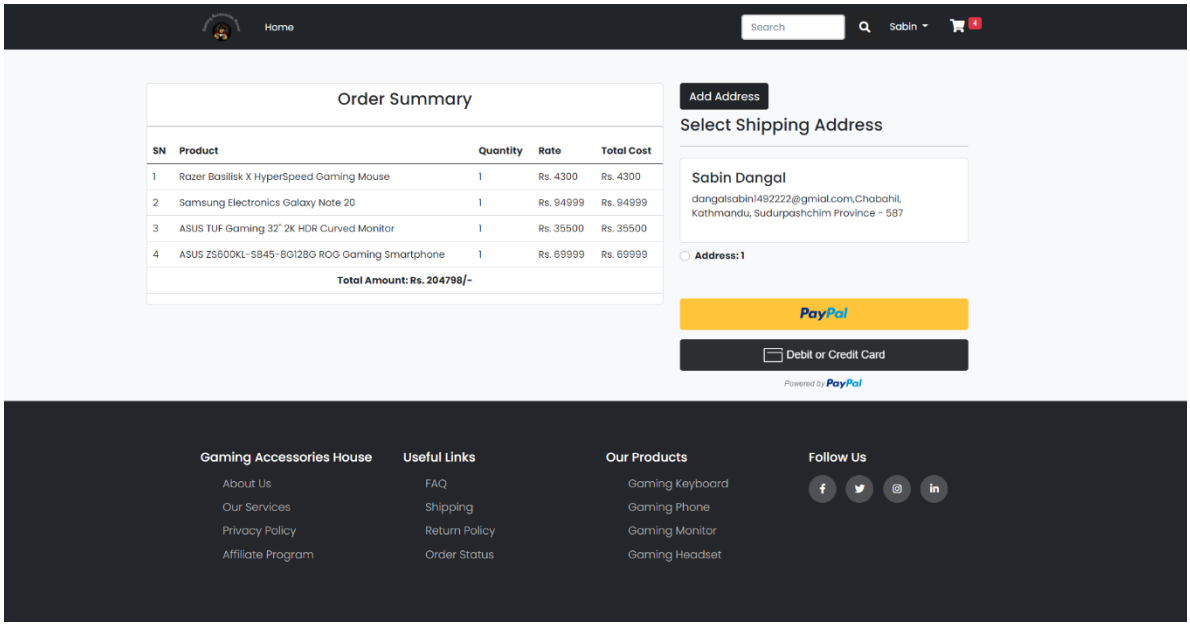

Figure 12 Payment Output

```
{% extends 'app/base.html' %} {% load static %} {% block title %} Product
Detail{% endblock title %} {% block main-content %}

<div class="container">
  <div class="row mt-5 pro-detail">
    <div class="col-sm-5 p-details card mb-5 p-4">
      <img src="{{product.product_image.url}}" alt="" class="p-image">
    </div>
    <div class="col-sm-6 offset-1 productdetails mt-5">
      <p class="p-title">{{product.title}}</p>
      <p>Brand: {{product.brand}}</p>

      <p>  <span class="text-decoration-line-through mprice text-dark"> Rs. {{product.marked_price}}</span>  <span
 class="sprice">Rs. {{product.selling_price}}</span> </p>
      <h5 class="font-weight-bold">Product Description</h5>
      <p>{{product.description}}</p>


      {% if request.user.is_authenticated %} {% if item_already_in_cart %}
      <a href="{% url 'showcart' %}" class="btn btn-primary px-5 py-2 pro-btn">Go To Cart</a>
      {% else %}
      <form action="/add-to-cart">
        <input type="hidden" name="prod_id" value="{{product.id}}" id="prod_id" />
        <button type="submit" class="btn btn-primary py-2 px-5 pro-btn">
          Add To Cart
        </button>
      </form>
      {% endif %} {% endif %} {% if not request.user.is_authenticated %}
      <form action="/add-to-cart d-inline">
        <input type="hidden" name="prod_id" value="{{product.id}}" id="prod_id" />
        <button type="submit" class="btn btn-primary py-2 px-5 pro-btn">
          Add To Cart
        </button>
      </form>

      {% endif %}

    </div>
  </div>
</div>

{% endblock main-content %}
```

Figure 13 Product Details Code

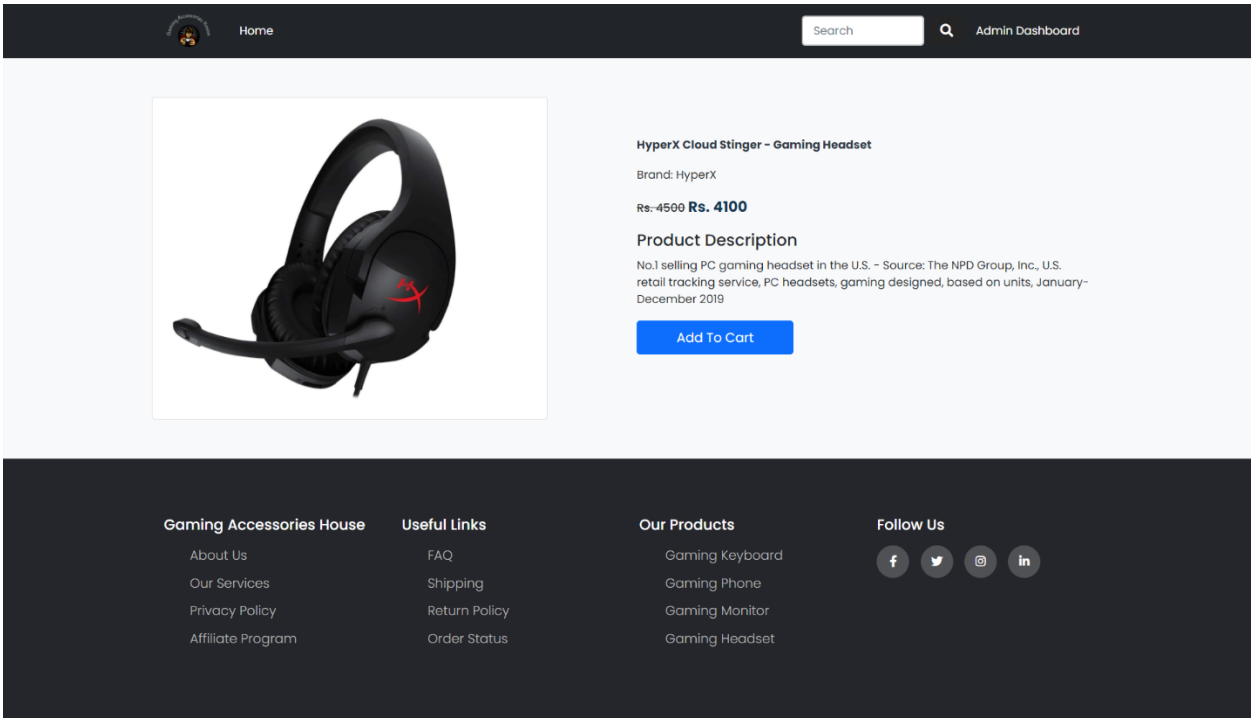

Figure 14 Product Details Output

```
{% extends 'app/base.html' %}
{% load static %}
{% load bootstrap %}
{% block title %}
<title>Update Profile</title>
{% endblock %}

{% block main-content %}
<div class="container">
    <div class="row">
        <div class="col">
            <h1>Current Details</h1>
            <div class="card">
                <div class="card-body">
                    <img src="{{request.user.profile.profile_pic.url}}" width="100" class=
"img-responsive rounded-circle">

                </div>

                <div class="card-footer">
                    <div class="py-3">Firstname : {{request.user.profile.firstname}}</div>
                    <div class="py-3">Lastname : {{request.user.profile.lastname}}</div>
                    <div class="py-3">Email : {{request.user.profile.email}}</div>
                    <div class="py-3">Phone : {{request.user.profile.phone}}</div>
                    <div class="py-3">Address : {{request.user.profile.address}}</div>
                </div>


            </div>
        </div>
        <div class="col">
            <div class="col-md-12">
                {% for message in messages %}
                {% if message.level == DEFAULT_MESSAGE_LEVELS.SUCCESS %}
                <h6 class="success_message py-3 px-1 my-2 text-center bg-success text-white">{{message}}</h6>
                {% endif %}

                {% if message.level == DEFAULT_MESSAGE_LEVELS.ERROR%}
                <h6 class="error_message py-3 px-1 my-2 text-center">{{message}}</h6>
                {% endif %}
                {% endfor %}
                <h1 class="text-center">Update Details</h1>
                <div class="card py-2 px-2">
                    <form class="form" action="" method="POST" enctype="multipart/form-data">
                        {% csrf_token %}
                        {{ form | bootstrap_horizontal}}
                        <button type="submit" class="btn btn-dark mt-2">Submit</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

Figure 15 Profile code

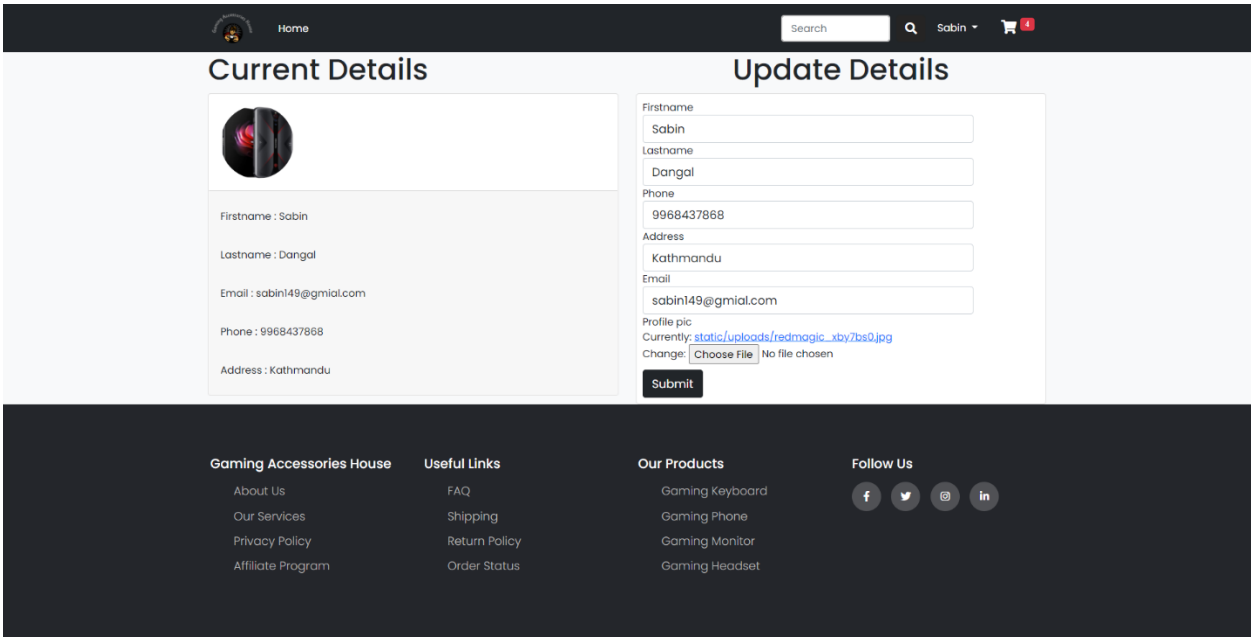

Figure 16 Profile Output

2.2.3 CSS Codes

```css
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700&display=swap');
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
body {
    font-family: 'Poppins', sans-serif;
    font-size: 14px;
}
/* navbar section */
.home {
    margin-left: 6px;
}
.main-logo {
    height: 2.5rem;
}
.cart-icon {
    font-size: 19px;
}
.search-input[type=text] {
    width: 150px;
    border: 2px solid #ccc;
    border-radius: 4px;
    font-size: 14px;
    transition: width 0.4s ease-in-out;
    margin-left: 5px;
}
.search-input[type=text]:focus {
    width: 312px;
}
.btn-se {
    border: none;
    outline: none;
    background-color: rgb(37, 36, 36);
    border-radius: 5px;
}
/* admin-dashboard */
.c-body a {
    text-decoration: none;
    color: #fff;
}
.c-body a:hover {
    color: orange;
}
.nav-dropdwn {
    margin-left: -200px;
}
/*  bannner section */
.carousel-item img {
    height: 615px !important;
}
/*products-detail section */
.pro-detail .p-title {
    color: #2d3748;
    font-size: 0.875;
    font-weight: 700;
}
.pro-detail .p-image {
    width: 80%;
    height: auto;
}
/* all products section */
.category-title {
    font-size: 1.5rem;
    font-weight: 700;
    color: #707b8e;
    text-align: center;
    margin: 0 auto 20px;
    position: relative;
    line-height: 60px;
}
.category-title:hover {
    color: black;
}
```

Figure 17 CSS Code 1

```css
.category-title::after {
    content: ' ';
    background: black;
    width: 80px;
    height: 5px;
    border-radius: 5px;
    position: absolute;
    bottom: 0;
    left: 50%;
    transform: translateX(-50%);
}
.all-products {

    width: 100%;
    margin: auto;
}
.mprice {
    color: #bdbdbd;
    font-weight: 500;
    text-decoration: line-through;
}
.sprice {
    color: #103755;
    font-size: 1.125rem;
    font-weight: 700;
}
.details .ptitle {
    color: #707b8e;
    font-size: 0.875rem;
}
.details .ptitle:hover {
    color: #103755;
    transform: scale(1);
}
.product-details {

    transition: transform .2s;
}
.all-products .product-img {
    width: 80%;

}
.product-details:hover {
    transform: translateY(-5px);

}
/* footer section */

.container {
    max-width: 1170px;
    margin: auto;
}
.row {
    display: flex;
    flex-wrap: wrap;
}
ul {
    list-style: none;
}
.footer {
    background-color: #24262b;
    padding: 70px 0;
    line-height: 1.5;
}
.footer-col {
    width: 25%;
    padding: 0 15px;
}
.footer-col h4 {
    font-size: 18px;
    color: #ffffff;
    text-transform: capitalize;
    margin-bottom: 15px;
    font-weight: 500;
    position: relative;
}
.footer-col ul li:not(:last-child) {
    margin-bottom: 10px;
}
```

Figure 18 CSS Code 2

```css
.footer-col ul li a {
    font-size: 16px;
    text-transform: capitalize;
    color: #ffffff;
    text-decoration: none;
    font-weight: 300;
    color: #bbbbbb;
    display: block;
    transition: all 0.3s ease;
}
.footer-col ul li a:hover {
    color: #ffffff;
    padding-left: 8px;
}
.footer-col .social-links a {
    display: inline-block;
    height: 40px;
    width: 40px;
    background-color: rgba(255, 255, 255, 0.2);
    margin: 0 10px 10px 0;
    text-align: center;
    line-height: 40px;
    border-radius: 50%;
    color: #ffffff;
    transition: all 0.5s ease;
}
.footer-col .social-links a:hover {
    color: #24262b;
    background-color: #ffffff;
}
/*responsive*/
@media(max-width: 767px) {
    .footer-col {
        width: 50%;
        margin-bottom: 30px;
    }
}
@media(max-width: 574px) {
    .footer-col {
        width: 100%;
    }
}
@media only screen and (max-width: 600px) {
    .pro-detail .card {
        width: 200px;
        margin-left: 10px;
    }
    .pro-btn {
        margin-bottom: 10px;
    }
    .all-products .row .card {
        width: 50%;

    }
    .all-products .row .card img {
        width: 70%;
    }
    .cp-btn {
        font-size: 12px;
        height: 50px;
    }
    .cpd-btn {
        font-size: 12px;
        height: 60px;

    }
    .gif {
        width: 400px;
    }
}
```

Figure 19 CSS Code 3

2.2.3 JavaScript Codes

```javascript
$('.plus-cart').click(function(){
    var id = $ (this).attr("pid").toString();
    var eml =  this.parentNode.children[2]
    $.ajax({
        type:'GET',
        url:'/pluscart',
        data:{
            prod_id:id
        },
        success:function(data){
            eml.innerText= data.quantity
            document.getElementById("amount").innerText=data.amount
            document.getElementById("totalamount").innerText=data.totalamount
        }
    })

})

$('.minus-cart').click(function(){
    var id = $ (this).attr("pid").toString();
    var eml =  this.parentNode.children[2]
    $.ajax({
        type:'GET',
        url:'/minuscart',
        data:{
            prod_id:id
        },
        success:function(data){
            eml.innerText= data.quantity
            document.getElementById("amount").innerText=data.amount
            document.getElementById("totalamount").innerText=data.totalamount

        }
    })

})

$('.remove-cart').click(function(){
    var id = $ (this).attr("pid").toString();
    var eml =  this
    $.ajax({
        type:'GET',
        url:'/removecart',
        data:{
            prod_id:id
        },
        success:function(data){
            document.getElementById("amount").innerText=data.amount
            document.getElementById("totalamount").innerText=data.totalamount
            eml.parentNode.parentNode.parentNode.parentNode.remove()

        }
    })

})
```

Figure 20 JavaScript Code

2.3 Importance of responsive design

Responsive design is the design that is done in the websites to make it compatible with different width screens. The key point for making the website responsive is to eliminate the need for needless resizing, floating, zooming, that happens on the webpages that are not mobile-friendly. If the website is not responsive, then users have to spend a lot of time on the website. Similarly, it will also affect the ranks of the search engines.

2.4 Responsive Check

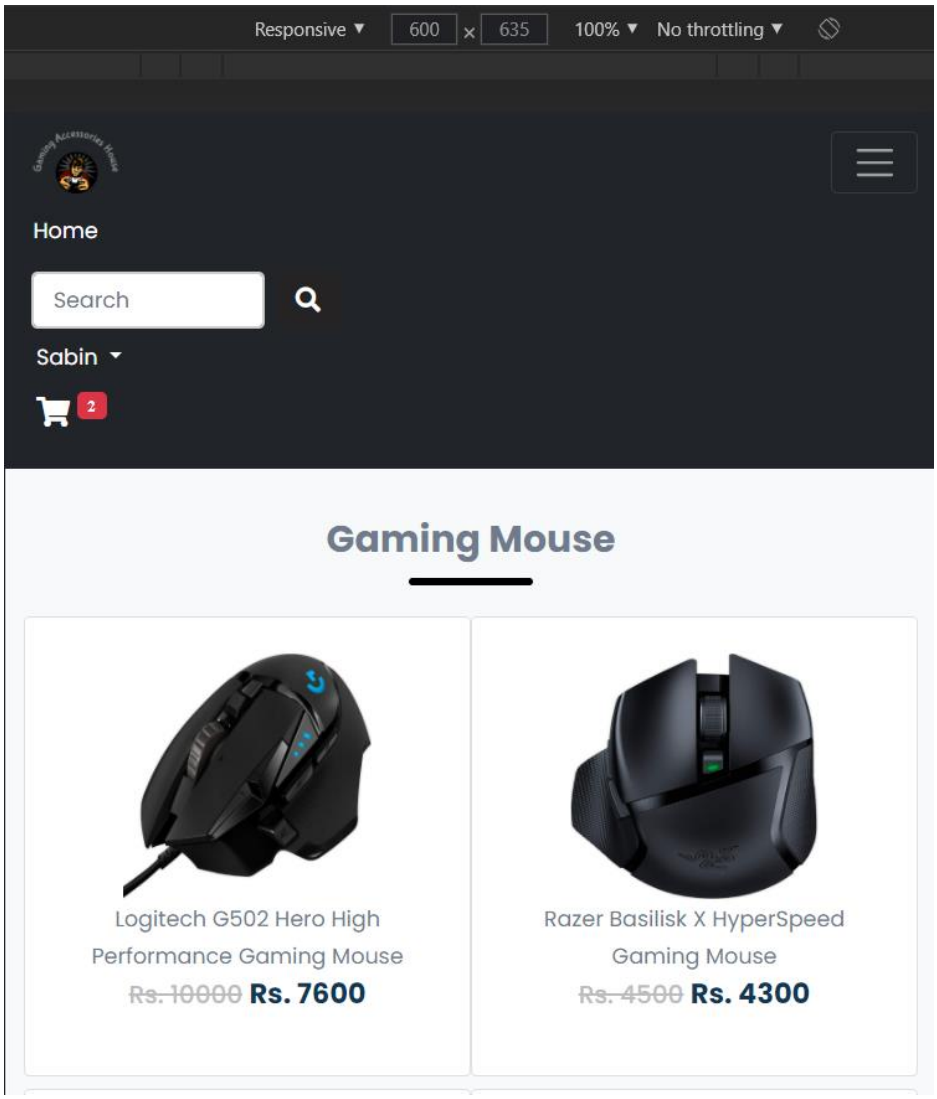

Figure 21 Responsive Check of Home Page

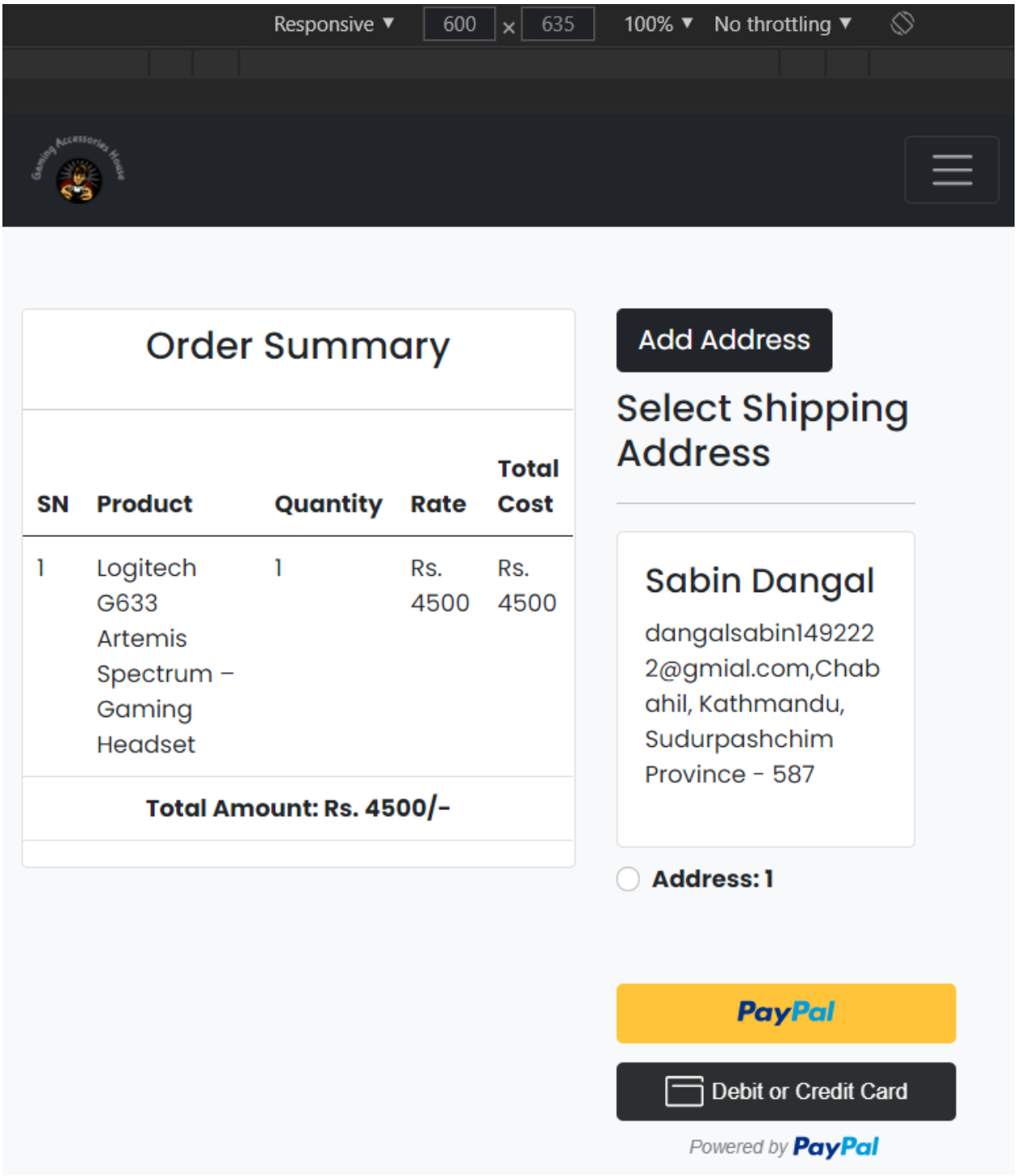

Figure 22 Responsive Check of Payment

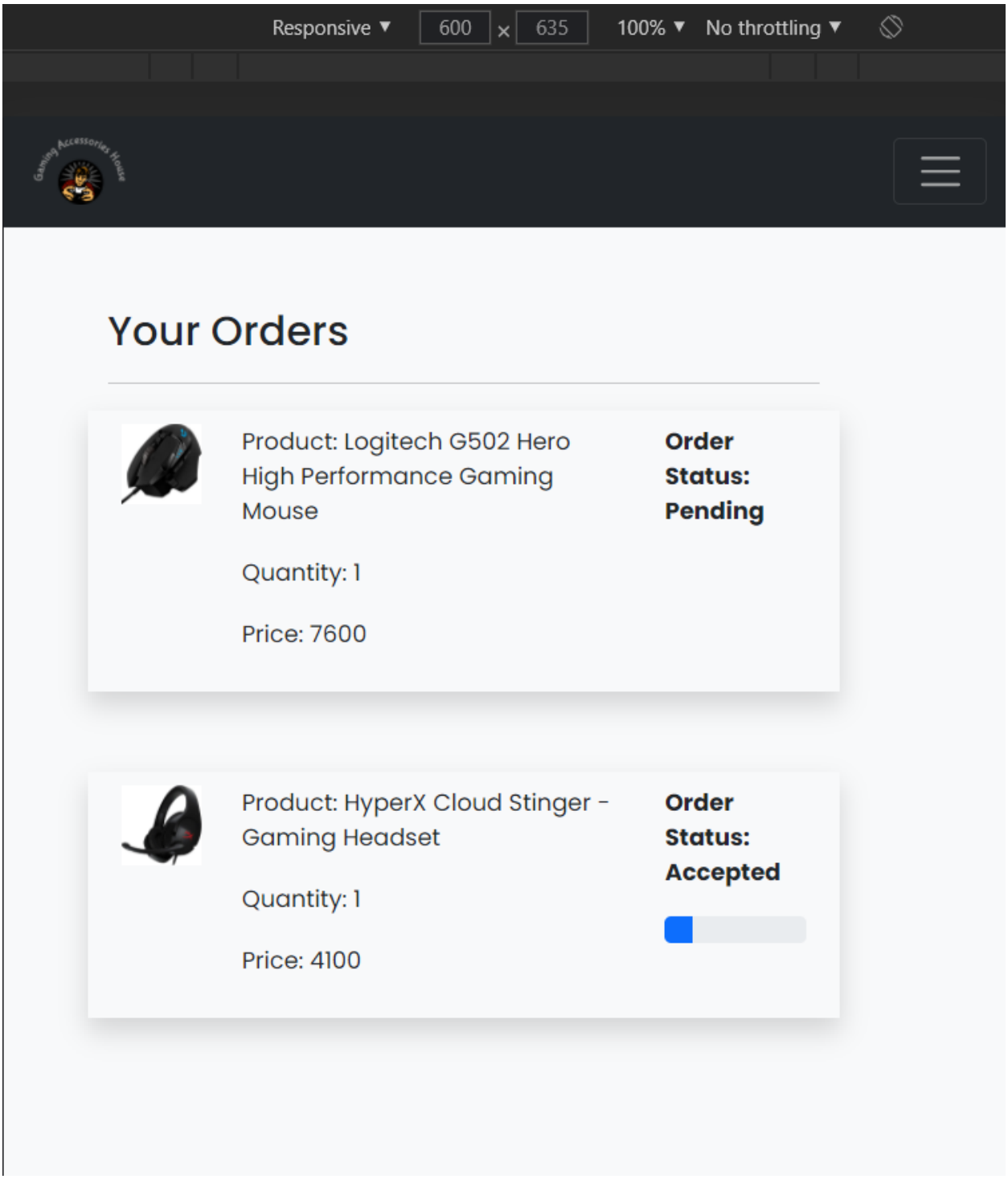

Figure 23 Responsive Check of Orders

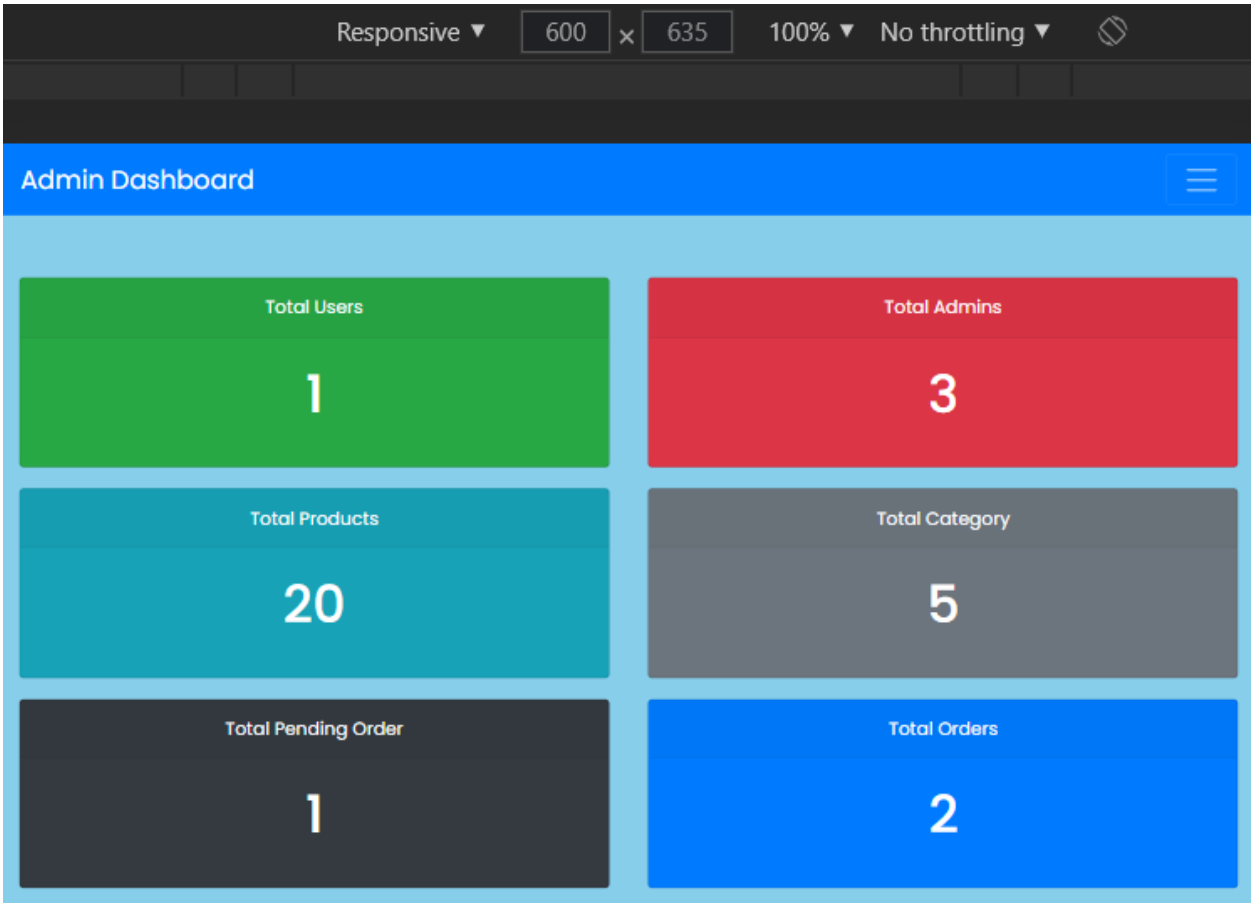

Figure 24 Responsive Check of Admin Dashboard

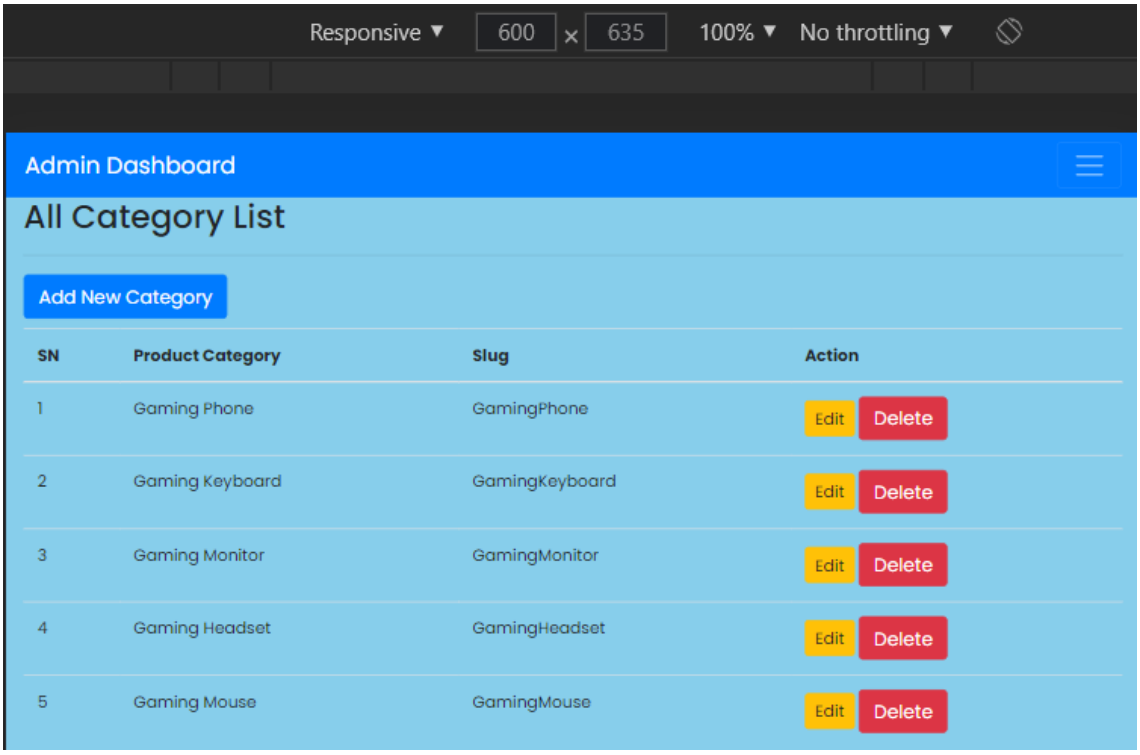

Figure 25 Responsive Check of Admin Category List

Chapter 3 Databases

3.1 Introduction

Database (Castano et al. 1995) is used to manage a large number of data so that it can be easily accessible and handle the data. In this project, I have used MySQL (MySQL 2001) database.

Advantages:

- It is a Relational Database Management System, so it saves and shows all the data in tabular form which are arranged in rows and columns.
- It is one of the secured databases which contains a data security layer.
- It contains a storage engine architecture that makes this database faster and reliable.

Disadvantages:

- It cannot be used for handling a large number of data.
- Like other paid databases, it does not contain debugging or developing tools.
- There is a chance that data will be corrupt while using it.

3.3 Screenshots

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'gaming_accessories_house',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'"
        }

    }
}
```

Figure 26 Database Configuration Code

```python
class Profile(models.Model):
    user = models.OneToOneField(User,  on_delete=models.CASCADE)
    firstname = models.CharField(max_length=200)
    lastname = models.CharField(max_length=200)
    phone = models.CharField(max_length=10)
    address = models.CharField(max_length=100)
    email = models.EmailField()
    username = models.CharField(max_length=200)
    profile_pic = models.FileField(
        upload_to='static/uploads', default='static/uploads/rog.png')
    created_date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return str(self.id)+" "+str(self.phone)+" "+self.address+" "+self.email+" "+str(self.profile_pic)

PROVINCE_CHOICES = (
    ('Province No. 1', 'Province No. 1'),
    ('Province No. 2', 'Province No. 2'),
    ('Bagmati Province', 'Bagmati Province'),
    ('Gandaki Province', 'Gandaki Province'),
    ('Lumbini Province', 'Lumbini Province'),
    ('Karnali Province', 'Karnali Province'),
    ('Sudurpashchim Province', 'Sudurpashchim Province')
)

class Category_choices(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField(unique=True)

    def __str__(self):
        return self.title

class Product(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField(unique=True)
    marked_price = models.IntegerField()
    selling_price = models.IntegerField()
    brand = models.CharField(max_length=100)
    description = models.TextField()
    category = models.ForeignKey(Category_choices, on_delete=models.CASCADE)
    product_image = models.ImageField(upload_to='productimg')

    def __str__(self):
        return str(self.id)+" "+self.title+" "+self.slug+" "+str(self.marked_price)+" "+str(self.selling_price)+
" "+(self.brand)+" " + self.description + str(self.category)+" "+str(self.product_image)
```

Figure 27 Models Code

```python
class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)

    def __str__(self):
        return str(self.id)

    @property
    def total_cost(self):
        return self.quantity * self.product.selling_price

STATUS_CHOICES = (
    ('Accepted', 'Accepted'),
    ('Packed', 'Packed'),
    ('On The Way', 'On The Way'),
    ('Delievered', 'Delievered'),
    ('Cancel', 'Cancel')
)

class Customer(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    email = models.EmailField()
    name = models.CharField(max_length=200)
    address = models.CharField(max_length=200)
    city = models.CharField(max_length=200)
    province = models.CharField(choices=PROVINCE_CHOICES, max_length=50)
    zipcode = models.IntegerField()
    def __str__(self):
        return str(self.id)+" "+self.email+" "+self.name+" "+self.address+" "+self.city+" "+self.province+" "+str(self.zipcode)


class OrderPlaced(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE, null=True)
    profile = models.ForeignKey(Profile, on_delete=models.CASCADE, null=True)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    ordered_date = models.DateTimeField(auto_now_add=True)
    status = models.CharField(max_length=50, choices=STATUS_CHOICES, default='Pending')
    def __str__(self):
        return "Order: " + str(self.id)
    @property
    def total_cost(self):
        return self.quantity * self.product.selling_price
```

Figure 28 Models Code

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	app_cart	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	app_category_choices	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	app_customer	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	app_orderplaced	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	80.0 KiB	-
☐	app_product	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	app_profile	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	auth_group	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	auth_group_permissions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	auth_permission	Browse Structure Search Insert Empty Drop	52	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	auth_user	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	32.0 KiB	-
☐	auth_user_groups	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	auth_user_user_permissions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	django_admin_log	Browse Structure Search Insert Empty Drop	67	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	django_content_type	Browse Structure Search Insert Empty Drop	13	InnoDB	utf8mb4_general_ci	48.0 KiB	-
☐	django_migrations	Browse Structure Search Insert Empty Drop	60	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	django_session	Browse Structure Search Insert Empty Drop	15	InnoDB	utf8mb4_general_ci	32.0 KiB	-
	16 tables	**Sum**	**242**	**InnoDB**	**utf8mb4_general_ci**	**656.0 KiB**	**0 B**

Figure 29 Database Tables

Chapter 4 Backend

4.1 Technologies:

Technologies that are used in the backend for creating this website are explained below:

Python:

Python (Van Rossum and Drake Jr 1995) is a high-level programming language created by Guido van Rossum in1991. It is mainly used for web development, mathematic, software development, and system scripting. It is a widely-used programming language as its syntax is easy to understand and it can be run into different platforms.

Django:

Django (Holovaty and Kaplan-Moss 2009) is a python open-source web framework which is consists of different components to develop websites much easier and faster. Currently, Django is used by big companies like Instagram, Spotify, Mozilla, etc. due to its various features like open-source, scalable, fully loaded, etc.

4.2 Screenshots

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app', 'bootstrapform', 'admins',
]
```

Figure 30 Installed Apps Code

```python
urlpatterns = [
    path('', include('app.urls')),
    path('admin-dashboard/',include('admins.urls')),
]
```

Figure 31 Main URLs Code

```python
class HomeView(TemplateView):
    def get(self,request):
        totalitem = 0
        if self.request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
        template_name = "app/home.html"
        allcategories=Category_choices.objects.all()
        context={'allcategories':allcategories,'totalitem':totalitem}
        return render(request,template_name,context)


class ProductDetailView(TemplateView):
    def get(self, request, **kwargs):
        url_slug = kwargs['slug']
        product = Product.objects.get(slug=url_slug)
        item_already_in_cart = False
        totalitem = 0
        if self.request.user.is_authenticated:
            totalitem = len(Cart.objects.filter(user=request.user))
            item_already_in_cart = Cart.objects.filter(
                Q(product=product.id) & Q(user=request.user)).exists()
        return render(request, 'app/productdetail.html', {'product': product, 'item_already_in_cart'
: item_already_in_cart, 'totalitem': totalitem})
```

Figure 32 Views Function Code

```python
def unauthenticated_user(view_function):
    def wrapper_function(request, *args, **kwargs):
        if request.user.is_authenticated:
            return redirect('/')
        else:
            return view_function(request, *args, **kwargs)
    return wrapper_function


def user_only(view_function):
    def wrapper_function(request, *args, **kwargs):
        if request.user.is_staff:
            return redirect('/admin-dashboard')
        else:
            return view_function(request, *args, **kwargs)
    return wrapper_function


def admin_only(view_function):
    def wrapper_function(request, *args, **kwargs):
        if request.user.is_staff:
            return view_function(request, *args, **kwargs)
        else:
            return redirect('/')
    return wrapper_function
```

Figure 33 Decorators Code

```python

@unauthenticated_user
def login_user(request):
    if request.user.is_authenticated:
        return render(request, 'app/home.html')
    else:
        if request.method == 'POST':
            form = LoginForm(request.POST)
            if form.is_valid():
                data = form.cleaned_data
                user = authenticate(request, username=data['username'],
                                    password=data['password'])
                if user is not None:
                    if not user.is_staff:
                        login(request, user)
                        return redirect('all-products')
                    elif user.is_staff:
                        login(request, user)
                        return redirect('/admin-dashboard')
                else:
                    messages.add_message(request, messages.ERROR,
                                         'Username or Password is Invalid')
                    return render(request, 'app/login.html', {'form': form})
    form = LoginForm()
    context = {
        'form': LoginForm
    }
    return render(request, 'app/login.html', context)


@unauthenticated_user
def CustomerRegistrationView(request):
    if request.method == 'POST':
        form = CustomerRegistrationForm(request.POST)
        if form.is_valid():
            user=form.save()
            Profile.objects.create(user=user,username=user.username)
            messages.add_message(request, messages.SUCCESS, 'User Registered Successfully')
            return redirect('/login')
        else:
            messages.add_message(request, messages.ERROR, 'Please Provide Correct Details')
            return render(request, "app/customerregistration.html", {'form': form})
    return render(request, 'app/customerregistration.html', {'form': CustomerRegistrationForm})
```

Figure 34 Views Function Code

```python
@user_only
@login_required
def shippingaddress(request):
    totalitem = 0
    if request.user.is_authenticated:
        totalitem = len(Cart.objects.filter(user=request.user))
    if request.method == 'POST':
        fm = CustomerAddressForm(request.POST)
        if fm.is_valid():
            usr = request.user
            name = fm.cleaned_data['name']
            email = fm.cleaned_data['email']
            address = fm.cleaned_data['address']
            city = fm.cleaned_data['city']
            province = fm.cleaned_data['province']
            zipcode = fm.cleaned_data['zipcode']
            reg = Customer(user=usr, name=name,email=email, address=address,
                           city=city, province=province, zipcode=zipcode)
            reg.save()
            fm = CustomerAddressForm()
    else:
        fm = CustomerAddressForm()

    stud = Customer.objects.filter(user=request.user)
    context={'form': fm, 'stu': stud,'totalitem':totalitem}
    return render(request, 'app/shippingaddress.html',context)

@user_only
@login_required
def delete_address(request,id):
    if request.method=='POST':
        pi=Customer.objects.get(pk=id)
        pi.delete()
        return HttpResponseRedirect('/shippingaddress')

@method_decorator(user_only,name='dispatch')
@method_decorator(login_required,name='dispatch')
class update_address(View):
    def get(self, request, id):
        pi = Customer.objects.get(pk=id)
        fm = CustomerAddressForm(instance=pi)
        return render(request, 'app/updateaddress.html', {'form': fm})

    def post(self, request, id):
        pi = Customer.objects.get(pk=id)
        fm = CustomerAddressForm(request.POST, instance=pi)
        if fm.is_valid():
            fm.save()
        return HttpResponseRedirect('/shippingaddress')
```

Figure 35 Views Function Code

```python
@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class AdminCategoryListView(ListView):
    template_name = "admins/admincategorylist.html"
    queryset = Category_choices.objects.all().order_by("-id")
    context_object_name = "allcategory"


@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class AdminCategoryCreateView(CreateView):
    template_name = "admins/admincategorycreate.html"
    form_class = CategoryForm
    success_url = reverse_lazy("admincategorylist")

    def form_valid(self, form):
        form.save()
        return super().form_valid(form)


@admin_only
@login_required
def delete_category(request, id):
    if request.method == 'POST':
        pi = Category_choices.objects.get(pk=id)
        pi.delete()
        return HttpResponseRedirect('/admin-dashboard/admin-category/list')


@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class update_category(View):
    def get(self, request, id):
        pi = Category_choices.objects.get(pk=id)
        fm = CategoryForm(instance=pi)
        return render(request, 'admins/adminupdatecategory.html', {'form': fm})

    def post(self, request, id):
        pi = Category_choices.objects.get(pk=id)
        fm = CategoryForm(request.POST, instance=pi)
        if fm.is_valid():
            fm.save()
        return HttpResponseRedirect('/admin-dashboard/admin-category/list')
```

Figure 36 Views Function Code

```python
@admin_only
@login_required
def AdminProductListView(request):
    template_name = "admins/adminproductlist.html"
    allproducts = Product.objects.all().order_by("-id")
    context = {
        'allproducts': allproducts
    }
    return render(request, template_name, context)


@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class AdminProductCreateView(CreateView):
    template_name = "admins/adminproductcreate.html"
    form_class = ProductForm
    success_url = reverse_lazy("adminproductlist")

    def form_valid(self, form):
        form.save()
        return super().form_valid(form)


@admin_only
@login_required
def delete_product(request, id):
    if request.method == 'POST':
        pi = Product.objects.get(pk=id)
        pi.delete()
        return HttpResponseRedirect('/admin-dashboard/admin-product/list')


@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class update_product(View):
    def get(self, request, id):
        pi = Product.objects.get(pk=id)
        fm = ProductForm(instance=pi)
        return render(request, 'admins/adminupdateproduct.html', {'form': fm})

    def post(self, request, id):
        pi = Product.objects.get(pk=id)
        fm = ProductForm(request.POST, instance=pi)
        if fm.is_valid():
            fm.save()
        return HttpResponseRedirect('/admin-dashboard/admin-product/list')
```

Figure 37 Admin Views Function Code

```python
@admin_only
@login_required
def AdminOrderListView(request):
    template_name = "admins/adminorderlist.html"
    allorders = OrderPlaced.objects.all().order_by("-id")
    context = {
        'allorders': allorders,
    }
    return render(request, template_name, context)


@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class AdminOrderDetailView(DetailView):
    template_name = "admins/adminorderdetail.html"
    model = OrderPlaced
    context_object_name = "ord_obj"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context["allstatus"] = STATUS_CHOICES
        return context


@admin_only
@login_required
def delete_order(request, id):
    if request.method == 'POST':
        pi = OrderPlaced.objects.get(pk=id)
        pi.delete()
        return HttpResponseRedirect('/admin-dashboard/admin-all-orders')


@method_decorator(login_required, name='dispatch')
@method_decorator(admin_only, name='dispatch')
class AdminOrderStatuChangeView(View):
    def post(self, request, *args, **kwargs):
        order_id = self.kwargs["pk"]
        order_obj = OrderPlaced.objects.get(id=order_id)

        new_status = request.POST.get("status")
        order_obj.status = new_status
        order_obj.save()
        return redirect(reverse_lazy("adminorderdetail", kwargs={"pk": order_id}))

```

Figure 38 Admin Views Function Code

```
urlpatterns = [
    path('', views.admin_dashboard, name='dashboard'),
    path('logout/', auth_views.LogoutView.as_view(next_page='login'), name='logout'),
    path('show-user', views.get_user, name='show-user'),
    path('show-admin', views.get_admin, name='show-admin'),
    path('update-user-to-admin/<int:user_id>', views.update_user_to_admin),
    path('register-user', views.register_user_admin),
    path("admin-product/list", views.AdminProductListView,
         name="adminproductlist"),
    path("admin-all-orders/", views.AdminOrderListView,
         name="adminorderlist"),
    path("admin-product/add/", views.AdminProductCreateView.as_view(),
         name="adminproductcreate"),
    path('deleteproduct/<int:id>/', views.delete_product, name='deleteproduct'),
    path('updateproduct<int:id>/',
         views.update_product.as_view(), name="updateproduct"),
    path("admin-category/list", views.AdminCategoryListView.as_view(),
         name="admincategorylist"),
    path("admin-category/add/", views.AdminCategoryCreateView.as_view(),
         name="admincategorycreate"),
    path('deletecategory/<int:id>/', views.delete_category, name='deletecategory'),
    path('updatecategory<int:id>/',
         views.update_category.as_view(), name="updatecategory"),
    path("admin-order/<int:pk>/", views.AdminOrderDetailView.as_view(),
         name="adminorderdetail"),
    path("admin-order-<int:pk>-change/",
         views.AdminOrderStatuChangeView.as_view(), name="adminorderstatuschange"),
    path("admin-pendingorder/", views.AdminHomeView.as_view(),
         name="adminpendingorder"),
    path('deleteorder/<int:id>/', views.delete_order, name='deleteorder'),
    path('deleteuser/<int:id>/', views.delete_user, name='deleteuser'),

]
```

Figure 39 Admins App URLs Code

```
urlpatterns = [
    path('', views.HomeView.as_view(), name='all-products'),
    path('product/<slug:slug>/', ProductDetailView.as_view(), name='productdetail'),
    path('add-to-cart/', views.add_to_cart, name='add-to-cart'),
    path('cart/', views.show_cart, name='showcart'),
    path('pluscart/', views.plus_cart, name='pluscart'),
    path('minuscart/', views.minus_cart, name='minuscart'),
    path('removecart/', views.remove_cart, name='removecart'),
    path('registration/', views.CustomerRegistrationView,
         name='customerregistration'),
    path('login/', views.login_user, name='login'),
    path('userprofile/', views.user_profile, name='userprofile'),
    path('shippingaddress/', views.shippingaddress, name='shippingaddress'),
    path('deleteaddress/<int:id>/', views.delete_address, name='deleteaddress'),
    path('updateaddress<int:id>/',views.update_address.as_view(),name="updateaddress"),
    path('checkout/', views.checkout, name='checkout'),
    path('orders/', views.orders, name='orders'),
    path('paymentdone/', views.payment_done, name='paymentdone'),
    path("search/", views.SearchView.as_view(), name="search"),
    path('passwordchange/', auth_views.PasswordChangeView.as_view(
        template_name='app/passwordchange.html', form_class=UserPasswordChangeForm, success_url=
'/passwordchangedone/'), name='passwordchange'),
    path('passwordchangedone/', auth_views.PasswordChangeView.as_view(
        template_name='app/passwordchangedone.html'), name='passwordchangedone'),
    path('password-reset/', auth_views.PasswordResetView.as_view(template_name='app/password_reset.html',
                                                                 form_class=UserPasswordResetForm), name=
'password_reset'),
    path('password-reset/done/', auth_views.PasswordResetDoneView.as_view(
        template_name='app/password_reset_done.html'), name='password_reset_done'),
    path('password-reset-confirm/<uidb64>/<token>/', auth_views.PasswordResetConfirmView.as_view(
        template_name='app/password_reset_confirm.html', form_class=UserSetPasswordForm), name=
'password_reset_confirm'),
    path('password-reset-complete/', auth_views.PasswordResetCompleteView.as_view(
        template_name='app/password_reset_complete.html'), name='password_reset_complete'),
    path('logout/', auth_views.LogoutView.as_view(next_page='login'), name='logout'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Figure 40 Main App URLs Code

```python
class ProductForm(forms.ModelForm):
    class Meta:
        model = Product
        fields = ["title", "slug", "category", "product_image","marked_price",
                  "selling_price", "description", "brand"]
        widgets = {
            "title": forms.TextInput(attrs={
                "class": "form-control",
                "placeholder": "Enter the product title here..."
            }),
            "slug": forms.TextInput(attrs={
                "class": "form-control",
                "placeholder": "Enter the unique slug here..."
            }),
            "category": forms.Select(attrs={
                "class": "form-control"
            }),

            "product_image": forms.ClearableFileInput(attrs={
                "class": "form-control"
            }),
            "marked_price": forms.NumberInput(attrs={
                "class": "form-control",
                "required":False,
                "placeholder": "Marked price of the product..."
            }),
            "selling_price": forms.NumberInput(attrs={
                "class": "form-control",
                "placeholder": "Selling price of the product..."
            }),
            "description": forms.Textarea(attrs={
                "class": "form-control",
                "placeholder": "Description of the product...",
                "rows": 5
            }),
            "brand": forms.TextInput(attrs={
                "class": "form-control",
                "placeholder": "Enter the product brand here..."
            })
        }



class CategoryForm(forms.ModelForm):
    class Meta:
        model = Category_choices
        fields = ["title", "slug"]
        widgets = {
            "title": forms.TextInput(attrs={
                "class": "form-control",
                "placeholder": "Enter the unique product here..."
            }),
            "slug": forms.TextInput(attrs={
                "class": "form-control",
                "placeholder": "Enter the unique slug here..."
            }),
        }
```

Figure 41 Admin App Form Code

```python
class CustomerRegistrationForm(UserCreationForm):
    password1 = forms.CharField(label='Password', widget=forms.PasswordInput(attrs={'class': 'form-control'}))
    password2 = forms.CharField(label='Confirm Password (again)',
                                widget=forms.PasswordInput(attrs={'class': 'form-control'}))
    email = forms.CharField(required=True, widget=forms.EmailInput(
        attrs={'class': 'form-control'}))

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
        labels = {'email': 'Email'}
        widgets = {'username': forms.TextInput(attrs={'class': 'form-control'})}

class LoginForm(forms.Form):
    username = UsernameField(widget=forms.TextInput(
        attrs={'autofocus': True, 'onfocus': "this.value=''", 'class': 'form-control', 'placeholder': 'Enter Your Username'}))
    password = forms.CharField(label=_("Password"), strip=False, widget=forms.PasswordInput(
        attrs={'autocomplete': 'current-password', 'onfocus': "this.value=''", 'class': 'form-control', 'placeholder': 'Enter Your Password'}))

class CustomerAddressForm(ModelForm):
    class Meta:
        model = Customer
        fields = ['name', 'email', 'address', 'city', 'province', 'zipcode']
        widgets = {'name': forms.TextInput(attrs={'class': 'form-control'}),"email":forms.EmailInput(
            attrs={'class': 'form-control'}), 'address': forms.TextInput(attrs={'class': 'form-control'}), 'city': forms.TextInput(
            attrs={'class': 'form-control'}), 'province': forms.Select(attrs={'class': 'form-control'}), 'zipcode': forms.NumberInput(attrs={'class': 'form-control'})}

class UserPasswordChangeForm(PasswordChangeForm):
    old_password = forms.CharField(label=_("Old Password"), strip=False, widget=forms.PasswordInput(
        attrs={'autocomplete': 'current-password', 'autofocus': True, 'class': 'form-control'}))
    new_password1 = forms.CharField(label=_("New Password"), strip=False, widget=forms.PasswordInput(
        attrs={'autocomplete': 'new-password', 'class': 'form-control'}), help_text=password_validation.password_validators_help_text_html())
    new_password2 = forms.CharField(label=_("Confirm New Password"), strip=False, widget=forms.PasswordInput(
        attrs={'autocomplete': 'new-password', 'class': 'form-control'}))

class UserPasswordResetForm(PasswordResetForm):
    email = forms.EmailField(label=('Email'), max_length=254, widget=forms.EmailInput(
        attrs={'autocomplete': 'email', 'class': 'form-control'}))

class UserSetPasswordForm(SetPasswordForm):
    new_password1 = forms.CharField(label=_("New password"), strip=False, widget=forms.PasswordInput(
        attrs={'autocomplete': 'new-password', 'class': 'form-control'}), help_text=password_validation.password_validators_help_text_html())
    new_password2 = forms.CharField(label=_("Confirm New Password"), strip=False, widget=forms.PasswordInput(
        attrs={'autocomplete': 'new-password', 'class': 'form-control'}))

class ProfileForm(ModelForm):
    class Meta:
        model = Profile
        fields = '__all__'
        exclude = ['user', 'username']
```

Figure 42 Main App Form Code

Chapter 5 Project Issues

5.1 Issues During the Project Development

I got many problems while doing this project. While doing the plus, minus, and remove function, ajax was not working. So, I had to see all the code repeatedly. Similarly, when I was working with the admin dashboard, the order status change view was not working so I had to take help from others.

5.2 Limitations of the Project

Some of the limitations of this project are listed below:

- **Customer Support:** This website does not contain this feature in which users can ask different questions or give feedback related to the website.
- **Wishlist feature:** Customer cannot add products to the wish list sections instead of that they must add products to the cart.
- **Static Footer:** Most of these links in the footer section is static. Links-related accounts, templates are not made.
- **Payment Options:** On this website, there is only one payment option i.e. PayPal, other payment options like master cards, credit cards, debit cards, etc. are not included.
- **Other Seller:** Like other e-commerce websites, it does not feature in which other sellers can sell their products using this website.

5.3 Future Works

There are a lot of things that can be done to improve this website. Some of them are mentioned below:

- **More Payment Options:** Many other payment options can be added so that customers can choose different payment methods.
- **Customer Support:** Customer Support can be added to it so that they can give reviews, feedbacks on the website.
- **Wishlist Feature:** This feature can also be added so that customers can add products to the wish list.

Conclusion

After much consideration, I've come to the realization that e-commerce has unquestionably been an integral aspect of our culture. The internet plays and will continue to play an important role in our everyday lives. From this module, I learned How we can make website using python web framework Django. Moreover, in this semester I have done ecommerce project after learning Django.

Reference

Castano, S., Fugini, M., Martella, G., and Samarati, P. (1995) *Database Securi* [online]: Addison—Wesley

Crockford, D. (2008) *JavaScript: The Good Parts: The Good Parts* [online]: " O'Reilly Media, Inc."

Duckett, J. (2011) *HTML & CSS: Design and Build Websites* [online]: Wiley Indianapolis, IN

Holovaty, A. and Kaplan-Moss, J. (2009) *The Definitive Guide to Django: Web Development done Right* [online]: Apress

Lubbers, P., Albers, B., Salim, F., and Pye, T. (2011) *Pro HTML5 Programming* [online]: Springer

MySQL, A. B. (2001) 'No Title'*. MySQL*

Spurlock, J. (2013) *Bootstrap: Responsive Web Development* [online]: " O'Reilly Media, Inc."

Van Rossum, G. and Drake Jr, F. L. (1995) *Python Tutorial* [online]: Centrum voor Wiskunde en Informatica Amsterdam

Django (2021) available from <https://www.youtube.com/playlist?list=PLbGui_ZYuhigchy8DTw4pX4duTTpvqlh6> [17 April 2021]