# INTRODUCTION

## INTRODUCTION

This project Furniture Management System, developed using Python and Django, is a comprehensive solution designed to streamline the operations of furniture stores. This project aims to provide an efficient platform for furniture store owners and administrators to manage their inventory, sales, and customer interactions seamlessly.

With features such as a user-friendly interface, product catalog management, real-time inventory tracking, order processing, secure user authentication, sales analytics, payment integration, and customer relationship management, Shop Cart empowers businesses to modernize their operations and enhance the customer shopping experience.

Its responsive design ensures accessibility on various devices, and robust security measures safeguard sensitive data. Shop Cart promises to revolutionize how furniture businesses manage their operations, increasing efficiency and customer satisfaction.

### 1.1 PROBLEM DEFINITION

The Furniture Management System aims to streamline and optimize the process of managing furniture inventory, sales, and customer orders using Python and Django.

This system will enable furniture store owners to efficiently track and update their inventory, process customer orders, manage supplier information, and monitor sales transactions.

Through an intuitive web interface, users will have real-time access to inventory data, enabling them to make informed decisions, minimize stockouts, and enhance customer satisfaction.

By automating these critical functions, the system seeks to reduce manual errors, improve operational efficiency, and ultimately contribute to the growth and success of furniture businesses.

**SYSTEM SPECIFICATION**

**SYSTEM SPECIFICATIONS**

**2.1 HARDWARE SPECIFICATION**

This section gives the details and specification of the hardware on which the system is developed.

Processor : Intel Core i5 11th Generation

RAM : 8 GB DDR4

Monitor : 19.5 LED Monitor

SSD : 512 GB

Monitor : Wide Screen

## 2.2 SOFTWARE SPECIFICATION

## This section gives the details of the software that are used for the development.

Front-End : HTML, CSS

Coding Language : PHP

Back-End : My-SQL

Operating System : Windows 11

# PROJECT DESCRIPTION

## PROJECT DESCRIPTION

### 3.1 INTRODUCTION

The Furniture Management System is designed to streamline the operations of a furniture store by providing a comprehensive and efficient platform built using Python and Django.

This system aims to address several critical challenges faced by furniture retailers, including inventory management, order processing, and customer interactions.

With features such as real-time inventory tracking, automated order processing, and customer relationship management, it empowers businesses to maintain optimal stock levels, ensure timely deliveries, and enhance customer satisfaction. Additionally, it offers detailed sales analytics, enabling data-driven decisions to optimize product offerings and pricing strategies.

By leveraging the power of Python and Django, this system promises a scalable, secure, and user-friendly solution that can transform furniture retail operations, reduce manual workload, and elevate the overall shopping experience for both customers and administrators.

## 3.2 DATA FLOW DIGRAM

Creating a Data Flow Diagram (DFD) for a furniture management system using Python and Django would typically involve various components and data flows. Below is a simplified DFD that outlines the primary processes and data interactions in such a system:

**Data flow diagram level 0**

Furniture management

system

Successfully login

admin

**Data flow diagram level 1**

admin

Login

Admin

Manage furniture

manage system

Change password

Admin

Admin Admin

## 3.3 DATABASE DESIGN

### 3.3.1 DATA BASE DESIGN FOR USER TABLE

Table name : User table

Table description : This table is used to maintain the user data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default |
| Id | Int | NO | PRI | NULL |
| Password | varchar (128) | NO |  | NULL |
| last\_login | datetime(6) | YES |  | NULL |
| is\_superuser | tinyint(1) | NO |  | NULL |
| Username | varchar (150) | NO | UNI | NULL |
| First\_name | varchar (150) | NO |  | NULL |
| Last\_name | varchar (150) | NO |  | NULL |
| Email | varchar (50) | NO |  | NULL |
| is\_staff | tinyint(1) | NO |  | NULL |
| is\_active | tinyint(1) | NO |  | NULL |
| Date\_joined | datetimr(6) | NO |  | NULL |

### 

**3.3.2 DATA BASE DESIGN FOR CATEGORY**

Table name  **:**  Category

Table description **:** This table is used to maintain the category of the product details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default |
| Id | bigint | NO | PRI | NULL |
| Name | varchar (50) | NO |  | NULL |

**3.3.3 DATABASE DESIGN FOR CUSTOMER**

Table name : User table

Table description : This table is used to maintain the customer details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default |
| Id | bigint | NO | PRI | NULL |
| First\_name | varchar (50) | NO |  | NULL |
| Last\_name | varchar (50) | NO |  | NULL |
| Phone | varchar(10) | NO |  | NULL |
| Email | varchar (50) | NO |  | NULL |
| Password | varchar (20) | NO |  | NULL |

**3.3.4 DATABASE DESIGN FOR ORDER TABLE**

Table name : User table

Table description : This table is used to maintain the order details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default |
| Id | bigint | NO | PRI | NULL |
| quantity | Int | NO |  | NULL |
| Price | datetime(6) | NO |  | NULL |
| address | varchar(50) | NO |  | NULL |
| Phone | varchar (10) | NO |  | NULL |
| Date | date | NO |  | NULL |
| Status | tinyint(1) | NO |  | NULL |
| Customer\_id | bigint | NO | MUL | NULL |
| Product\_id | bigint | NO | MUL | NULL |

**3.3.5 DATABASE DESIGN FOR PRODUCT**

Table name : User table

Table description : This table is used to maintain the product details.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** |
| Id | Bigint | NO | PRI | NULL |
| name | varchar(50) | NO |  | NULL |
| Price | Int | NO |  | NULL |
| Description | varchar(50) | YES |  | NULL |
| Image | varchar (50) | NO |  | NULL |
| Category\_id | Bigint | NO | MUL | NULL |

## 3.4 MODULE DESCRIPTION

### 3.4.1 USER MODULE

Creating a user module for a furniture management system is essential for facilitating user interactions, purchases, and account management. Below is a module description outlining the key features and functionalities of the user module:

**1. User Registration:**

- New users can create accounts by providing necessary information, such as name, email address, password, and contact details.

- Email verification or captcha for security.

**2. User Login:**

- Registered users can log in using their email/username and password.

- Implement secure authentication mechanisms like password hashing.

**3. User Profile Management:**

- Users can view and edit their profile information, including contact details and shipping address.

- Upload and manage profile pictures or avatars.

**5. Product Details:**

- Users can view detailed product information, including specifications, reviews, and availability.

- Add products to the shopping cart.

**6. Shopping Cart**:

- Users can add, remove, or update items in their shopping cart.

- Calculate and display the total price.

- Proceed to checkout from the cart.

**7. Checkout and Payment:**

- Users can enter shipping information and select payment methods.

- Implement secure payment processing through various payment gateways.

- Provide order summary and confirmation.

**8. Wishlist and Favorites:**

- Users can create and manage wishlists of products they intend to purchase.

- Save favorite products for future reference.

**9. Support and Help Center:**

- Provide resources for users to seek assistance or report issues.

- Offer a contact form or chat support for inquiries.

**10. Review and Rating:**

- Allow users to leave reviews and ratings for products.

- Display average product ratings and customer reviews.

**11. Logout Functionality:**

- Users can securely log out of their accounts.

## 3.4.2 ADMIN MODULE

Creating an admin module for a furniture management system is crucial for managing the system's backend, overseeing operations, and ensuring everything runs smoothly. Below is a description of the key features and functionalities you might include in the admin module:

**1. Admin Authentication and Authorization:**

- Admin login: Secure authentication for authorized administrators.

- Role-based access control: Assign different admin roles (e.g., super admin, inventory manager) with specific permissions.

1. **User Management:**

- Add, edit, or delete user accounts (employees, customers, other admins).

- Assign roles and permissions to users.

- Monitor user activity and login history.

**3. Product Management:**

- Add, update, or delete furniture products.

- Manage product categories and attributes.

- Monitor product availability and stock levels.

**4. Inventory Management:**

- Track real-time inventory status.90

- Receive alerts for low stock levels.

- Initiate restocking requests.

- Handle product returns and adjustments.

**5. Order Management:**

- View and manage all customer orders.

- Update order status (e.g., confirm, ship, cancel).

- Process refunds and returns.

- Generate invoices and packing slips.

**6. Customer Management:**

- Access and modify customer profiles.

- View customer purchase history.

- Manage customer inquiries and issues.

**7. Sales and Financial Management:**

- Generate sales reports and analytics.

- Track revenue, profits, and expenses.

- Monitor payment transactions and invoices.

- Set pricing and discounts.

**8. Supplier Management:**

- Manage supplier information and contacts.

- Handle purchase orders and supplier relationships.

- Track deliveries and supplier performance.

**9. Maintenance and Updates:**

- Perform system maintenance tasks.

- Install software updates and security patches.

**10. Logout Functionality:**

- Allow admins to securely log out of their accounts.

**SAMPLE CODING**

**SAMPLE CODING**

**CODING FOR USER MODULE**

**USER REGISTRATION**

def u\_register(request):

if request.method == 'POST':

data1 = RegisterForm(request.POST)

if data1.is\_valid():

data1.save()

messages.error(request,'Invalid Details...')

return redirect('/register')

data1 = RegisterForm()

return render(request,'shopkart/register.html',context={'details':data1})

**USER LOGIN**

def u\_login(request):

if request.method == "POST":

form = AuthenticationForm(request, data=request.POST)

if form.is\_valid():

username = form.cleaned\_data.get("username")

password = form.cleaned\_data.get("password")

user = authenticate(username=username , password=password)

if user is not None:

login(request,user)

return redirect('/')

else:

messages.error(request,"Invalid username (or) password...")

return redirect('/login')

else:

messages.error(request,"Invalid username (or) password...")

return redirect('/login')

form = AuthenticationForm()

return render(request,'shopkart/login.html' , context={'login1':form})

**USER PROFILE**

def u\_profile(request):

det = request.user

return render(request,'shopkart/profile.html',context={'user':det})

**USER PROFILE UPDATION**

def profup(request):

if request.method =='POST':

up\_frm = ProfileUpdateForm(request.POST,request.FILES,instance=request.user.profile)

if up\_frm.is\_valid():

up\_frm.save()

messages.success(request,"Updated successfully !!")

else:

messages.error(request,'invalid form')

up\_frm = ProfileUpdateForm(instance=request.user.profile)

return render (request,'shopkart/updateprf.html',context = {'up\_frm' : up\_frm})

**ADMIN MODULE**

**ADD PRODUCT**

def addp(request):

if request.method=="POST":

p = productaddform(request.POST)

if p.is\_valid():

p.save()

p = productaddform()

return render(request,"shopkart/addp.html",context={'form':p})

**ADD CATEGORY**

def addc(request):

if request.method=="POST":

p = categoryaddform(request.POST)

if p.is\_valid():

p.save()

p = categoryaddform()

return render(request,"shopkart/addc.html",context={'form':p})

**CART DETAILS REPORT**

def cart\_details\_report(request):

# Get the current user (assuming you have authenticated users)

user = request.user

# Retrieve the user's cart based on their session data

cart = request.session.get('cart', {})

# Initialize an empty list to store cart details

cart\_details = []

# Loop through the items in the cart and gather details

for product\_id, quantity in cart.items():

product = Products.objects.get(pk=product\_id)

cart\_details.append({

'product\_name': product.name,

'quantity': quantity,

'added\_by': user.username, # You can customize this based on your user model

})

# Render the cart details report template with the data

return render(request, 'shopkart/report.html', {'cart\_details': cart\_details})

**HTML**

**REGISTRATION**

{% extends 'shopkart/header.html'%}

{% load static %}

{% block content %}

<html>

<title>user registration form</title>

<div class='register'>

<h1 style="text-align: center; color: #ff523b;">Register</h1>

<form method="POST">

{%csrf\_token%}

<h4 style="margin-top:20px;">Username : {{details.username}}</h4><br>

<h4 style="margin-top:20px;">Email : {{details.email}}</h4><br>

<h4 style="margin-top:20px;">Password : {{details.password1}}</h4><br>

<h4 style="margin-top:20px;">Password(again) : {{details.password2}}</h4><br>

<div>

</div>

{% for message in messages %}

<p style="color: red;">{{message}}<p>

{% endfor %}

<button style="border: none;" class="btn" type="submit">register</button><script>onsubmit("you have successfully registered")</script>

</form>

<br>

<h4>If you already have an account. <a href ="/login" style="color: blue;">login</a></h4>

</div>

</div>

</html>

{% endblock %}

**LOGIN**

{% extends 'shopkart/header.html'%}

{% load static %}

{% block content %}

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

<div class="loginpg">

<h1 style="text-align: center; color: #ff523b;">Login</h1>

<br>

<form method="POST">

{% csrf\_token %}

<h4 style="margin-top:20px;">Username : {{login1.username}}</h4><br>

<h4 style="margin-top:20px;">Password : {{login1.password}}</h4><br>

<div>

{% for message in messages %}

<p style="color: red;">{{message}}<p>

{% endfor %}

</div>

<button style="border: none;" class="btn" type="submit">Login</button>

</form>

<br>

<p >Don't have an account? <a href="/register" style="color: blue;">Create an account</a></p>

</div>

{% endblock %}

**PROFILE**

{% extends 'shopkart/header.html'%}

{% load static %}

{% block content %}

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

{% if user.is\_authenticated %}

<center>

<div class="prf">

<img class='logo1' style="margin-top: 10px; margin-bottom: 8px;border-radius: 150px;height: 100px;width: 100px;" src="{{user.profile.image.url}}">

<h2 style="margin-right:60px; margin-bottom: 10px;" >welcome {{user}}</h2>

<p> NAME :{{user}}</p></br>

<p> EMAIL : {{user.email}}</p></br>

<p> ADDRESS: {{user.profile.add}}</p></br>

<p> MOBILE : {{user.profile.phone}}</p></br>

<p> D O B : {{user.profile.dob}}</p></br>

<a href="{% url 'update'%}"><button class="btn" style="border: none;" type="submit">Update</button></a>

<a href="{% url 'logout'%}"><button style="border: none;" class="btn" type="submit">Logout</button></a>

{% else %}

<h2>please login to view your profile </h2>

<a style="border: none;" href="{% url 'login'%}">login</a>

{% endif %}

</div>

</center>

{% endblock %}

**UPDATE PROFILE**

{% extends 'shopkart/header.html'%}

{% load static %}

<html>

{% block content %}

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

{% if user.is\_authenticated %}

<div class="register">

<form method="POST" enctype="multipart/form-data">

{% csrf\_token %}

<img style="margin-top: 10px; margin-bottom: 8px;border-radius: 150px;height: 100px;width: 100px;" src="{{user.profile.image.url}}">

<p >Address : {{ up\_frm.add }}</p>

<p class="regf">Contact : {{ up\_frm.phone }}</p>

<p class="regf">Date Of Birth : {{ up\_frm.dob }}</p>

<p class="regf">{{ up\_frm.image }}</p>

{% for m in messages %}

<p style="font-weight: bold;color: darkgreen;">{{m}}</p>

{% endfor %}

<button class="btn" style="margin-top: 15px; border: none;" type="submit">Update</button></a>

</form>

</div>

{% else %}

<h3>Please Login</h3>

<a href="/login"><input type="submit" value="login"></a>

{% endif %}

{% endblock %}

</html>

**ADD PRODUCT**

{% extends 'shopkart/header.html'%}

{% load static %}

{% block content %}

<html>

<title>Product add form</title>

<div class='register'>

<h1 style="text-align: center; color: #ff523b;">Add Product</h1>

{% if request.user.is\_staff %}

<form method="POST" enctype="multipart/form-data">

{%csrf\_token%}

<p style="margin-top:20px;">Name : {{form.name}}</p><br>

<p style="margin-top:20px;">Price : {{form.price}}</p><br>

<p style="margin-top:20px;">Category : {{form.category}}</p><br>

<br><p class="regf">{{ form.image }}</p>

<button style="border: none;" class="btn" type="submit">Add</button>

</form>

{% else %}

<p>you doesn't have an admin access</p>

{% endif %}

<br>

</div>

</div>

</html>

{% endblock %}

**ADD CATEGORY**

{% extends 'shopkart/header.html'%}

{% load static %}

{% block content %}

<html>

<title>Category add form</title>

<div class='addc'>

<h2 style="text-align: center; color: #ff523b;">Add Category<h2>

{% if request.user.is\_staff %}

<form method="POST">

{% csrf\_token %}

{{form.as\_p}}

<input type="submit" value="Add" class="btn" style="border: none; text-align: center;">

</form>

{% else %}

<p>you doesn't have an admin access</p>

{% endif %}

<br>

</div>

</div>

</html>

{% endblock %}

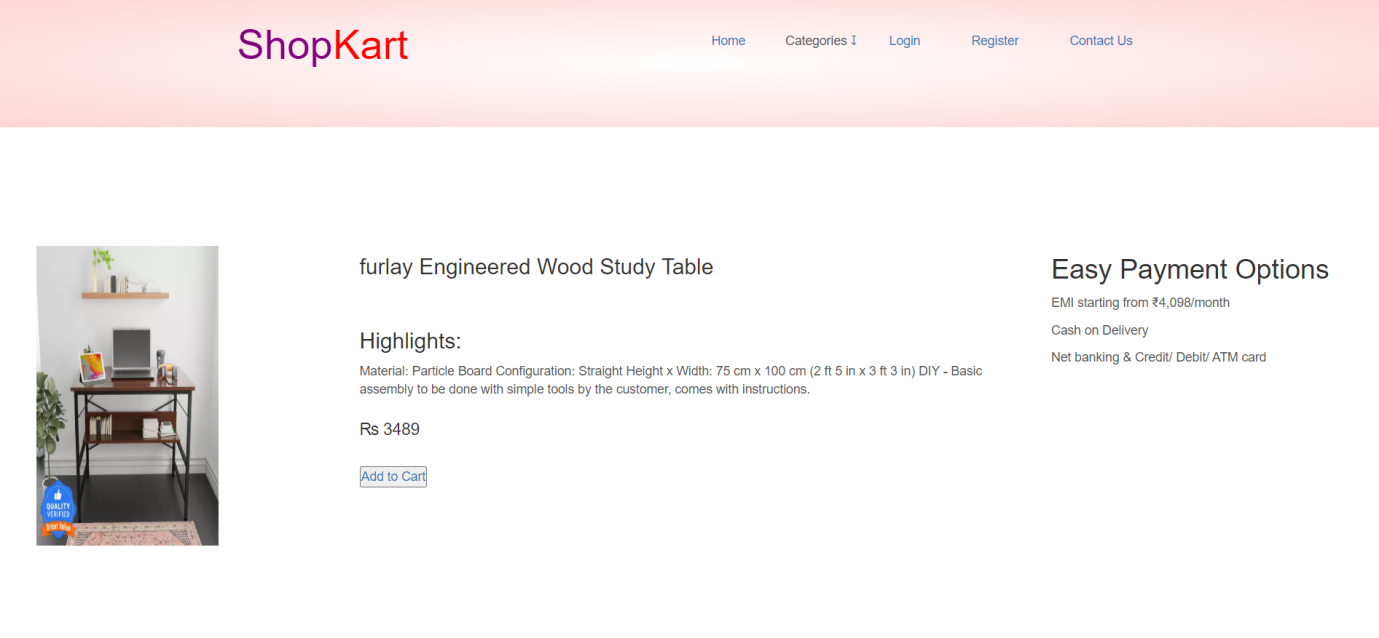
**SCREEN SHOTS**

**SCREEN SHOTS**

**5.1 INPUTFORMS /OUTPUT FORMS**

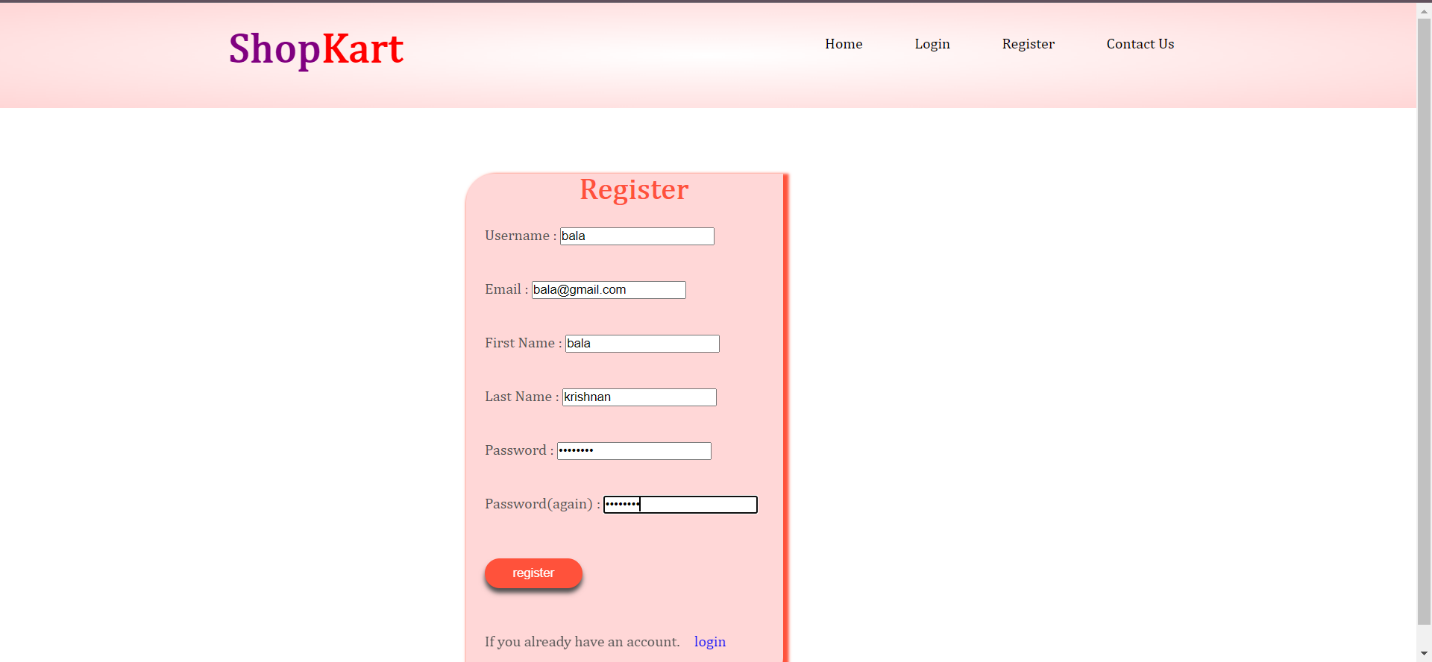
**5.1.1 HOME**

Description :Home page providing users with essential information and navigation options

****

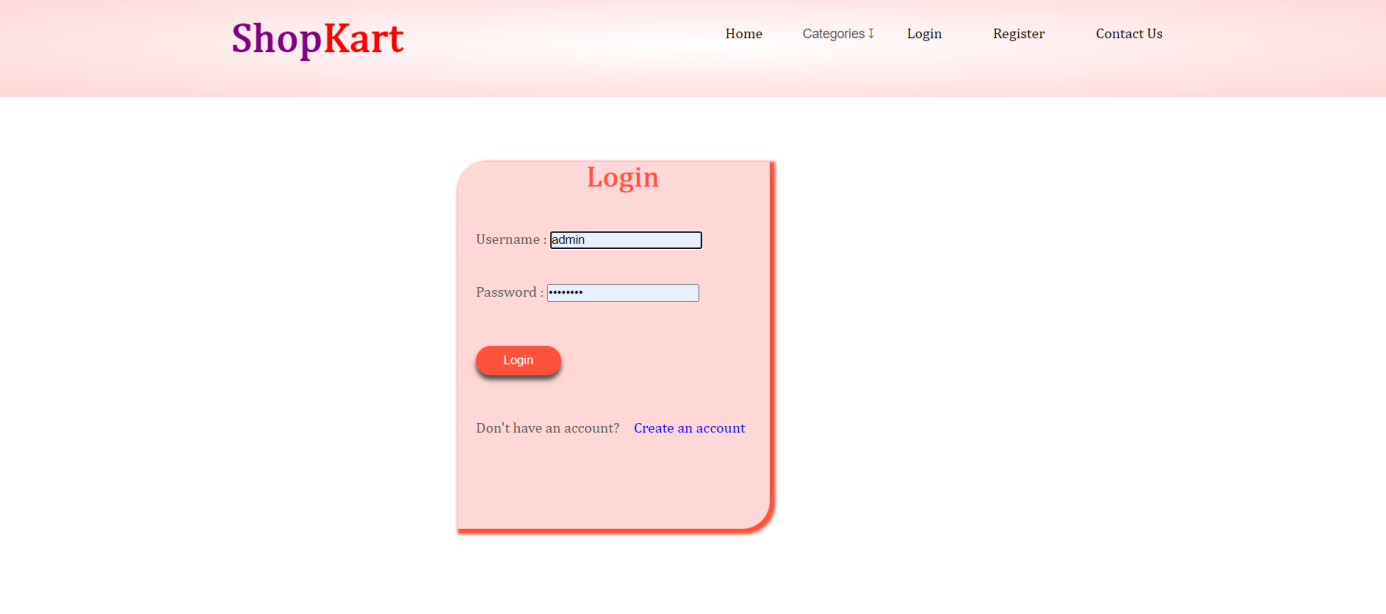
**5.1.2 REGISTRATION**

Description: Register page allows new users to create accounts.



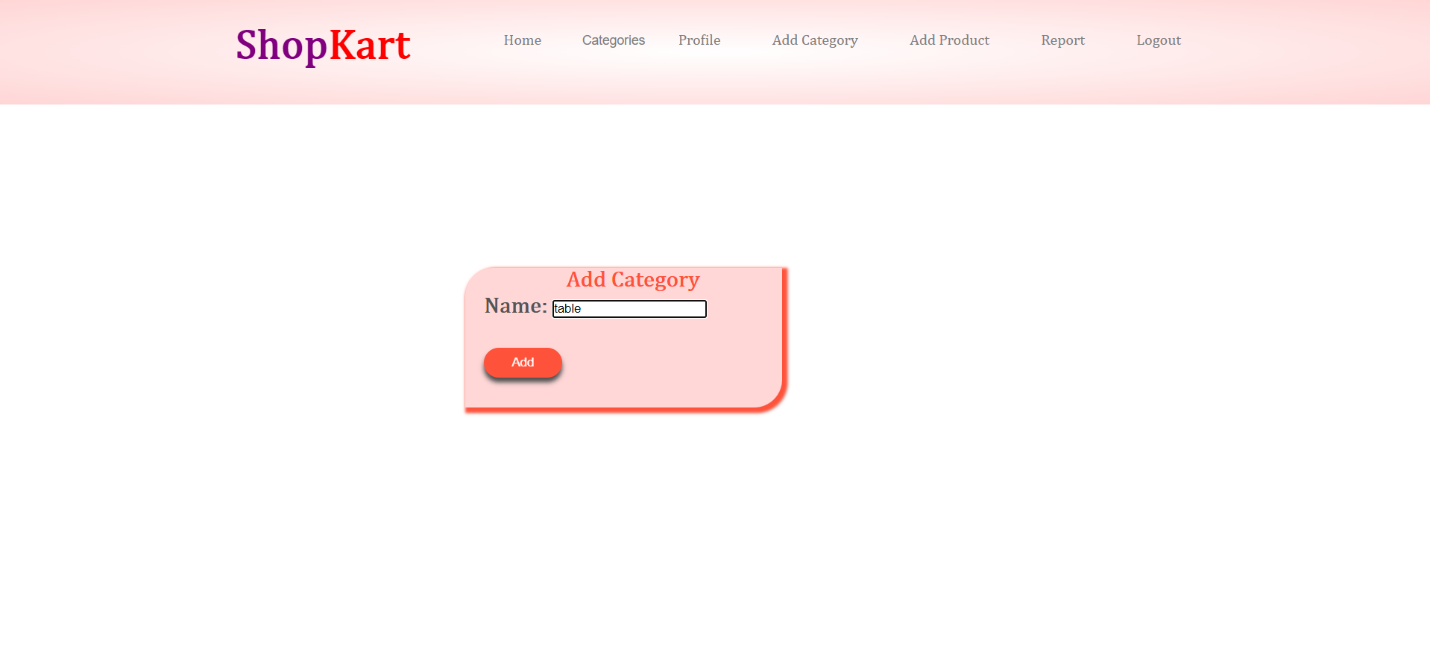
**5.1.3 LOGIN**

Description Login page provides a secure gateway for authorized user to access the system.

****

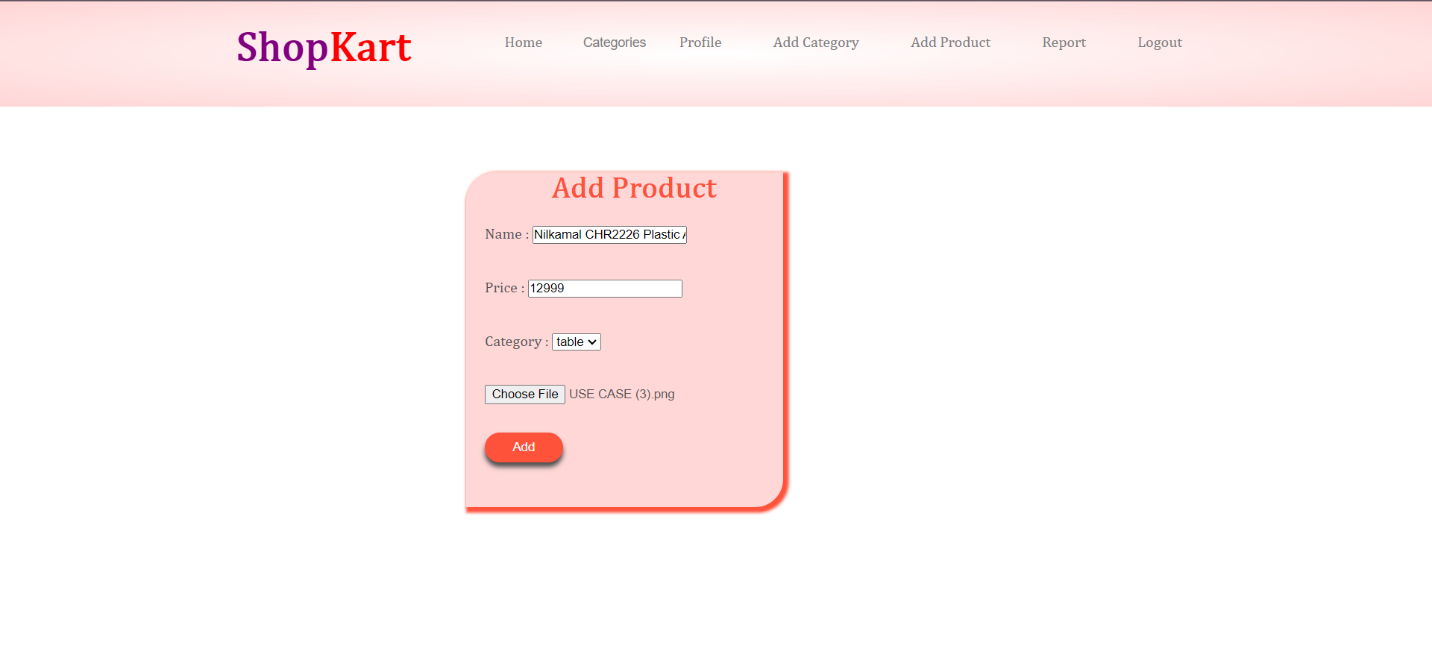
**5.1.4 ADD CATEGORY**

Description : Add category is used to the admin to add category of the furniture.



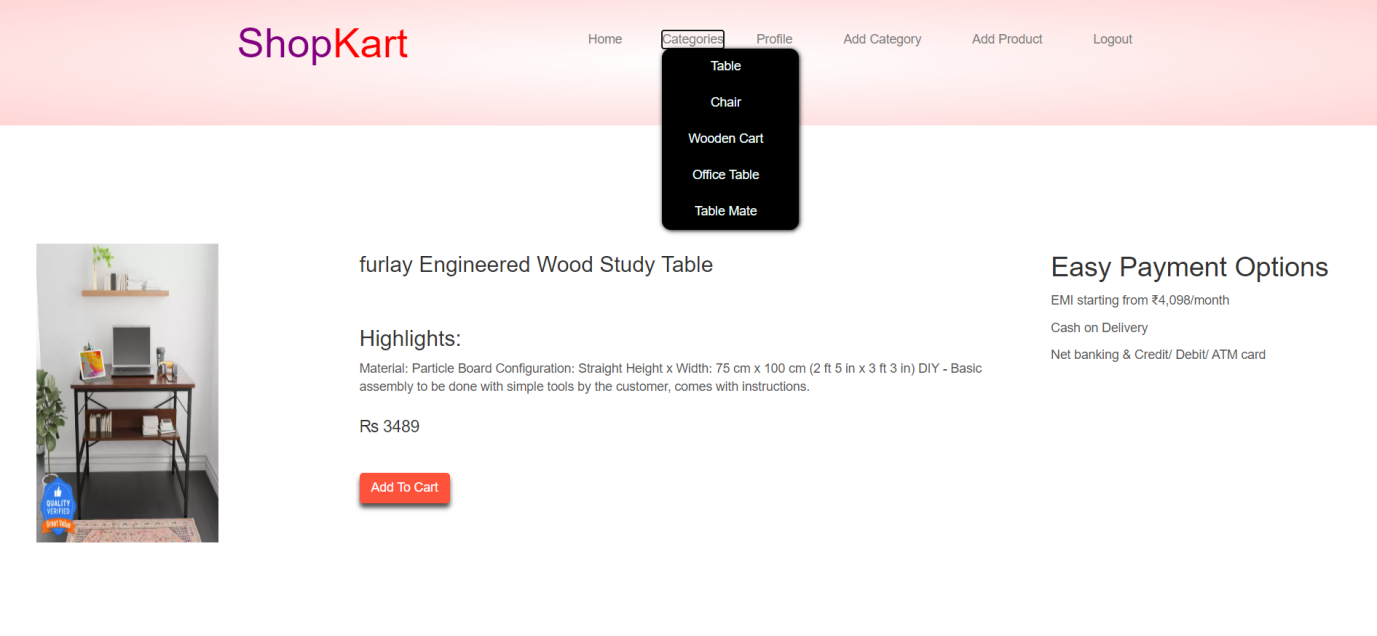
**5.1.5 ADD PRODUCT**

Description : Add product is used to the admin to add furniture



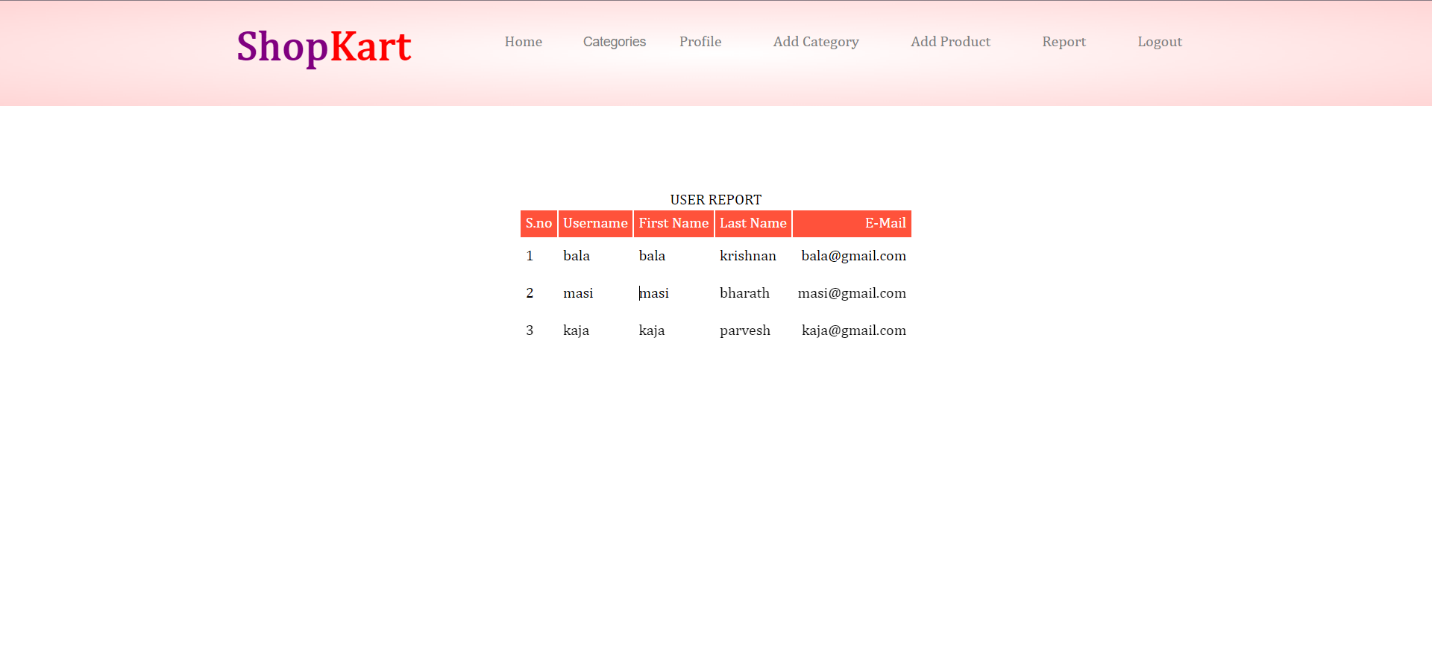
**5.5.6 CATEGORY**

Description : Category option is used to find the furniture in categor type.

****

**5.2 REPORTS**

Description :admin can generate user report.



**CONCLUSION AND FUTURE ENCHANCEMENT**

**CONCLUSION AND FUTURE ENHANCEMENT**

### CONCLUSION

In conclusion, the Furniture Management System project, "Shop Cart," represents a significant leap forward in the way furniture businesses operate and serve their customers. Developed using Python and Django, this robust and versatile system offers an array of features that streamline inventory management, sales processing, and customer engagement. It provides both customers and administrators with a user-friendly and efficient platform for their respective needs. For customers, it simplifies the shopping experience, allowing them to browse, select, and purchase furniture seamlessly. For administrators, it empowers them with tools to manage products, monitor inventory in real-time, and make data-driven decisions. With responsive design and strong security measures, Shop Cart not only improves operational efficiency but also ensures a safe and enjoyable shopping environment. Overall, this project revolutionizes the way furniture businesses function, fostering growth, efficiency, and customer satisfaction.

1. **Efficient Furniture Management:**

The FMS has enabled us to efficiently manage our furniture inventory, track sales, and optimize our supply chain operations.

**2 .Improved Customer Experience:**

Customers now have a user-friendly platform to browse and purchase furniture products,

resulting in increased customer satisfaction and loyalty.

1. **Enhanced Inventory Control:**

The system's inventory management features have allowed us to reduce stockouts, minimize

overstocking, and improve overall inventory accuracy.

**4 .Streamlined Order Processing:**

Our order management functionalities have streamlined order processing, reducing errors and

improving order fulfillment efficiency.

**5 .Security and Data Protection:**

The system has been designed with security in mind, protecting user data and ensuring secure payment processing.

### FUTURE ENHANCEMENT

Looking ahead, the future enhancements for the Furniture Management System project, "Shop Cart," hold the potential to elevate its capabilities even further. One promising avenue is the integration of advanced data analytics and machine learning algorithms to provide predictive insights into inventory management, helping store owners optimize stock levels and minimize wastage. Additionally, incorporating a recommendation engine based on customer preferences and browsing history can enhance the personalized shopping experience, boosting sales and customer loyalty. Expansion into a mobile application version would tap into a broader user base, allowing customers to shop on-the-go, while implementing more robust reporting and business intelligence features will empower administrators with deeper insights into their operations. Lastly, exploring sustainability features such as eco-friendly product tagging and carbon footprint tracking can align the system with modern environmental concerns, reflecting a commitment to responsible business practices. These future enhancements promise to keep "Shop Cart" at the forefront of innovation in the furniture retail industry, offering both customers and administrators an ever-improving platform for their needs.

1. **Mobile App:**

Develop a mobile application to provide customers with a convenient and responsive shopping experience on their smartphones and tablets.

1. **Integration with E-commerce Platforms:**

Integrate the FMS with popular e-commerce platforms like Commerce to expand

our online presence and reach a broader audience

.

1. **Expanded Payment Options:**

Offer additional payment options, such as digital wallets and installment plans, to cater to a

wider range of customers.

1. **Customer Engagement Tools:**

Implement features like live chat support, product reviews, and a customer loyalty program to

enhance customer engagement.

**5 . Sustainability and Eco-Friendly Products:**

Include information about the sustainability and eco-friendly products to meet growing

consumer demands for environmentally responsible choices.

# BIBLIOGRAPHY

**BIBLIOGRAPHY**

**BOOK REFERENCES**

:

1. "**Python Crash Course**" by Eric Matthes

2. "**Learning Python**" by Mark Lutz

3. "**SQL For Dummies**" by Allen G. Taylor

4. "**Learning SQL**" by Alan Beaulieu

**WEBSITE REFERENCES**

1. [www.pythonreferencebook.com](http://www.phpreferencebook.com/)
2. [www.tutorialspoint.com/python/python\_function\_reference.htm](http://www.tutorialspoint.com/php/php_function_reference.htm)
3. [www.elated.com/articles/python-references/](http://www.elated.com/articles/php-references/)