CS254 - Database System Lab

Online Retail Furniture Shopping Site

Team Members:

1. N Nagabhushanam - 221CS231
2. N Yaswanth - 221CS232
3. P Abhishek Kumar - 221CS233
4. D Pramod Chaitanya - 221CS235

Contents

Introduction

In the continuous changing world, where consumer preferences are always changing and technology is developing at an extraordinary rate, there have been major changes to the retail scene. The emergence of e-commerce has fundamentally altered consumer purchasing habits by offering unparalleled convenience, accessibility, and variety to customers worldwide. The retail furniture industry, among many others, has embraced the digital revolution, utilizing online platforms to engage with clients and boost revenue [1]. For product display and sales support, the retail furniture sector has historically depended on physical storefronts. But furniture retailers now need to adjust to the digital world as e-commerce grows and customer preferences shift to online shopping. There are various benefits to purchasing furniture online as opposed to using traditional retail techniques. It saves clients from having to make in-person store visits by enabling them to conveniently browse and buy furniture from their homes comfortably [2].

Online stores also have a larger product assortment, giving buyers the opportunity to look through a wide range of designs, styles, and price ranges. Also, retailers can expand their customer base and get over regional barriers by offering furniture for sale online [3]. Through the utilization of digital marketing tactics and social media platforms, retailers can proficiently advertise their merchandise and interact with prospective clients from diverse demographic backgrounds. A unique set of difficulties arises when switching to online shopping, though. In a market where competition is fierce, maintaining and gaining new customers depends on offering a smooth and intuitive shopping experience. Furthermore, gaining the confidence and credibility of internet customers depends on dependable logistics and safe payment processing [3].

Our project, the Online Retail Furniture Shopping Site, develops an innovative online platform that transforms the furniture shopping experience for customers, with a particular emphasis on innovation, user experience, and business efficiency.

Literature Survey

This literature survey explores leading furniture retailers IKEA, Wayfair, Ashley Furniture, and Overstock. By analyzing their operations and shortcomings, we gain valuable insights into the dynamics of the online furniture retail sector.

IKEA [4]

IKEA is a leading global retailer of furniture, providing stylish and reasonably priced home furnishings in many different nations. IKEA makes furniture that is easy to assemble and ships at a low cost, making its furniture more accessible to customers across the globe. But some customers might find it difficult to self-assemble, especially if they don't have the necessary tools or assembly skills. Furthermore, customers looking for distinctive furniture pieces may find fewer personalization options in IKEA's product range. IKEA furniture's quality and longevity have also come under scrutiny, as some customers have reported problems over time [4]. IKEA is still a well-liked option for consumers on a tight budget looking for contemporary furniture designs in spite of these disadvantages.

Wayfair [5]

Wayfair functions as an all-inclusive virtual marketplace for household goods, offering an extensive variety of furniture pieces from multiple brands. The website's advanced search capabilities and user-friendly interface make it easier for customers to browse and find products. Yet, the wide range of suppliers can cause variations in product quality, which can result in inconsistent sales and possibly unsatisfied consumers [5]. Furthermore, a few customers have experienced delivery delays, particularly for items with longer lead times or backorders, which has negatively impacted their overall shopping experience. Wayfair's return policy may also pose difficulties, as the convenience of online shopping is diminished by restocking fees and return shipping charges [5]. Wayfair's wide selection of products and affordable prices make it a popular choice for home furnishings even with these disadvantages.

Ashley Furniture [6]

Ashley Furniture is a well-known furniture retailer with a large selection of merchandise and physical retail stores spread throughout multiple locations. The company appeals to a wide range of customers with its varied furniture collections, which suit a variety of budgets and styles. However, consumers on a tight budget looking for less expensive options might be turned off by Ashley Furniture's higher price point. For customers looking for high-quality furniture and reliable brands, Ashley Furniture is still a popular option. Furthermore, customers with particular design preferences may not be able to personalize Ashley Furniture products due to

the limited customization options [6]. A few consumers have also voiced dissatisfaction with customer service, citing challenges in resolving product flaws and delivery delays, which have an effect on general satisfaction.

Overstock [7]

Overstock is a well-known online retailer that draws in customers on a tight budget by providing discounted furniture and home decor at affordable prices. Cost-conscious shoppers continue to favour overstock, but it's advisable to proceed with caution when making purchases. However, there have been reports of flaws and inconsistencies, raising questions about the quality of the product [7]. There have also been reports of delivery delays, possibly as a result of inventory problems. There may be costs and inconveniences associated with the return process. Problems with customer service, like delayed responses and improper fixes, have made shopping even less enjoyable.

In summary, even though IKEA, Wayfair, Ashley Furniture and Overstock are excellent in many areas, they have drawbacks like hard to assemble items, poor quality, slow delivery, and little customization. By prioritizing simple assembly, guaranteeing product quality, simplifying delivery procedures, offering a wide range of customization options with proper communication between consumer and retailer, and providing first-rate customer service, we hope to reduce these drawbacks in our project. We hope to give our users a more seamless and fulfilling online furniture shopping experience by taking on these challenges head-on.

Objectives

The project aims to create a secure, user-friendly online furniture shopping platform. Objectives include implementing authentication, optimizing performance, integrating inventory management, ensuring smooth transactions, and enhancing user experience with intuitive design.

1. Implement a secure authentication and authorization system to ensure data privacy and user account protection, utilizing industry-standard encryption protocols and multi-factor authentication methods.
2. Integrate a robust inventory management system to track product availability, manage stock levels, and synchronize real-time inventory updates across multiple channels, minimizing the risk of overselling or stock outs.
3. Reduce page load times and increase overall site speed by using strategies like lazy loading, code compression, and caching mechanisms to optimize responsiveness and performance on websites.
4. Implement seamless integration with third-party payment gateways and shipping providers to facilitate secure transactions, ensuring smooth order processing, payment processing, and shipping fulfillment.
5. Integrate easy-to-use social media sharing tools, enabling users to effortlessly share purchases and reviews on popular platforms, amplifying social proof and attracting new customers through word-of-mouth marketing.
6. Develop an intuitive and efficient user interface (UI) design that prioritizes ease of navigation, accessibility, and visual appeal, enhancing the overall user experience (UX) of the online furniture shopping platform.
7. Develop a comprehensive data backup and disaster recovery plan to protect critical business data, ensure continuity of operations, and minimize downtime in the event of system failures, cyber attacks, or natural disasters

Features

User Registration and Authentication:

1. Allow users to register and login to the website securely.
2. Implement authentication mechanisms to ensure user account protection.

Product Browsing and Filtering:

1. Provide users with the ability to browse and filter products based on different categories.
2. Implement search functionality for users to easily find specific items.

Shopping Cart Management:

1. Enable users to add multiple items to their shopping cart.
2. Allow users to adjust the quantity of items in the cart.
3. Implement functionalities for adding, removing, and updating items in the cart.

Order Management:

1. Allow users to view their order history and details.
2. Provide information on the shipping status of the ordered products.

Admin Panel:

1. Create an admin interface for managing products, inventory, and orders.
2. Enable admins to add, update, and remove products from the store.
3. Implement inventory management functionalities for the admin to track stock levels.

Email Notifications:

1. Send email notifications to users upon successful registration.
2. Notify users via email when they place an order or when their order is shipped.
3. Send email notifications to users when out-of-stock items become available again.

Requirement specifications

A. Front End Development :

In frontend development, we are using a combination of HTML, CSS with Bootstrap, JavaScript, and React JS to craft a responsive and engaging user interface.

HTML (Hypertext Markup Language):

HTML will serve as the backbone of our web pages, providing the structural elements necessary for displaying content. We will use semantic HTML markup to ensure accessibility and search engine optimization.

CSS (Cascading Style Sheets) with Bootstrap :

CSS will be used to style the HTML elements, defining layout, typography, colors, and responsiveness. Bootstrap will augment CSS by providing pre-designed components and utilities, facilitating rapid development and consistent styling across devices.

JavaScript :

JavaScript will enhance user interaction and dynamic content on the website. We will employ it for form validation, interactive features, asynchronous data fetching ensuring a seamless user experience.

React JS :

React JS will power our UI components, enabling us to build reusable and modular elements. It will manage the application's state efficiently, facilitate component composition, and enable us to create complex user interfaces with ease.

B. Flask :

Flask is a lightweight and extensible web application framework for Python, responsible for managing server-side logic, data processing, and interaction with the database. Its routing system efficiently maps URL endpoints to view functions, enabling the handling of HTTP requests and rendering of appropriate responses. Upon receiving incoming HTTP requests, Flask extracts data and executes logic to interact with the database, facilitating seamless data processing. Additionally, Flask integrates seamlessly with Jinja2, allowing for dynamic HTML generation through embedded Python code. Furthermore, Flask provides robust session management capabilities, enabling the storage and retrieval of user data across multiple requests, thereby facilitating features such as user authentication and personalized experiences.

C. Backend Development with MySQL Database :

Our application's persistent storage layer, the MySQL database, will hold important information like user profiles, product specifications, and order histories. We will create a relational database schema, specifying tables, columns, primary keys, foreign keys, and constraints to guarantee data integrity and effective querying, in order to correctly represent the application's data model. We will work with the MySQL database using SQLAlchemy ORM to carry out a variety of data manipulation operations, such as adding, editing, removing, and querying records. By enabling us to work with Python objects rather than raw SQL queries, SQLAlchemy's abstraction layer streamlines database interactions and improves the readability and maintainability of code. We'll also use strong input validation and sanitization methods to guard against SQL injection attacks and preserve the integrity of user data.
To prevent unwanted access to user accounts, passwords will be safely hashed and stored using cryptographic hashing algorithms. Flask will use the database URL configuration in SQLAlchemy to create and maintain connections to the MySQL database. For a smooth integration with the backend logic, this also entails setting connection parameters like host, port, username, password, and database name.

This comprehensive approach to front end development, Flask and MySQL database integration will result in a robust and scalable online retail furniture shopping platform, offering a seamless user experience and efficient data management capabilities.

Schema Diagram

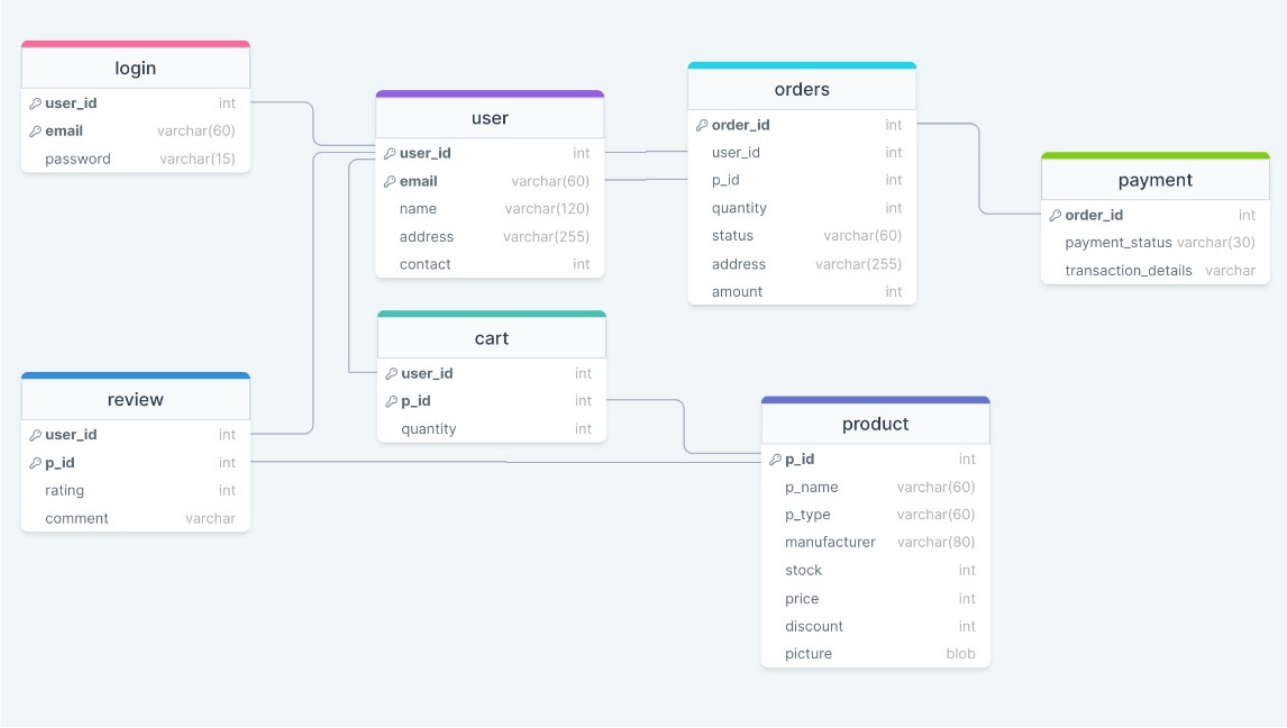

Figure 1: Database schema design

ER Diagram

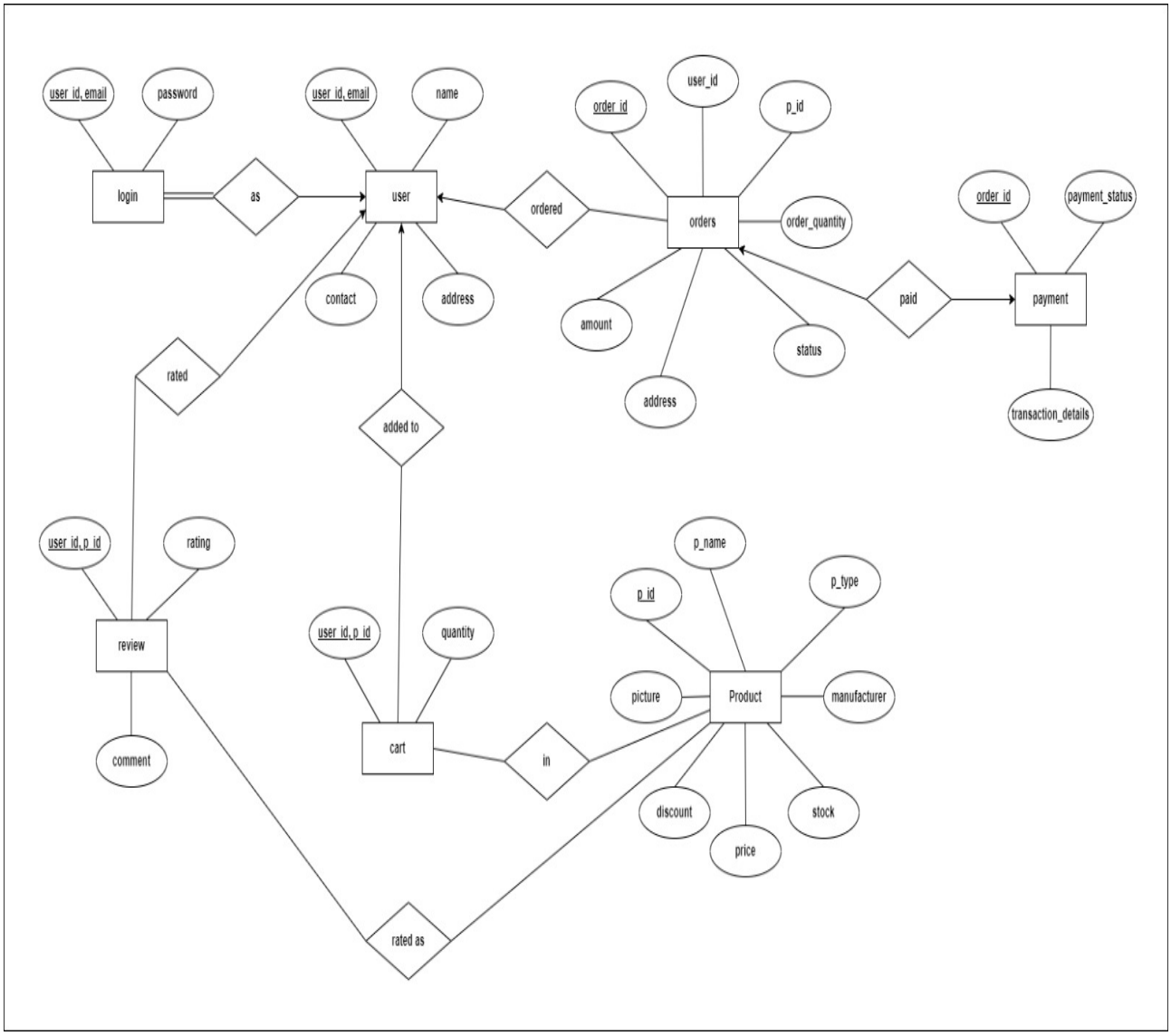

Figure 2: ER diagram

Normalization process

First Normal Form (1NF)

1. A relation is in 1NF if it contains only atomic values and there are no repeating groups or arrays of values.
2. Example: Ensuring that each attribute of a table contains only indivisible values without any repeating groups or arrays.
3. Our tables are all in 1NF as they contain atomic values in each cell and do not have repeating groups.

Second Normal Form (2NF)

1. A relation is in 2NF if it is in 1NF and every non-prime attribute is fully functionally dependent on the whole primary key.
2. Example: Ensuring that every non-prime attribute depends on the entire primary key, not just part of it.
3. Our tables are all in 2NF as all non-prime attributes depend on the whole primary key in each table.

Third Normal Form (3NF)

1. A relation is in 3NF if it is in 2NF and there are no transitive dependencies.
2. Example: Removing non-key attributes that depend on other non-key attributes (transitive dependencies), ensuring that each attribute directly depends only on the primary key.
3. Our tables are all in 3NF, non-key attributes depend only on the primary key, not on other non-key attributes and each attribute in a table is directly related to the primary key and not indirectly through another non-key attribute.

Our tables are confirmed to be in Thrid Normal Form (3NF), which ensures that they are free from transitive dependencies. This normalization level helps to maintain data integrity and eliminates certain types of redundancy in the database schema.

Tables

```
mysql> desc cart;
+----------+------+------+-----+---------+-------+
| Field    | Type | Null | Key | Default | Extra |
+----------+------+------+-----+---------+-------+
| user_id  | int  | NO   | PRI | NULL    |       |
| p_id     | int  | NO   | PRI | NULL    |       |
| quantity | int  | YES  |     | NULL    |       |
+----------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 3: cart table

```
mysql> desc login;
+---------+--------------+------+-----+---------+----------------+
| Field   | Type         | Null | Key | Default | Extra          |
+---------+--------------+------+-----+---------+----------------+
| user_id | int          | NO   | PRI | NULL    | auto_increment |
| email   | varchar(255) | NO   | PRI | NULL    |                |
| pwd     | varchar(255) | NO   |     | NULL    |                |
+---------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

Figure 4: login table

```
mysql> desc orders;
+----------+---------------+------+-----+---------+----------------+
| Field    | Type          | Null | Key | Default | Extra          |
+----------+---------------+------+-----+---------+----------------+
| order_id | int           | NO   | PRI | NULL    | auto_increment |
| user_id  | int           | YES  | MUL | NULL    |                |
| p_id     | int           | YES  | MUL | NULL    |                |
| quantity | int           | YES  |     | NULL    |                |
| status   | varchar(50)   | YES  |     | NULL    |                |
| address  | varchar(255)  | YES  |     | NULL    |                |
| amount   | decimal(10,2) | YES  |     | NULL    |                |
+----------+---------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figure 5: order table

```
mysql> desc payment;
+---------------------+--------------+------+-----+---------+-------+
| Field               | Type         | Null | Key | Default | Extra |
+---------------------+--------------+------+-----+---------+-------+
| order_id            | int          | NO   | PRI | NULL    |       |
| payment_status      | varchar(50)  | YES  |     | NULL    |       |
| transaction_details | varchar(255) | YES  |     | NULL    |       |
+---------------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 6: payment table

```
mysql> desc product;
+--------------+---------------+------+-----+---------+----------------+
| Field        | Type          | Null | Key | Default | Extra          |
+--------------+---------------+------+-----+---------+----------------+
| p_id         | int           | NO   | PRI | NULL    | auto_increment |
| p_name       | varchar(255)  | NO   |     | NULL    |                |
| p_type       | varchar(100)  | YES  |     | NULL    |                |
| manufacturer | varchar(255)  | YES  |     | NULL    |                |
| stock        | int           | YES  |     | NULL    |                |
| price        | decimal(10,2) | YES  |     | NULL    |                |
| discount     | decimal(10,2) | YES  |     | NULL    |                |
| picture      | longblob      | YES  |     | NULL    |                |
+--------------+---------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

Figure 7: product table

```
mysql> desc review;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| user_id | int          | NO   | PRI | NULL    |       |
| p_id    | int          | NO   | PRI | NULL    |       |
| rating  | decimal(3,1) | YES  |     | NULL    |       |
| comment | text         | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Figure 8: review table

```
mysql> desc user;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| user_id | int          | NO   | PRI | NULL    |       |
| email   | varchar(255) | NO   | PRI | NULL    |       |
| name    | varchar(255) | NO   |     | NULL    |       |
| address | varchar(255) | YES  |     | NULL    |       |
| contact | varchar(20)  | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 9: user table

Implementation

Design and Database Tools

1. Draw.io: This web-based diagramming tool was instrumental in creating the Entity-Relationship (ER) diagram for our project. The ER diagram served as a visual representation of the database structure,

illustrating how entities such as users, songs, and playlists are interconnected. draw.io's user-friendly interface and extensive feature set made it an excellent choice for designing complex database schemas without the need for specialized software.

2. OneCompiler: An online platform for coding and executing SQL queries, OneCompiler played a crucial role in the initial creation and testing of our database. It allowed us to write, run, and debug SQL scripts in a cloud-based environment, facilitating a hassle-free setup of the database schema and manipulation of data without the need for local database server installations.

3. SQL Workbench: This powerful, cross-platform SQL query tool was used to further refine our database schema and manage the database. With SQL Workbench, we were able to visually design the schema, perform advanced SQL queries, and manage the database effectively. Its comprehensive features for database administration and development ensured that our database was optimized for performance and scalability.

Front End Development

Our front-end work was concentrated on creating a visually beautiful and user-friendly interface through the usage of JavaScript, React JS, HTML, CSS with Bootstrap, and JavaScript.

1. HTML (Hypertext Markup Language): HTML served as the foundation of our web pages, providing the structure for content presentation and user interaction.

2. CSS (Cascading Style Sheets) with Bootstrap : The user interface elements were styled and designed using CSS and Bootstrap to provide uniformity, responsiveness, and visual appeal on a range of screens and devices.

3. JavaScript : We are improving website's functionality and user engagement using JavaScript. It was applied to form validation, dynamic content changes, and the addition of interactive features to improve user experience.

4. React JS : We will design modular and reusable user interface components thanks to React JS, which makes it easier to construct and maintain complicated user interfaces quickly. It allowed for smooth data flow and rendering optimizations by effectively managing the application state.

Flask

Although we have not yet implemented Flask in our project, it is planned to be utilized as the back end framework for handling server-side logic, data processing, and integration with the MySQL database.

1. Routing System: Flask's routing system will be leveraged to efficiently map URL endpoints to corresponding view functions, enabling seamless handling of HTTP requests and responses.

2. Data Management: Flask will facilitate interaction with the MySQL database, allowing for efficient storage, retrieval, and manipulation of application data. SQLAlchemy ORM will be employed for database operations, ensuring data integrity and security.

3. Session Management: Flask's session management capabilities will be utilized for maintaining user sessions and managing authentication, ensuring secure access to restricted areas of the application.

4. Template Rendering: Flask will enable dynamic HTML generation through template rendering, allowing for the creation of personalized and interactive user interfaces using Jinja2 templates.

MySQL Database

The MySQL database will serve as the persistent data storage layer for the application, storing critical information such as user profiles, product details, and order histories.

1. Database Schema: A relational database schema will be designed to define the structure of the MySQL database, specifying tables, columns, primary keys, foreign keys, and constraints.

2. Data Manipulation: MySQL will facilitate a variety of data manipulation operations, including adding, editing, removing, and querying records. SQLAlchemy ORM will be used to interact with the database, providing a Pythonic interface for database operations.

3. Data Security: To ensure data security, sensitive information such as user passwords will be securely hashed and stored using cryptographic hashing algorithms.

4. Connection Configuration: Flask will utilize the database URL configuration in SQLAlchemy to establish connections to the MySQL database, specifying connection parameters such as host, port, username, password, and database name.

Code snippets

```
def calculate_total_amount(cart_items):
        total_amount = 0
        for item in cart_items:
            total_amount += float(item[2]) * item[3]
        return total_amount

print(mysql)

class User():
    def __init__(self) -> None:
        pass

    def check_existence_user(username):
        cur = mysql.cursor()
        cur.execute("SELECT COUNT(*) FROM login WHERE email=%s", (username,))
        res = cur.fetchone()
        print(res)
        print(type(res))
        cur.close()
        if res == (0,):
            print("User does not exist")  # Debugging message
            return False
        return True

    def check_user_pwd(username, password):
        cur = mysql.cursor()
        cur.execute("SELECT COUNT(*) FROM login WHERE email=%s AND pwd=%s",
                    (username, password))
        res = cur.fetchall()
        print(res)
        print(type(res))

        cur.close()
        if res == [(0,)]:
            print("Invalid user credentials")  # Debugging message
            return False
        return True
```

```
def register_new_user(name, email, address, mobile, pincode, password):
    cur = mysql.connection.cursor()
    cur.execute("INSERT INTO registration (name, email, address, mobile, pin_code, password) VALUES (%s, %s, %s, %s, %s, %s)",
                (name, email, address, mobile, pincode, password))
    mysql.connection.commit()
    cur.close()
    print("Registration successful:", name, email)  # Debugging message

def get_user_cart_items(userid):
    cur = mysql.cursor()
    cur.execute("SELECT c.p_id, p.p_name, p.price, c.quantity FROM cart c JOIN product p ON c.p_id=p.p_id WHERE user_id=%s", (userid,))
    res = cur.fetchall()
    cur.close()
    print("Cart retrieval successful:", userid)  # Debugging message
    return res

def get_user_id(username):
    cur = mysql.cursor()
    cur.execute("SELECT user_id FROM login WHERE email=%s", (username,))
    res = cur.fetchall()
    cur.close()
    print("User ID retrieval successful:", username)
    return res[0][0]
```

```python
def add_to_cart(user_id, p_id):
    cur = mysql.cursor()
    try:
        cur.execute("SELECT quantity FROM cart WHERE user_id=%s AND p_id=%s", (user_id, p_id))
        result = cur.fetchone()
        if result:
        # Update the existing quantity
            new_quantity = result[0] + 1
            cur.execute("UPDATE cart SET quantity=%s WHERE user_id=%s AND p_id=%s", (new_quantity, user_id, p_id))
        else:
        # Insert a new row
            cur.execute("INSERT INTO cart (user_id, p_id, quantity) VALUES (%s, %s, %s)", (user_id, p_id, 1))
        mysql.commit()
    except mysql.connector.Error as error:
        print("Error: ", error)
    finally:
        cur.close()
        print("Product added to cart successfully:",p_id)


def get_user_details(user_id):
    cur = mysql.cursor()
    cur.execute("SELECT name, email, address, mobile FROM registration WHERE id=%s", (user_id,))
    res = cur.fetchone()
    cur.close()
    print("User details retrieval successful:", user_id)  # Debugging message
    return res

def clear_cart(user_id):
    cur = mysql.cursor()
    cur.execute("DELETE FROM cart WHERE user_id=%s", (user_id,))
    mysql.commit()
    cur.close()
    print("Cart cleared successfully:", user_id)  # Debugging message
```

```python
class Admin:
    def remove_product(p_id):
        cur = mysql.cursor()
        cur.execute("DELETE FROM product WHERE p_id=%s", (p_id,))
        mysql.commit()
        cur.close()
        print("Product removed successfully:", p_id)

    def update_product(p_name, p_type, manufacturer, stock, price):
        cur = mysql.cursor()
        cur.execute("UPDATE product SET p_type=%s, manufacturer=%s, stock=%s, price=%s WHERE p_name=%s",
                    (p_type, manufacturer, stock, price, p_name))
        mysql.commit()
        cur.close()
        print("Product updated successfully:", p_name)

    def get_shipped_products():
        cur = mysql.cursor()
        cur.execute("SELECT p.transaction_details, o.p_id, o.user_id, o.address, o.quantity, o.amount, o.status "
                    "FROM orders o JOIN payment p ON o.order_id=p.order_id WHERE o.status='shipped'")
        res = cur.fetchall()
        cur.close()
        print("Retrieval of shipped products successful")
        return res

    def get_unshipped_products():
        cur = mysql.cursor()
        cur.execute("SELECT p.p_id, p.p_name, o.user_id, o.address, o.quantity, o.amount, o.status "
                    "FROM orders o JOIN product p ON o.p_id=p.p_id WHERE o.status='unshipped'")
        res = cur.fetchall()
        cur.close()
        print("Retrieval of unshipped products successful")
        return res
```

```python
def get_returned_products():
    cur = mysql.cursor()
    cur.execute("SELECT p.p_id, p.p_name, o.user_id, o.address, o.quantity, o.amount, o.status "
                "FROM orders o JOIN product p ON o.p_id=p.p_id WHERE o.status='returned'")
    res = cur.fetchall()
    cur.close()
    print("Retrieval of returned products successful")
    return res

def process_order(user_id, cart_items, user_details):
    cur = mysql.cursor()
    order_id = None

    try:
        # Insert order details
        cur.execute("INSERT INTO orders (user_id, address, amount, status) VALUES (%s, %s, %s, %s)",
                    (user_id, user_details[2], calculate_total_amount(cart_items), 'unshipped'))
        order_id = cur.lastrowid

        # Insert order items
        for item in cart_items:
            cur.execute("INSERT INTO order_items (order_id, p_id, quantity) VALUES (%s, %s, %s)",
                        (order_id, item[0], item[3]))

        # Insert payment details (assuming successful payment)
        cur.execute("INSERT INTO payment (order_id, payment_status, transaction_details) VALUES (%s, %s, %s)",
                    (order_id, 'successful', 'Transaction details'))

        mysql.commit()
    except Exception as e:
        mysql.rollback()
        print("Error processing order:", str(e))
    finally:
        cur.close()

    return order_id
```

```python
def get_order_details(order_id):
    cur = mysql.cursor()
    cur.execute("SELECT o.order_id, o.user_id, o.address, o.amount, o.status, p.payment_status, p.transaction_details "
                "FROM orders o JOIN payment p ON o.order_id=p.order_id WHERE o.order_id=%s", (order_id,))
    res = cur.fetchone()
    cur.close()
    print("Order details retrieval successful:", order_id)
    return res

def calculate_total_amount(cart_items):
    total_amount = 0
    for item in cart_items:
        total_amount += float(item[2]) * item[3]
    return total_amount
def get_all_products():
    cur = mysql.cursor()
    cur.execute("select p_id,p_name,p_type,price,stock from product") #=========================
    res=cur.fetchall()
    cur.close()
    print("Retrival of products successful")
    return res
```

Website Pages

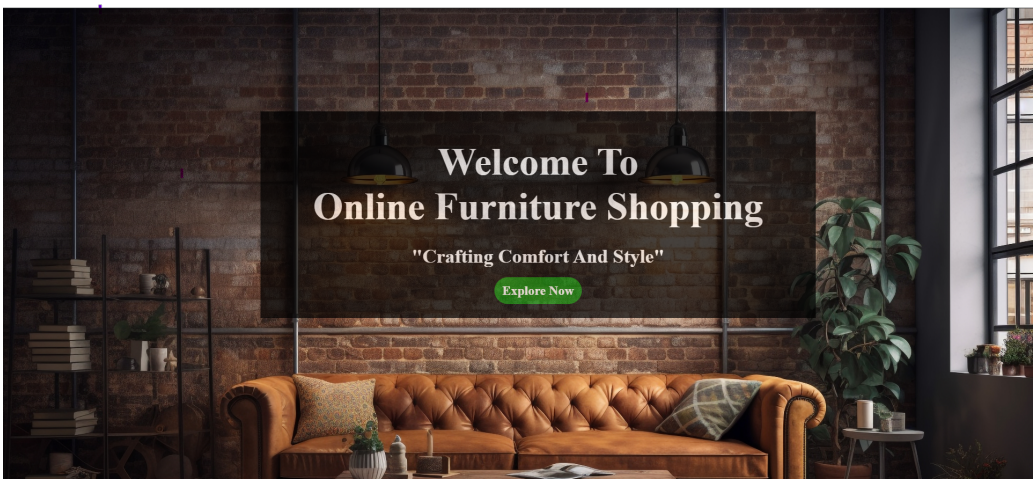

Figure 10: Main Page

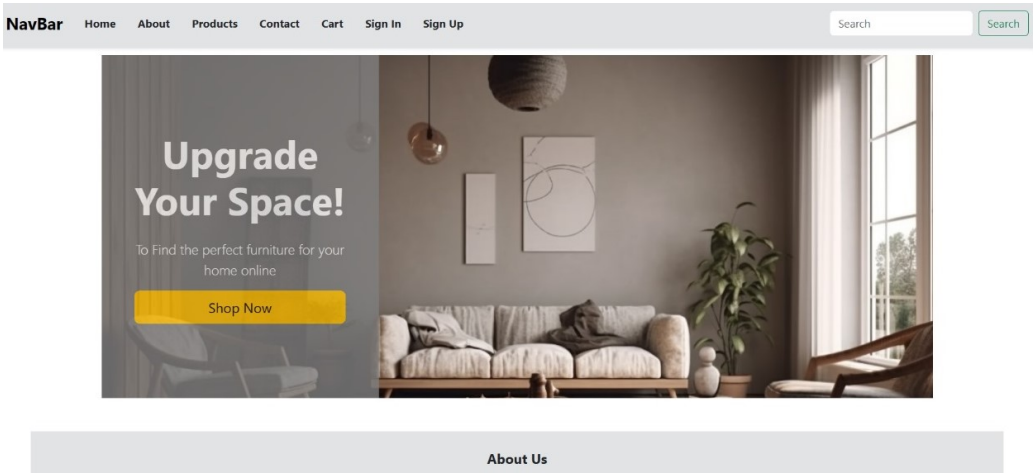

Figure 11: Main Page

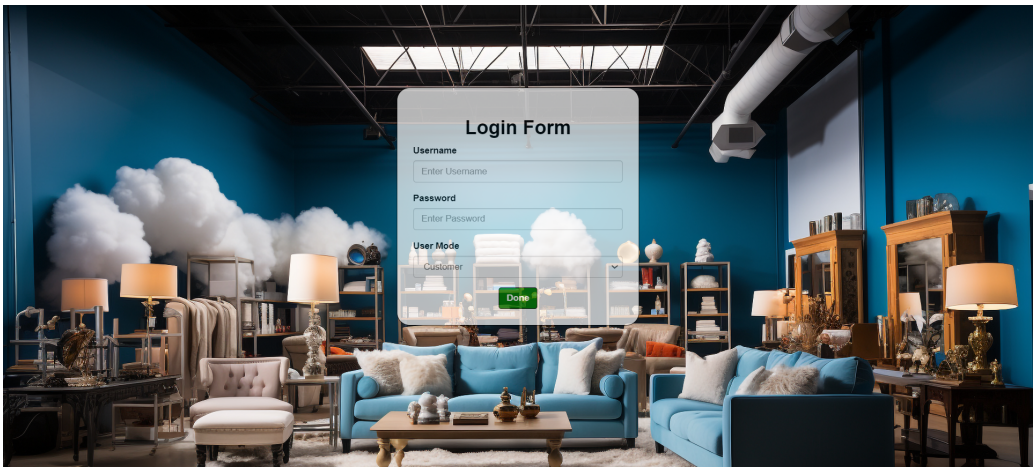

Figure 12: Login Page

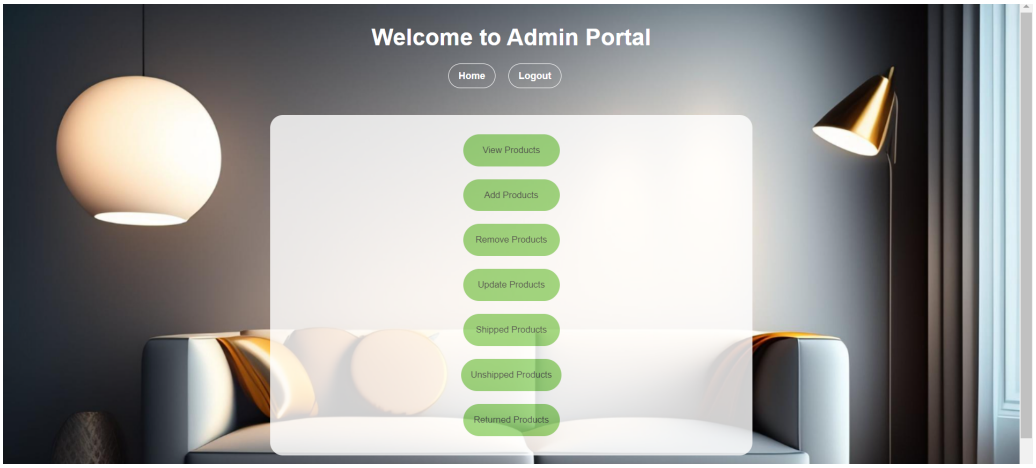

Figure 13: Admin Portal

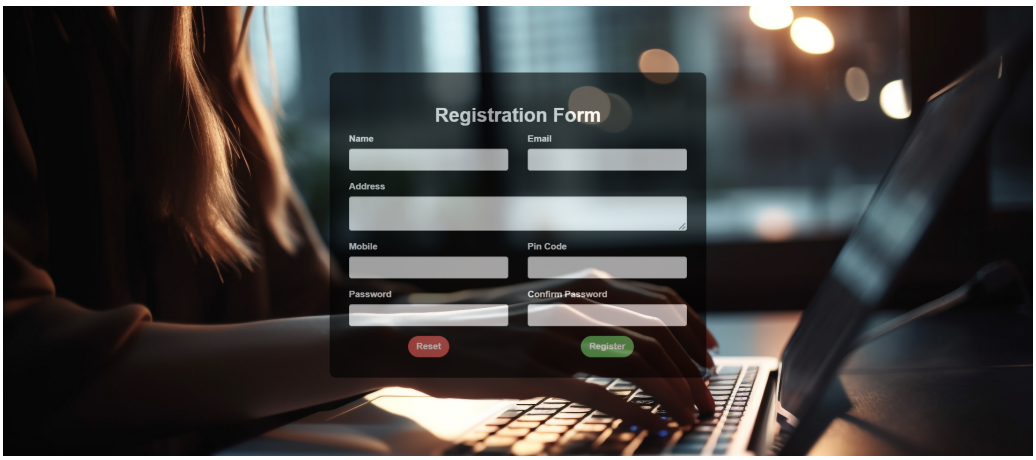

Figure 14: Registration

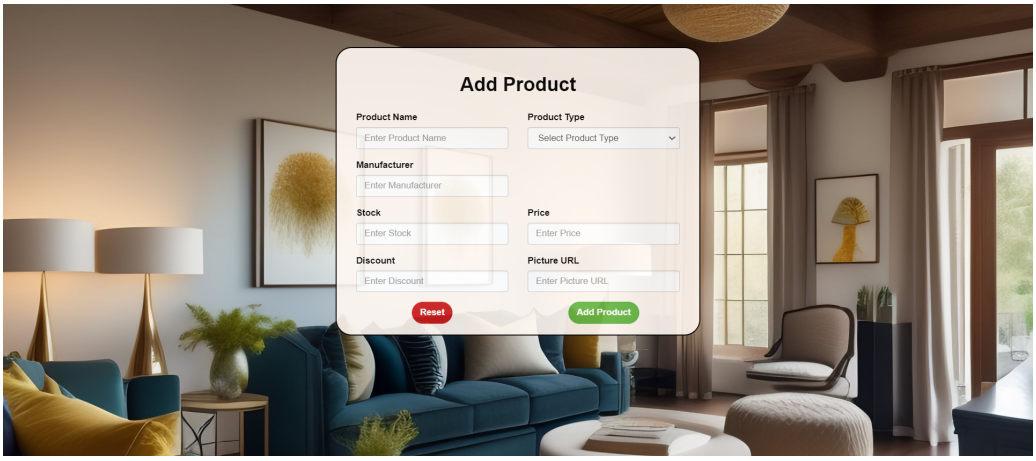

Figure 15: Add Product

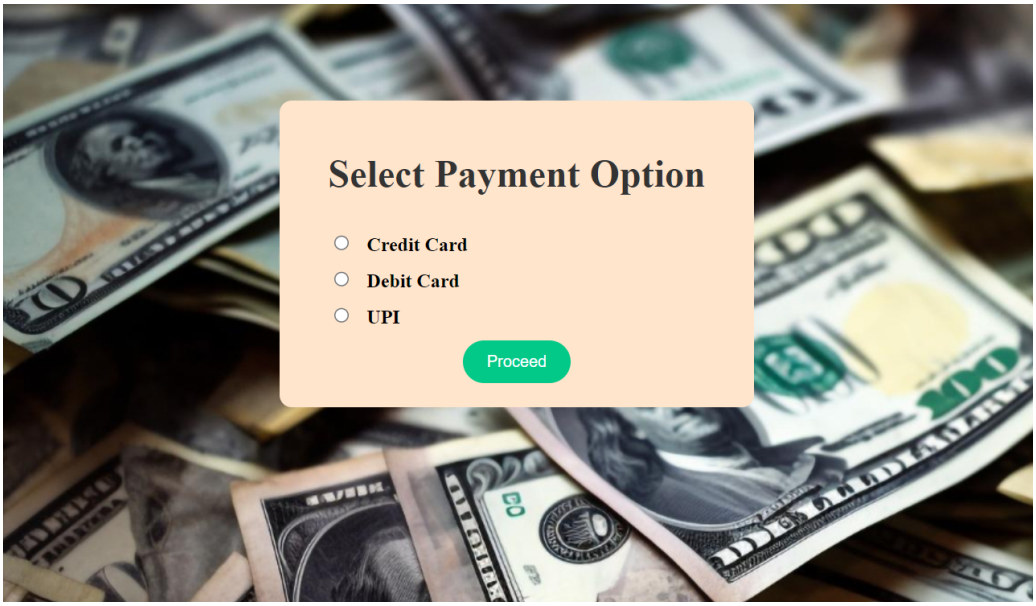

Figure 16: Select Payment Type

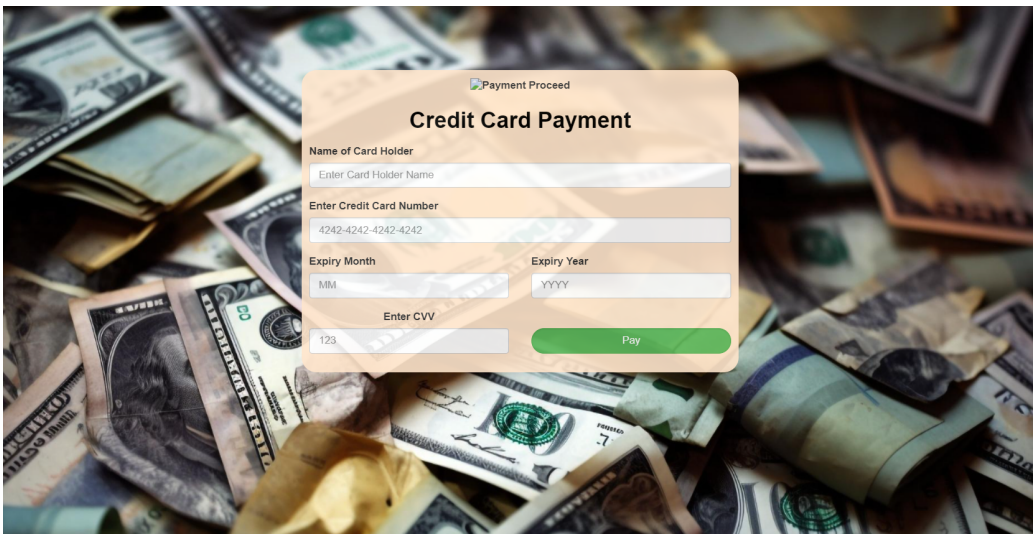

Figure 17: Payment Through Credit Card

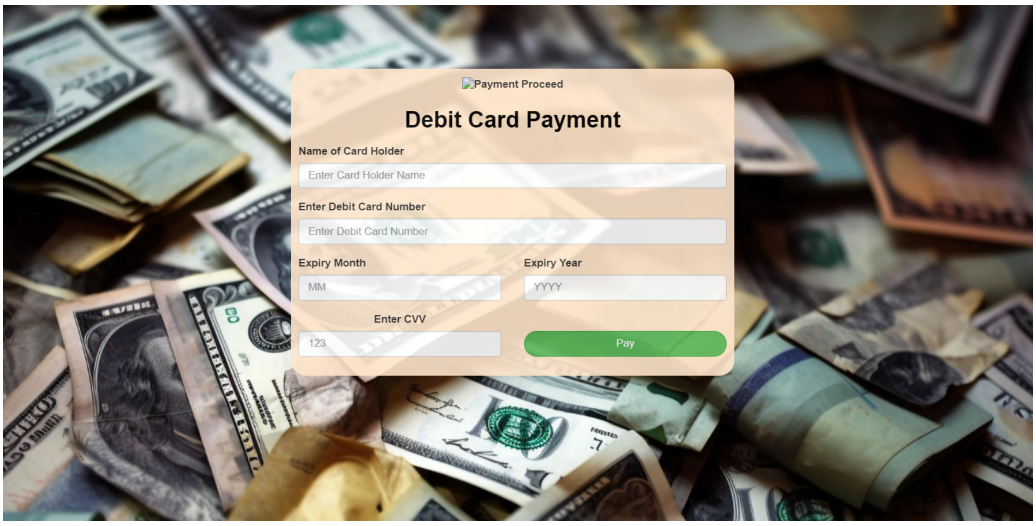

Figure 18: Payment Through Debit Card

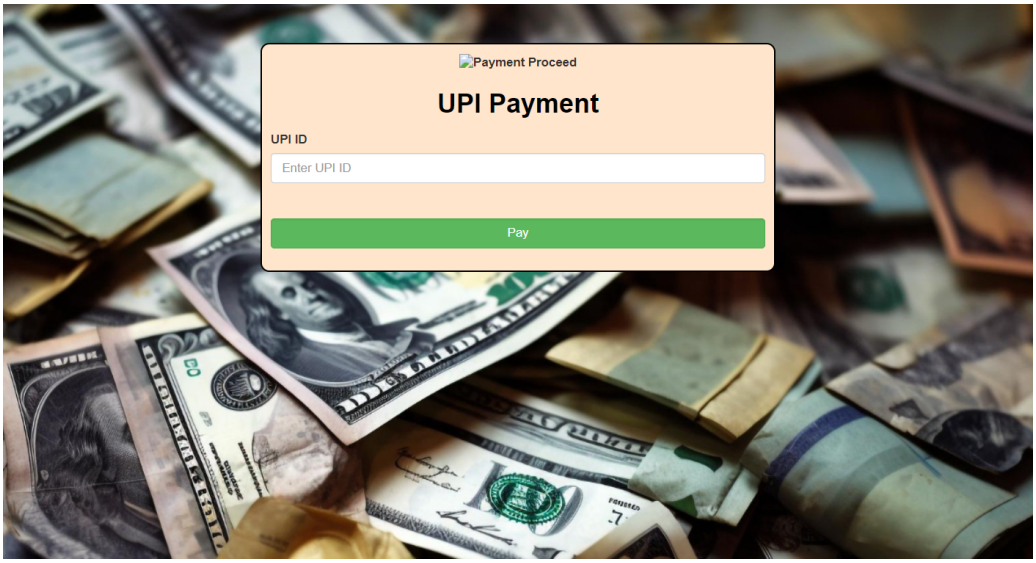

Figure 19: Payment Through UPI

Summary

This Online Retail Furniture Shopping Site project aimed to revolutionize the furniture shopping experience by developing a secure, user-friendly online platform. Leveraging a combination of front-end technologies such as HTML, CSS with Bootstrap, JavaScript, and React JS, alongside the robust back-end functionalities of Flask and MySQL database, the project focused on providing seamless navigation, product browsing, secure transactions, and efficient order management. The integration of MySQL as the database management system ensured reliable data storage, retrieval, and manipulation, enabling smooth user interactions and streamlined business operations.

Achievements

1. Implemented a robust authentication system to safeguard user accounts and ensure data privacy.
2. Employed strategies like lazy loading and code compression to enhance site speed and responsiveness.
3. Facilitated secure transactions through seamless integration with third-party payment gateways.
4. Developed an intuitive user interface with features like product filtering and social media sharing to improve user engagement.
5. Achieved secure data handling through MySQL's encryption capabilities, ensuring that sensitive user information such as passwords remained protected from unauthorized access.
6. Utilized MySQL's query optimization techniques to enhance database performance, resulting in faster response times for data retrieval and processing.
7. Implemented a robust inventory management system using MySQL, enabling real-time updates of product availability and minimizing the risk of stockouts or overselling.
8. Leveraged MySQL's scalability features to accommodate a growing user base and ensured database reliability through data backups and disaster recovery plans.

Lessons Learned

1. Prioritizing intuitive design and seamless navigation is crucial for enhancing user satisfaction and retention.
2. Implementing robust authentication mechanisms and encryption protocols is essential to protect user data from unauthorized access.
3. Clear communication and collaboration among team members are vital for successful project execution, ensuring alignment with project objectives and timelines..
4. Learned the importance of maintaining data integrity and implementing robust security measures within the database management system to protect sensitive information.
5. Gained insights into optimization strategies for database performance, including index optimization, query tuning, and efficient data modeling.
6. Understood the significance of designing a scalable database architecture to accommodate future growth and increasing data volumes.
7. Emphasized the importance of effective communication and collaboration among team members, particularly in coordinating database design, schema modifications, and query optimizations.
8. Acknowledged the need for continuous learning and staying updated with best practices in DBMS and MySQL to adapt to evolving business requirements and technological advancements.

Overall, the project provided valuable insights into the complexities of developing an online retail platform and underscored the importance of addressing user needs, ensuring data security, and embracing technological advancements to deliver a seamless shopping experience.

References

[1] S. Lim, M. Kim, H. Youn, The impact of website design on online furniture shopping behavior: A study of visual aesthetics and usability, Journal of Business Research 118 (2020) 377–387.
URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3991033

[2] H. Min, R. Xu, Online furniture retailing: A comprehensive review of recent trends, challenges, and opportunities, Journal of Retailing and Consumer Services 43 (2018) 275–284.
URL https://onlinelibrary.wiley.com/doi/full/10.1111/ijmr.12129

[3] R. Sharma, P. Mahajan, Impact of user-generated content on online shopping intention: A study on online furniture stores in india, Journal of Internet Commerce 19 (4) (2020) 305–327.
URL https://ideas.repec.org/a/ids/ijidsc/v11y2019i3p209-233.html

[4] A. S. Ons Al-Shamaileh, The effect of website interactivity and repeated exposure on user experience.
URL https://dl.acm.org/doi/abs/10.1145/2382176.2382178

[5] P. S. Cohan, Furniture.
URL https://link.springer.com/chapter/10.1007/978-1-4842-6519-2_6

[6] C. L. I. M. . C. P. Yamen Elsaid, Sarah Hanson, Driving profitability at the hungry dragon.
URL http://pubs.mumacasereview.org/2020/MCR-05-07-HungryDragon-Hanson-p1-26.pdf

[7] B. Schulz, Overstock goes 'beyond' with acquisition, relaunch.
URL https://go.gale.com/ps/i.do?id=GALE%7CA759459601&sid=googleScholar&v=2.1&it=r&linkaccess=abs&issn=07347456&p=AONE&sw=w&userGroupName=anon%7Ed99ffa27&aty=open-web-entry