**University of North Texas**

**Denton, Texas**

**Computer Science**

**CSCE 5350(Section -006) – Fundamentals of Database Systems**

# **Online Multi-Vendor Marketplace System**

**Team Members**

**1.Sahith Goud Athelli**

**2.Meghana Atthipalli**

**3.Pravallika Chava**

**4.Sekhar Reddy Kandula**

**5.Sai Priya Pandeti Vasantha Kumar**

**6.Manisha Singam**

1.Introduction

2. Goals and Scope

3. Updated ER-Diagram

4. ER Mapping

5. Normalization

6. SQL statements to create tables

7. Sample Data Screenshots

8.Functional Requirements and Queries

9.Steps for running the system

10. UI Screenshots

11. GitHub Link

**1. Introduction**

The Online Multi-vendor Marketplace System (OMVMS) resembles a cutting-edge shopping center platform online. It's where numerous merchants can settle in to sell their things, and individuals who need to purchase can undoubtedly buy various items. Sellers can easily deal with their products, and purchasers can make accounts, track their orders, and make secure buys. The framework keeps everything coordinated, from the amount of an item available and it also incorporates what customers think in reviews. There's even a way for vendors to publicize their items on the platform. The framework is intended to be easy to utilize, safe, and can develop as additional users participate. The diagram shows how everything, from vendors to products and customers, is associated with making the framework work smoothly.

**2. Goals and Scope**.  
The goal of the online Multi-vendor Marketplace System (OMVMS) is to give a user-friendly and safe platform where merchants can list and sell products, and clients can easily browse and make buys. The project intends to change the Internet shopping experience by encouraging a powerful commercial center that focuses on versatility and data integrity.

3. **Updated ER-diagram**

**4. ER mapping**1. Vendor Table:

- Attributes:

- `vendor\_id` (AutoField, Primary Key)

- `vendor\_name` (CharField)

- `vendor\_email` (EmailField)

- `password` (CharField)

2. Customer Table:

- Attributes:

- `customer\_id` (AutoField, Primary Key)

- `customer\_name` (CharField)

- `customer\_email` (EmailField)

- `password` (CharField)

3. Admin Table:

- Attributes:

- `admin\_id` (AutoField, Primary Key)

- `admin\_name` (CharField)

- `email` (EmailField)

- `password` (CharField)

4. Category Table:

- Attributes:

- `category\_id` (AutoField, Primary Key)

- `category\_name` (CharField)

- `description` (TextField)

5. SubCategory Table:

- Attributes:

- `subcategory\_id` (AutoField, Primary Key)

- `category` (ForeignKey to Category)

- `subcategory\_name` (CharField)

6. Product Table:

- Attributes:

- `product\_id` (AutoField, Primary Key)

- `vendor` (ForeignKey to Vendor)

- `category` (ForeignKey to Category)

- `subcategory` (ForeignKey to SubCategory)

- `product\_name` (CharField)

- `price` (DecimalField)

- `description` (TextField)

- `quantity` (IntegerField)

- `image` (ImageField)

7. Order Table:

- Attributes:

- `order\_id` (AutoField, Primary Key)

- `customer` (ForeignKey to Customer)

- `shipping\_address` (ForeignKey to ShippingAddress)

- `billing\_address` (ForeignKey to BillingAddress)

- `date` (DateField)

- `total\_amount` (DecimalField)

8. Transaction Table:

- Attributes:

- `transaction\_id` (AutoField, Primary Key)

- `order` (ForeignKey to Order)

- `payment\_method` (ForeignKey to PaymentMethod)

- `amount` (DecimalField)

- `date` (DateField)

- `status` (CharField)

9. Review Table:

- Attributes:

- `review\_id` (AutoField, Primary Key)

- `product` (ForeignKey to Product)

- `customer` (ForeignKey to Customer)

- `rating` (IntegerField)

- `comment` (TextField)

10. Advertisement Table:

- Attributes:

- `ad\_id` (AutoField, Primary Key)

- `vendor` (ForeignKey to Vendor)

- `ad\_slot` (ForeignKey to AdSlot)

- `ad\_content` (TextField)

- `duration` (IntegerField)

11. PaymentGateway Table

- Attributes:

- `gateway\_id` (AutoField, Primary Key)

- `gateway\_name` (CharField)

- `credentials` (TextField)

12. ShoppingCart Table:

- Attributes:

- `cart\_id` (AutoField, Primary Key)

- `customer` (ForeignKey to Customer)

- `product` (ForeignKey to Product)

- `quantity` (PositiveIntegerField)

13. Wishlist Table:

- Attributes:

- `wishlist\_id` (AutoField, Primary Key)

- `customer` (ForeignKey to Customer)

- `total\_items` (IntegerField)

14. PaymentMethod Table:

- Attributes:

- `payment\_method\_id` (AutoField, Primary Key)

- `method\_name` (CharField)

- `details` (TextField)

15. ShippingAddress Table:

- Attributes:

- `shipping\_address\_id` (AutoField, Primary Key)

- `customer` (ForeignKey to Customer)

- `address\_line` (CharField)

- `city` (CharField)

- `state` (CharField)

- `postal\_code` (CharField)

16.BillingAddress Table:

- Attributes:

- `billing\_address\_id` (AutoField, Primary Key)

- `customer` (ForeignKey to Customer)

- `address\_line` (CharField)

- `city` (CharField)

- `state` (CharField)

- `postal\_code` (CharField)

17. Return Table:

- Attributes:

- `return\_id` (AutoField, Primary Key)

- `order` (ForeignKey to Order)

- `reason` (TextField)

- `status` (CharField)

- `date` (DateField)

18. Refund Table:

- Attributes:

- `refund\_id` (AutoField, Primary Key)

- `return\_item` (ForeignKey to Return)

- `amount` (DecimalField)

- `date` (DateField)

19. AdSlot Table:

- Attributes:

- `ad\_slot\_id` (AutoField, Primary Key)

- `slot\_position` (CharField)

- `duration` (IntegerField)

- `price` (DecimalField)

20. DiscountPromotion Table:

- Attributes:

- `discount\_id` (AutoField, Primary Key)

- `vendor` (ForeignKey to Vendor)

- `discount\_code` (CharField)

- `amount` (DecimalField)

- `validity` (DateField)

**5.Normalization  
Functional Dependencies:**

* + **app\_admin**: No apparent issues.
  + **app\_adslot**: No apparent issues.
  + **app\_advertisement**: Dependent on **ad\_slot\_id** and **vendor\_id**.
  + **app\_billingaddress**: Dependent on **customer\_id**.
  + **app\_customer**: No apparent issues.
  + **app\_discountpromotion**: Dependent on **vendor\_id**.
  + **app\_order**: Dependent on **billing\_address\_id**, **customer\_id**, and **shipping\_address\_id**.
  + **app\_paymentgateway**: No apparent issues.
  + **app\_paymentmethod**: No apparent issues.
  + **app\_product**: Dependent on **category\_id**, **subcategory\_id**, and **vendor\_id**.
  + **app\_refund**: Dependent on **return\_item\_id**.
  + **app\_return**: Dependent on **order\_id**.
  + **app\_review**: Dependent on **customer\_id** and **product\_id**.
  + **app\_shippingaddress**: Dependent on **customer\_id**.
  + **app\_shoppingcart**: Dependent on **customer\_id** and **product\_id**.
  + **app\_subcategory**: Dependent on **category\_id**.
  + **app\_transaction**: Dependent on **order\_id** and **payment\_method\_id**.
  + **app\_vendor**: No apparent issues.
  + **app\_wishlist**: Dependent on **customer\_id**.

**Normalization Steps:**

1. **First Normal Form (1NF):** Ensure all atomic values.
2. **Second Normal Form (2NF):** Remove partial dependencies.
3. **Third Normal Form (3NF):** Eliminate transitive dependencies.

**Normalized Tables:**

1. Customer Information:

* **app\_customer**:
  + **customer\_id** (PK)
  + **customer\_name**
  + **customer\_email**
  + **password**

2. Address Information:

* **app\_address**:
  + **address\_id** (PK)
  + **address\_line**
  + **city**
  + **state**
  + **postal\_code**
  + **customer\_id** (FK to **app\_customer**)

3. Order Information:

* **app\_order**:
  + **order\_id** (PK)
  + **date**
  + **total\_amount**
  + **billing\_address\_id** (FK to **app\_address**)
  + **shipping\_address\_id** (FK to **app\_address**)
  + **customer\_id** (FK to **app\_customer**)

4. Product Information:

* **app\_product**:
  + **product\_id** (PK)
  + **product\_name**
  + **price**
  + **description**
  + **category\_id** (FK to **app\_category**)
  + **subcategory\_id** (FK to **app\_subcategory**)
  + **vendor\_id** (FK to **app\_vendor**)
  + **image**
  + **quantity**

5. Advertisement Information:

* **app\_advertisement**:
  + **ad\_id** (PK)
  + **ad\_content**
  + **duration**
  + **ad\_slot\_id** (FK to **app\_adslot**)
  + **vendor\_id** (FK to **app\_vendor**)

6. Payment Information:

* **app\_payment**:
  + **payment\_id** (PK)
  + **amount**
  + **date**
  + **order\_id** (FK to **app\_order**)
  + **payment\_method\_id** (FK to **app\_paymentmethod**)

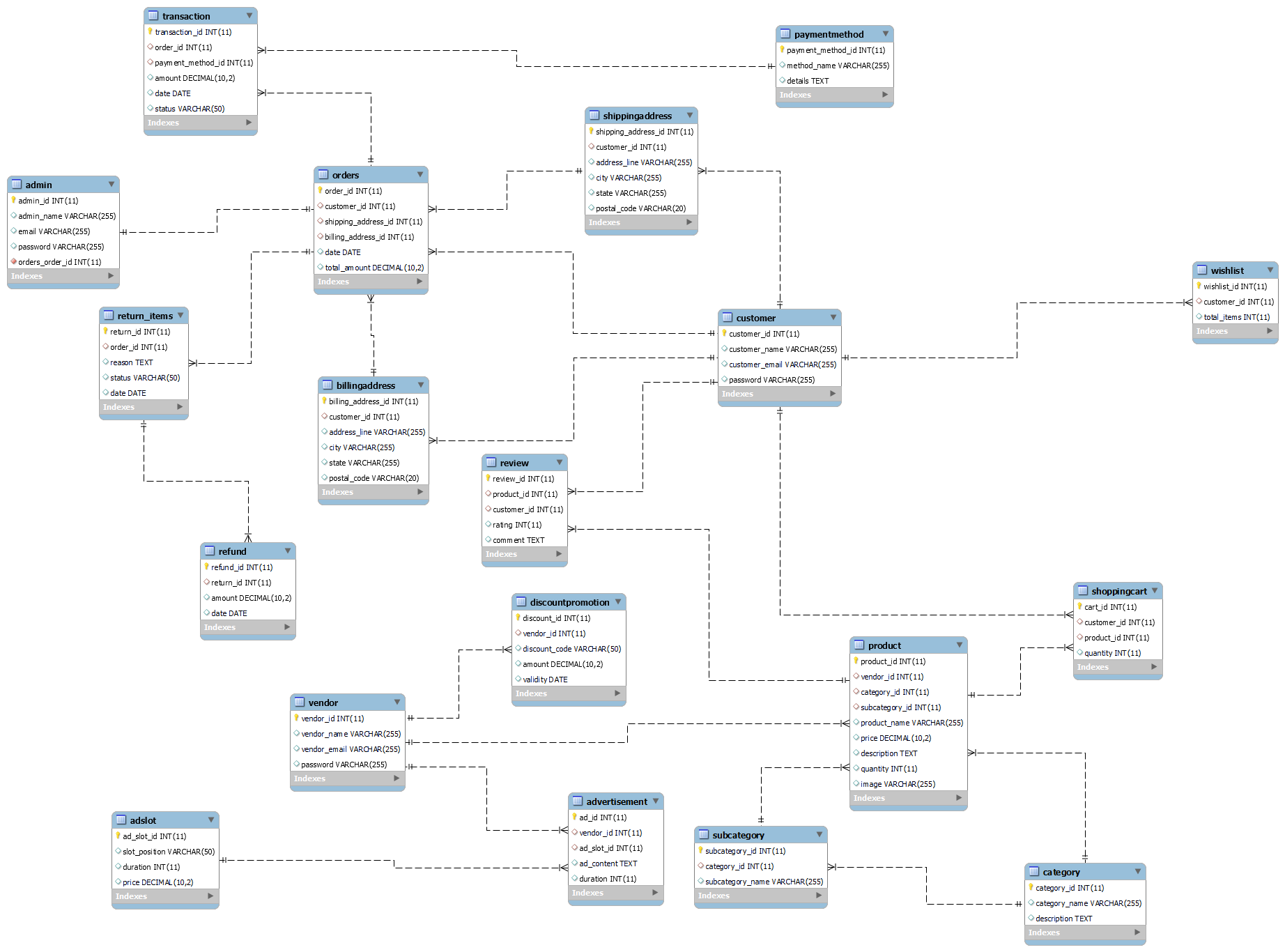
7. Review Information:

* **app\_review**:
  + **review\_id** (PK)
  + **rating**
  + **comment**
  + **customer\_id** (FK to **app\_customer**)
  + **product\_id** (FK to **app\_product**)

8. Wishlist Information:

* **app\_wishlist**:
  + **wishlist\_id** (PK)
  + **total\_items**
  + **customer\_id** (FK to **app\_customer**)

These tables aim to minimize redundancy and maintain data integrity by splitting data logically based on functional dependencies.



**6.SQL statements to create table**-- Create Vendor Table

CREATE TABLE vendor (

vendor\_id SERIAL PRIMARY KEY,

vendor\_name VARCHAR(255),

vendor\_email VARCHAR(255) UNIQUE,

password VARCHAR(255)

);

-- Create Customer Table

CREATE TABLE customer (

customer\_id SERIAL PRIMARY KEY,

customer\_name VARCHAR(255),

customer\_email VARCHAR(255) UNIQUE,

password VARCHAR(255)

);

-- Create Admin Table

CREATE TABLE admin (

admin\_id SERIAL PRIMARY KEY,

admin\_name VARCHAR(255),

email VARCHAR(255) UNIQUE,

password VARCHAR(255)

);

-- Create Category Table

CREATE TABLE category (

category\_id SERIAL PRIMARY KEY,

category\_name VARCHAR(255),

description TEXT

);

-- Create SubCategory Table

CREATE TABLE subcategory (

subcategory\_id SERIAL PRIMARY KEY,

category\_id INTEGER REFERENCES category(category\_id),

subcategory\_name VARCHAR(255)

);

-- Create Product Table

CREATE TABLE product (

product\_id SERIAL PRIMARY KEY,

vendor\_id INTEGER REFERENCES vendor(vendor\_id),

category\_id INTEGER REFERENCES category(category\_id),

subcategory\_id INTEGER REFERENCES subcategory(subcategory\_id),

product\_name VARCHAR(255),

price DECIMAL(10, 2),

description TEXT,

quantity INTEGER,

image VARCHAR(255)

);

-- Create Order Table

CREATE TABLE "order" (

order\_id SERIAL PRIMARY KEY,

customer\_id INTEGER REFERENCES customer(customer\_id),

shipping\_address\_id INTEGER REFERENCES shippingaddress(shipping\_address\_id),

billing\_address\_id INTEGER REFERENCES billingaddress(billing\_address\_id),

date DATE,

total\_amount DECIMAL(10, 2)

);

-- Create Transaction Table

CREATE TABLE transaction (

transaction\_id SERIAL PRIMARY KEY,

order\_id INTEGER REFERENCES "order"(order\_id),

payment\_method\_id INTEGER REFERENCES paymentmethod(payment\_method\_id),

amount DECIMAL(10, 2),

date DATE,

status VARCHAR(50)

);

-- Create Review Table

CREATE TABLE review (

review\_id SERIAL PRIMARY KEY,

product\_id INTEGER REFERENCES product(product\_id),

customer\_id INTEGER REFERENCES customer(customer\_id),

rating INTEGER,

comment TEXT

);

-- Create Advertisement Table

CREATE TABLE advertisement (

ad\_id SERIAL PRIMARY KEY,

vendor\_id INTEGER REFERENCES vendor(vendor\_id),

ad\_slot\_id INTEGER REFERENCES adslot(ad\_slot\_id),

ad\_content TEXT,

duration INTEGER

);

-- Create PaymentGateway Table

CREATE TABLE paymentgateway (

gateway\_id SERIAL PRIMARY KEY,

gateway\_name VARCHAR(255),

credentials TEXT

);

-- Create ShoppingCart Table

CREATE TABLE shoppingcart (

cart\_id SERIAL PRIMARY KEY,

customer\_id INTEGER REFERENCES customer(customer\_id),

product\_id INTEGER REFERENCES product(product\_id),

quantity INTEGER

);

-- Create Wishlist Table

CREATE TABLE wishlist (

wishlist\_id SERIAL PRIMARY KEY,

customer\_id INTEGER REFERENCES customer(customer\_id),

total\_items INTEGER

);

-- Create PaymentMethod Table

CREATE TABLE paymentmethod (

payment\_method\_id SERIAL PRIMARY KEY,

method\_name VARCHAR(255),

details TEXT

);

-- Create ShippingAddress Table

CREATE TABLE shippingaddress (

shipping\_address\_id SERIAL PRIMARY KEY,

customer\_id INTEGER REFERENCES customer(customer\_id),

address\_line VARCHAR(255),

city VARCHAR(255),

state VARCHAR(255),

postal\_code VARCHAR(20)

);

-- Create BillingAddress Table

CREATE TABLE billingaddress (

billing\_address\_id SERIAL PRIMARY KEY,

customer\_id INTEGER REFERENCES customer(customer\_id),

address\_line VARCHAR(255),

city VARCHAR(255),

state VARCHAR(255),

postal\_code VARCHAR(20)

);

-- Create Return Table

CREATE TABLE "return" (

return\_id SERIAL PRIMARY KEY,

order\_id INTEGER REFERENCES "order"(order\_id),

reason TEXT,

status VARCHAR(50),

date DATE

);

-- Create Refund Table

CREATE TABLE refund (

refund\_id SERIAL PRIMARY KEY,

return\_item\_id INTEGER REFERENCES "return"(return\_id),

amount DECIMAL(10, 2),

date DATE

);

-- Create AdSlot Table

CREATE TABLE adslot (

ad\_slot\_id SERIAL PRIMARY KEY,

slot\_position VARCHAR(50),

duration INTEGER,

price DECIMAL(10, 2)

);

-- Create DiscountPromotion Table

CREATE TABLE discountpromotion (

discount\_id SERIAL PRIMARY KEY,

vendor\_id INTEGER REFERENCES vendor(vendor\_id),

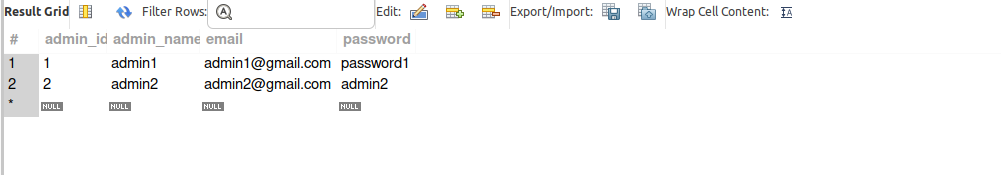
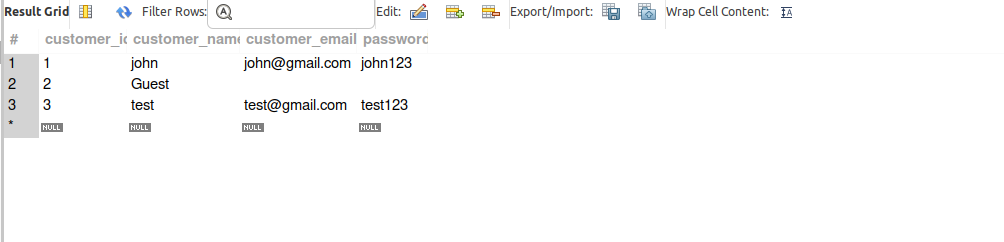
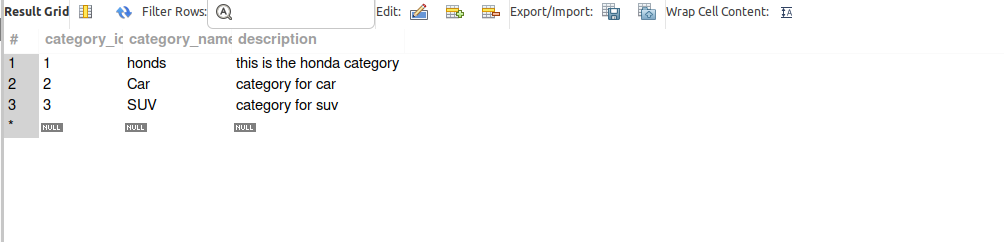
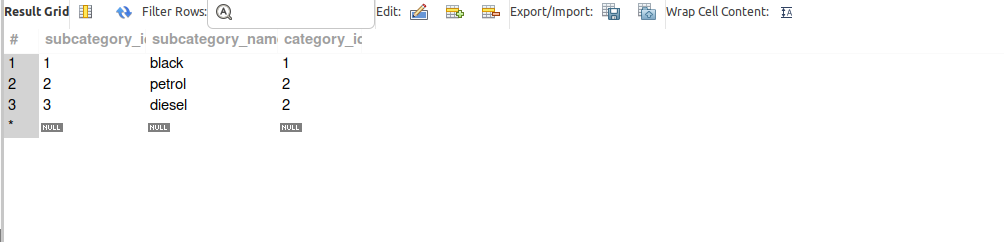
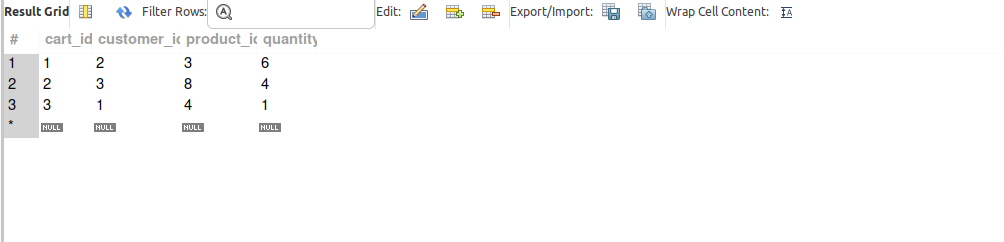
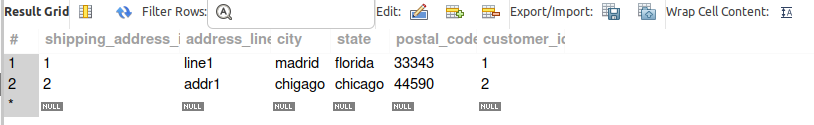
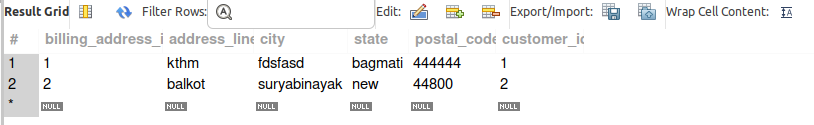
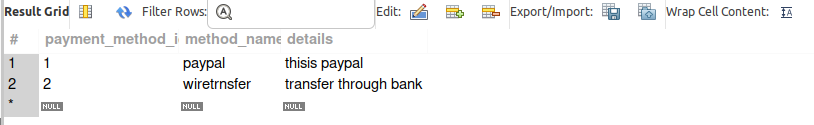
discount\_code VARCHAR(50),

amount DECIMAL(10, 2),

validity DATE

);

**7.Sample data Screenshots**

1. **Admin  
   **
2. **Products  
   **
3. **Customers  
   **
4. **Categories  
   **
5. **subCategories  
   **
6. **Cart  
   **
7. **Reviews  
   **
8. **Shipping address  
   **
9. **BIlling address  
   **
10. **Payment gateways  
    **

**8. Functional Requirements and Queries  
a.Adding product**Admins can add the product through the system dashboard with details like category, subcategory, product name, price, quantity and image.  
  
Query:INSERT INTO product (vendor\_id, category\_id, subcategory\_id, product\_name, price, description, quantity, image)

VALUES (1, 1, 1, 'Example Product', 29.99, 'This is an example product description.', 50, 'example\_product.jpg');

**b.Registering customers** Admin can add the customers who can sell and purchase the products from the system

Query:

INSERT INTO customer (customer\_name, customer\_email, password)

VALUES ('John Doe', 'john.doe@example.com', 'customer password');

**c. Adding the product to the cart**The system provides the functionality to add the product they want to buy to the shopping cart.

query:

INSERT INTO shoppingcart (customer\_id, product\_id, quantity)

VALUES (1, 1, 2);

**d.Vendor Registration**The system is capable of adding different vendors which sell different types of products.

Query:  
INSERT INTO vendor (vendor\_name, vendor\_email, password)

VALUES ('NewVendor', 'newvendor@email.com', 'newvendorpassword');

**9. Steps for running the system**Step 1: Download the Django project code to your local machine.  
Step 2: Make sure that Python and pip are installed on your machine.  
Step 3: Install the dependencies by using the command “**pip install -r requirements.txt”**.

Step 4: Open the project on any IDE and configure the database for MySQL in settings.py file.

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'your\_database\_name',

'USER': 'your\_database\_user',

'PASSWORD': 'your\_database\_password',

'HOST': 'localhost',

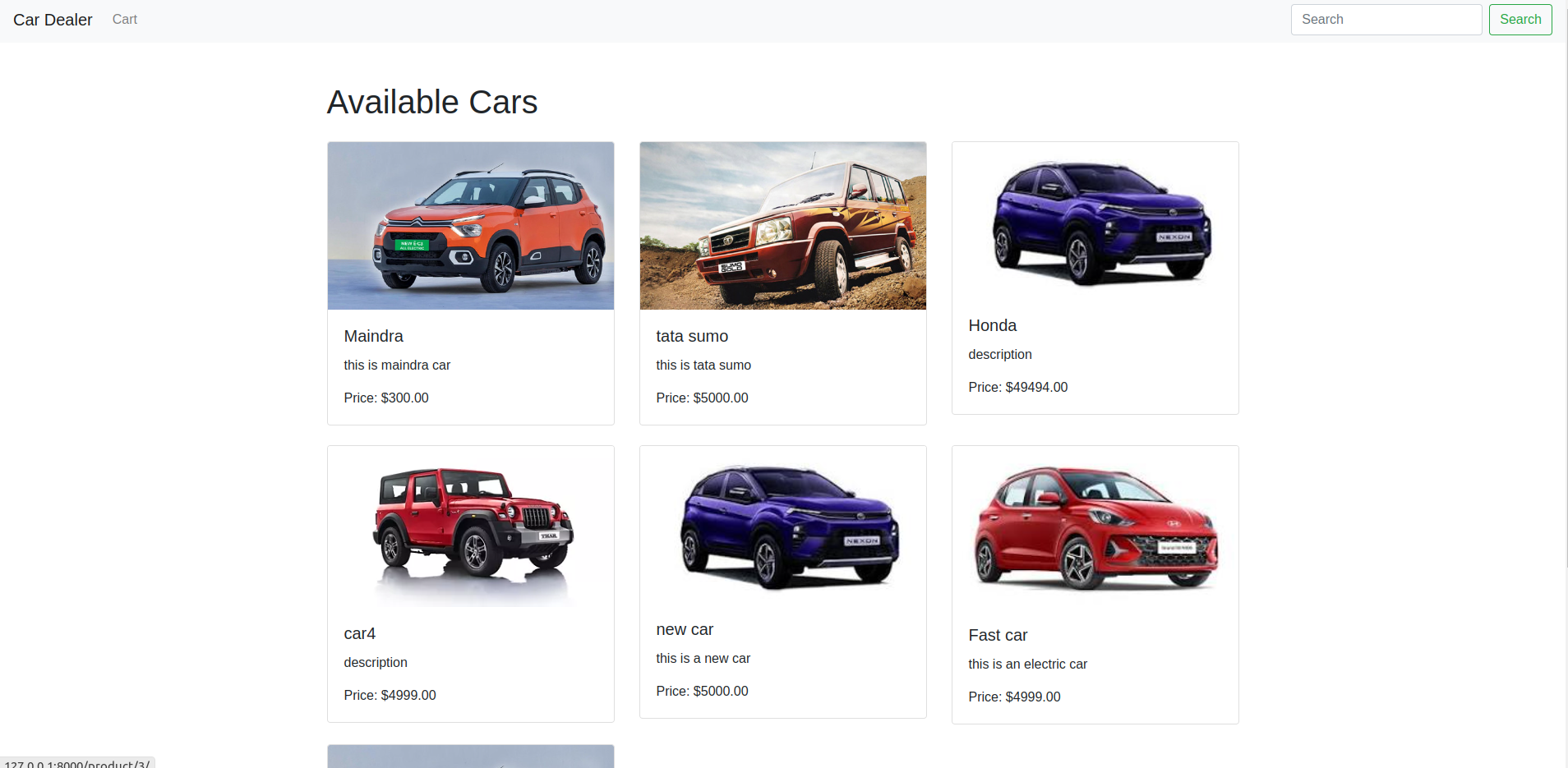
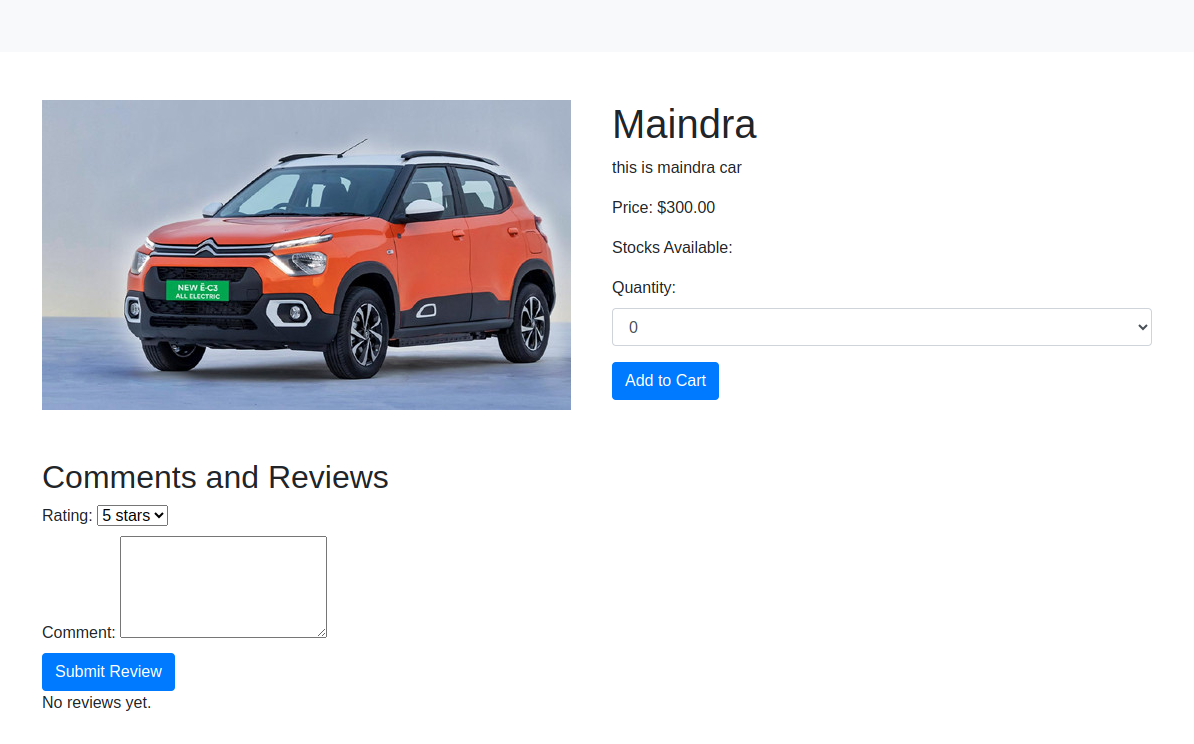
'PORT': '3306',

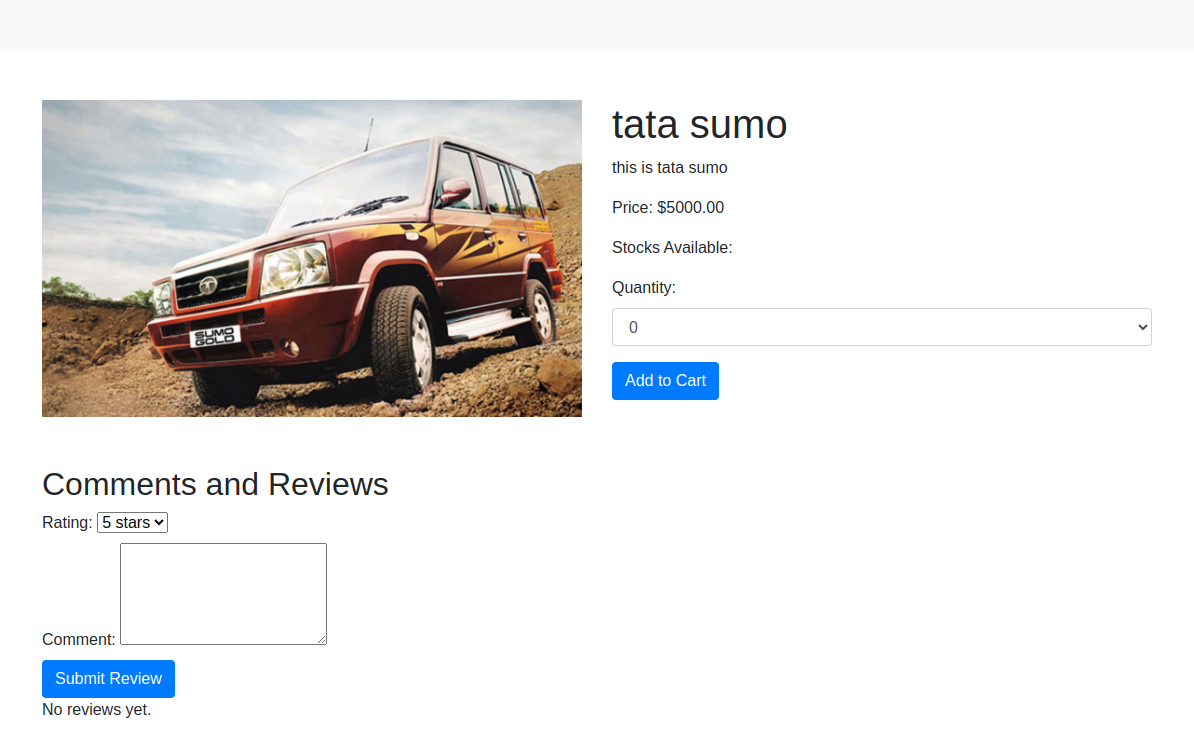
}

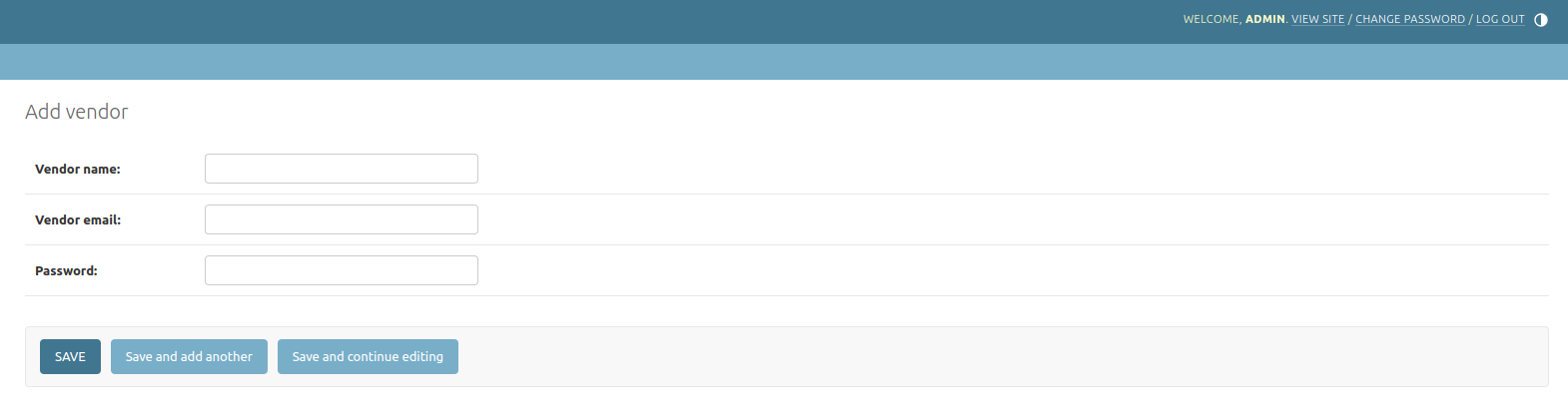
}

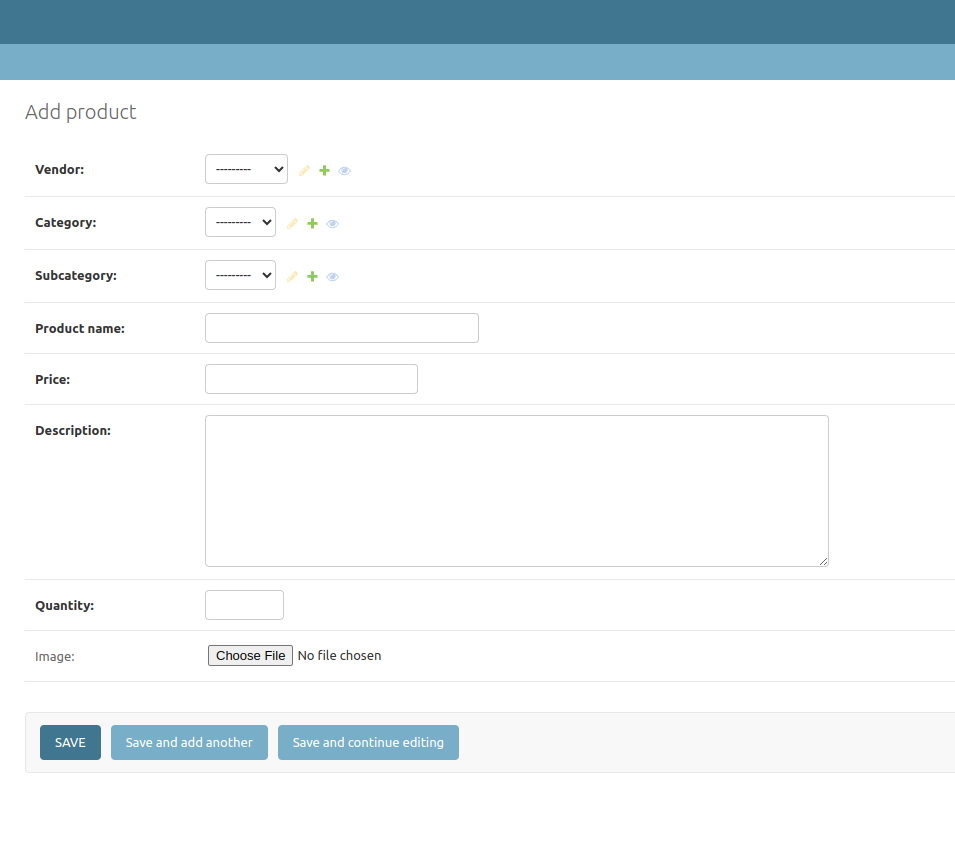
Step 5: Create a superuser/admin by using the command “**python manage.py createsuperuser”**Step 6: Migrate the database using the command **“python manage.py migrate”**

Step 7: Run the development server using the command **“python manage.py run server”**Step 8: Access the system through the link **“**[**http://127.0.0.1:8000/**](http://127.0.0.1:8000/)**”.**Step 9: Access the admin through the link “[**http://127.0.0.1:8000/admin**](http://127.0.0.1:8000/admin)**”**

**10.UI Screenshots  
  
  
  
**

****

****

****

11. GitHub Link

https://github.com/Sekhar0799/FDB-Project---8