CSE370 : Database Systems
Project Report
Project Title : Project Title Here

Group No : 10, CSE370 Lab Section : 05, Spring 2024		
ID	**Name**	**Contribution**
24141183	Razin Sufian	Backend and Frontend

Table of Contents

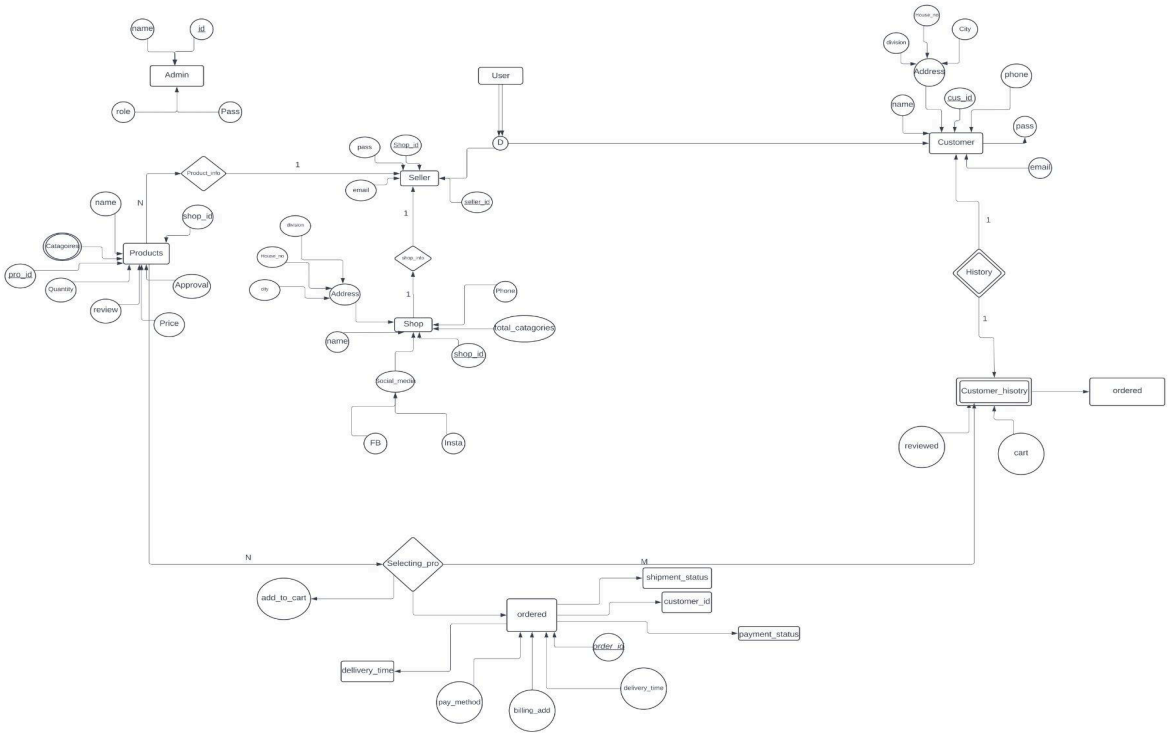
Introduction

"I'm creating an online marketplace platform where sellers can easily set up their own shops to sell their products. It's like giving them their own storefront in a bustling virtual mall. Customers can visit these shops, browse through the products, and purchase what they like. I want to make it convenient for both sellers and customers to interact and transact seamlessly, offering features like secure payments and order management. Essentially, I'm bringing together sellers and buyers in a vibrant online community, making shopping and selling online an enjoyable experience for everyone involved."

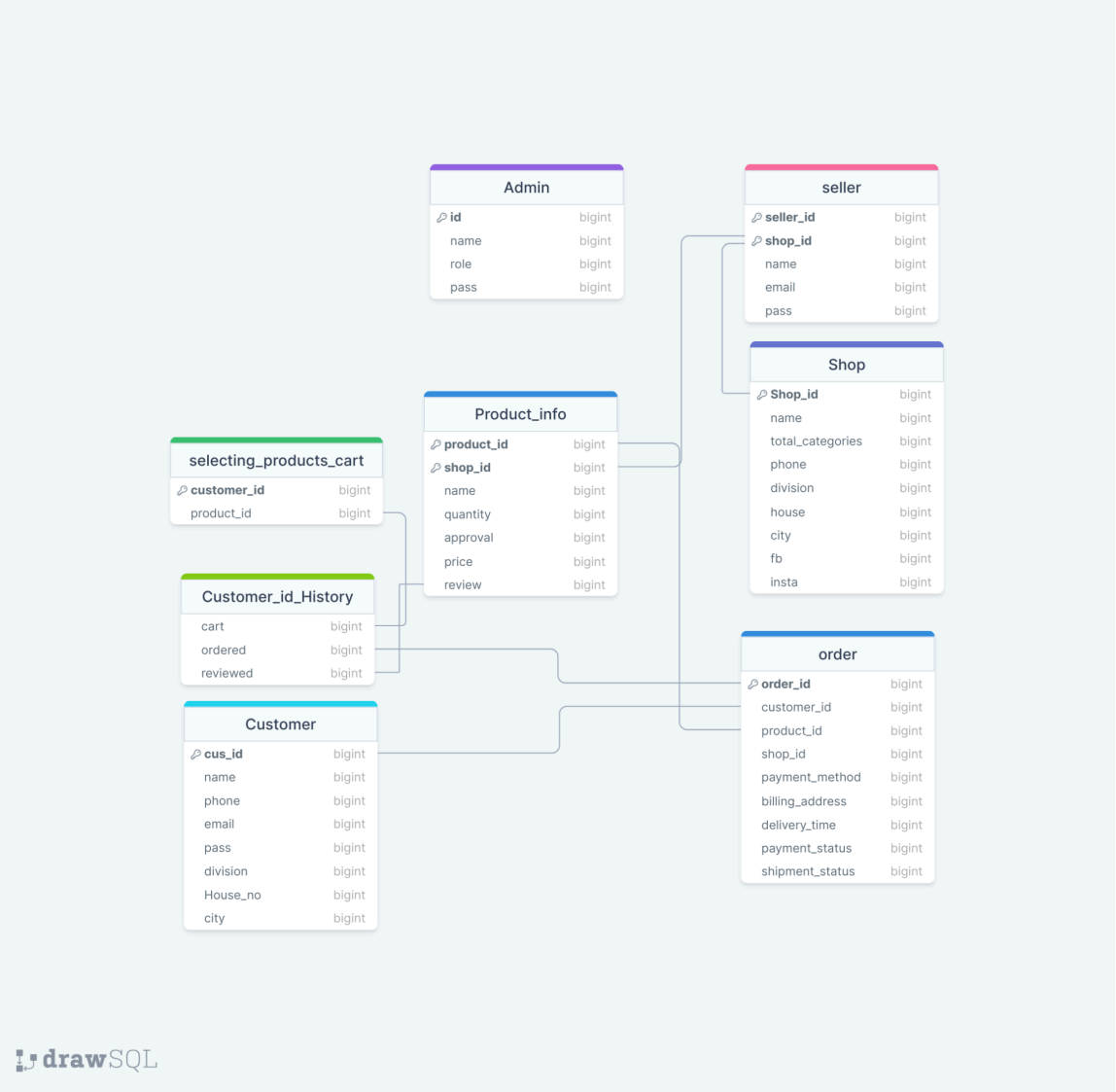
Project Features

User and profile : Create and modify user account
Seller and profile: Create and modify seller account
seller page management: can modify products and deliveries data
Product listing page: Customer point of view on search result
Specific shop page : Show all public data of the individual shop
add to cart/ buy / check out page

ER/EER Diagram



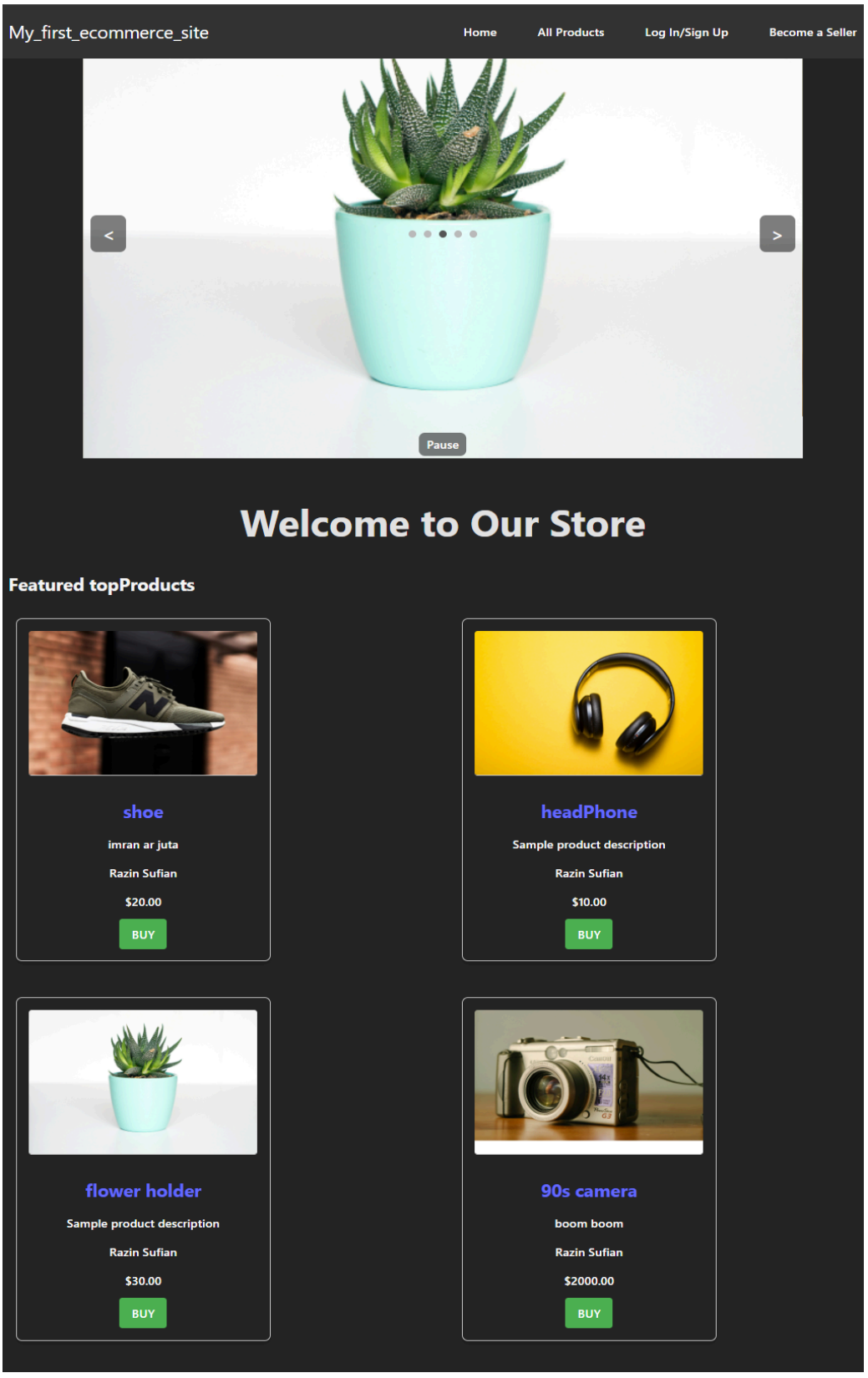
Schema Diagram



Frontend Development

"In my frontend development, I've organized my code into various directories to maintain a clean and modular architecture.

API: This is where I handle communication with the backend server. I've written functions to fetch data and interact with server endpoints.
AppStore: Here, I manage the application state using Redux or a similar library. I define reducers, actions, and slices to control how data is stored and updated throughout the app.
Authentication: In this directory, I implement user authentication features like login and signup pages. I use React components to create user-friendly forms and connect them to authentication APIs.
Components: This is where I keep reusable React components used across the app. It includes basic UI elements like buttons and input fields, as well as more complex components like product cards.
Features: Here, I organize code based on specific app features like user management and product listing. Each feature has its own directory containing related components, pages, and logic.
Layout: This directory contains layout components defining the overall structure and styling of the app. I create layouts for main pages like the home page and product listing page to ensure consistency in UI design.
Pages: Here, I define individual pages of the app such as the home page and user profile page. Each page is implemented as a React component with its own logic and styling.
Seller: This is where I focus on features tailored specifically for sellers, like shop management and product editing. I create separate components and pages to provide sellers with a dedicated experience.
Shared: In this directory, I keep shared utility functions, constants, and styles used across the app. It helps in maintaining consistency and promoting code reusability.


My_first_ecommerce_site
Home
All Products
Log In/Sign Up
Become a Seller
<
>
Pause
Welcome to Our Store
Featured topProducts
shoe
imran ar juta
Razin Sufian
$20.00
BUY
headPhone
Sample product description
Razin Sufian
$10.00
BUY
flower holder
Sample product description
Razin Sufian
$30.00
BUY
90s camera
boom boom
Razin Sufian
$2000.00
BUY

My_first_ecommerce_site

Home | All Products | Log In/Sign Up | Become a Seller

All Products

headPhone

Razin Sufian

$10.00

BUY

shoe

Razin Sufian

$20.00

BUY

flower holder

Razin Sufian

$30.00

BUY

car

Razin Sufian

$40.00

BUY

smart watch

Razin Sufian

$50.00

BUY

black car

Razin Sufian

$1234.00

BUY

90s camera

Razin Sufian

$2000.00

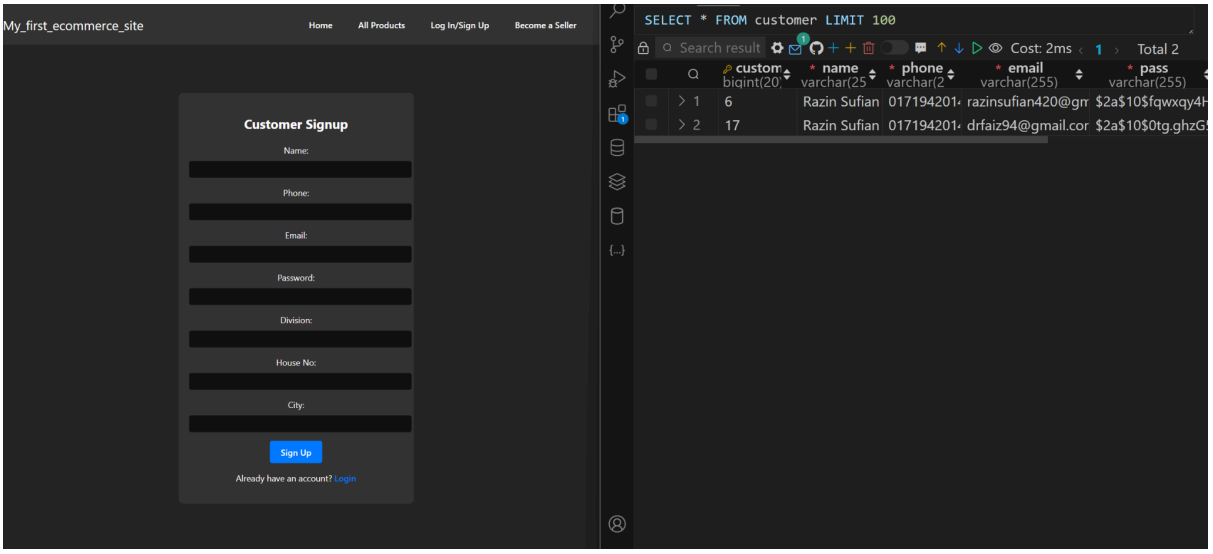
BUY

watch

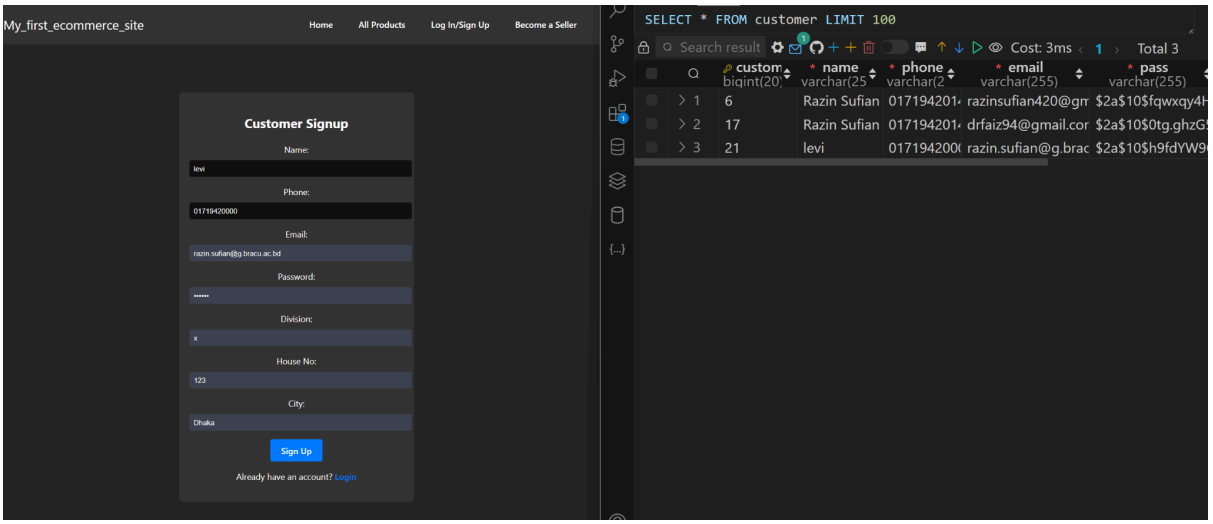
messi

$2000.00

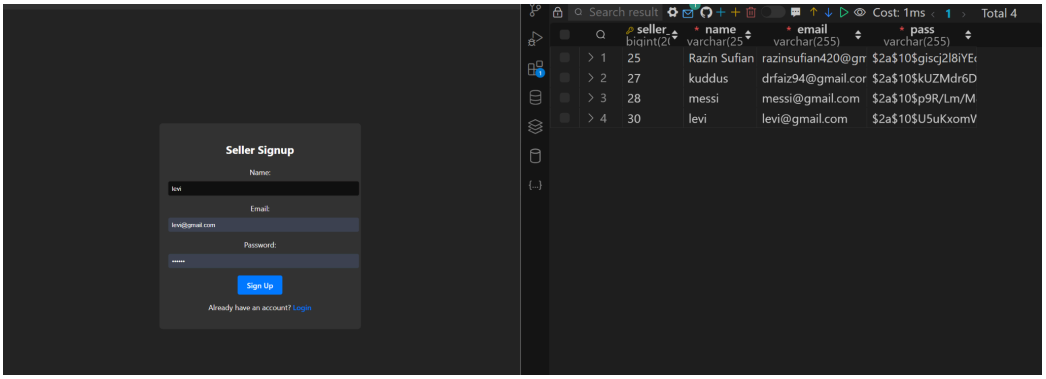
BUY

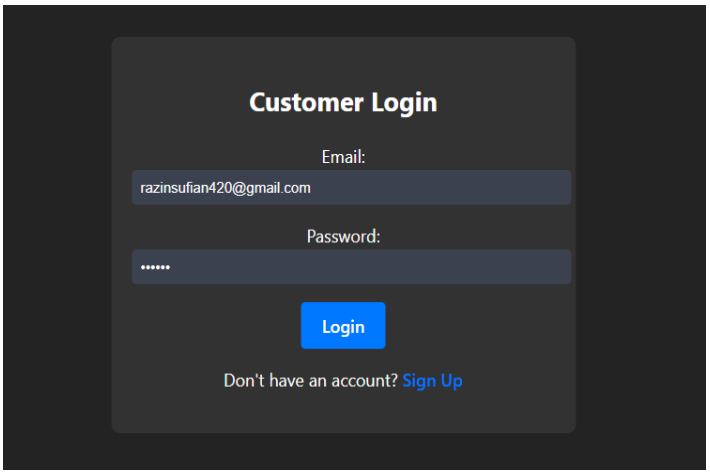


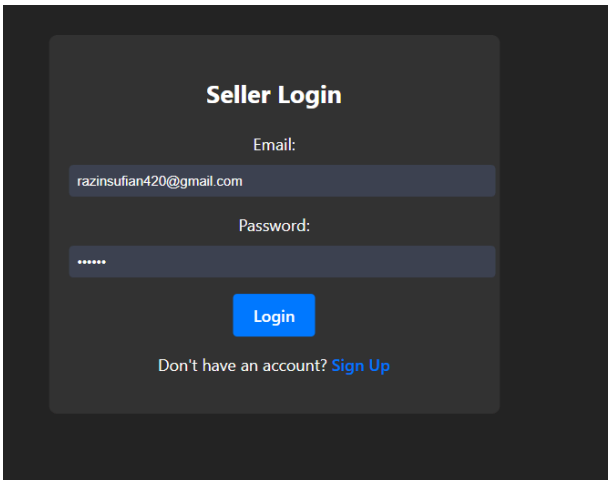
As we can see currently there are only 2 customers signed in , now if i sing up a new customer, the database will be update:



Same thing will happen it the seller sign up page:

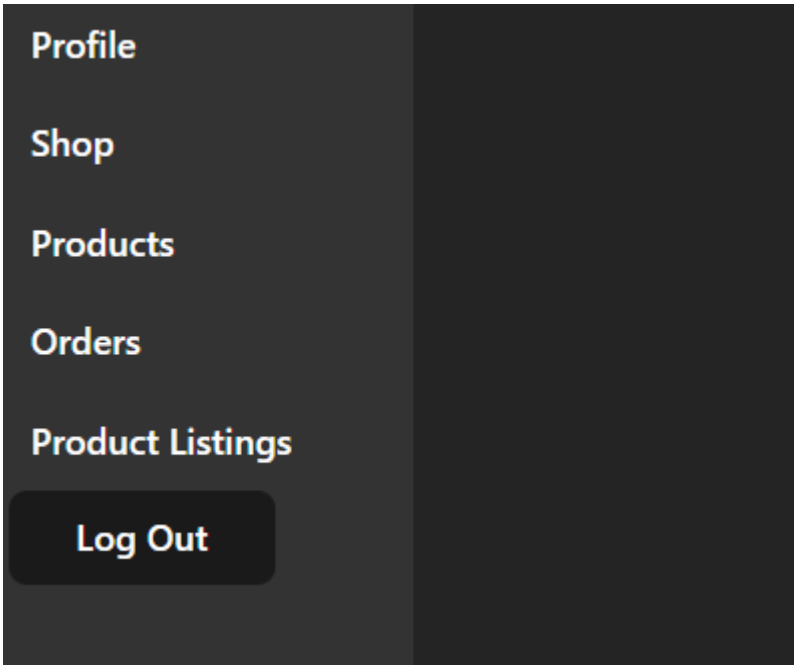




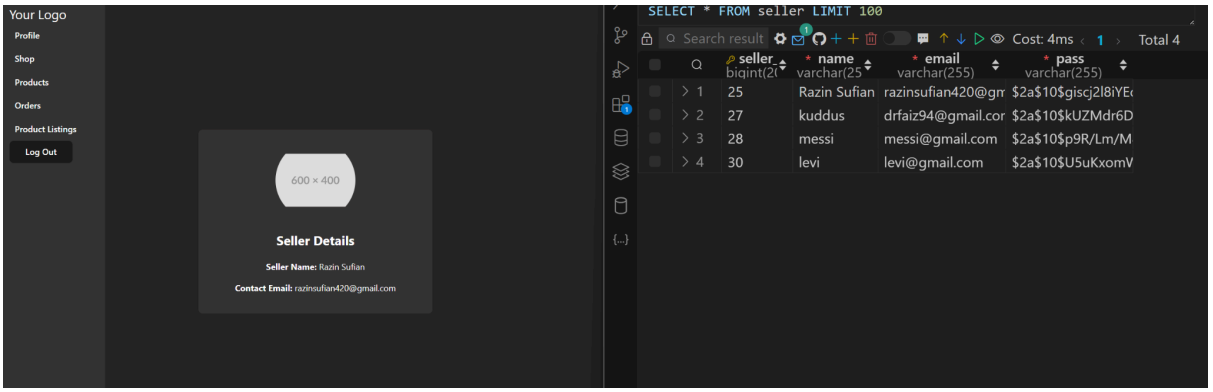


Seller and Customer login page:

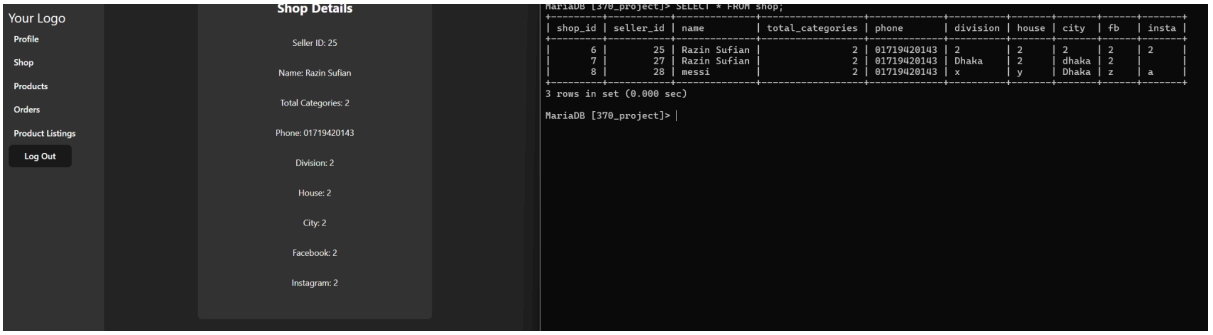
Seller login view:



Seller profile:

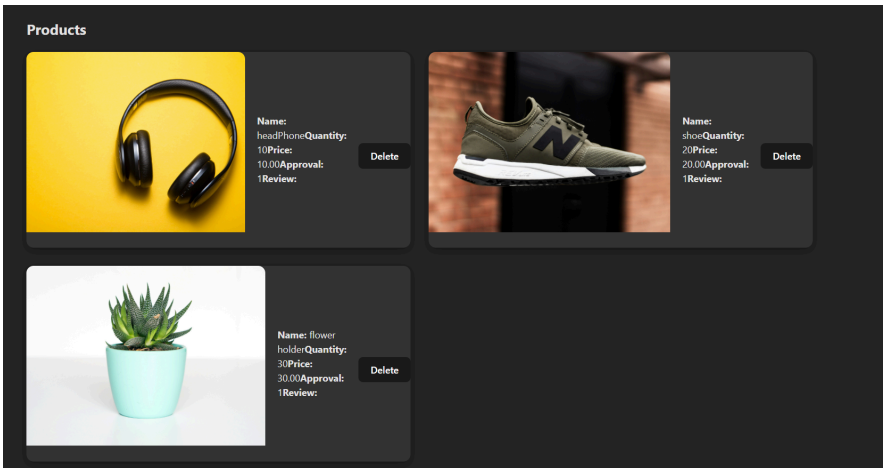


SHOP DATA:



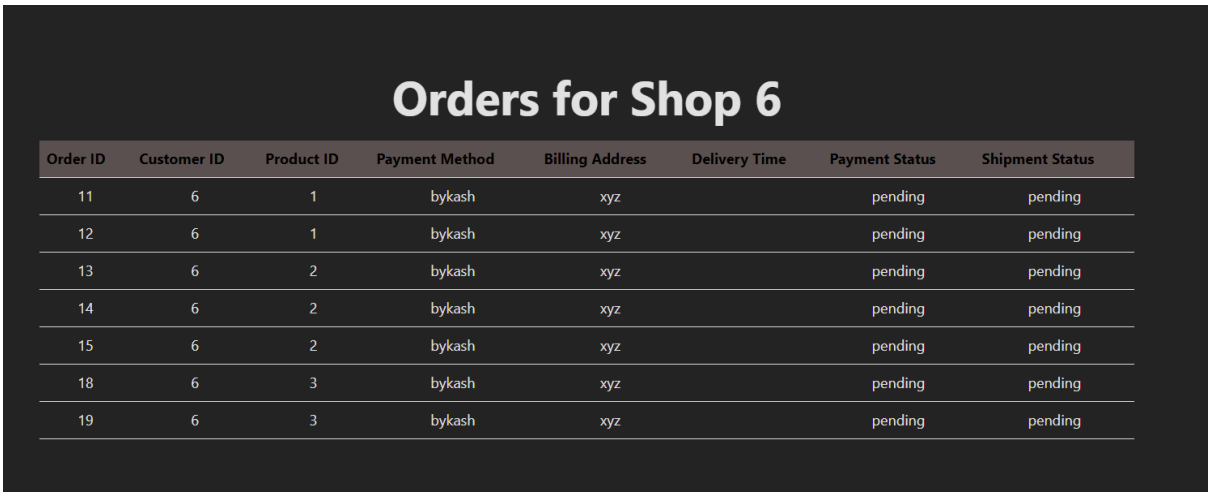
Products Page (seller View):

```
MariaDB [370_project]> SELECT * FROM product_info;
+------------+---------+---------------+----------+----------+---------+--------+-----------------------------+---------------------------+
| product_id | shop_id | name          | quantity | approval | price   | review | image_url                   | product_description       |
+------------+---------+---------------+----------+----------+---------+--------+-----------------------------+---------------------------+
|          1 |       6 | headPhone     |       10 |        1 |   10.00 | NULL   | https://i.ibb.co/vqjSZ9K/1.png | Sample product description |
|          2 |       6 | shoe          |       20 |        1 |   20.00 | NULL   | https://i.ibb.co/RcVmkWW/2.png | imran ar juta             |
|          3 |       6 | flower holder |       30 |        1 |   30.00 | NULL   | https://i.ibb.co/N3Q4mD3/3.png | Sample product description |
|          4 |       7 | car           |       40 |        1 |   40.00 | NULL   | https://i.ibb.co/VqwG9C0/4.png | Sample product description |
|          5 |       7 | smart watch   |       50 |        1 |   50.00 | NULL   | https://i.ibb.co/sWsPcys/6.png | Sample product description |
|          8 |       7 | black car     |        1 |        0 | 1234.00 | das    | https://i.ibb.co/dj6h0Df/7.png | posh car                  |
|         12 |       7 | 90s camera    |        1 |        0 | 2000.00 | 5      | https://i.ibb.co/gMmx1N3/8.png | boom boom                 |
|         13 |       8 | watch         |       10 |        0 | 2000.00 | 5      | https://i.ibb.co/XFZGv9q/5.png | lol                       |
+------------+---------+---------------+----------+----------+---------+--------+-----------------------------+---------------------------+
8 rows in set (0.000 sec)
```



Orders Page:

```
MariaDB [370_project]> SELECT * FROM `order` WHERE shop_id = 6;
+----------+-------------+------------+---------+----------------+-----------------+---------------------+----------------+-----------------+
| order_id | customer_id | product_id | shop_id | payment_method | billing_address | delivery_time       | payment_status | shipment_status |
+----------+-------------+------------+---------+----------------+-----------------+---------------------+----------------+-----------------+
|       11 |           6 |          1 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
|       12 |           6 |          1 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
|       13 |           6 |          2 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
|       14 |           6 |          2 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
|       15 |           6 |          2 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
|       18 |           6 |          3 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
|       19 |           6 |          3 |       6 | bykash         | xyz             | 0000-00-00 00:00:00 | pending        | pending         |
+----------+-------------+------------+---------+----------------+-----------------+---------------------+----------------+-----------------+
7 rows in set (0.000 sec)
```

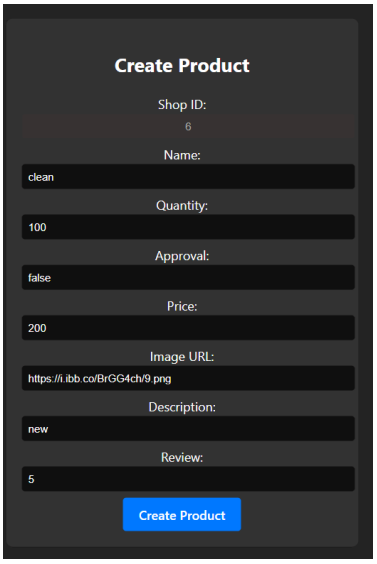


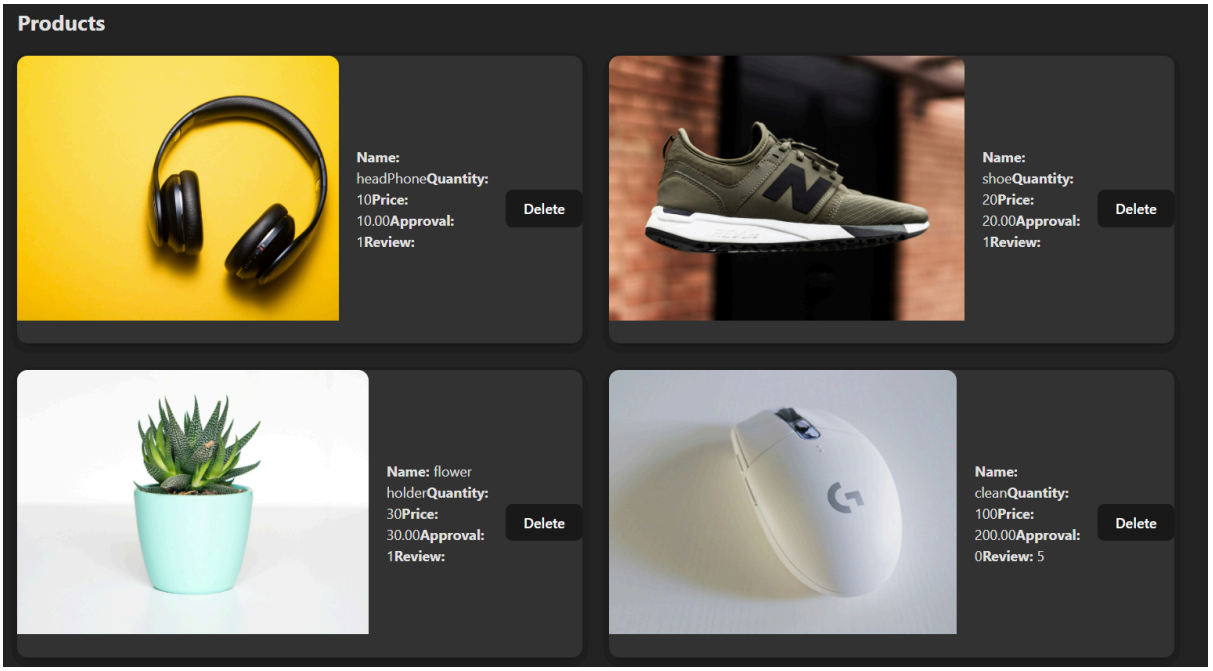
Add products:

Before creating product for shopId 6:

```
MariaDB [370_project]> SELECT * FROM product_info WHERE shop_id = 6;
+------------+---------+---------------+----------+----------+-------+--------+-------------------------------+---------------------------+
| product_id | shop_id | name          | quantity | approval | price | review | image_url                     | product_description       |
+------------+---------+---------------+----------+----------+-------+--------+-------------------------------+---------------------------+
|          1 |       6 | headPhone     |       10 |        1 | 10.00 | NULL   | https://i.ibb.co/vqjSZ9K/1.png | Sample product description |
|          2 |       6 | shoe          |       20 |        1 | 20.00 | NULL   | https://i.ibb.co/RcVmkWW/2.png | imran ar juta             |
|          3 |       6 | flower holder |       30 |        1 | 30.00 | NULL   | https://i.ibb.co/N3Q4mD3/3.png | Sample product description |
+------------+---------+---------------+----------+----------+-------+--------+-------------------------------+---------------------------+
3 rows in set (0.000 sec)
```

After entering new product:





Update in the database:

```
MariaDB [370_project]> select * from product_info where shop_id = 6;
+------------+---------+---------------+----------+----------+--------+--------+-------------------------------+---------------------------+
| product_id | shop_id | name          | quantity | approval | price  | review | image_url                     | product_description       |
+------------+---------+---------------+----------+----------+--------+--------+-------------------------------+---------------------------+
|          1 |       6 | headPhone     |       10 |        1 |  10.00 | NULL   | https://i.ibb.co/vqjSZ9K/1.png | Sample product description |
|          2 |       6 | shoe          |       20 |        1 |  20.00 | NULL   | https://i.ibb.co/RcVmkWW/2.png | imran ar juta             |
|          3 |       6 | flower holder |       30 |        1 |  30.00 | NULL   | https://i.ibb.co/N3Q4mD3/3.png | Sample product description |
|         14 |       6 | clean         |      100 |        0 | 200.00 | 5      | https://i.ibb.co/BrGG4ch/9.png | new                       |
+------------+---------+---------------+----------+----------+--------+--------+-------------------------------+---------------------------+
4 rows in set (0.000 sec)
```

It can also be deleted from the DataBase by clicking delete button.

So this is basically how my fronted is connecting with backend and making the changes in the database:

Backend Development

My backend is divided into app, controllers, middleware, models and routers:

app: This is the core directory where I've organized all the backend code for my project. It contains subdirectories for controllers, middleware, models, routers, and tests.
controllers: Here, I define functions to handle various HTTP requests and interact with the data models. Each controller corresponds to a specific route or endpoint in the API.
middleware: In this directory, I store middleware functions that intercept and process incoming requests before they reach the route handlers. Middleware functions can perform tasks like authentication, validation, or logging.
models: This directory contains data models that define the structure and behavior of our application's data. Each model represents a specific entity in our database, and it includes methods for querying and manipulating data.
router: Here, I define route handlers using Express.js routers. Each router corresponds to a group of related routes and is responsible for mapping incoming requests to the appropriate controller functions.
tests: This directory houses test scripts written to ensure the correctness and reliability of our backend code. I write unit tests and integration tests to validate the functionality of controllers, middleware, and models.

Here is the my database schema:--

```
-- Admin Table
CREATE TABLE IF NOT EXISTS `admin` (
    `id` BIGINT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL,
    `role` VARCHAR(255) NOT NULL,
    `pass` VARCHAR(255) NOT NULL, -- This should
be hashed
    PRIMARY KEY (`id`)
);

-- Customer Table
CREATE TABLE IF NOT EXISTS `customer` (
```

```
    `customer_id` BIGINT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL,
    `phone` VARCHAR(20) NOT NULL,
    `email` VARCHAR(255) NOT NULL UNIQUE,
    `pass` VARCHAR(255) NOT NULL,
    `division` VARCHAR(255) NOT NULL,
    `house_no` VARCHAR(255) NOT NULL,
    `city` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`customer_id`)
);

-- Seller Table
CREATE TABLE IF NOT EXISTS `seller` (
    `seller_id` BIGINT NOT NULL AUTO_INCREMENT,
    `name` VARCHAR(255) NOT NULL,
    `email` VARCHAR(255) NOT NULL UNIQUE,
    `pass` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`seller_id`)
);

-- Shop Table
CREATE TABLE IF NOT EXISTS `shop` (
    `shop_id` BIGINT NOT NULL AUTO_INCREMENT,
    `seller_id` BIGINT NOT NULL,
    `name` VARCHAR(255) NOT NULL,
    `total_categories` INT NOT NULL,
    `phone` VARCHAR(20) NOT NULL,
    `division` VARCHAR(255) NOT NULL,
    `house` VARCHAR(255) NOT NULL,
    `city` VARCHAR(255) NOT NULL,
```

```
    `fb` VARCHAR(255),
    `insta` VARCHAR(255),
    PRIMARY KEY (`shop_id`),
    FOREIGN KEY (`seller_id`) REFERENCES
`seller`(`seller_id`)
);

-- Product_info Table
CREATE TABLE IF NOT EXISTS `product_info` (
    `product_id` BIGINT NOT NULL AUTO_INCREMENT,
    `shop_id` BIGINT NOT NULL,
    `name` VARCHAR(255) NOT NULL,
    `quantity` INT NOT NULL,
    `approval` BOOLEAN NOT NULL,
    `price` DECIMAL(10,2) NOT NULL,
    `review` TEXT,
    PRIMARY KEY (`product_id`),
    FOREIGN KEY (`shop_id`) REFERENCES
`shop`(`shop_id`)
);

-- Order Table
CREATE TABLE IF NOT EXISTS `order` (
    `order_id` BIGINT NOT NULL AUTO_INCREMENT,
    `customer_id` BIGINT NOT NULL,
    `product_id` BIGINT NOT NULL,
    `shop_id` BIGINT NOT NULL,
    `payment_method` VARCHAR(255) NOT NULL,
    `billing_address` VARCHAR(255) NOT NULL,
    `delivery_time` DATETIME NOT NULL,
```

```
    `payment_status` VARCHAR(255) NOT NULL,
    `shipment_status` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`order_id`),
    FOREIGN KEY (`customer_id`) REFERENCES
`customer`(`customer_id`),
    FOREIGN KEY (`product_id`) REFERENCES
`product_info`(`product_id`),
    FOREIGN KEY (`shop_id`) REFERENCES
`product_info`(`shop_id`)
);

-- selecting_products_cart Table
CREATE TABLE IF NOT EXISTS
`selecting_products_cart` (
    `customer_id` BIGINT NOT NULL,
    `product_id` BIGINT NOT NULL,
    PRIMARY KEY (`customer_id`, `product_id`),
    FOREIGN KEY (`customer_id`) REFERENCES
`customer`(`customer_id`),
    FOREIGN KEY (`product_id`) REFERENCES
`product_info`(`product_id`)
);


ALTER TABLE `product_info`
ADD COLUMN `image_url` VARCHAR(255),
ADD COLUMN `product_description` VARCHAR(255);

START TRANSACTION;
```

```
ALTER TABLE `order`
MODIFY COLUMN `customer_id` BIGINT NULL;

ALTER TABLE `order`
DROP FOREIGN KEY `order_ibfk_1`;

ALTER TABLE `order`
ADD CONSTRAINT `order_fk_customer_id`
FOREIGN KEY (`customer_id`) REFERENCES `customer`
(`customer_id`)
ON DELETE SET NULL;

COMMIT;
```

models.I have written my sql quarries in the controllers file and then connected with the routers file , then all the navigate link are called in the server.js files:

Connection with the database:

```
import mysql from "mysql2"
import dotenv from "dotenv"
dotenv.config()

//Create a connection pool to the database
const pool = mysql.createPool({
  host: process.env.MYSQL_HOST,
  user: process.env.MYSQL_USER,
  password: process.env.MYSQL_PASSWORD,
  database: process.env.MYSQL_DATABASE,
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0
}).promise();
export {
    pool
  };
```

Here are some quarries:

To fetch the TOP products:

```
const getTopProducts = async () => {
    const [rows] = await pool.query(`
        SELECT
            oi.product_id,
            pi.name AS product_name,
            pi.product_description AS product_description,
            s.name AS shop_name,
            pi.image_url AS product_image_url,
            pi.price,
            SUM(1) AS quantity_sold
        FROM
            \`order\` o
        INNER JOIN
            \`product_info\` oi ON o.product_id = oi.product_id
        INNER JOIN
            \`product_info\` pi ON oi.product_id = pi.product_id
        INNER JOIN
            \`shop\` s ON oi.shop_id = s.shop_id
        GROUP BY
            oi.product_id, pi.name, pi.product_description,
s.name, pi.image_url, pi.price
        ORDER BY
            quantity_sold DESC
        LIMIT 15;
    `);
    return rows;
};
```

Get products by shop name:

```
export const getProductByShopName = async
                (shop_name) => {
        const [rows] = await pool.execute(`
            SELECT pi.*, s.name AS shop_name
                  FROM product_info pi
            INNER JOIN shop s ON pi.shop_id =
                    s.shop_id
                     WHERE s.name = ?
                  `, [shop_name]);
                     return rows;
                         }
```

Seller controller:

```
export const getAllSellers = async () => {
    const [rows] = await pool.execute('SELECT * FROM seller');
    return rows;
}
export const createSeller = async (name, email, pass) => {
    const [result] = await pool.execute(`INSERT INTO seller
(name, email, pass) VALUES (?, ?, ?)`, [name, email, pass]);
    return result;
};

export const getSellerById = async (seller_id) => {
    const [rows] = await pool.execute(`SELECT * FROM seller
WHERE seller_id = ?`, [seller_id]);
    return rows;
};
```

```
export const getSellerByToken = async (token) => {
    const user = jwt.verify(token, process.env.JWT_SECRET);
    console.log(user);
    const [rows] = await pool.execute(`SELECT * FROM seller
WHERE seller_id = ?`, [user.seller_id]);

    return rows;
}
export const updateSeller = async (seller_id, name, email,
pass) => {
    const [result] = await pool.execute(`UPDATE seller SET
name = ?, email = ?, pass = ? WHERE seller_id = ?`, [name,
email, pass, seller_id]);
    return result;
};

export const deleteSeller = async (seller_id) => {
    const [result] = await pool.execute(`DELETE FROM seller
WHERE seller_id = ?`, [seller_id]);
    return result;
}
```

Then i have called this query in the router files that connected it with the frontend:

Source Code Repository

https://github.com/RazinSufian/370_Project

Conclusion

"In this project, I've built an e-commerce marketplace platform with separate frontend and backend components. The frontend, developed using React, provides users with an intuitive interface for browsing products, making purchases, and managing accounts. The backend, powered by Node.js and Express.js, handles data storage, user authentication, and API interactions with the frontend. By organizing code into directories like controllers, models, and routers, I've created a scalable and maintainable architecture. Overall, this project showcases my skills in full-stack development and demonstrates my ability to create functional and user-friendly web applications."

References

Youtube Playlist:
https://www.youtube.com/@chaiaurcode

Full stack developer:
https://github.com/IstiakImran