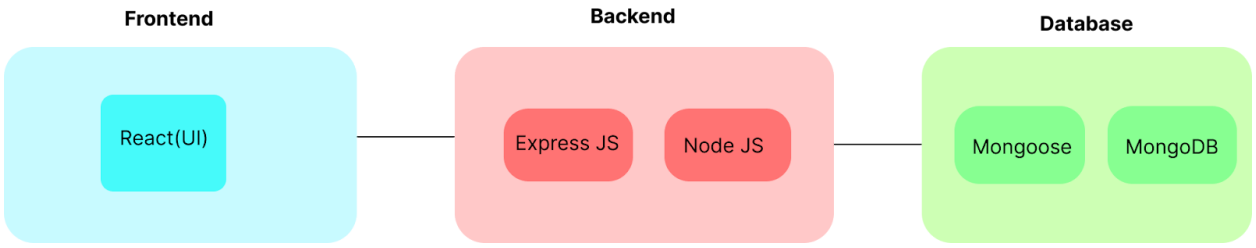
SHOPEZ : E-commerce Application

Description :

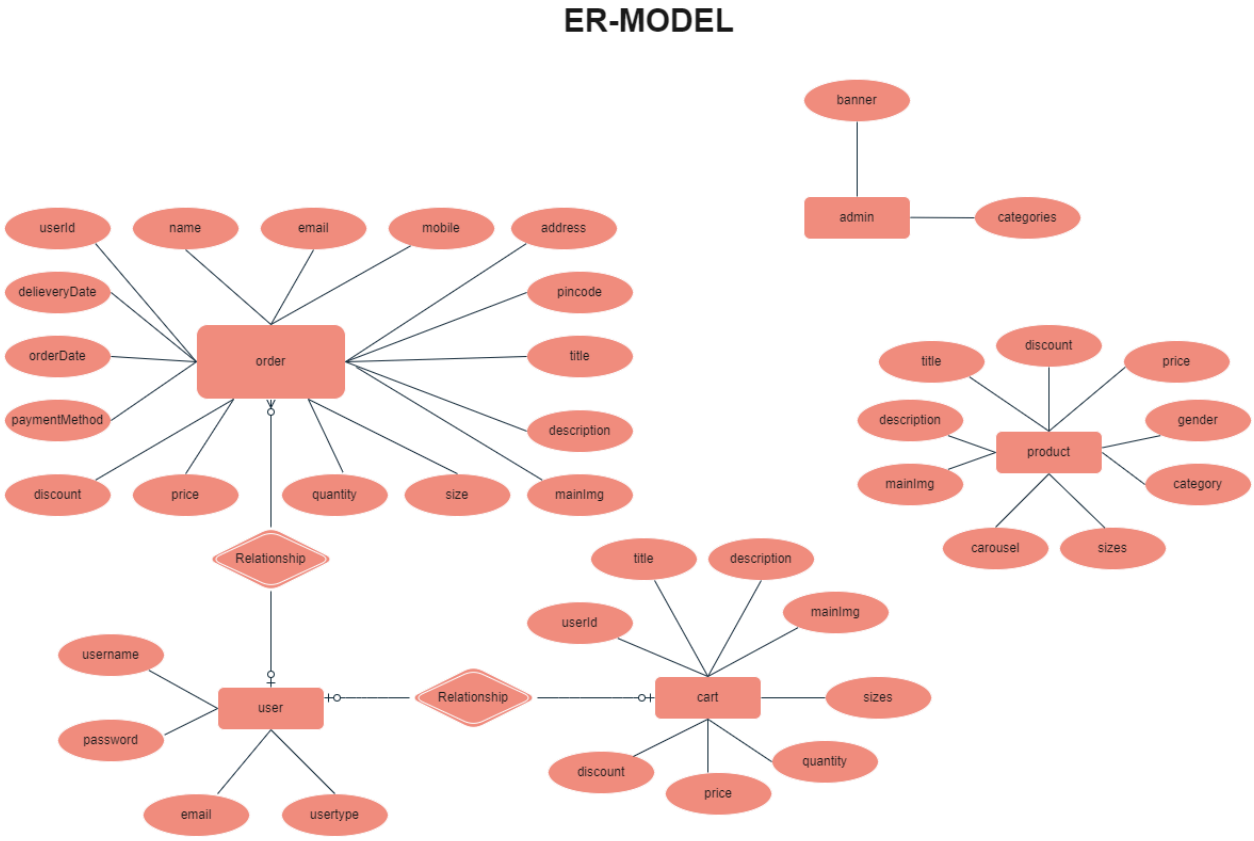


The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Cart, Products, Profile, Admin dashboard, etc.,

• The backend is represented by the "Backend" section, consisting of API endpoints for Users, Orders, Products, etc. It also includes Admin Authentication and an Admin Dashboard.

• The Database section represents the database that stores collections for Users, cart, Orders and Product.

ER - Diagram :



The ShopEZ ER-diagram represents the entities and relationships involved in an e-commerce system. It illustrates how users, products, cart, and orders are interconnected. Here is a breakdown of the entities and their relationships:

USER: Represents the individuals or entities who are registered in the platform.

Admin: Represents a collection with important details such as Banner image and Categories.

Products: Represents a collection of all the products available in the platform.

Cart: This collection stores all the products that are added to the cart by users. Here, the elements in the cart are differentiated by the user Id.

Orders: This collection stores all the orders that are made by the users in the platform.

Features :

1. Comprehensive Product Catalog: ShopEZ boasts an extensive catalog of products, offering a diverse range of items and options for shoppers. You can effortlessly explore and discover various products, complete with detailed descriptions, customer reviews, pricing, and available discounts, to find the perfect items for your needs.

2. Shop Now Button: Each product listing features a convenient "Shop Now" button. When you find a product that aligns with your preferences, simply click on the button to initiate the purchasing process.

3. Order Details Page: Upon clicking the "Shop Now" button, you will be directed to an order details page. Here, you can provide relevant information such as your shipping address, preferred payment method, and any specific product requirements.

4. Secure and Efficient Checkout Process: ShopEZ guarantees a secure and efficient checkout process. Your personal information will be handled with the utmost security, and we strive to make the purchasing process as swift and trouble-free as possible.

5. Order Confirmation and Details: After successfully placing an order, you will receive

a confirmation notification. Subsequently, you will be directed to an order details page, where you can review all pertinent information about your order, including shipping details, payment method, and any specific product requests you specified.

Roles & Responsibilities :

USER: Represents the individuals or entities who are registered in the platform.

Admin: Represents a collection with important details such as Banner image and Categories.

Products: Represents a collection of all the products available in the platform.

Cart: This collection stores all the products that are added to the cart by users. Here, the elements in the cart are differentiated by the user Id.

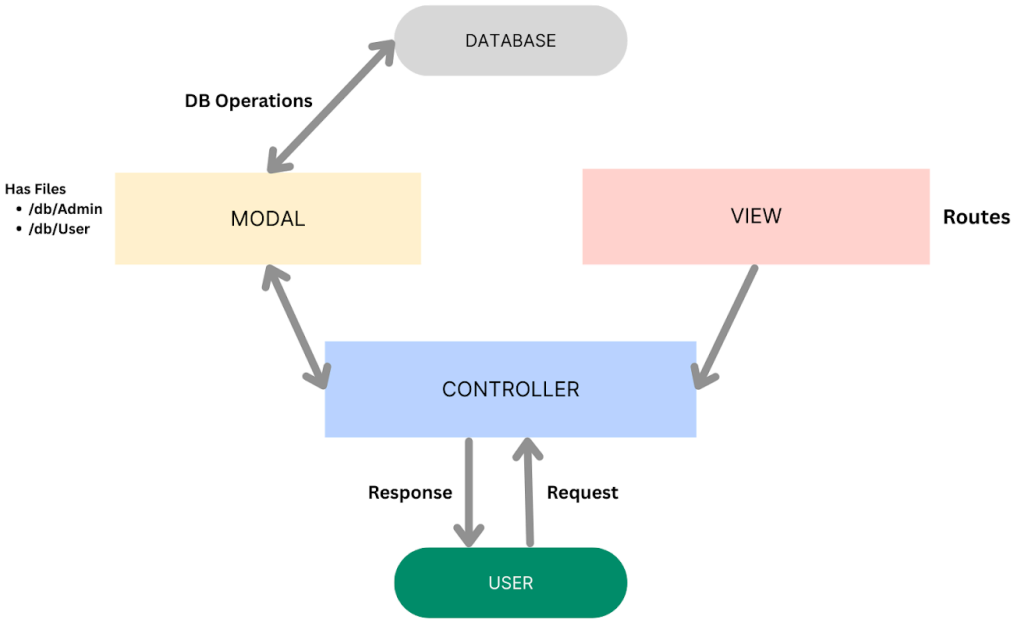
Orders: This collection stores all the orders that are made by the users in the platform.

User Flow:

Users start by registering for an account.
After registration, they can log in with their credentials.
Once logged in, they can check for the available products in the platform.
Users can add the products they wish to their carts and order.
They can then proceed by entering address and payment details.

After ordering, they can check them in the profile section.

MVC Pattern:



The Shop-Ez backend application follows the Model-View-Controller (MVC) architectural pattern, a software design approach that separates an application into three interconnected layers. This separation allows for modularity, easier maintenance, and scalability.

Model Layer (Data Layer)

The Model layer is responsible for handling all data-related logic. This includes the definition of data schemas and the operations performed on the database using those schemas. The models are implemented using Mongoose, which provides a schema-based solution to model application data for MongoDB.

Controller Layer

The Controller layer acts as an intermediary between the view (routes) and the model. It receives incoming requests, processes the input (which may include validation or transformation), calls the appropriate methods from the model, and then returns a response to the client.

View Layer (Routing Layer)

In the context of a backend REST API, the View is implemented as the routing layer, where various endpoints are defined. These endpoints determine how the backend responds to different HTTP requests (GET, POST, PUT, DELETE) and are responsible for invoking the appropriate controller functions.

Advantages of Using MVC in This Project

Separation of Concerns: Each layer has a clearly defined responsibility, improving readability and maintainability.

Scalability: New features can be added easily by creating new routes, controllers, and models.

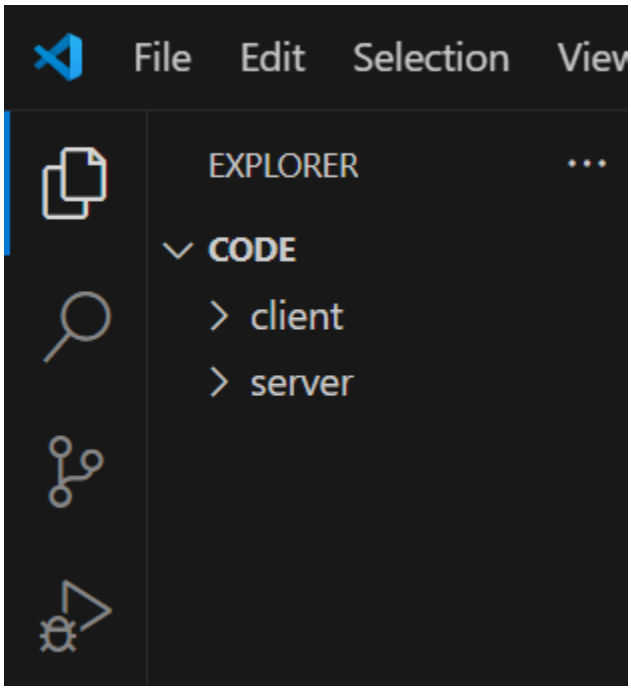
Reusability: Logic in controllers and models can be reused across multiple parts of the application.

Testing: Each layer can be tested independently, especially the controllers and models.

Collaboration-Friendly: Multiple developers can work simultaneously on different layers without conflict.

Project SETUP & Configuration:

Creating project folder



Create a new folder with your <project name>.

2. **Inside that folder create two new folders.**
3. **Name one as Client.**
4. **Name another one as Server.**
5. **Now open that folder in VS Code.**

Client setup (installing react app)

Open the Client folder in the terminal of VScode.

```
npm create vite@latest . -- --template react
```

Select React framework from the given options.

```
Select a framework:
| React
```

Select JavaScript variant from the given options.

```
Select a variant:
```

```
| JavaScript
```

Now lets navigate to the client folder by giving the following command.

```
cd client
```

To install all the packages run the following command.

```
npm install
```

To start the React server type the following command.

```
npm run dev
```

Server setup (npm init)

Open Server folder in terminal of VScode.

```
npm init -y
```

Create files:

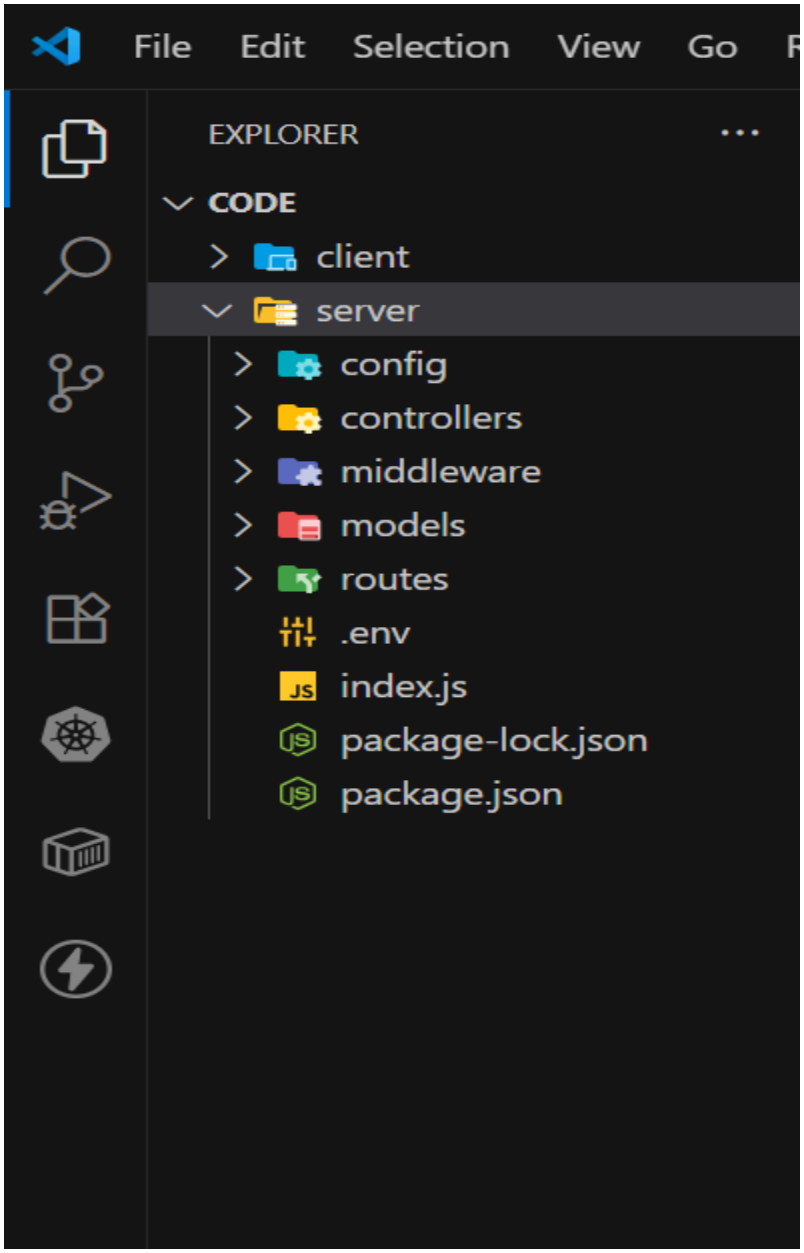
```
server.js
```

Create folders:

```
models
controllers
Routes
```

BACKEND DEVELOPMENT

BACKEND STRUCTURE :



Developement And Explanation :

1.Setup express server:

• Create index.js file.

• Create an express server on your desired port number.

• Define API's

2. Database Configuration:

• Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.

• Create a database and define the necessary collections for admin, users, products, orders and other relevant data.

3. Create Express.js Server:

• Set up an Express.js server to handle HTTP requests and serve API endpoints.

• Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

4. Define API Routes:

• Create separate route files for different API functionalities such as users, orders, and authentication.

• Define the necessary routes for listing products, handling user registration and login,managing orders, etc.

• Implement route handlers using Express.js to handle requests and interact with the database.

5. Implement Data Models:

• Define Mongoose schemas for the different data entities like products, users, and orders.

• Create corresponding Mongoose models to interact with the MongoDB database.

• Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

6. User Authentication:

• Create routes and middleware for user registration, login, and logout.

• Set up authentication middleware to protect routes that require user authentication.

7. Handle new products and Orders:

• Create routes and controllers to handle new product listings, including fetching products data from the database and sending it as a response.

• Implement ordering(buy) functionality by creating routes and controllers to handle order requests, including validation and database updates.

8. Admin Functionality:

• Implement routes and controllers specific to admin functionalities such as adding products, managing user orders, etc.

• Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

9. Error Handling:

• Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.

Database Development

Configure MongoDB

Install Mongoose

In your Node.js project directory, run:

```
Npm install mongoose
```

CREATE DATABASE CONNECTION

Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below

Create Schema and models

Schema.js

server > Schema.js > productSchema

```
1   import mongoose from "mongoose";
2
3   const userSchema = new mongoose.Schema({
4       username: {type: String},
5       password: {type: String},
6       email: {type: String},
7       usertype: {type: String}
8   });
9
10  const adminSchema = new mongoose.Schema({
11      banner: {type: String},
12      categories: {type: Array}
13  });
14
15  const productSchema = new mongoose.Schema({
16      title: {type: String},
17      description: {type: String},
18      mainImg: {type: String},
19      carousel: {type: Array},
20      sizes: {type: Array},
21      category: {type: String},
22      gender: {type: String},
23      price: {type: Number},
24      discount: {type: Number}
25  })
26
```

JS Schema.js

server > JS Schema.js > productSchema

```
const orderSchema = new mongoose.Schema({
    userId: {type: String},
    name: {type: String},
    email: {type: String},
    mobile: {type: String},
    address: {type: String},
    pincode: {type: String},
    title: {type: String},
    description: {type: String},
    mainImg: {type: String},
    size: {type: String},
    quantity: {type: Number},
    price: {type: Number},
    discount: {type: Number},
    paymentMethod: {type: String},
    orderDate: {type: String},
    deliveryDate: {type: String},
    orderStatus: {type: String, default: 'order placed'}
})

const cartSchema = new mongoose.Schema({
    userId: {type: String},
    title: {type: String},
    description: {type: String},
    mainImg: {type: String},
    size: {type: String},
    quantity: {type: String},
    price: {type: Number},
    discount: {type: Number}
})


export const User = mongoose.model('users', userSchema);
export const Admin = mongoose.model('admin', adminSchema);
export const Product = mongoose.model('products', productSchema);
export const Orders = mongoose.model('orders', orderSchema);
export const Cart = mongoose.model('cart', cartSchema);

```

Frontend Development :

Frontend Structure

Development and execution

1. Setup React Application:

• Create a React app in the client folder.

• Install required libraries

• Create required pages and components and add routes.

2.Design UI components:

• Create Components.

• Implement layout and styling.

• Add navigation.

3.Implement frontend logic:

• Integration with API endpoints.

• Implement data binding.

Project Execution

Steps For Execution

Step 1: Set Up the Frontend (React App):

a) Open a terminal and navigate into the client folder:

cd client

b) Once installation is done, start the React development server:

npm run dev

c) The app should now be running on:

http://localhost:5173

Step 2: Set Up the Backend (Express Server)

a) Open a new terminal tab/window or split the terminal.

b) Navigate into the server folder:

cd ../server

Step 3: Configure Environment Variables

Inside the server folder, create a new file named .env (no file extension).

In that .env file, add your MongoDB connection string:

MONGO_URI=mongodb://localhost:27017/<— dbname—>

Step 4: Start the Backend Server:

Inside the same server folder, run the backend server using:

nodemon index.js

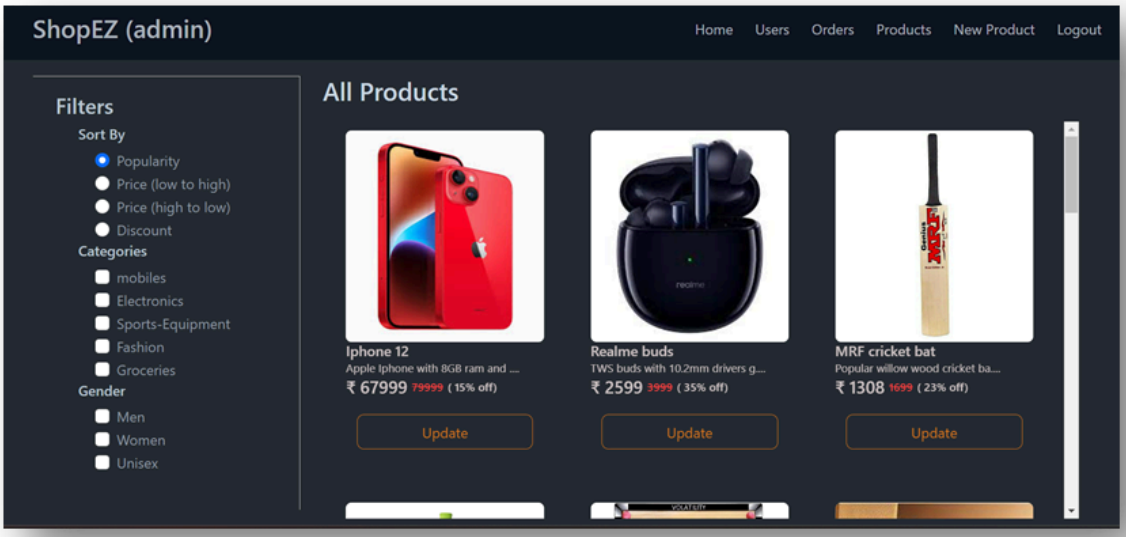
The server should start on:

http://localhost:8000

Demo Screenshorts

Landing page : This is the landing page of the application .

All Products : This is the all products page , where admins can display all the available products .



New Product Page: This is the new product page where admin can add new products.