✅ 1. **Project Structure**

ecommerce/

├── models/

│   ├── category.model.js

│   ├── product.model.js

│   ├── order.model.js

│ └── cart.model.js

│   └── user.model.js

├── controllers/

│   ├── category.controller.js

│   ├── product.controller.js

│   ├── cart.controller.js

│   ├── order.controller.js

│   └── user.controller.js

├── routes/

│   ├── category.routes.js

│   ├── product.routes.js

│   └── cart.routes.js

│   ├── order.routes.js

│   └── user.routes.js

├── config/

│   └── db.js

├── app.js

└── package.json

2. **Step-by-Step Modules**

**A. Category Module**

**models/category.model.js**

const mongoose = require('mongoose');

const categorySchema = new mongoose.Schema({

  name: { type: String, required: true, unique: true },

  description: String

}, { timestamps: true });

module.exports = mongoose.model('Category', categorySchema);

controllers/category.controller.js

const Category = require('../models/category.model');

// Get all categories

exports.getCategories = async (req, res) => {

  try {

    const categories = await Category.find();

    res.json(categories);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Create a new category

exports.createCategory = async (req, res) => {

  try {

    const category = new Category(req.body);

    await category.save();

    res.status(201).json(category);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Update a category

exports.updateCategory = async (req, res) => {

  try {

    const { id } = req.params;

    const updatedCategory = await Category.findByIdAndUpdate(id, req.body, {

      new: true,

      runValidators: true,

    });

    if (!updatedCategory) {

      return res.status(404).json({ error: 'Category not found' });

    }

    res.json(updatedCategory);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Delete a category

exports.deleteCategory = async (req, res) => {

  try {

    const { id } = req.params;

    const deletedCategory = await Category.findByIdAndDelete(id);

    if (!deletedCategory) {

      return res.status(404).json({ error: 'Category not found' });

    }

    res.json({ message: 'Category deleted successfully' });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

routes/category.routes.js

const express = require('express');

const router = express.Router();

const { getCategories,createCategory,updateCategory,deleteCategory,} = require('../controllers/category.controller');

// Get all categories

router.get('/', getCategories);

// Create a new category

router.post('/', createCategory);

// Update a category by ID

router.put('/:id', updateCategory);

// Delete a category by ID

router.delete('/:id', deleteCategory);

module.exports = router;

#### B. ****Product Module****

##### models/product.model.js

const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({

  name: { type: String, required: true },

  category: { type: mongoose.Schema.Types.ObjectId, ref: 'Category' },

  price: Number,

  stock: Number,

  description: String,

  image: String

}, { timestamps: true });

module.exports = mongoose.model('Product', productSchema);

controllers/product.controller.js

const Product = require('../models/product.model');

// Get all products with populated category

exports.getProducts = async (req, res) => {

  try {

    const products = await Product.find().populate('category');

    res.json(products);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Create a new product

exports.createProduct = async (req, res) => {

  try {

    const product = new Product(req.body);

    await product.save();

    res.status(201).json(product);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Update a product by ID

exports.updateProduct = async (req, res) => {

  try {

    const { id } = req.params;

    const updatedProduct = await Product.findByIdAndUpdate(id, req.body, {

      new: true,

      runValidators: true,

    }).populate('category');

    if (!updatedProduct) {

      return res.status(404).json({ error: 'Product not found' });

    }

    res.json(updatedProduct);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Delete a product by ID

exports.deleteProduct = async (req, res) => {

  try {

    const { id } = req.params;

    const deletedProduct = await Product.findByIdAndDelete(id);

    if (!deletedProduct) {

      return res.status(404).json({ error: 'Product not found' });

    }

    res.json({ message: 'Product deleted successfully' });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

routes/product.routes.js

const express = require('express');

const router = express.Router();

const {

  getProducts,createProduct,updateProduct,  deleteProduct

} = require('../controllers/product.controller');

// Get all products

router.get('/', getProducts);

// Create a new product

router.post('/', createProduct);

// Update a product by ID

router.put('/:id', updateProduct);

// Delete a product by ID

router.delete('/:id', deleteProduct);

module.exports = router;

C. **Cart Model**

models/cart.model.js

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

// Cart schema

const cartSchema = new Schema({

  user: {

    type: Schema.Types.ObjectId,

    ref: 'User', // Assuming you have a User model

    required: true

  },

  items: [{

    product: {

      type: Schema.Types.ObjectId,

      ref: 'Product', // Assuming you have a Product model

      required: true

    },

    quantity: {

      type: Number,

      required: true,

      min: 1

    },

    price: {

      type: Number,

      required: true

    }

  }],

  totalAmount: {

    type: Number,

    default: 0

  },

  createdAt: {

    type: Date,

    default: Date.now

  },

  updatedAt: {

    type: Date,

    default: Date.now

  }

});

// Middleware to update totalAmount and updatedAt

cartSchema.pre('save', function(next) {

  this.totalAmount = this.items.reduce((total, item) => total + item.price \* item.quantity, 0);

  this.updatedAt = Date.now();

  next();

});

const Cart = mongoose.model('Cart', cartSchema);

module.exports = Cart;

controllers/cart.controller.js

const Cart = require('../models/cart.model');

// Get cart for a user

exports.getCart = async (req, res) => {

  try {

    const cart = await Cart.findOne({ user: req.user.id }).populate('items.product');

    if (!cart) {

      return res.status(404).json({ error: 'Cart not found' });

    }

    res.json(cart);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Add item to the cart

exports.addToCart = async (req, res) => {

  try {

    const { productId, quantity, price } = req.body;

    let cart = await Cart.findOne({ user: req.user.id });

    if (!cart) {

      // If the cart doesn't exist, create one

      cart = new Cart({ user: req.user.id, items: [] });

    }

    const existingItem = cart.items.find(item => item.product.toString() === productId);

    if (existingItem) {

      // If item exists, update quantity

      existingItem.quantity += quantity;

    } else {

      // If item doesn't exist, add it to the cart

      cart.items.push({ product: productId, quantity, price });

    }

    await cart.save();

    res.status(201).json(cart);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Remove item from the cart

exports.removeFromCart = async (req, res) => {

  try {

    const { productId } = req.params;

    const cart = await Cart.findOne({ user: req.user.id });

    if (!cart) {

      return res.status(404).json({ error: 'Cart not found' });

    }

    cart.items = cart.items.filter(item => item.product.toString() !== productId);

    await cart.save();

    res.json(cart);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Clear cart

exports.clearCart = async (req, res) => {

  try {

    const cart = await Cart.findOneAndDelete({ user: req.user.id });

    if (!cart) {

      return res.status(404).json({ error: 'Cart not found' });

    }

    res.json({ message: 'Cart cleared successfully' });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

routes/cart.routes.js

const express = require('express');

const router = express.Router();

const {

  getCart,

  addToCart,

  removeFromCart,

  clearCart

} = require('../controllers/cart.controller');

// Assuming you're using authentication middleware to get `req.user`

const authMiddleware = require('../middlewares/auth');

// Get current user's cart

router.get('/', authMiddleware, getCart);

// Add item to cart

router.post('/', authMiddleware, addToCart);

// Remove a specific product from cart

router.delete('/:productId', authMiddleware, removeFrom

D. **Order Module**

models/order.model.js

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

// Order schema

const orderSchema = new Schema({

  user: {

    type: Schema.Types.ObjectId,

    ref: 'User',

    required: true

  },

  cartId: {

    type: Schema.Types.ObjectId,

    ref: 'Cart',

    required: true

  },

  status: {

    type: String,

    enum: ['pending', 'paid', 'shipped', 'delivered', 'canceled'],

    default: 'pending'

  },

  paymentMethod: {

    type: String,

    enum: ['COD', 'card', 'wallet'],

    required: true

  },

  shippingAddress: {

    type: String,

    required: true

  },

  totalAmount: {

    type: Number,

    required: true

  },

  createdAt: {

    type: Date,

    default: Date.now

  },

  updatedAt: {

    type: Date,

    default: Date.now

  }

});

// Middleware to update `updatedAt` before saving

orderSchema.pre('save', function(next) {

  this.updatedAt = Date.now();

  next();

});

const Order = mongoose.model('Order', orderSchema);

module.exports = Order;

controllers/order.controller.js

const Order = require('../models/order.model');

const Cart = require('../models/cart.model');

// Create an order from cart

exports.createOrder = async (req, res) => {

  try {

    const cart = await Cart.findOne({ user: req.user.id });

    if (!cart || cart.items.length === 0) {

      return res.status(400).json({ error: 'Cart is empty' });

    }

    const { paymentMethod, shippingAddress } = req.body;

    const order = new Order({

      user: req.user.id,

      cartId: cart.\_id,

      paymentMethod,

      shippingAddress,

      totalAmount: cart.totalAmount

    });

    await order.save();

    // Optionally, clear the cart after creating the order

    await Cart.findOneAndDelete({ user: req.user.id });

    res.status(201).json(order);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Get orders for a user

exports.getOrders = async (req, res) => {

  try {

    const orders = await Order.find({ user: req.user.id }).populate('cartId');

    res.json(orders);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Update an order (e.g., change status)

exports.updateOrder = async (req, res) => {

  try {

    const { id } = req.params;

    const { status } = req.body;

    const updatedOrder = await Order.findByIdAndUpdate(id, { status }, { new: true });

    if (!updatedOrder) {

      return res.status(404).json({ error: 'Order not found' });

    }

    res.json(updatedOrder);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Delete an order

exports.deleteOrder = async (req, res) => {

  try {

    const { id } = req.params;

    const deletedOrder = await Order.findByIdAndDelete(id);

    if (!deletedOrder) {

      return res.status(404).json({ error: 'Order not found' });

    }

    res.json({ message: 'Order deleted successfully' });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

routes/order.routes.js

const express = require('express');

const router = express.Router();

const {

  createOrder,

  getOrders,

  updateOrder,

  deleteOrder

} = require('../controllers/order.controller');

const authMiddleware = require('../middlewares/auth');

// Create a new order from the current user's cart

router.post('/', authMiddleware, createOrder);

// Get all orders for the current user

router.get('/', authMiddleware, getOrders);

// Update an order (e.g., status change)

router.put('/:id', authMiddleware, updateOrder);

// Delete an order

router.delete('/:id', authMiddleware, deleteOrder);

module.exports = router;

models/user.model.js

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

  name: String,

  email: { type: String, unique: true },

  password: String, // In production, hash this!

  isAdmin: { type: Boolean, default: false }

}, { timestamps: true });

module.exports = mongoose.model('User', userSchema);

controllers/user.controller.js

const User = require('../models/user.model');

// Get all users

exports.getUsers = async (req, res) => {

  try {

    const users = await User.find();

    res.json(users);

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

// Create a new user

exports.createUser = async (req, res) => {

  try {

    const user = new User(req.body); // Password hashing recommended!

    await user.save();

    res.status(201).json(user);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Update a user by ID

exports.updateUser = async (req, res) => {

  try {

    const { id } = req.params;

    const updatedUser = await User.findByIdAndUpdate(id, req.body, {

      new: true,

      runValidators: true,

    });

    if (!updatedUser) {

      return res.status(404).json({ error: 'User not found' });

    }

    res.json(updatedUser);

  } catch (err) {

    res.status(400).json({ error: err.message });

  }

};

// Delete a user by ID

exports.deleteUser = async (req, res) => {

  try {

    const { id } = req.params;

    const deletedUser = await User.findByIdAndDelete(id);

    if (!deletedUser) {

      return res.status(404).json({ error: 'User not found' });

    }

    res.json({ message: 'User deleted successfully' });

  } catch (err) {

    res.status(500).json({ error: err.message });

  }

};

routes/user.routes.js

const express = require('express');

const router = express.Router();

const { getUsers, createUser, deleteUser , updateUser  } = require('../controllers/user.controller');

router.get('/', getUsers);

router.post('/', createUser);

// Update a product by ID

router.put('/:id', updateUser);

// Delete a product by ID

router.delete('/:id', deleteUser);

module.exports = router;

✅ Integrate All Routes in app.js

const express = require('express');

const connectDB = require('./config/db');

const app = express();

// Connect to MongoDB

connectDB();

// Middleware

app.use(express.json());

// Routes

app.use('/api/categories', require('./routes/category.routes'));

app.use('/api/products', require('./routes/product.routes'));

app.use('/api/users', require('./routes/user.routes'));

app.use('/api/cart, require('./routes/cart.routes'));

app.use('/api/orders', require('./routes/order.routes'));

// Start Server

const PORT = 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

// config/db.js

const mongoose = require('mongoose');

const connectDB = async () => {

  try {

    const conn = await mongoose.connect('mongodb://localhost:27017/ecommerce', {

        useUnifiedTopology: true,

    });

    console.log(`MongoDB Connected: ${conn.connection.host}`);

  } catch (error) {

    console.error(`Error: ${error.message}`);

    process.exit(1);

  }

};

module.exports = connectDB;