MERN Ecommerce

Description

An ecommerce store built with MERN stack, and utilizes third party API's. This ecommerce store enable three main different flows or implementations :

1. Buyers browse the store categories, products and brands
2. Sellers or Merchants manage their own brand component
3. Admins manage and control the entire store components

Features:

- Node provides the backend environment for this application
- Express middleware is used to handle requests, routes
- Mongoose schemas to model the application data
- React for displaying UI components
- Redux to manage application's state
- Redux Thunk middleware to handle asynchronous redux actions

Docker Guide

To run this project locally you can use docker compose provided in the repository. Here is a guide on how to run this project locally using docker compose.

Then simply start the docker compose:

```
docker-compose build
```

```
docker-compose up
```

Database Seed

- The seed command will create an admin user in the database
- The email and password are passed with the command as arguments
- Like below command, replace brackets with email and password.
- For more information, see code here

```
const chalk = require('chalk');
const bcrypt = require('bcryptjs');
const mongoose = require('mongoose');
const { faker } = require('@faker-js/faker');

const setupDB = require('./db');
const { ROLES } = require('../constants');
const User = require('../models/user');
const Brand = require('../models/brand');
const Product = require('../models/product');
const Category = require('../models/category');

const args = process.argv.slice(2);
const email = args[0];
const password = args[1];

const NUM_PRODUCTS = 100;
const NUM_BRANDS = 10;
const NUM_CATEGORIES = 10;

const seedDB = async () => {
  try {
    let categories = [];
    console.log(`${chalk.blue('✓')} ${chalk.blue('Seed database started')}`);
    if (!email || !password) throw new Error('Missing arguments');
    const existingUser = await User.findOne({ email });
    if (!existingUser) {
      console.log(`${chalk.yellow('!')} ${chalk.yellow('Seeding admin user...')}`);
      const user = new User({
        email,
        password,
        firstName: 'admin',
        lastName: 'admin',
        role: ROLES.Admin
      });
      const salt = await bcrypt.genSalt(10);
      const hash = await bcrypt.hash(user.password, salt);
      user.password = hash;
      await user.save();
      console.log(`${chalk.green('✓')} ${chalk.green('Admin user seeded.')}`);
    } else {
      console.log(`${chalk.yellow('!')} ${chalk.yellow('Admin user already exists, skipping seeding for admin user.')}`);
    }
    const categoriesCount = await Category.countDocuments();
    if (categoriesCount >= NUM_CATEGORIES) {
      console.log(`${chalk.yellow('!')} ${chalk.yellow('Sufficient number of categories already exist, skipping seeding for categories.')}`);
      categories = await Category.find().select('_id');
    } else {
      for (let i = 0; i < NUM_CATEGORIES; i++) {
        const category = new Category({
          name: faker.commerce.department(),
          description: faker.lorem.sentence(),
          isActive: true
        });
        await category.save();
        categories.push(category);
      }
      console.log(`${chalk.green('✓')} ${chalk.green('Categories seeded.')}`);
    }
    const brandsCount = await Brand.countDocuments();
    if (brandsCount >= NUM_BRANDS) {
      console.log(`${chalk.yellow('!')} ${chalk.yellow('Sufficient number of brands already exist, skipping seeding for brands.')}`);
    } else {
      for (let i = 0; i < NUM_BRANDS; i++) {
        const brand = new Brand({
          name: faker.company.name(),
          description: faker.lorem.sentence(),
          isActive: true
        });
        await brand.save();
      }
      console.log(`${chalk.green('✓')} ${chalk.green('Brands seeded.')}`);
    }
    const productsCount = await Product.countDocuments();
    if (productsCount >= NUM_PRODUCTS) {
      console.log(`${chalk.yellow('!')} ${chalk.yellow('Sufficient number of products already exist, skipping seeding for products.')}`);
    } else {
      const brands = await Brand.find().select('_id');
      for (let i = 0; i < NUM_PRODUCTS; i++) {
        const randomCategoryIndex = faker.number.int(categories.length - 1);
        const product = new Product({
          sku: faker.string.alphanumeric(10),
          name: faker.commerce.productName(),
          description: faker.lorem.sentence(),
          quantity: faker.number.int({ min: 1, max: 100 }),
          price: faker.commerce.price(),
          taxable: faker.datatype.boolean(),
          isActive: true,
          brand: brands[faker.number.int(brands.length - 1)]._id,
          category: categories[randomCategoryIndex]._id
        });
        await product.save();
        await Category.updateOne({ _id: categories[randomCategoryIndex]._id }, { $push: { products: product._id } });
      }
      console.log(`${chalk.green('✓')} ${chalk.green('Products seeded and associated with categories.')}`);
    }
  } catch (error) {
    console.log(`${chalk.red('x')} ${chalk.red('Error while seeding database')}`);
    console.log(error);
    return null;
  } finally {
    await mongoose.connection.close();
    console.log(`${chalk.blue('✓')} ${chalk.blue('Database connection closed!')}`);
  }
};
(async () => {
  try {
    await setupDB();
    await seedDB();
  } catch (error) {
    console.error(`Error initializing database: ${error.message}`);
  }

})();
```

npm run seed:db [email-***@****.com] [password-******] // This is just an example.

Install :

npm install in the project root will install dependencies in both client and server

Some basic Git commands are:

cd project

npm install

ENV

Create .env file for both client and server. See examples:

Frontend ENV:

```
API_URL=http://localhost:3000/api
```

Backend ENV:

```
PORT=3000
MONGO_URI=mongodb://127.0.0.1:27017/mern_ecommerce
JWT_SECRET=
MAILCHIMP_KEY=
MAILCHIMP_LIST_KEY=
MAILGUN_KEY=
MAILGUN_DOMAIN=
MAILGUN_EMAIL_SENDER=
GOOGLE_CLIENT_ID=
GOOGLE_CLIENT_SECRET=
GOOGLE_CALLBACK_URL=http://localhost:3000/api/auth/google/callback
FACEBOOK_CLIENT_ID=
FACEBOOK_CLIENT_SECRET=
FACEBOOK_CALLBACK_URL=http://localhost:3000/api/auth/facebook/callback
CLIENT_URL=http://localhost:8080
BASE_API_URL=api
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_REGION=us-east-2
AWS_BUCKET_NAME=
```

Start development:

```
npm run dev
```