CAPSTONE PROJECT REPORT

ON

SHOPFORHOME

Submitted By:

RAJESH KILLANA

DASARI JAYASHREE

CHINTADA BHARGAVI

RAJINI SUPE

MANIKANTA SIRIMALLA

Table of Contents

1. Problem statement

ShopForHome is a popular Store in the market for shopping the home décor stuff. Due to Covid 19 all the offline shopping stopped. So, the store wants to move to the online platforms and wants their own web application.

There are 2 users on the application: -

1. User
2. Admin

User Stories –

1. As a user I should be able to login, Logout and Register into the application.
2. As a user I should be able to see the products in different categories.
3. As a user I should be able to sort the products.
4. As a user I should be able to add the products into the shopping cart.
5. As a user I should be able to increase or decrease the quantity added in the cart.
6. As a user I should be able to add “n” number of products in the cart.
7. As a user I should be able to get the Wishlist option where I can add those products which I want but don’t want to order now
8. As a user I should get different discount coupons.

Admin Stories –

1. As an Admin I should be able to login, Logout and Register into the application.
2. As an Admin I should be able to perform CRUD on Users.
3. As an Admin I should be able to Perform CRUD on the products.
4. As an Admin I should be able to get bulk upload option to upload a csv for products details
5. As an Admin I should be able to get the stocks.
6. As an Admin I should be able to mail if any stock is less than 10.
7. As an Admin I should be able to get the sales report of a specific duration.
8. As an Admin I should be able to set the discount coupons for the specific set of users

2. List of Figures

3. Introduction

This project is a web-based shopping system for an existing shop. The project objective is to deliver the Goods through online application.

Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the internet. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products from anywhere through internet.

Project Overview

The concept of the application is to allow the customer to shop virtually using the internet and allow customers to buy the items and services of their desire from the store. The information pertaining to the products are stores on a MongoDB at the server side.

The server process the customers and the items are shipped to the address submitted by them. The application was designed into two modules first is for the customers who wish to buy the products. Second is the admin who maintains and updates the information pertaining to the articles and those of the customers.

The details of the items are bought forward from the database for the customer view based on the selection through the menu and the database of all the products are updated at the end of each transaction.

4. Design

Frontend: In the frontend side, we would be using React as the frontend library. We would use Redux for state management. We would use React Bootstrap library for designing of the interface.

Backend: For the Backend side, we would be using the Express library on top of Nodejs. We would use MongoDB as the NOSQL database to store our data as documents in JSON format. we would user mongoose to connect to our MongoDB database.

We Created REST APIs with Express and use these endpoints in the React frontend to interact with out backend part.

Technologies Used:

- MongoDB
- Express
- React
- Nodejs
- Devops
- Swagger
- Postman

Database Schema

In our shopping website we have two Entities they are user and the product.

The schema for those entities as written as below.

Product Schema

```
const productSchema = new mongoose.Schema({
  name: {
    type: String,
    required: [true, 'Please enter product name'],
    trim: true,
    maxLength: [100, 'Product name cannot exceed 100 characters'],
  },
  price: {
    type: Number,
    required: [true, 'Please enter product price'],
    maxLength: [5, 'Product name cannot exceed 5 characters'],
    default: 0.0,
  },
  description: {
    type: String,
    required: [true, 'Please enter product description'],
  },
```

```
ratings: {
  type: Number,
  default: 0,
},
images: [
  {
    public_id: {
      type: String,
      required: true,
    },
    url: {
      type: String,
      required: true,
    },
  },
],
category: {
  type: String,
  required: [true, 'Please select category for this product'],
  enum: {
    values: [
      'Electronics',
      'Cameras',
      'Laptops',
      'Accessories',
```

```
      'Headphones',
      'Food',
      'Books',
      'Clothes/Shoes',
      'Beauty/Health',
      'Sports',
      'Outdoor',
      'Home',
    ],
    message: 'Please select correct category for product',
  },
},
seller: {
  type: String,
  required: [true, 'Please enter product seller'],
},
stock: {
  type: Number,
  required: [true, 'Please enter product stock'],
  maxLength: [5, 'Product name cannot exceed 5 characters'],
  default: 0,
},
numOfReviews: {
  type: Number,
  default: 0,
```

```
},
reviews: [
  {
    user: {
      type: mongoose.Schema.ObjectId,
      ref: 'User',
      required: true,
    },
    name: {
      type: String,
      required: true,
    },
    rating: {
      type: Number,
      required: true,
    },
    comment: {
      type: String,
      required: true,
    },
  },
],
user: {
  type: mongoose.Schema.ObjectId,
  ref: 'User',
```

```
    required: true,
  },
  createdAt: {
    type: Date,
    default: Date.now,
  },
});
```

User Schema:

```
const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: [true, 'Please enter your name'],
        maxLength: [30, 'Your name cannot exceed 30 characters']
    },
    email: {
        type: String,
        required: [true, 'Please enter your email'],
        unique: true,
        validate: [validator.isEmail, 'Please enter valid email address']
    },
    password: {
        type: String,
        required: [true, 'Please enter your password'],
        minlength: [6, 'Your password must be longer than 6 characters'],
```

```
        select: false
    },
    avatar: {
        public_id: {
            type: String,
            required: true
        },
        url: {
            type: String,
            required: true
        }
    },
    role: {
        type: String,
        default: 'user'
    },
    createdAt: {
        type: Date,
        default: Date.now
    },
    resetPasswordToken: String,
    resetPasswordExpire: Date

})
```

E-R Diagram

E-R [Entity Relationship] diagrams are used to model and design relational databases, in terms of logic and business rules.

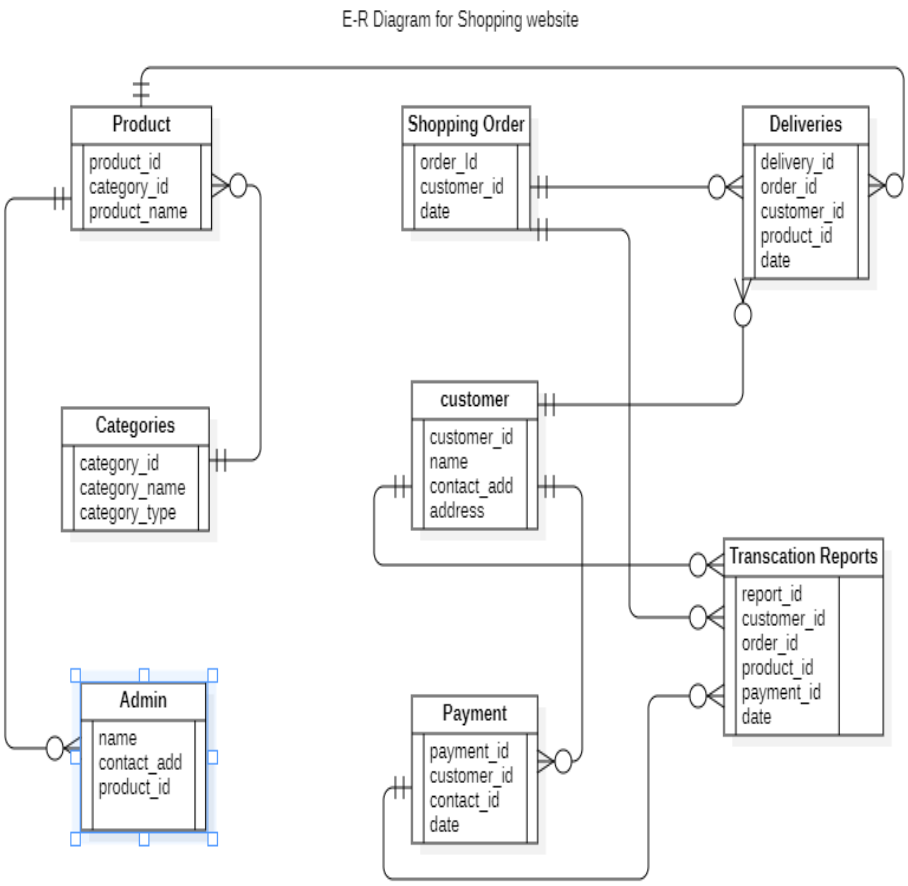

Fig 3.1 E-R diagram for ShopForHome

Use Case Diagram

A use case is a methodology used in system analysis to identify, clarify, organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

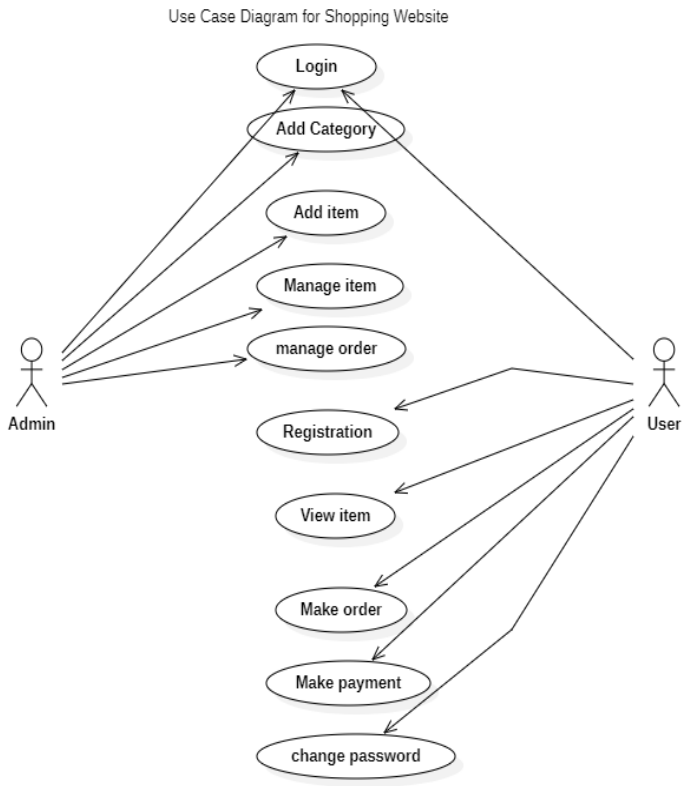

Fig 3.2 Use Case Diagram for ShopForHome

5. Implementation

The system after analysis has been identified to be presented with the following modules and roles.

The modules involved are:

- Admin
- User

Admin

The admin is the super user of this application. Only admin have access into the admin page. Admin may be the owner of the shop. The admin has all the information about all the users and about all the products.

This module is sub divided into different sub-modules.

- Manage Products.
- Manage users.
- Manage orders.

Manage Products

Add products: The ShopForHome project contains different kind of products. The products can be classified into different categories by name. Admin can add new products into the existing system with all its details including an image.

Delete Products: Admin can delete the products based on the stock of the product.

Search Products: Admin will have a list view of all the existing products. He can also search for a particular product by name.

Manage Users

View users: The admin have a list view of all the users registered in the system. Admin can view all the details of each user in the list except password.

Add Users: Admin has privileges to add a user directly by providing the details.

Delete Users: Admin has a right to delete a user.

Manage Orders

View order: Admin can view the orders which is generated by the users. He can verify the details of the purchase.

Delete order: Admin can delete order from the orders list when the product is taken for delivery.

User

User can register and login to the website and see the products available in the application and can verify the details of the product and can order the product from anywhere. He can pay through online, he can give review to the products.

Registration: A new user will have to register in the system by providing essential details to view the products in the system.

Login: A user must login with his username and password to the system after registration.

View Products: User can view the list of products based on their names after successful login. A detailed description of a particular product with product name, product details, product image, price can be viewed by users.

Search Product: Users can research for a particular product in the list by name.

Add to Cart: The user can add the desired product into his cart by clicking add to cart option on the product.

He can view his cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove.

Submit Cart: After confirming the items in the cart the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty.

Orders: In the orders the user will have a view of pending orders.

Edit Profile: The user can view and edit the profile.

code snippet

frontend > src > JS index.js

```
1   import React from 'react';
2   import ReactDOM from 'react-dom';
3   import App from './App';
4
5   import { Provider } from 'react-redux'
6   import store from './store'
7
8   import { positions, transitions, Provider as AlertProvider } from 'react-alert';
9   import AlertTemplate from 'react-alert-template-basic'
10
11  const options = {
12    timeout: 5000,
13    position: positions.BOTTOM_CENTER,
14    transition: transitions.SCALE
15  }
16
17  ReactDOM.render(
18    <Provider store={store} >
19      <AlertProvider template={AlertTemplate} {...options}>
20        <App />
21      </AlertProvider>
22    </Provider>,
23    document.getElementById('root')
24  );
25
```

Fig 5.1 index.js

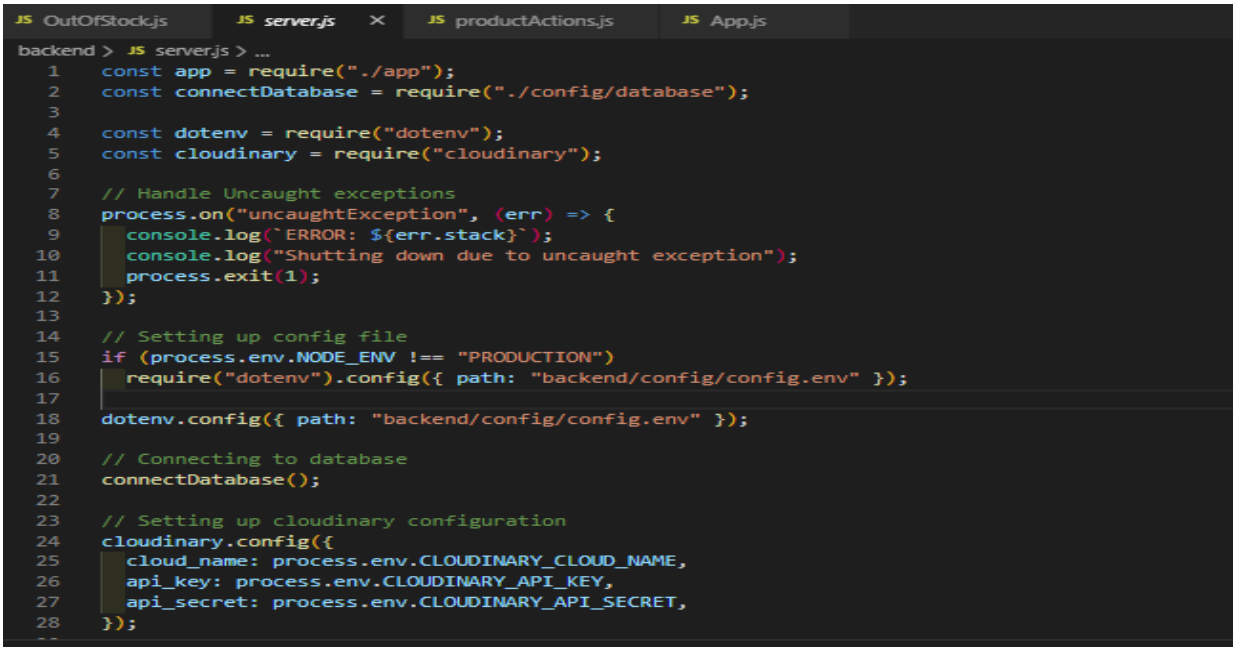

Fig 5.2 Server.js

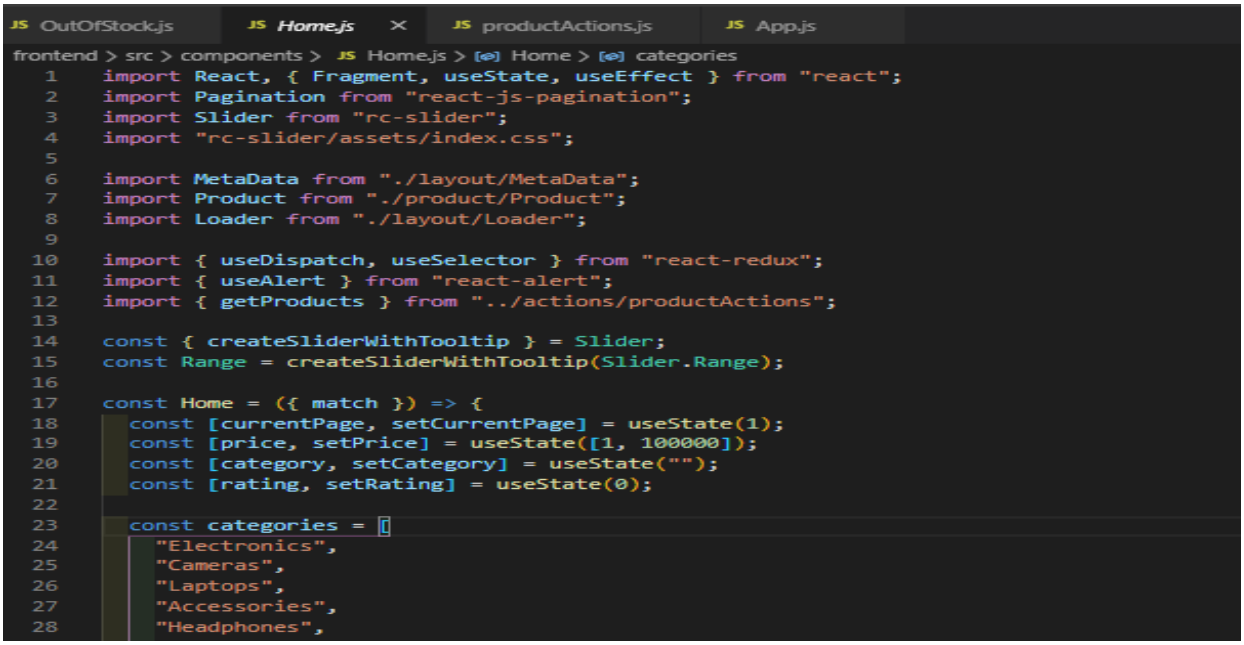

Fig 5.3 home.js

6. Output Screen

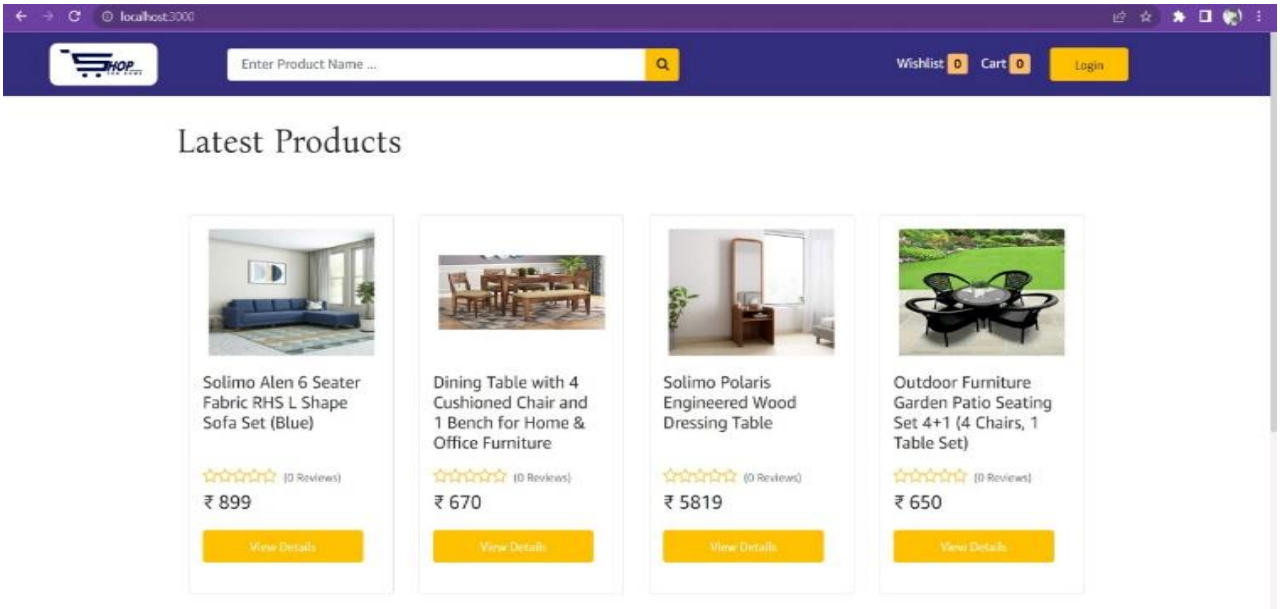

Fig 6.1 Home Page

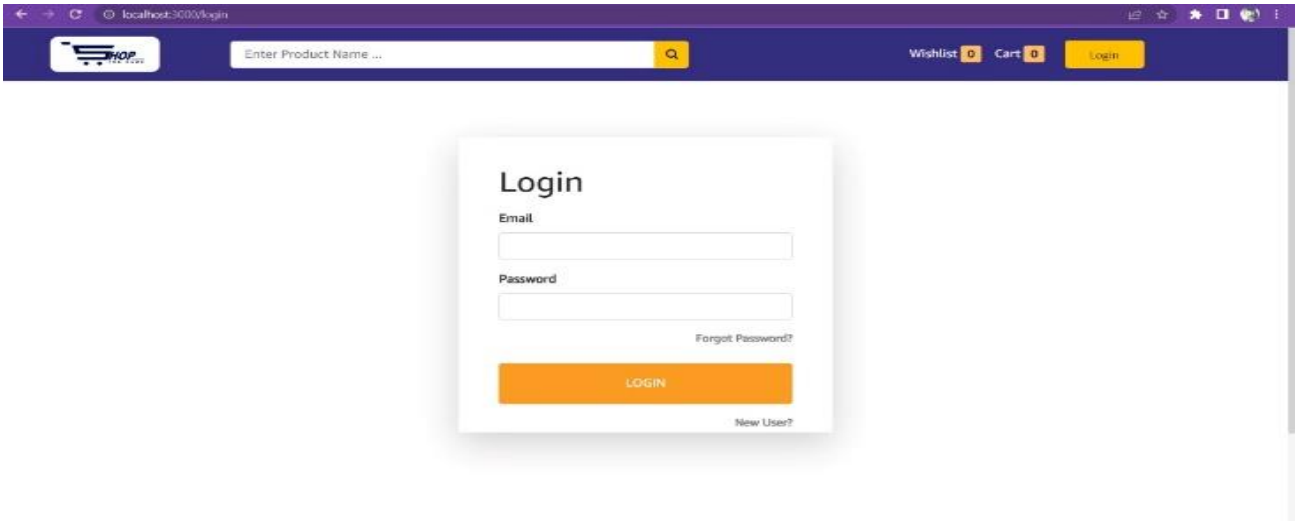

Fig 6.2 Login Page

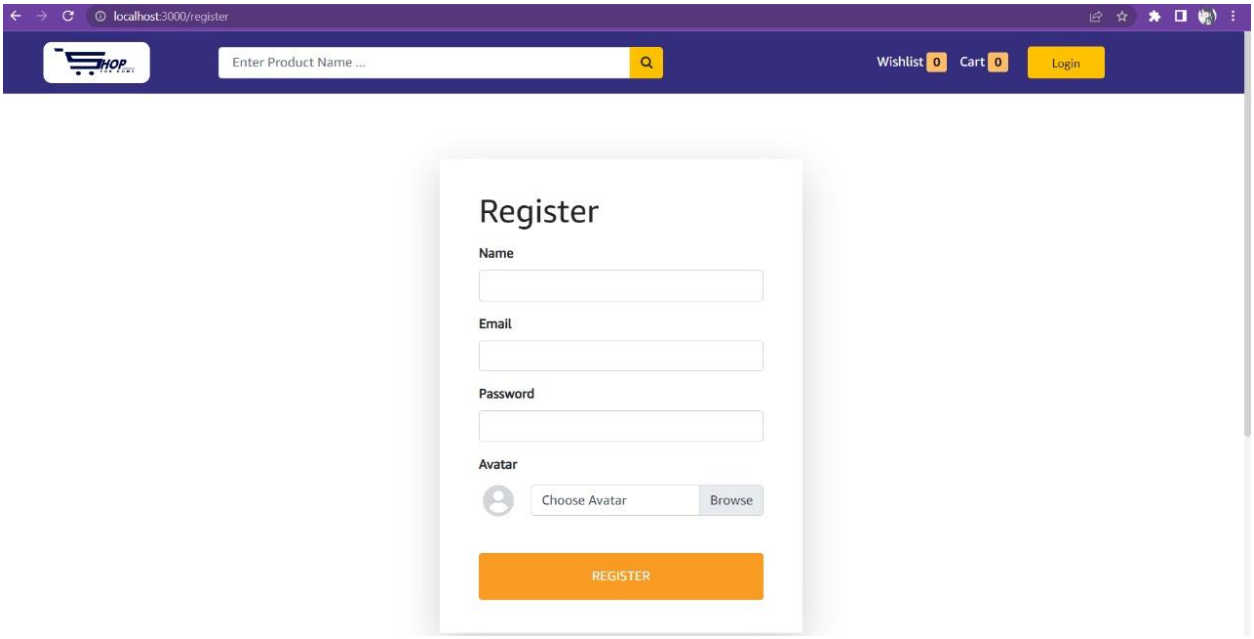

Fig 6.3 Registration Page

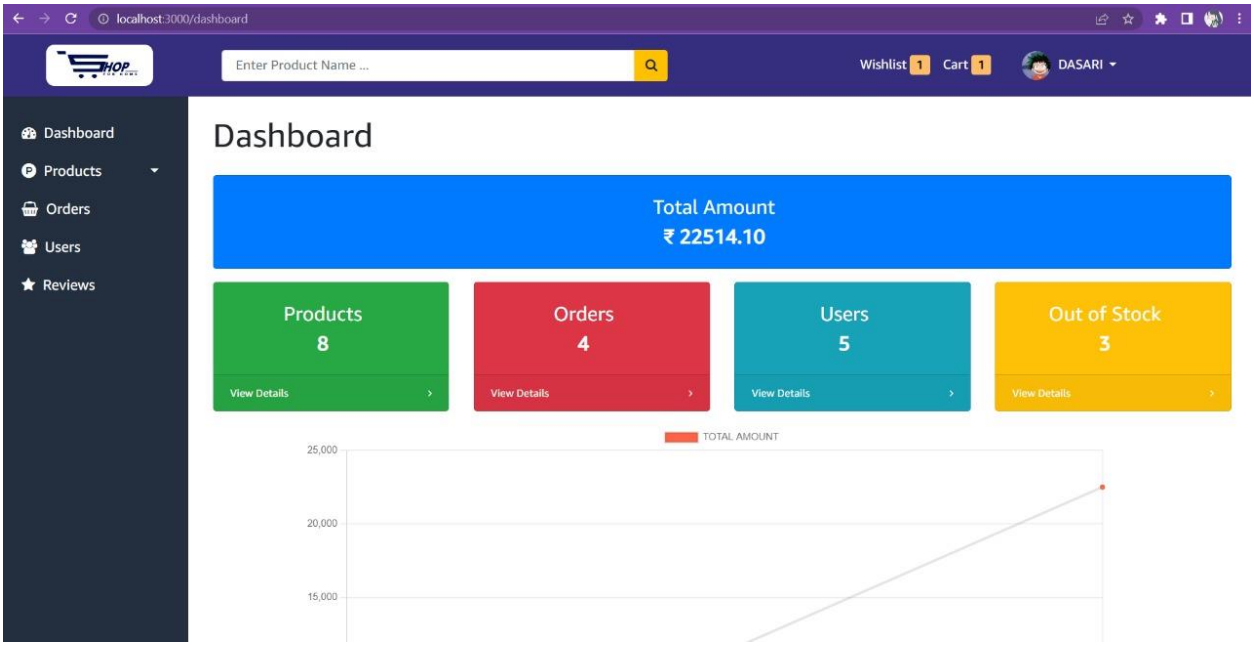

Fig 6.4 Admin Dashboard

Fig 6.5 Sales Report

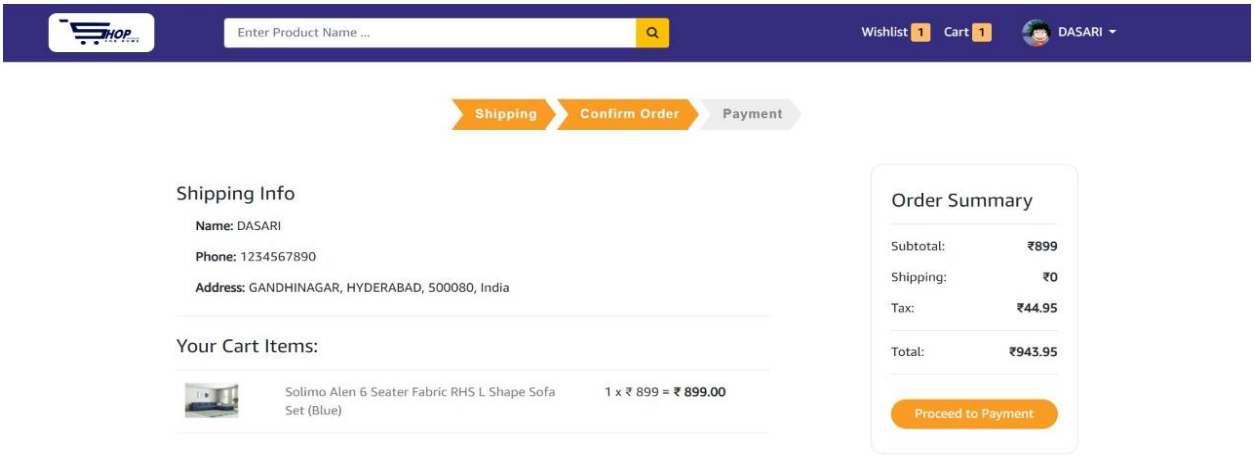

Fig 6.6 Payment Page

7. Git Repository

We store our website code in our git repo.

Following is our git repo link.

https://github.com/Capstone-Project-G8/ShopForHome

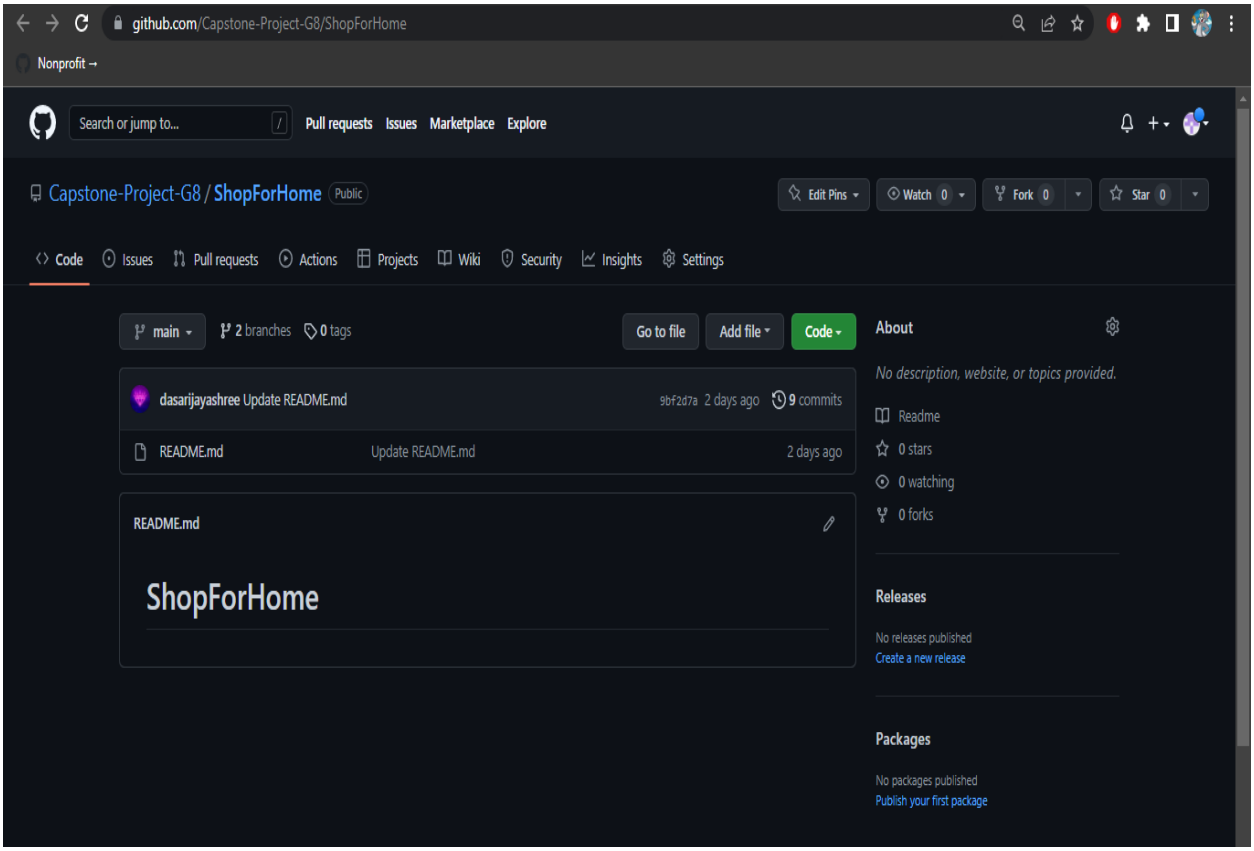

Fig 7.1 Git repo page

8. Swagger

Swagger allows you to describe the structure of your APIs so that machines can read them. The ability of APIs to describe their own structure is the root of all awesomeness in Swagger.

Swagger runs on : http://localhost:8000/api-docs/

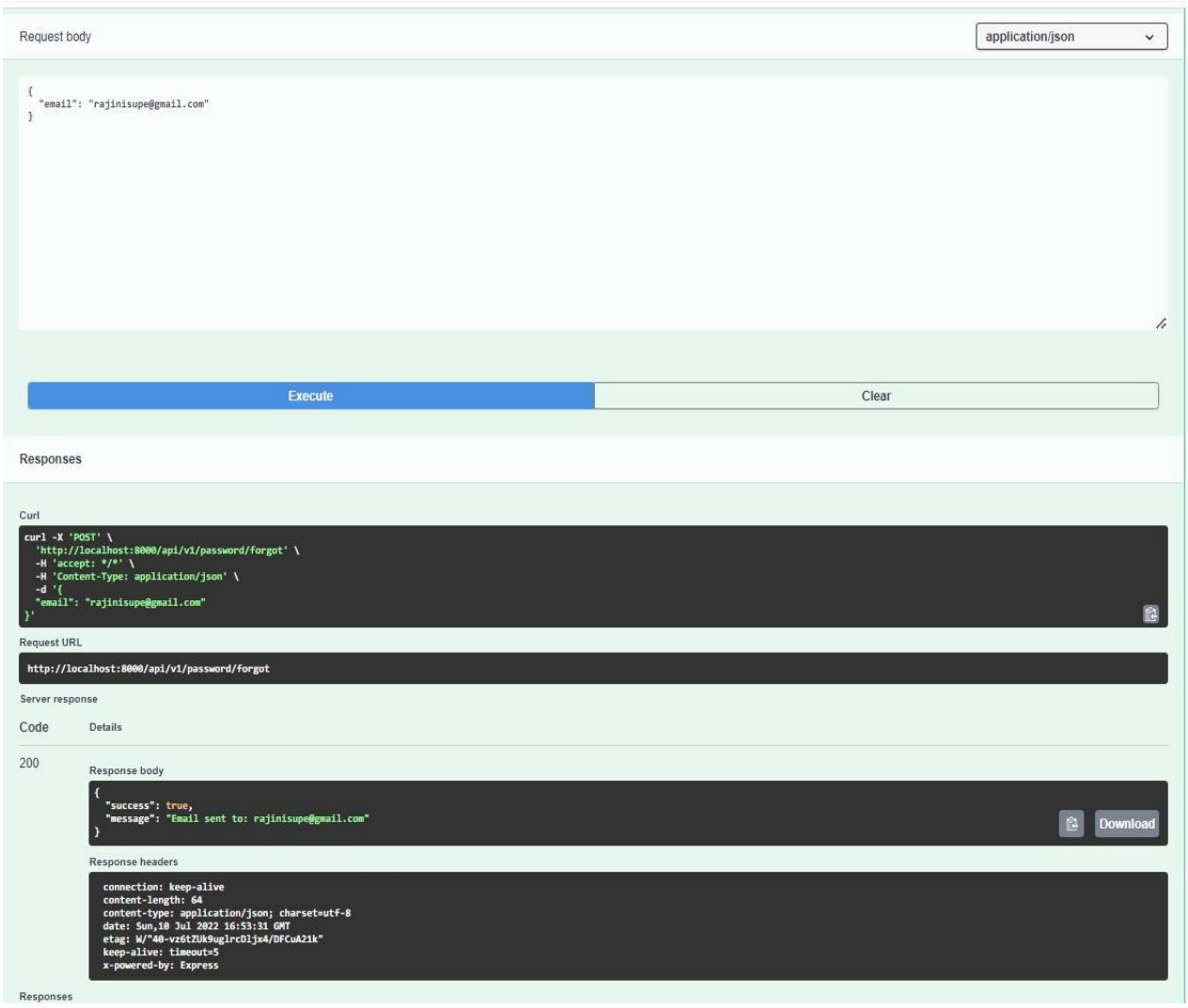

Fig 8.1 Swagger response

8. Conclusion

- The Web based Online shopping system is developed with the help of different tools such as React, MongoDB, Nodejs, Express, and devops and some agile methods.
- The developed system has met the objectives. Moreover, this system has high operational speed, and it is user-friendly.
- Being able to buy anytime anywhere, and widespread effects on economy and e-commerce.
- The system is valuable and usable in the perspective of any user.