Final Assessment

Ordering System for a Wholesale Shop

Faculty of Humanities & Social Sciences

Department of ICT

ICTC 3101.3

Web Application Development

AR97903

AF/19/15251

Table of Contents

Table of Figures

Introduction

The project documentation is about an ordering process management system developed especially for a selected wholesale grocery shop where grocery products are distributed to other small-scale retail shops around the area. Therefore, the web application is crafted to address the needs of the retail shop owners and optimize order processing and management of the business. The main functionalities of the web application are user login and registration, viewing products and browsing, filtering and searching products, adding to cart, checkout, and payment done by users whereas the admin is given the privileges of creating categories, products, updating and deleting products, managing order status. The website can be accessed from anywhere with the given domain. The web application is created based on MERN technology, the frontend website is built using React, and the backend of the project is a combination of ExpressJS and NodeJS, where MongoDB is used as the database to store all the related details. Finally, the completed website is deployed in AWS (Amazon Web Services) which is a widely used web hosting provider for websites.

Motivation

The web application is developed as an order management system for a wholesale shop to reduce the inconveniences caused during manual order processing and the limitations bounded by the lack of online presence. The main objective is to automate the ordering process and deliver the orders for the buyers without them having to physically come to the wholesale shop. The business processes can be carried out more efficiently this way and can cross-check every product and quantity of the order items. The business goal is to reduce confusion and making errors in order processing with the automation of processes. Apart from that, on the customer side, it is another user experience and a convenient process to order needed items delivered to their retail shops, which all can be done in one place and sitting. Thus, the web application optimizes the efficiency and user experience of the customers. With the higher volumes of order items, the wholesale business should be able to process the orders and payments within a short period without much deliberation of human labor. The need to build a website for the wholesale business arises from the competition among other wholesalers and to deliver products efficiently and effectively to a larger base of other small-scale businesses and grow the customer base through effective technology.

Problem Statement

Manual order processing has been a tiresome process that takes a long time and is inefficient in many ways such as taking time to write down the orders that are bulky because it is wholesale, require most of the labor, need good communication between the people who take down the order and the ones who load the products to the trucks since even if an item is loaded or not loaded by mistake it is considered to be a vital error in the business as it costs a lot of money for the business and the customer. Therefore, multiple copies of the same order need to be taken for each employee, at the cashier, at the warehouse, etc. The customers will have to come by themselves and spend a lot of time ordering products, doing transactions, and waiting till the products are loaded in the premises which is hectic and hinder the business and employees of the business. To minimize the inconveniences and limitations of the existing manual and local order management system, the need for an online web-based order management system arose.

Objectives

The main objective of this project is to overcome the challenges and limitations faced by the business to increase profit, and efficiency and thus to provide customers the best service from the business.

- Optimize efficiency - Streamline internal processes, such as order fulfillment, updating and deleting products, creating new categories and products, and delivery management, to improve operational efficiency and reduce costs. Efficiency is to be enhanced throughout the business processes from easy handling of ordering processes.
- Increase sales and revenue - Enable customers to easily browse and explore products, leading to increased sales. Implement features like filtering products and categories and searching products to drive customer engagement. Lack of errors in order management is a vital feature that needs to be embedded into the application.
- Better order process management - Automate the order processing workflow, reducing manual intervention and potential errors. Enable real-time tracking of order status to the customers. Need to be able to update orders status by the administrator.
- Provide satisfactory service to the customers - Enhance the overall shopping experience for customers by providing a user-friendly and intuitive interface, easy product browsing, and a smooth order placement process.

- Reduce unnecessary workload and labor – enable customers to browse and place orders, then the application should be able to give the checkout option displaying the total with the card transaction portal to process orders smoothly. The employees should be able to view the successfully placed orders and therefore start processing them.
- Prevent miscommunication and mistakes - enable seamless processing of orders by connecting and updating each employee with the respective orders placed by customers.
- Online transaction processing – card payments can be done directly through the web application and therefore is easy for both parties in handling transactions mostly of higher amounts.

System Overview

The main functionality of the order process management web application is to manage seamless and efficient processes from placing orders to payment by the user and on the other hand controlling those orders by the business. The purpose of this website is to enable small-scale retail shops to browse and place orders from the wholesale grocery shop regularly with ease from a remote location, get payments done by card and get the products delivered to them without having to visit the wholesale warehouse themselves.

The users of the system and their roles

1. Admin – admin has the privileges to manage the database and access to database, order status management, create, update, and delete products, manage the overall website, assign employees of the shop access to perform CRUD operations on the products
2. Users – users are any potential customer who has not yet registered to the shop, or an existing customer that can log in and purchase products. Logged-in customers can update their profiles and delivery address
3. Employees of the business – employees have the same access to perform on products such as creating, updating, and deleting products

The main functionalities identified in the system are as follows.

1. Browse the products – the interface is designed to provide intuitive navigation and guidance to choose products carefully with filters such as filter by category and price
2. Authentication – login, register, reset password
3. Search for products – can search for products with the same name or have in the description of the product
4. Dashboard of users – update profile, check the status of orders and previous orders
5. Dashboard of admin/employees – CRUD operations of products take place here. Employees or the admin can create categories and products, view products, update products, and delete products.
6. Order management – admin or employees can change the status of orders such as processing, delivered, and canceled.
7. Add to cart – users can add products to the cart and get a total of items
8. Payment portal – users are given the chance to pay using a card to fulfill the transaction

The Homepage/ Landing Page

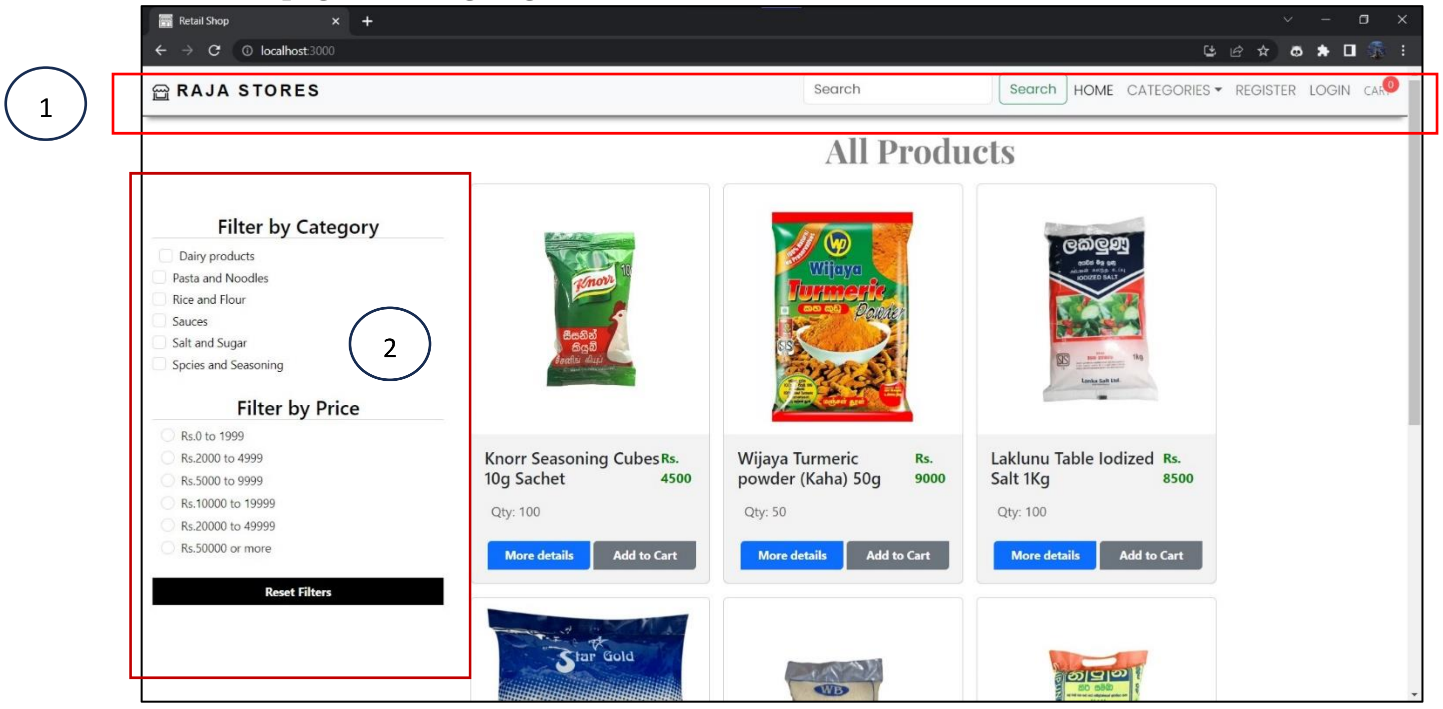


Figure 1: Homepage

The homepage consists of:

- Navigation bar or header
- Filters
- Products
- Footer at the bottom

<u>Header</u>

The header section acts as the navigation bar for the website where users can navigate between different web pages they want to navigate to.

1. Search bar is used to search products using a keyword

2. Home is the homepage

3. Categories allow having a dropdown to select products among the available categories

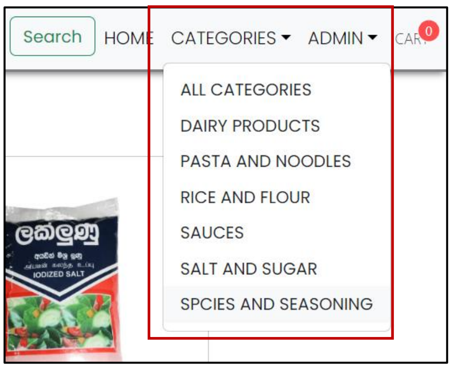


Figure 2: categories dropdown

When the user selects all categories, they are navigated to the page below and can see the available categories and choose accordingly.

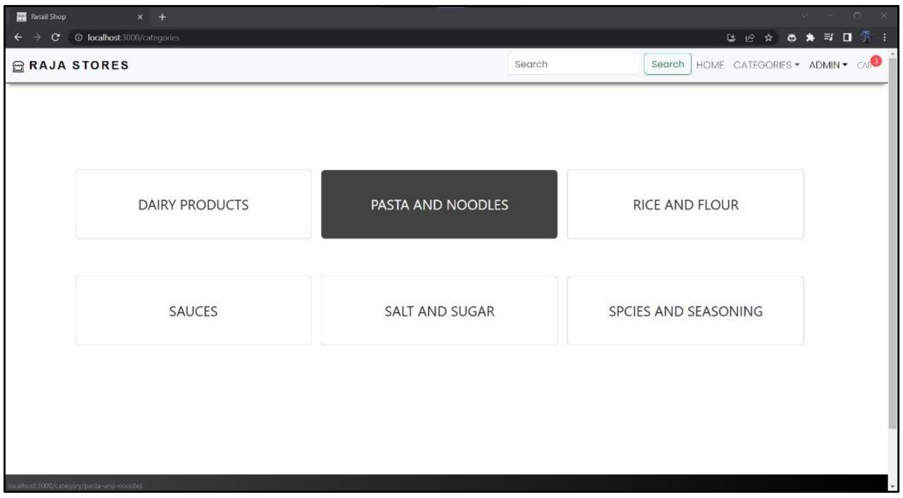


Figure 3: all categories

Upon choosing one category either from the cards from above or from the dropdown list, users can be able to categorize products according to their category and available product counts per category are also shown.

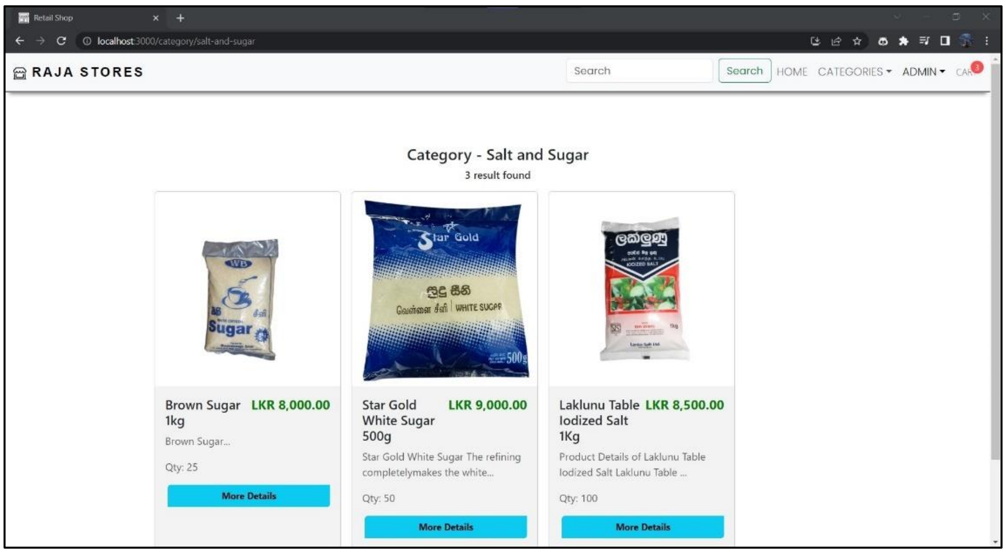

Retail Shop
localhost:3000/category/salt-and-sugar
RAJA STORES
Search
Search
HOME
CATEGORIES
ADMIN
CART
Category - Salt and Sugar
3 result found
Brown Sugar 1kg
LKR 8,000.00
Brown Sugar...
Qty: 25
More Details
Star Gold White Sugar 500g
LKR 9,000.00
Star Gold White Sugar The refining completelymakes the white...
Qty: 50
More Details
Laklunu Table Iodized Salt 1Kg
LKR 8,500.00
Product Details of Laklunu Table Iodized Salt Laklunu Table ...
Qty: 100
More Details

Figure 4: one category

Users can click more details to see details about products and add them to the cart if needed.

4. If the user is not logged in and is a guest then there are 2 options to choose as register or login. If the user is logged in then the user's name is displayed along with the dropdown list to select from the dashboard and logout. Users are directed to the login page if logout is selected.

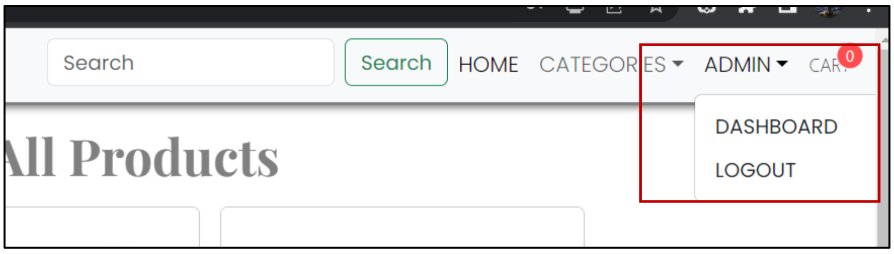


Figure 5: user dropdown menu

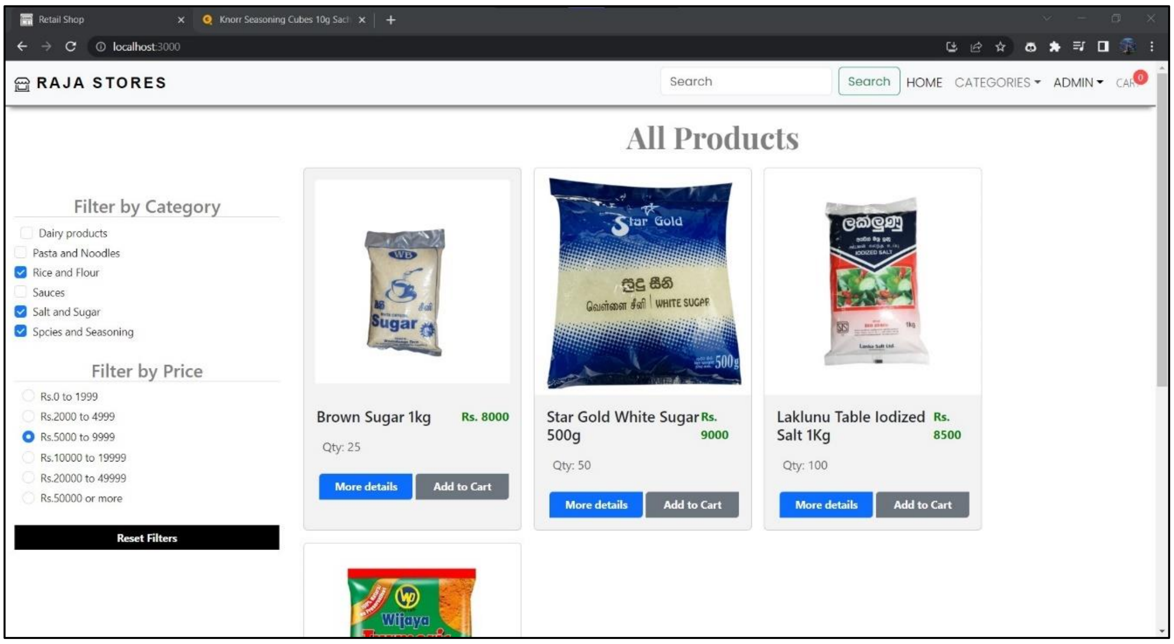
<u>Filters</u>

Filters are used to provide users more chances to find the products they want and act as an optimization method to ease the browsing process.

There are 2 filters as

1. Filter by category
2. Filter by Price

Therefore, users can choose from more than one category according to a desired price range.



Filter by Category

- [ ] Dairy products
- [ ] Pasta and Noodles
- [x] Rice and Flour
- [ ] Sauces
- [x] Salt and Sugar
- [x] Species and Seasoning

Filter by Price

- Rs.0 to 1999
- Rs.2000 to 4999
- [✓ Rs.5000 to 9999]
- Rs.10000 to 19999
- Rs.20000 to 49999
- Rs.50000 or more

Reset Filters

Brown Sugar 1kg Rs. 8000
Qty: 25
More details Add to Cart

Star Gold White Sugar 500g Rs. 9000
Qty: 50
More details Add to Cart

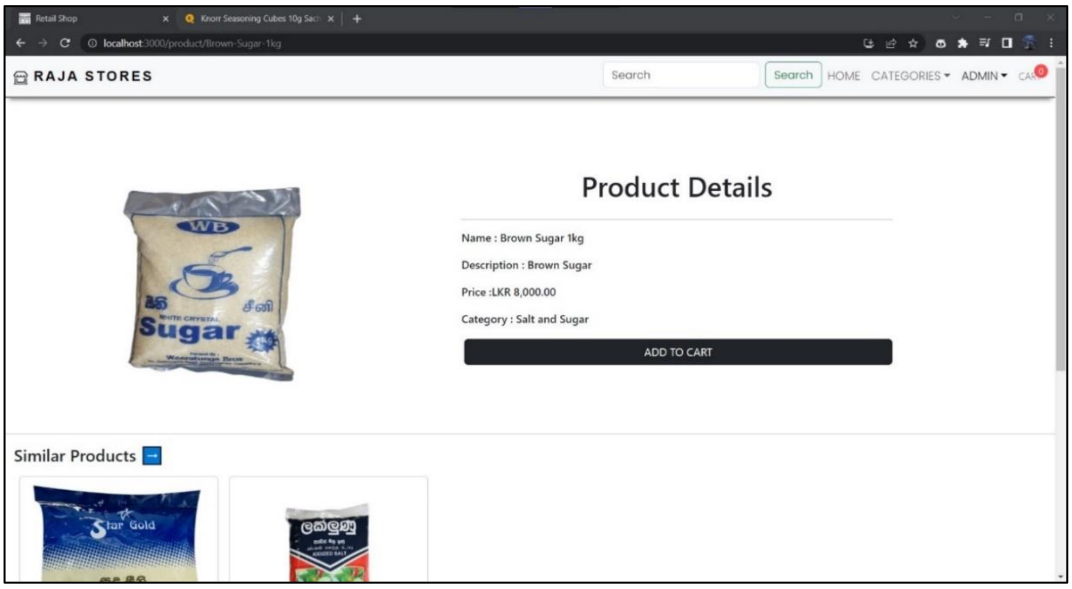
Laklunu Table Iodized Salt 1Kg Rs. 8500
Qty: 100
More details Add to Cart

Figure 6: filters

Products

Product cards are used to envelop the main product details needed and users can click on more details button to check more details about the product. Users can add products to the cart using add to cart button directly otherwise. Mainly the product name, the quantity provided per each product as a whole, and the overall price of those quantities are displayed on the card.

When the user clicks on more details, he/she is directed to the page below. There also the user can add items to the cart and view similar products to the product given above with the same category of products.



Product Details

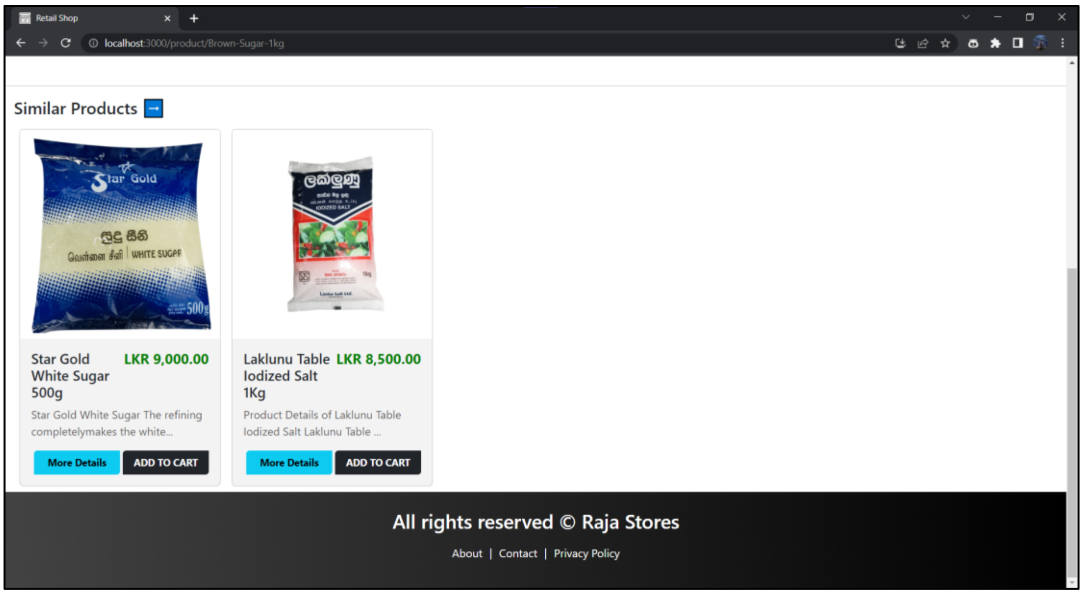
Name : Brown Sugar 1kg

Description : Brown Sugar

Price :LKR 8,000.00

Category : Salt and Sugar

Figure 7: More details page



Star Gold White Sugar 500g LKR 9,000.00

Star Gold White Sugar The refining completelymakes the white...

Laklunu Table Iodized Salt 1Kg LKR 8,500.00

Product Details of Laklunu Table Iodized Salt Laklunu Table ...

All rights reserved © Raja Stores

Figure 8: similar products

To optimize the loading process of the webpage, the products loaded to the homepage are limited to 6 products, and upon reaching the end the user can click on load more button to expand and load other products.

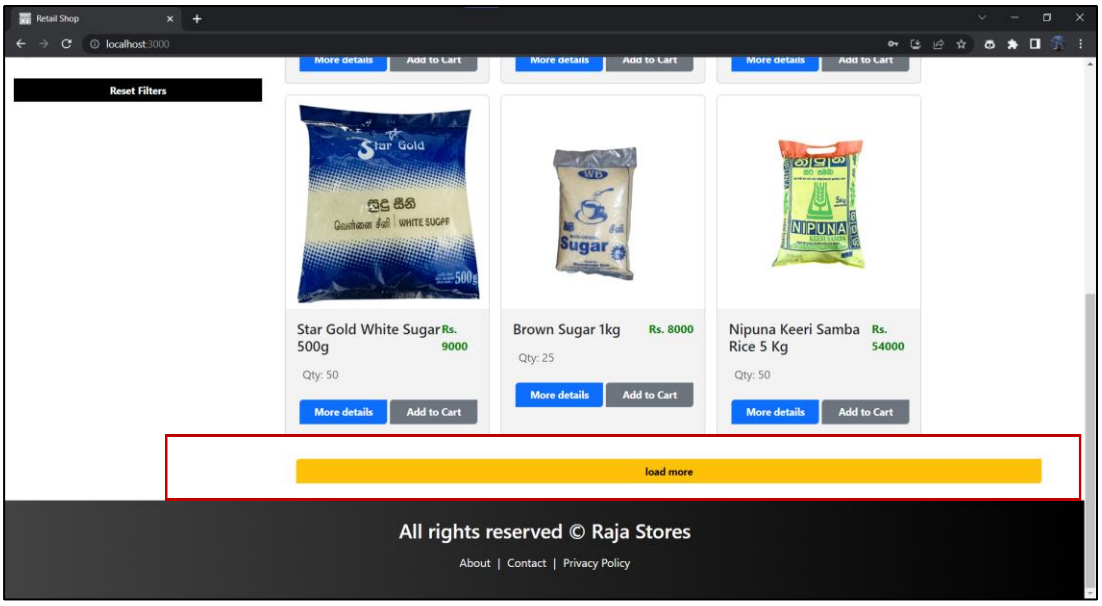
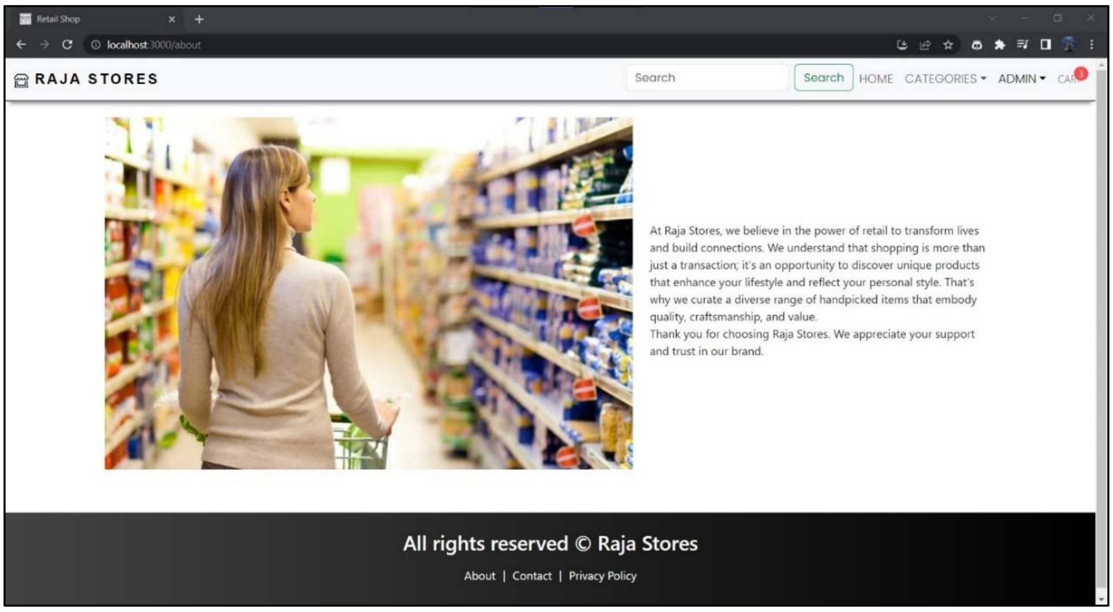


Figure 9: load more

Footer

The footer consists of pages about the general details, contact details, and privacy policy of the website.

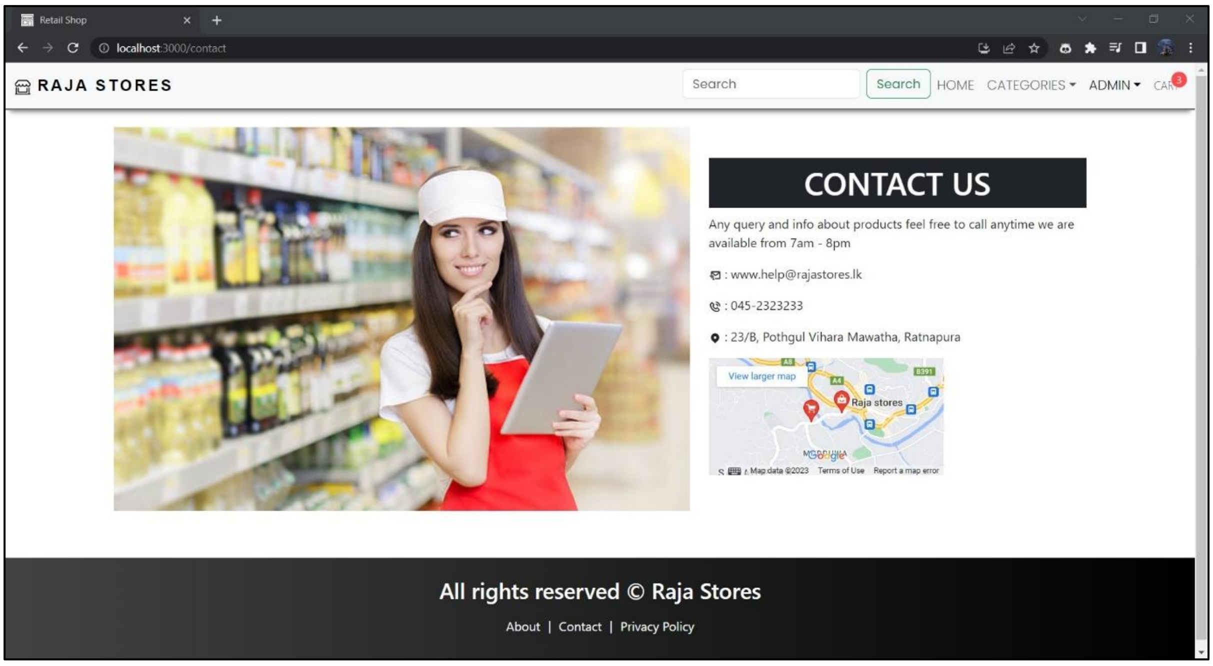
About



At Raja Stores, we believe in the power of retail to transform lives and build connections. We understand that shopping is more than just a transaction; it's an opportunity to discover unique products that enhance your lifestyle and reflect your personal style. That's why we curate a diverse range of handpicked items that embody quality, craftsmanship, and value.
Thank you for choosing Raja Stores. We appreciate your support and trust in our brand.

Figure 10: About

Contact



CONTACT US

Any query and info about products feel free to call anytime we are available from 7am - 8pm

: www.help@rajastores.lk

: 045-2323233

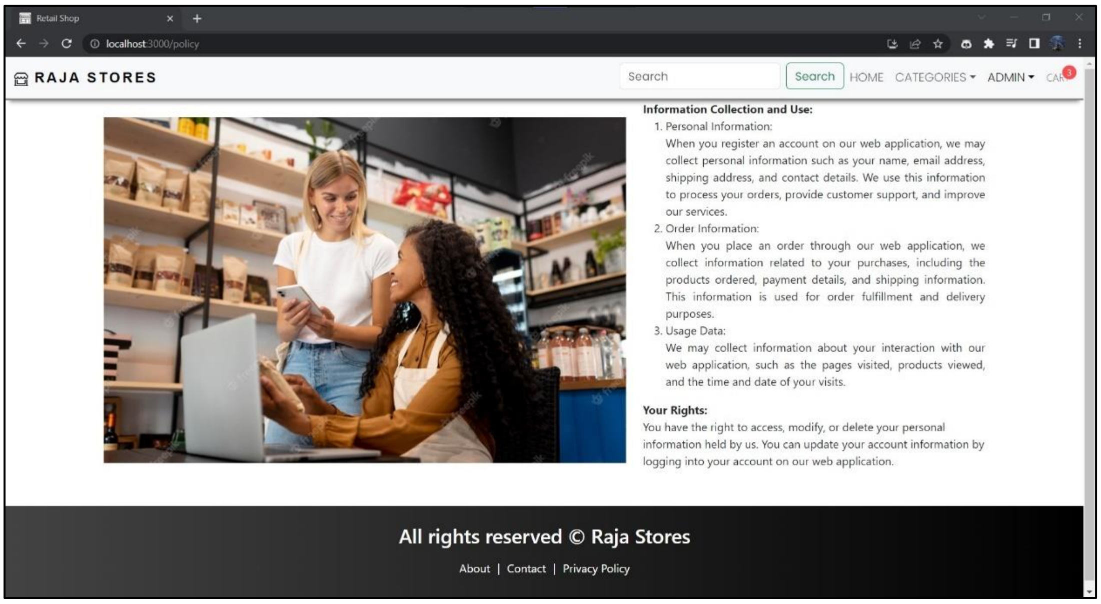
: 23/B, Pothgul Vihara Mawatha, Ratnapura

All rights reserved © Raja Stores

About | Contact | Privacy Policy

Figure 11: Contact

Privacy Policy



Information Collection and Use:

1. Personal Information:
   When you register an account on our web application, we may collect personal information such as your name, email address, shipping address, and contact details. We use this information to process your orders, provide customer support, and improve our services.
2. Order Information:
   When you place an order through our web application, we collect information related to your purchases, including the products ordered, payment details, and shipping information. This information is used for order fulfillment and delivery purposes.
3. Usage Data:
   We may collect information about your interaction with our web application, such as the pages visited, products viewed, and the time and date of your visits.

Your Rights:

You have the right to access, modify, or delete your personal information held by us. You can update your account information by logging into your account on our web application.

All rights reserved © Raja Stores

About | Contact | Privacy Policy

Figure 12: privacy policy

Authentication

The authentication process allows users to register with the business and then log in with the correct authorizations and place their orders to pay securely. Also, it prevents unauthorized access to the system. Users can store their details and summary details of all the transactions securely stored on the website therefore if anything goes wrong, they have valid evidence of every transaction securely stored. Another functionality of authentication is to assign access to certain functionalities and have distinct privileges to certain users. Therefore, employees and admins of the system can manage orders and perform CRUD operations on products that are not accessible to customers.

Registration

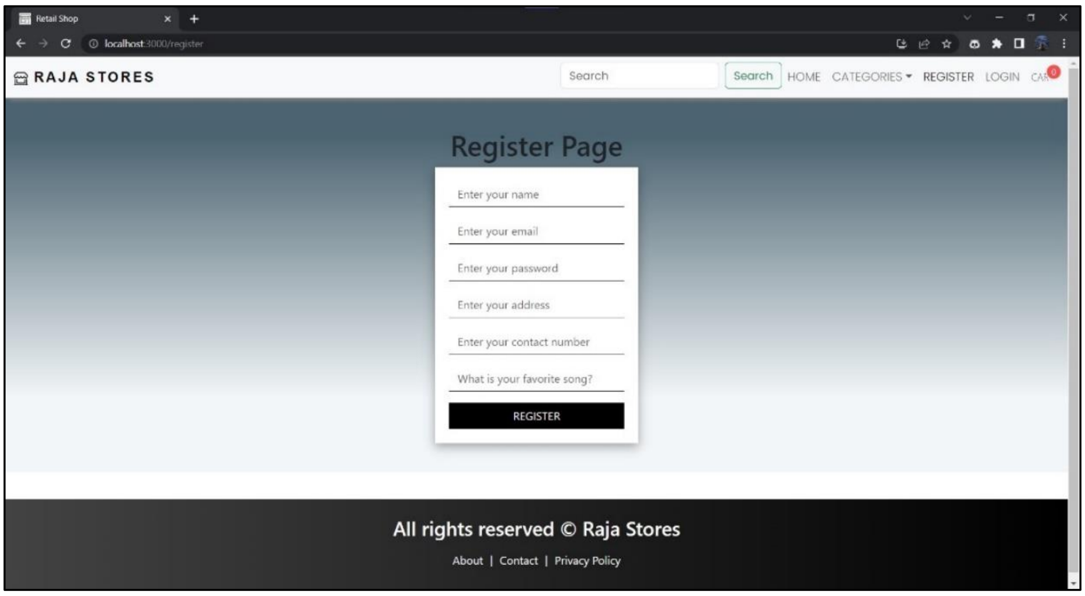


Figure 13: registration page

Users should fill in the given fields to register. The last field is used as a reference when resetting the password. Other fields can be filled as normal and cannot be kept empty as it is tested with the register button click if all the fields are empty or not. The users should provide a valid email according to the basic email structure.

1. Name of the user
2. Email address
3. Password
4. Address – delivery location can be changed afterward also
5. Contact number
6. Answer – to be used when resetting the password

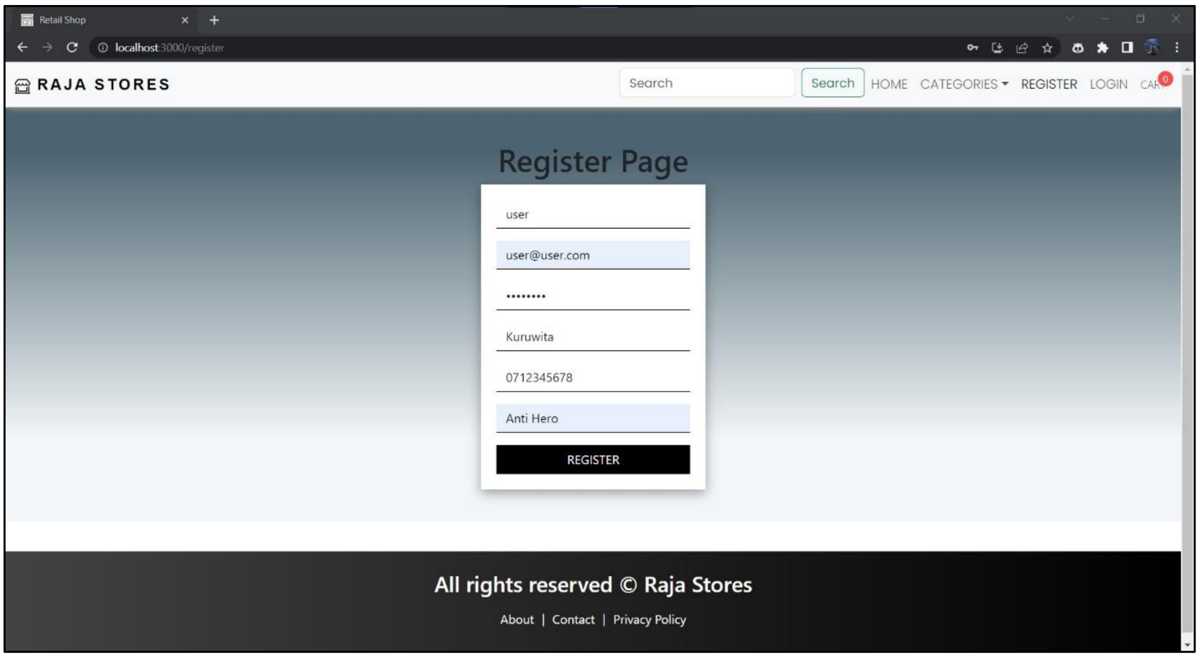


Figure 14: Registration filled

<u>Login</u>

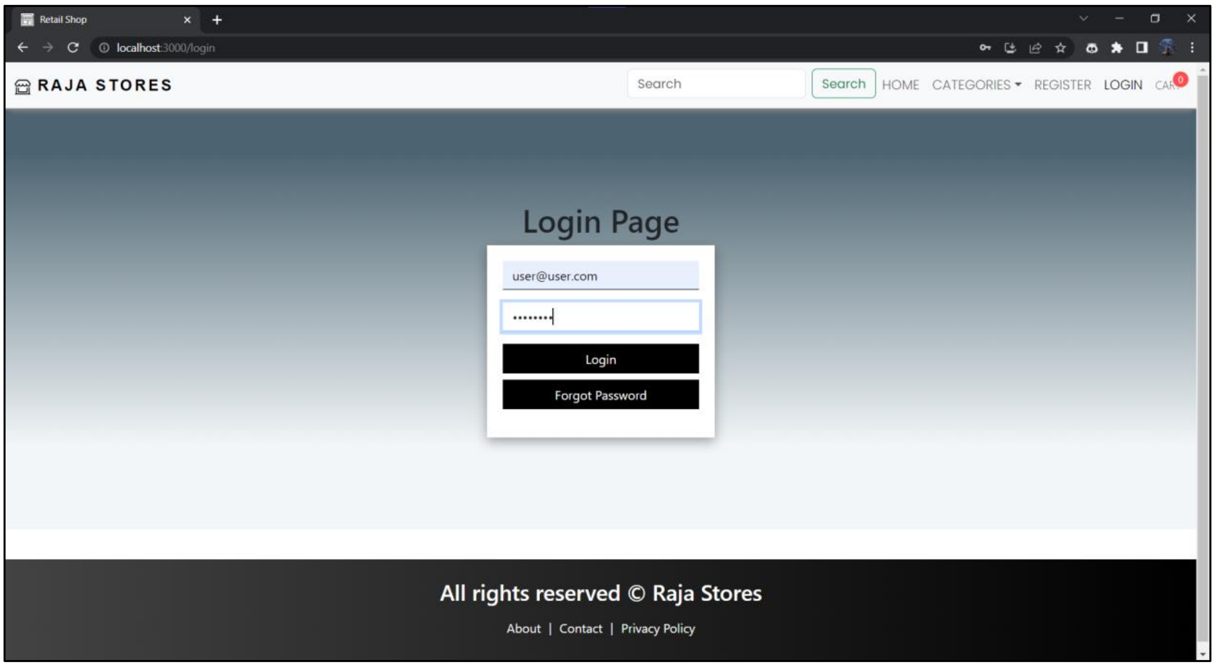


Figure 15: Login page

The login page consists of 2 fields email address and password. Once the user enters the correct email address and password, he/she is directed to the page that they were browsing first. For example, if the user was on the cart page before login, then he/she is directed to the cart page. Likewise, the login functionality is developed.

Forgot password button is used to direct to the reset password page.

Reset Password

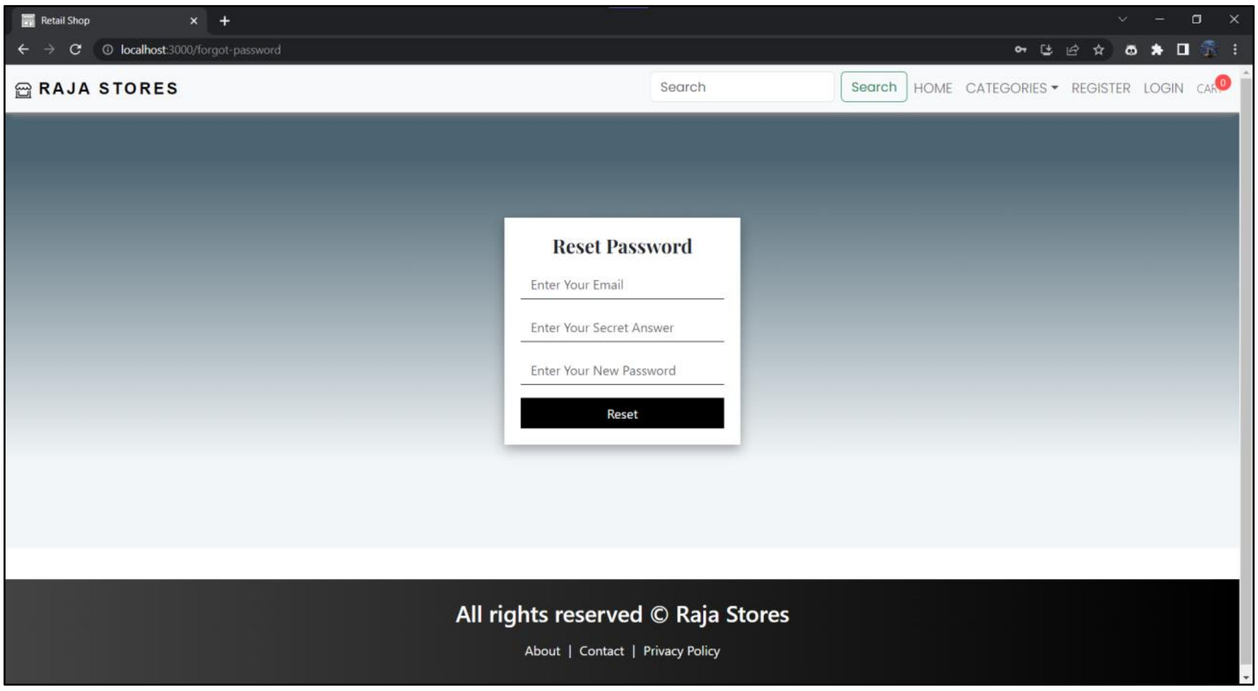


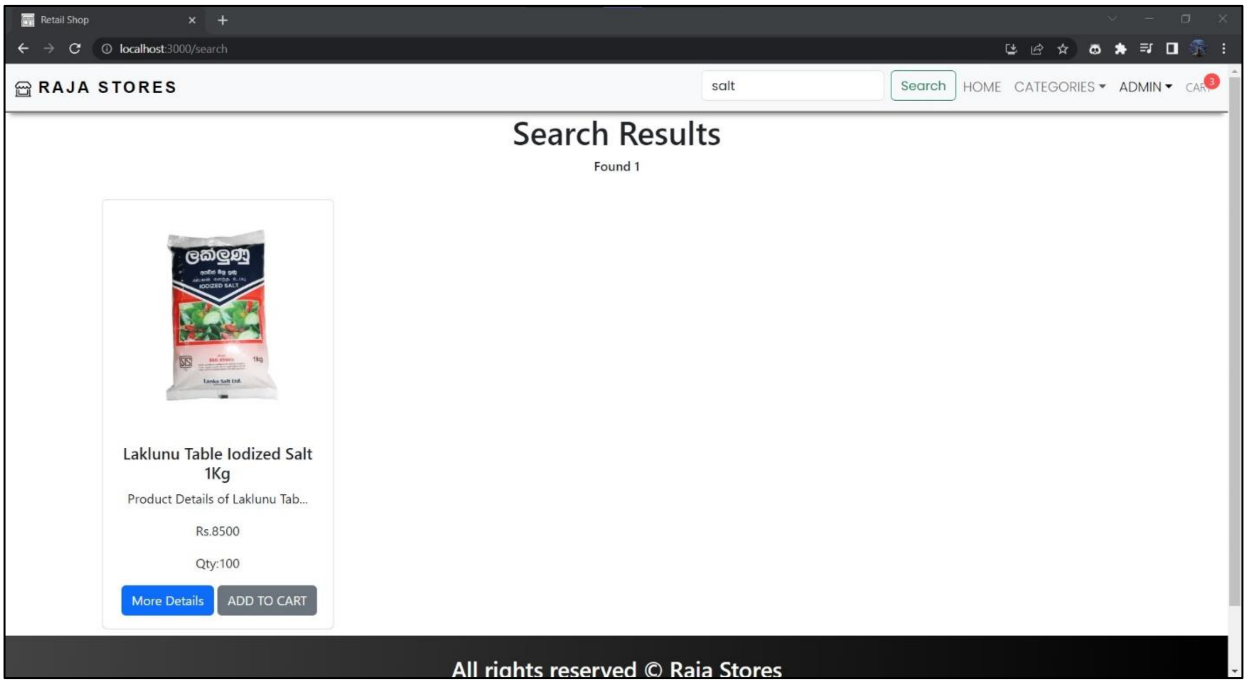
Figure 16: Reset password page

User needs to provide the email address, answer that is the question asked when registering as "What is your favorite song", and the new password to be used to reset the fields. Then the user can click on the reset button to reset the new password. The user is directed to the login page to once again enter the email address and the newly updated password to successfully log in.

The users can log out from the website by clicking the dropdown menu under the username. By clicking logout, the user is directed to the login page.

Search Products

Users can search for any product using the search bar in the header. The keyword entered is then searched against identifying if there are any products equal to the keyword or the description entered. Then the user can see the results of the products. This is another optimization functionality used to save the time of the customers of the business.



Search Results

Found 1

Laklunu Table Iodized Salt 1Kg

Product Details of Laklunu Tab...

Rs.8500

Qty:100

Figure 17: Search page

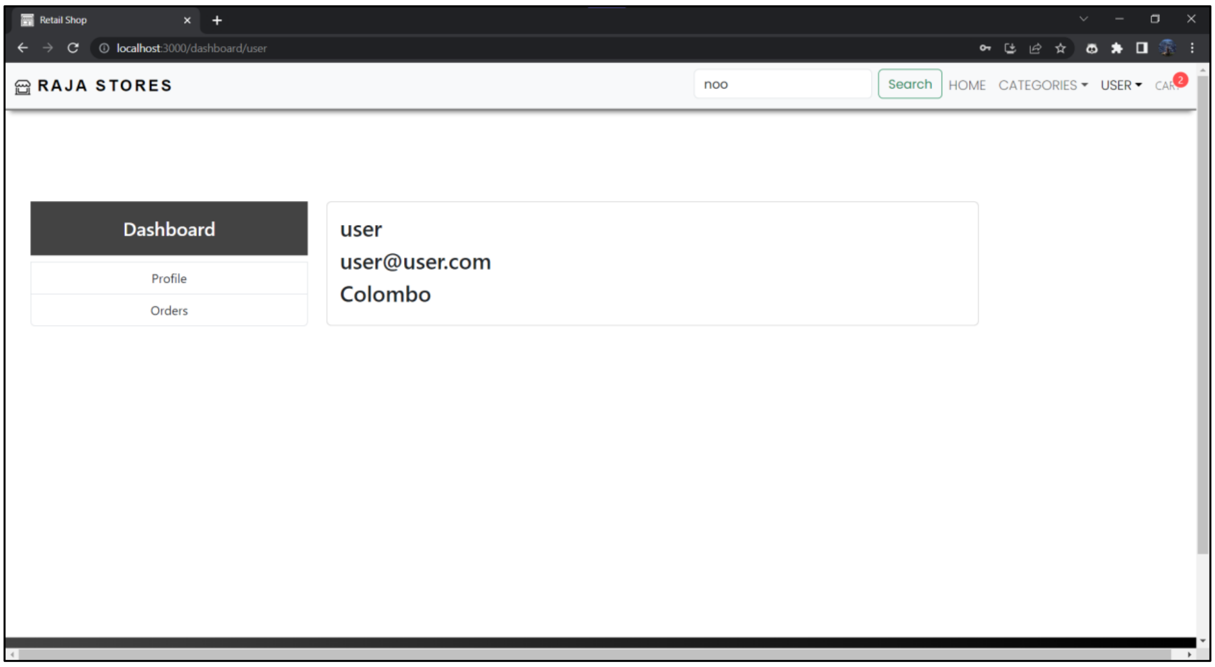
Dashboard of Users

The users who have already registered customers to the website and also who are already logged in and have access to the dashboard functionality. Upon selecting the user name in the header, the user can select the dashboard from the dropdown menu.

The user dashboard has 2 navigation panes

1. Profile
2. Orders

The following is the first screen user sees upon entering the dashboard. Then from the side pane user can select any functionality.

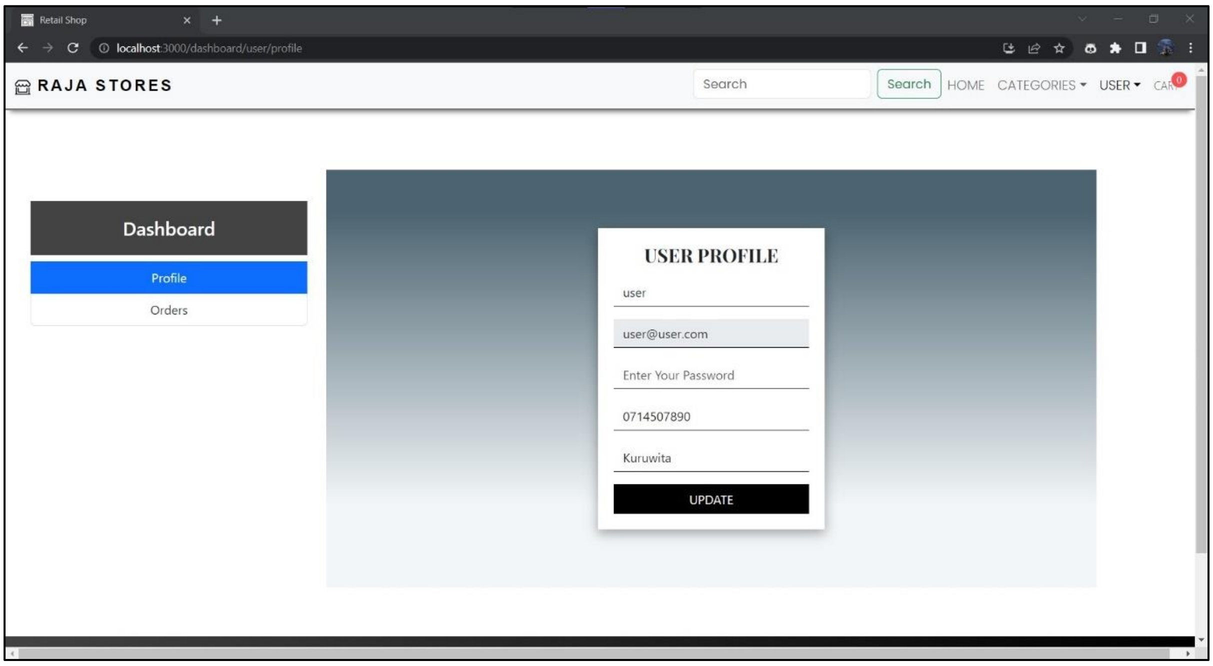


user
user@user.com
Colombo

Figure 18: User dashboard

1. Profile

Users can update any field of personal information including passwords using the profile panel.

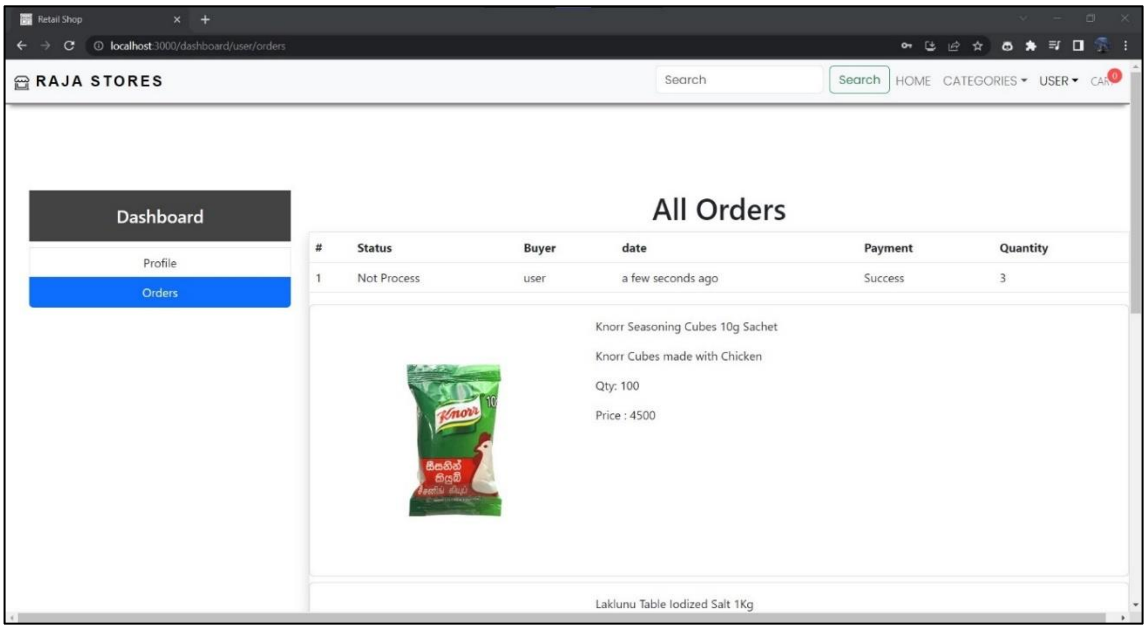


user
user@user.com
Enter Your Password
0714507890
Kuruwita

Figure 19: User profile

2. Orders

Every order transaction can be viewed in this panel. It records the history of records and the user can track the orders in real-time checking the status of the order.



All Orders

#	Status	Buyer	date	Payment	Quantity
1	Not Process	user	a few seconds ago	Success	3

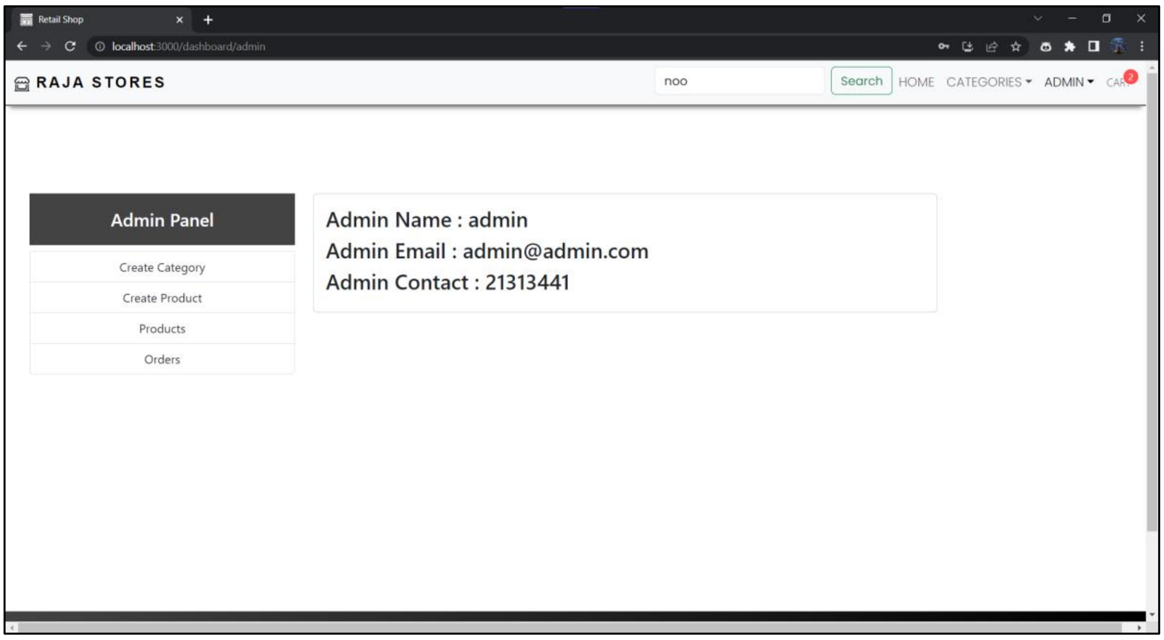
Knorr Seasoning Cubes 10g Sachet
Knorr Cubes made with Chicken
Qty: 100
Price : 4500

Laklunu Table Iodized Salt 1Kg

Figure 20: User orders

Dashboard of Admin/Employees

The dashboard of admin/employees is different and has more features than a typical user with more access to the system. All CRUD (Create, Read, Update, Delete) operations of products are done here. Apart from that order management is also done by admin/employees here.

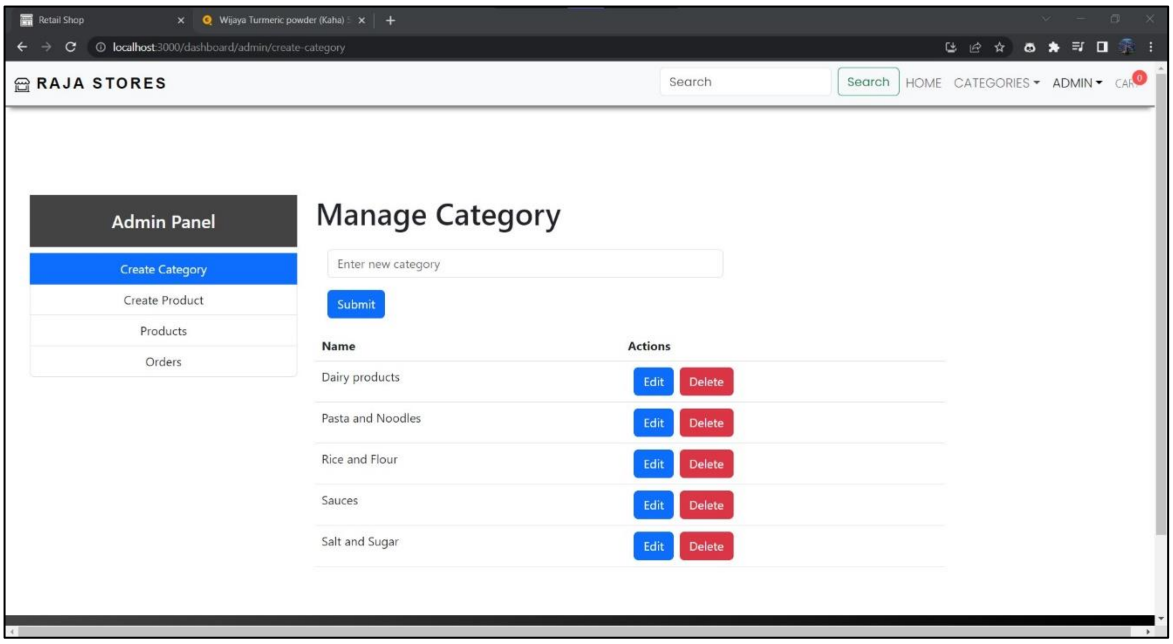


Admin Name : admin
Admin Email : admin@admin.com
Admin Contact : 21313441

Figure 21: Admin dashboard

Figure 22: Admin dashboard

1. Create a Category



Manage Category

Name	Actions
Dairy products	Edit Delete
Pasta and Noodles	Edit Delete
Rice and Flour	Edit Delete
Sauces	Edit Delete
Salt and Sugar	Edit Delete

Figure 23: Create a Category

Admin/employee can view the available categories below and add any other new category if needed by entering the category needed in the “Enter new category” text field and clicking on the “Submit” button.

The category names can be edited or a category can be entirely deleted using the “Edit” and “Delete” buttons corresponding to each category.

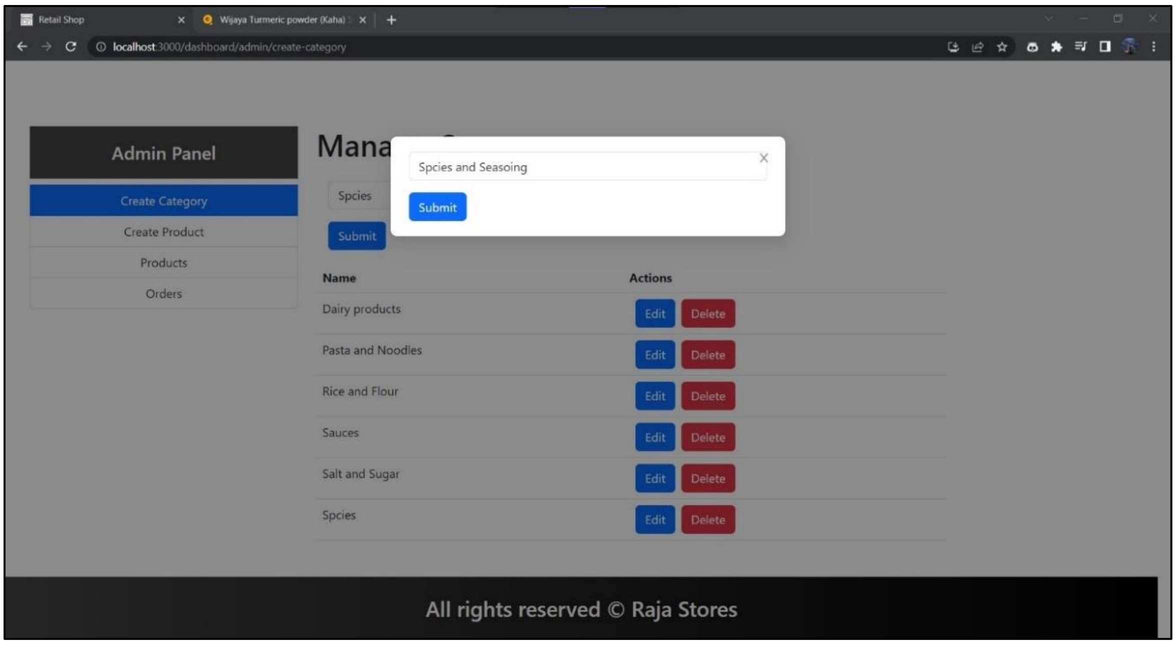


Figure 24: Update Category

2. Create a Product

This functionality allows to creation new products under a category.

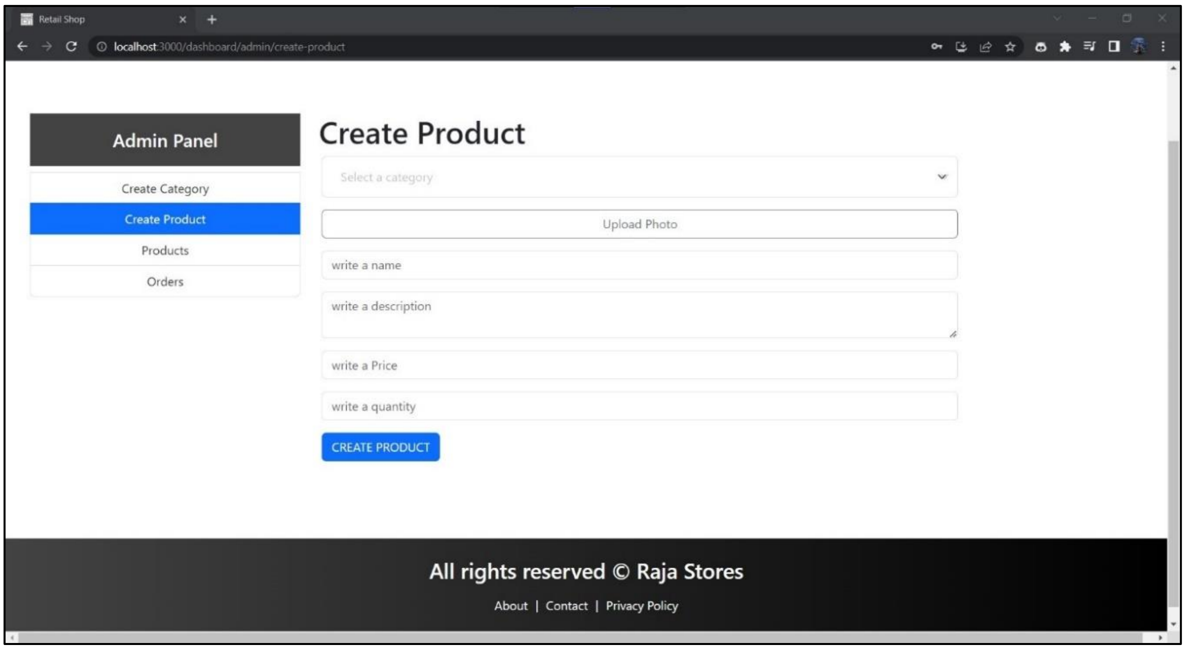
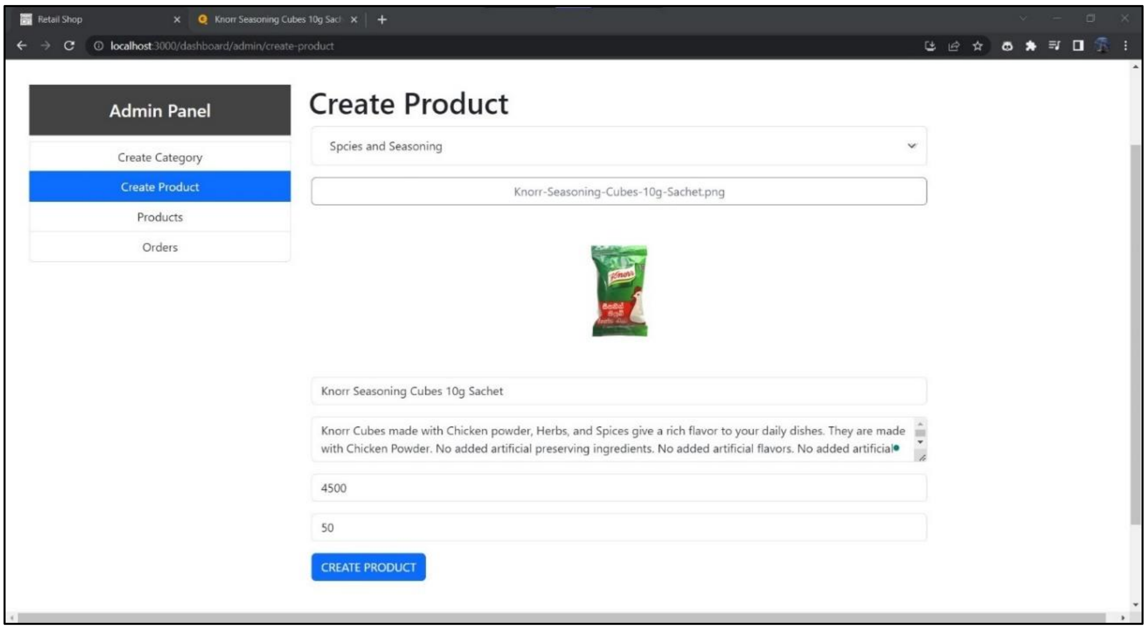


Figure 25: Create a product page

Upon entering the required details and attaching a suitable picture under 1MB size as the photo, a new product can be created. The admin/employee is directed to the Product panel once the new product is created. Then once the screen is refreshed, the newly created product can also be viewed.


Retail Shop
Knorr Seasoning Cubes 10g Sach
localhost:3000/dashboard/admin/create-product
Admin Panel
Create Category
Create Product
Products
Orders

Create Product

Spcies and Seasoning

Knorr-Seasoning-Cubes-10g-Sachet.png

Knorr Seasoning Cubes 10g Sachet

Knorr Cubes made with Chicken powder, Herbs, and Spices give a rich flavor to your daily dishes. They are made with Chicken Powder. No added artificial preserving ingredients. No added artificial flavors. No added artificial

4500

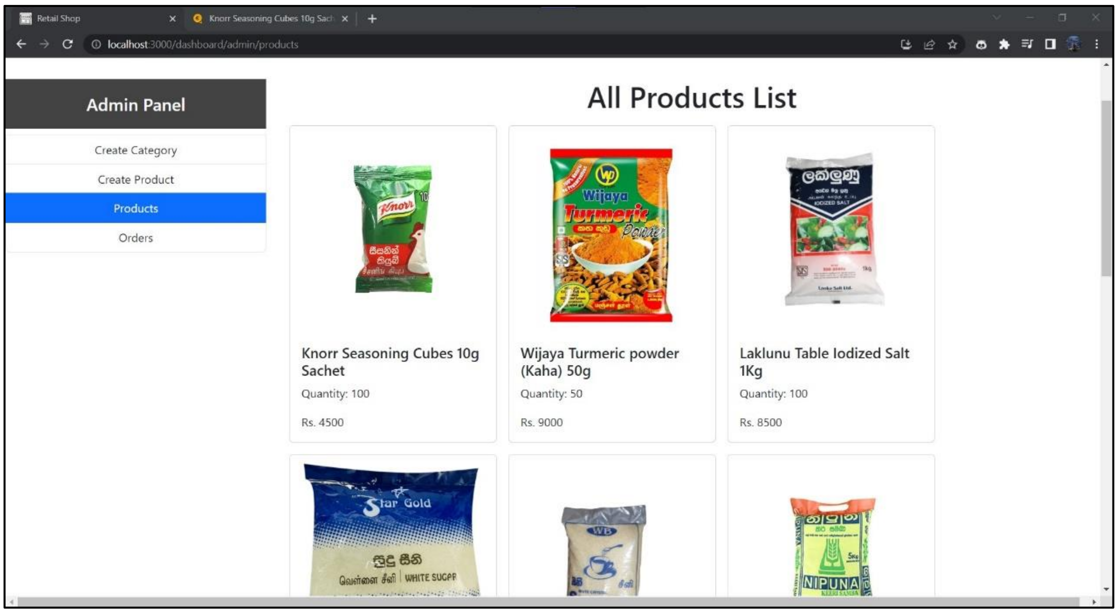
50

CREATE PRODUCT

Figure 26: Create product

3. View Products

This functionality allows the admin/employee to view products stored on the database.


Retail Shop
Knorr Seasoning Cubes 10g Sach
localhost:3000/dashboard/admin/products
Admin Panel
Create Category
Create Product
Products
Orders

All Products List

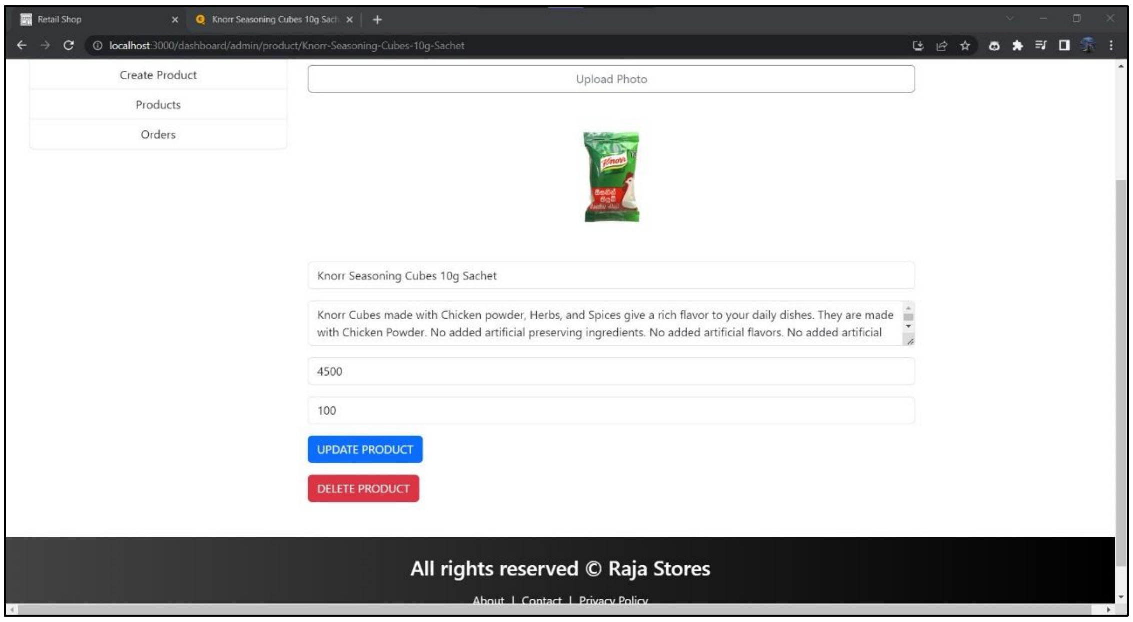
Knorr Seasoning Cubes 10g Sachet
Quantity: 100
Rs. 4500

Wijaya Turmeric powder (Kaha) 50g
Quantity: 50
Rs. 9000

Laklunu Table Iodized Salt 1Kg
Quantity: 100
Rs. 8500

Figure 27: List products

The name, quantity provided per product, and price of the whole package can be viewed in this panel. Once the admin/employee, clicks on one of the products, then they are directed to the following page.


Retail Shop
Knorr Seasoning Cubes 10g Sachet
localhost:3000/dashboard/admin/product/Knorr-Seasoning-Cubes-10g-Sachet
Create Product
Products
Orders
Upload Photo
UPDATE PRODUCT
DELETE PRODUCT
All rights reserved © Raja Stores
About | Contact | Privacy Policy

Knorr Seasoning Cubes 10g Sachet

Knorr Cubes made with Chicken powder, Herbs, and Spices give a rich flavor to your daily dishes. They are made with Chicken Powder. No added artificial preserving ingredients. No added artificial flavors. No added artificial

4500

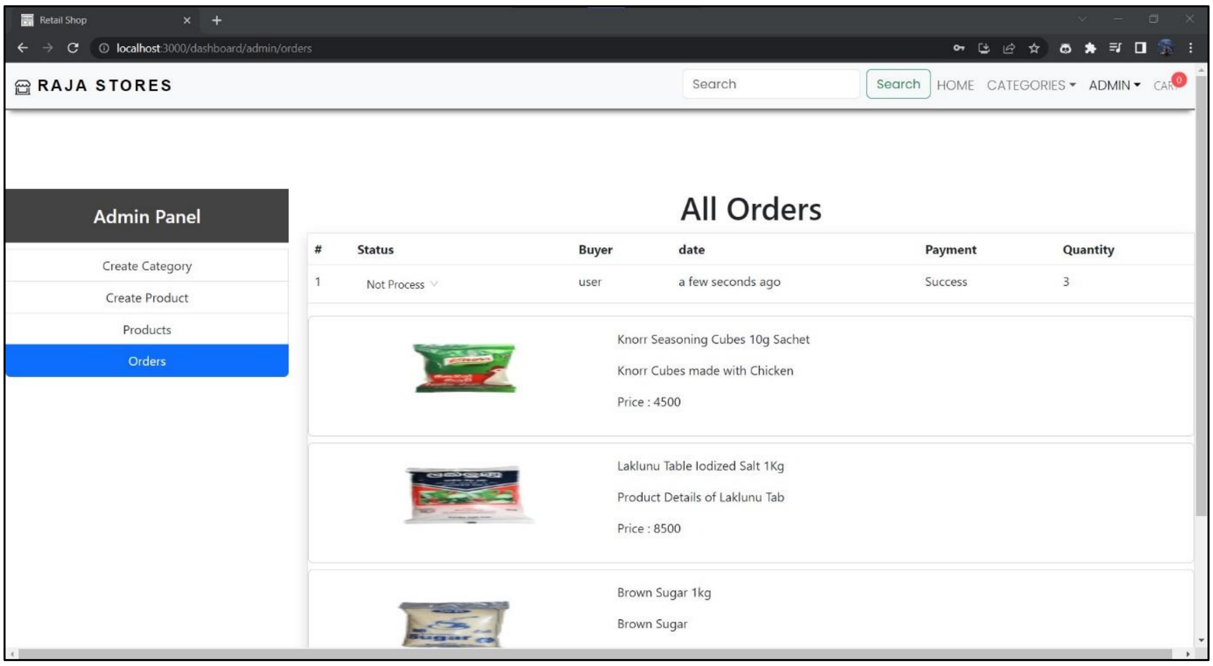
100

Figure 28: Update, and delete the product

Here, the admin/employee can update and delete the products and they are directed to the products page again and need to refresh the page to see the updated product or deleted product.

Orders Management

Order management is another functionality available for admin/employees. They can view all the orders placed by customers and start by processing them and updating the status of each order.



All Orders

#	Status	Buyer	date	Payment	Quantity
1	Not Process	user	a few seconds ago	Success	3

Knorr Seasoning Cubes 10g Sachet

Knorr Cubes made with Chicken

Price : 4500

Laklunu Table Iodized Salt 1Kg

Product Details of Laklunu Tab

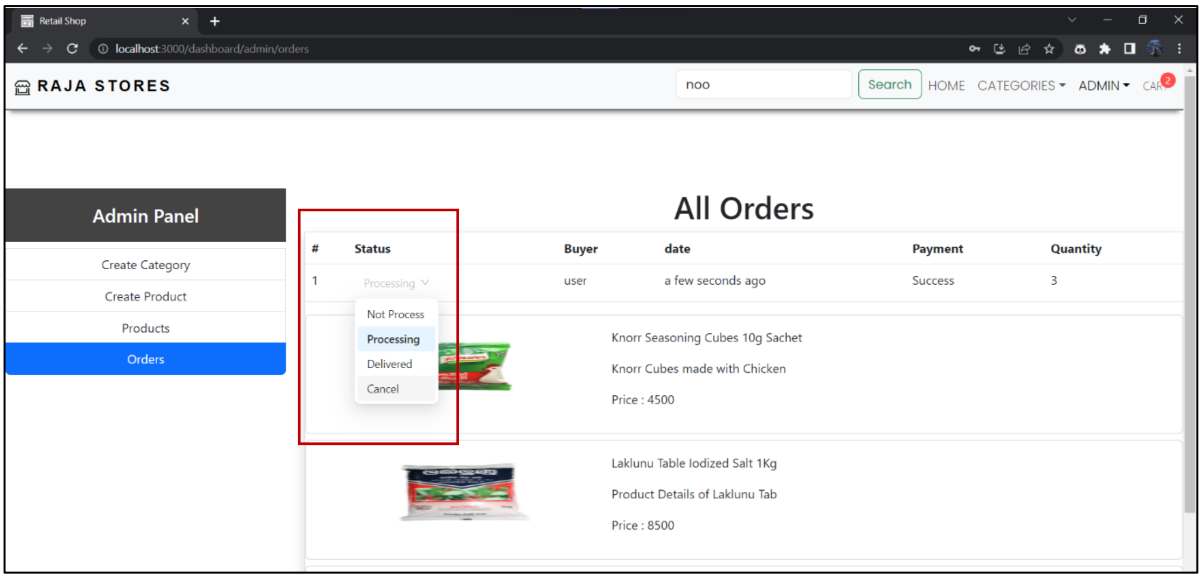
Price : 8500

Brown Sugar 1kg

Brown Sugar

Figure 29: Admin orders

There are 4 statuses to orders such as Not Process, Processing, Delivered, Cancel



All Orders

#	Status	Buyer	date	Payment	Quantity
1	Processing	user	a few seconds ago	Success	3

Knorr Seasoning Cubes 10g Sachet
Knorr Cubes made with Chicken
Price : 4500

Laklunu Table Iodized Salt 1Kg
Product Details of Laklunu Tab
Price : 8500

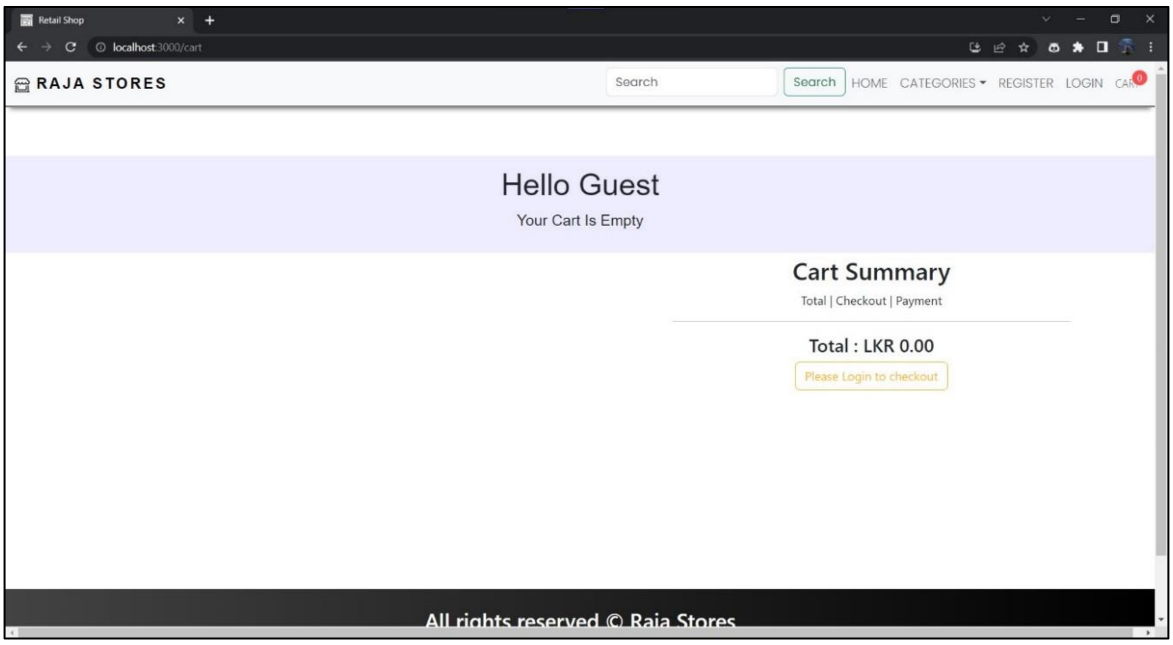
Figure 30: Order status

Once the admin/employee changes the status of the order, the customer can view the changed status.

Add to Cart

The cart is viewed differently by guest users and logged-in users differently.

1. Guest User



Hello Guest
Your Cart Is Empty

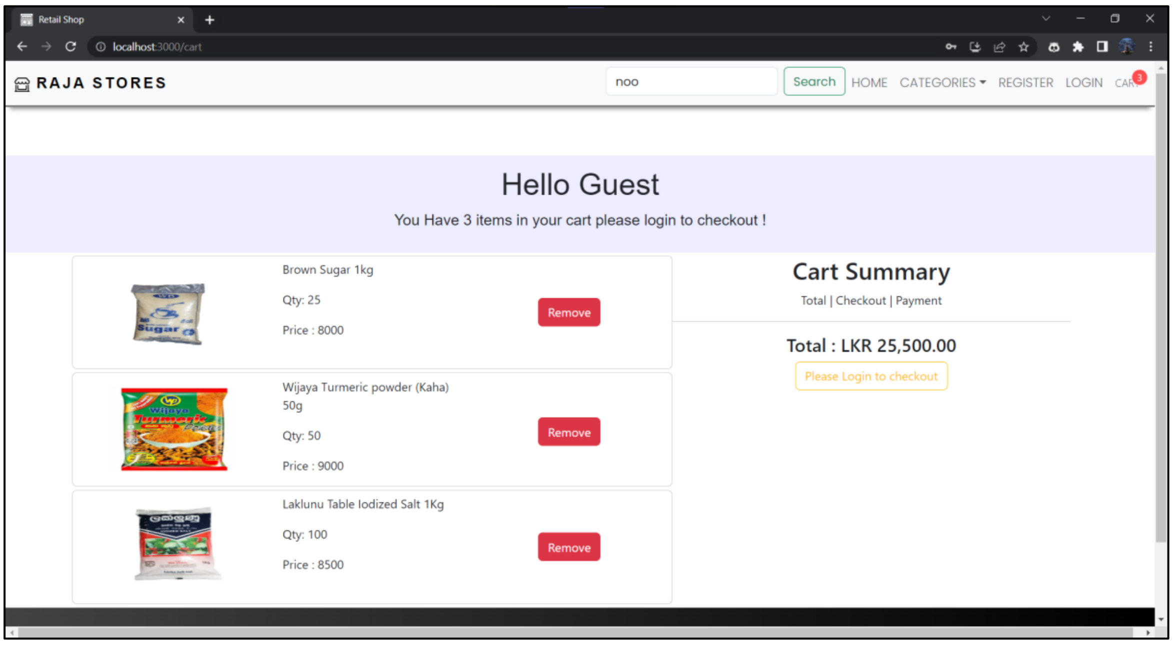
Cart Summary
Total | Checkout | Payment

Total : LKR 0.00

All rights reserved © Raja Stores

Figure 31: Guest empty cart

This is how the empty cart is shown to a guest user and they are welcomed as “guest”. Still, guests can add items to their cart and browse products.


Hello Guest

You Have 3 items in your cart please login to checkout !

Brown Sugar 1kg
Qty: 25
Price : 8000

Wijaya Turmeric powder (Kaha) 50g
Qty: 50
Price : 9000

Laklunu Table Iodized Salt 1Kg
Qty: 100
Price : 8500

Cart Summary

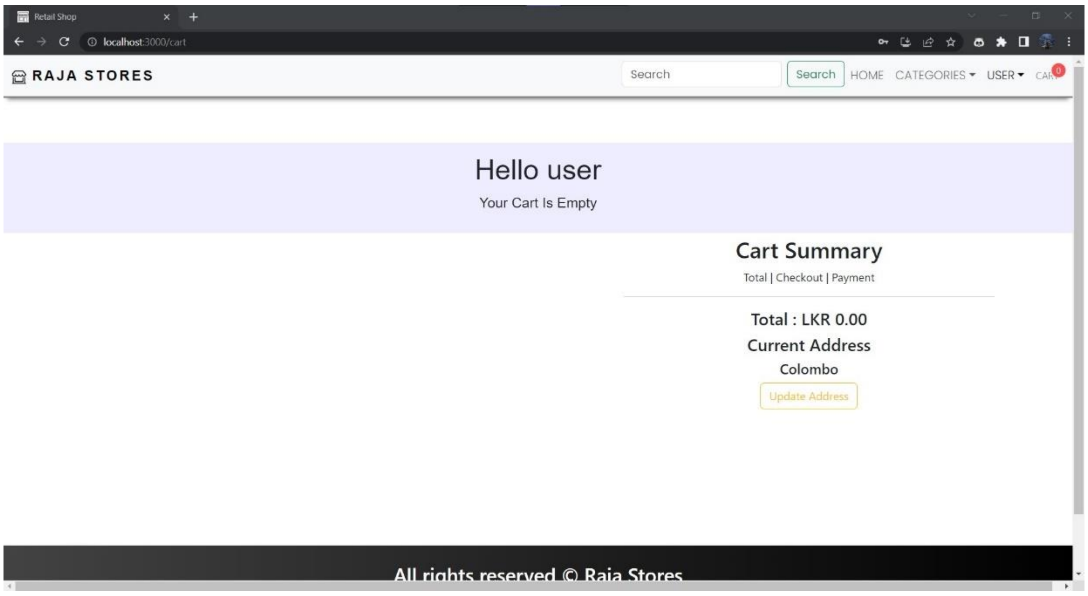
Total | Checkout | Payment

Total : LKR 25,500.00

Figure 32: Guest cart

Guests are asked to log in to the website first to checkout. They can click on the yellow button to be redirected to the login page. When once logged in they are directly navigated to the cart page.

2. Logged in User


Hello user

Your Cart Is Empty

Cart Summary

Total | Checkout | Payment

Total : LKR 0.00

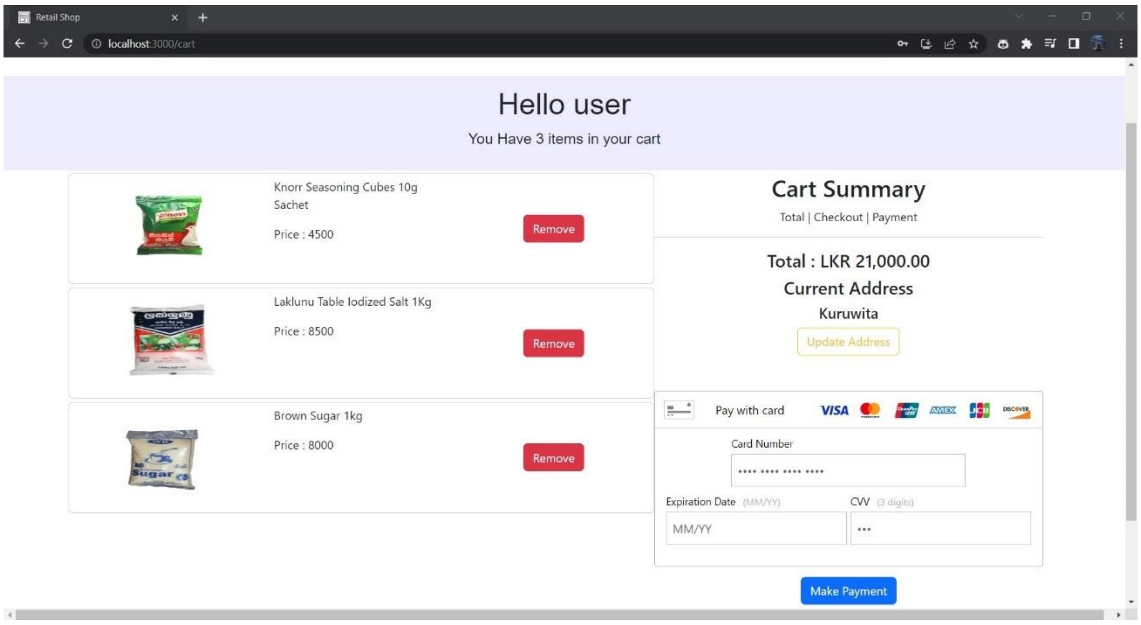
Current Address

Colombo

All rights reserved © Raja Stores

Figure 33: User empty cart

The users can add items to the cart they prefer upon using any of the above-mentioned ways in the document. They can check the cart icon in the header bar to see how many product items are added to the cart. The cart page is viewed below when different product items are added.



Hello user

You Have 3 items in your cart

Knorr Seasoning Cubes 10g Sachet

Price : 4500

Laklunu Table Iodized Salt 1Kg

Price : 8500

Brown Sugar 1kg

Price : 8000

Cart Summary

Total | Checkout | Payment

Total : LKR 21,000.00

Current Address

Kuruwita

Card Number

Expiration Date (MM/YY)

MM/YY

CVV (3 digits)

Figure 34: User cart

The user can change the delivery address using the update address feature. Using the remove button user can remove any product unwanted. The total of items can be displayed. Then under that payment portal is available.

Payment Portal

The payment portal is used to make payments using credit or debit cards. Upon successful payment, the user is directed he order page to see the status of orders.

Database Structure

In the MERN stack, MongoDB is used to layer database structure. MongoDB is a cross-platform document-oriented database program that is flexible and does not need to adjust for strict constrictions like in SQL.

The project database is stored in MongoDB atlas and is used to connect with the application using the URL string obtained unique to the database.

In the order process management web application, the database has 4 collections.

Collections – there can be several collections in one database. It is equivalent to tables in a relative database. Documents are collected as collections.

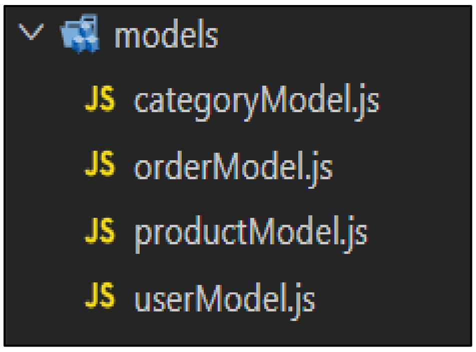
Documents – MongoDB stores data records as documents and these documents are flexible in storing data.

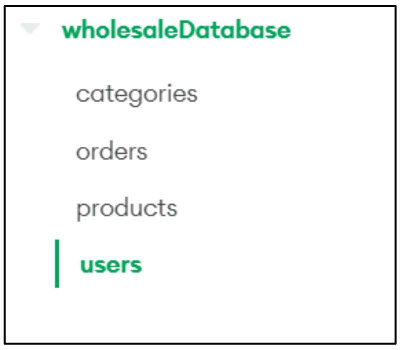
In the project, the database is named as “wholesaleDatabase” in MongoDB Atlas.

Then this database has 4 collections as

1. users
2. categories
3. products
4. orders

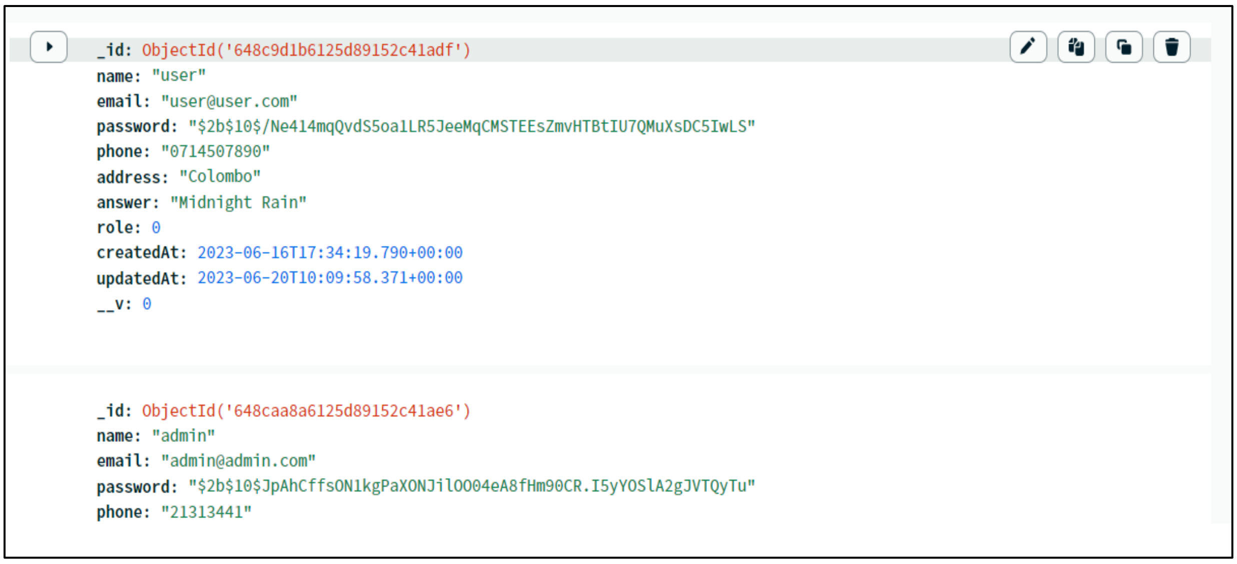
On the server side, models are created accordingly to provide structure on how to store and what fields are to be included in each collection with their relative data types and other needed information.


```
models
  categoryModel.js
  orderModel.js
  productModel.js
  userModel.js
```
Figure 35: models


```
wholesaleDatabase
  categories
  orders
  products
  users
```
Figure 36: MongoDB database collections

users collection and model

Below are some examples of documents of users collection stored in the MongoDB database. This data is used to validate users of the website to log in and provide access to different functionalities.

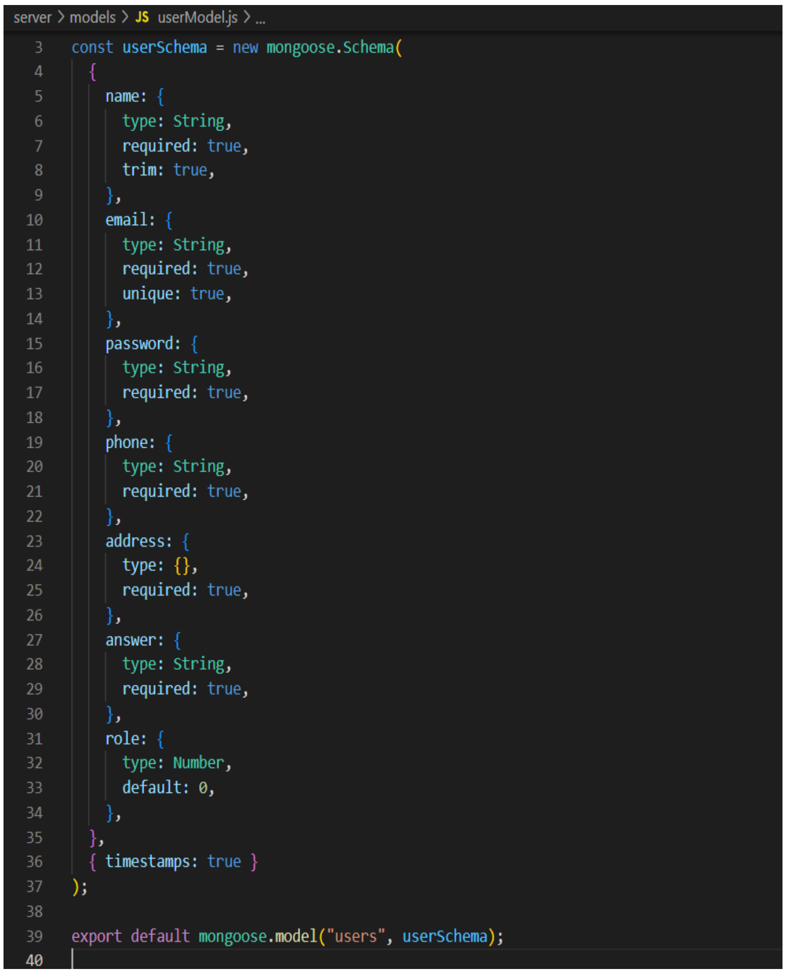


```
_id: ObjectId('648c9d1b6125d89152c41adf')
name: "user"
email: "user@user.com"
password: "$2b$10$/Ne414mqQvdS5oalLR5JeeMqCMSTEEsZmvHTBtIU7QMuXsDC5IwLS"
phone: "0714507890"
address: "Colombo"
answer: "Midnight Rain"
role: 0
createdAt: 2023-06-16T17:34:19.790+00:00
updatedAt: 2023-06-20T10:09:58.371+00:00
__v: 0


_id: ObjectId('648caa8a6125d89152c41ae6')
name: "admin"
email: "admin@admin.com"
password: "$2b$10$JpAhCffsON1kgPaXONJilOO04eA8fHm90CR.I5yYOSlA2gJVTQyTu"
phone: "21313441"
```

Figure 37: User collection document

User model

The user model comprises data fields as above. The role data field is used to assign admin access to selected users. Users with 1 as a role are given admin privileges and other normal users have 0 as a role.



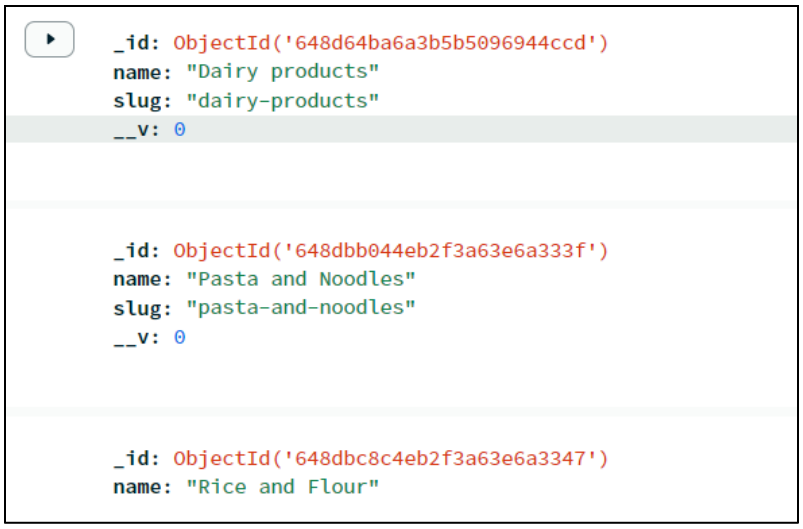
server > models > JS userModel.js > ...

```
const userSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
      trim: true,
    },
    email: {
      type: String,
      required: true,
      unique: true,
    },
    password: {
      type: String,
      required: true,
    },
    phone: {
      type: String,
      required: true,
    },
    address: {
      type: {},
      required: true,
    },
    answer: {
      type: String,
      required: true,
    },
    role: {
      type: Number,
      default: 0,
    },
  },
  { timestamps: true }
);

export default mongoose.model("users", userSchema);
```

Figure 38: User model

<u>categories collection and model</u>

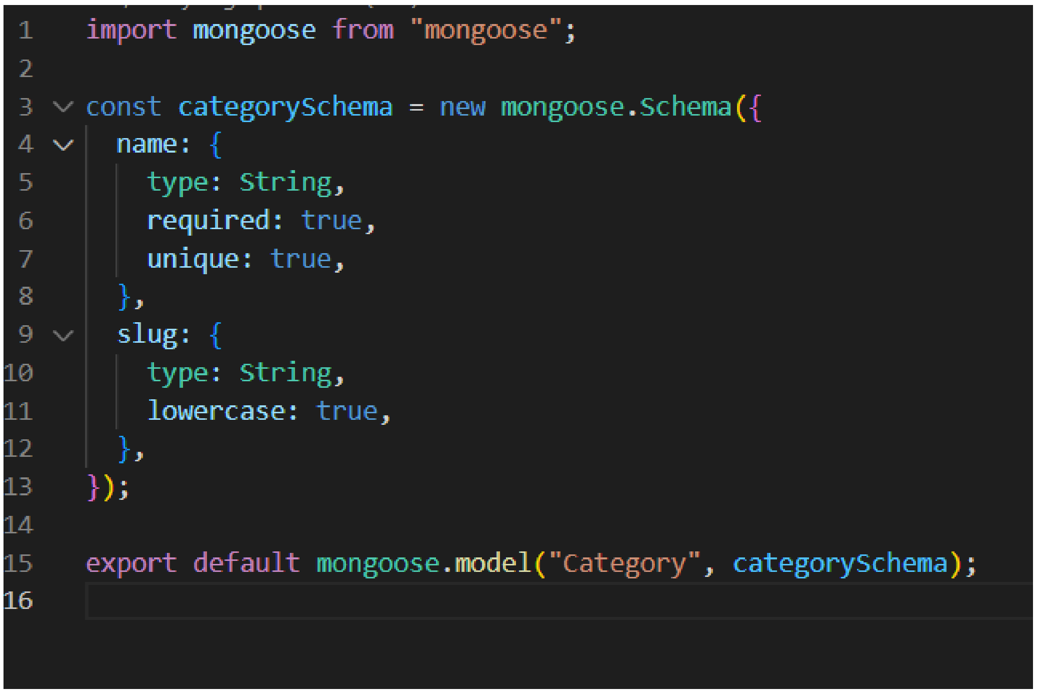
Categories collection is used to store details about categories of products.



```
_id: ObjectId('648d64ba6a3b5b5096944ccd')
name: "Dairy products"
slug: "dairy-products"
__v: 0

_id: ObjectId('648dbb044eb2f3a63e6a333f')
name: "Pasta and Noodles"
slug: "pasta-and-noodles"
__v: 0

_id: ObjectId('648dbc8c4eb2f3a63e6a3347')
name: "Rice and Flour"
```

Figure 39: category collection document

Categories model



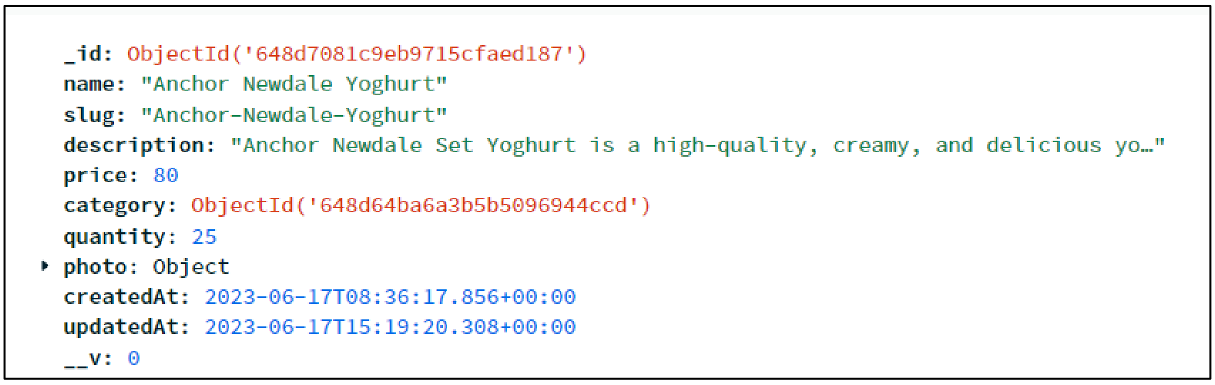
```
import mongoose from "mongoose";

const categorySchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    unique: true,
  },
  slug: {
    type: String,
    lowercase: true,
  },
});

export default mongoose.model("Category", categorySchema);
```

Figure 40: category model

The slug is used to combine the words with hyphens removing whitespaces since users can type anything and to retain consistency slug is used. Every word is converted to lowercase.

<u>product collection and model</u>

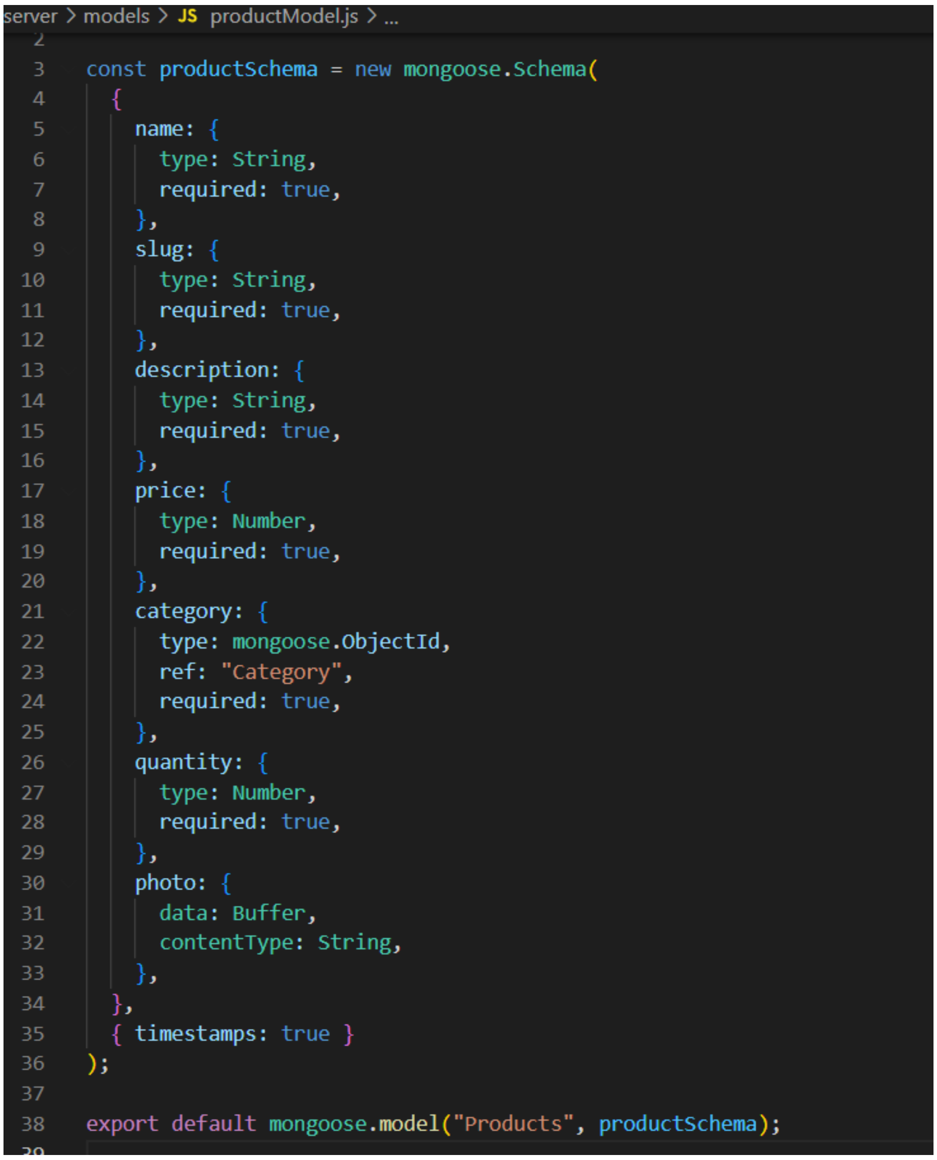
Product details are stored here. Category id is referenced to integrate the details about categories and products.



```
  _id: ObjectId('648d7081c9eb9715cfaed187')
  name: "Anchor Newdale Yoghurt"
  slug: "Anchor-Newdale-Yoghurt"
  description: "Anchor Newdale Set Yoghurt is a high-quality, creamy, and delicious yo…"
  price: 80
  category: ObjectId('648d64ba6a3b5b5096944ccd')
  quantity: 25
▸ photo: Object
  createdAt: 2023-06-17T08:36:17.856+00:00
  updatedAt: 2023-06-17T15:19:20.308+00:00
  __v: 0
```

Figure 41: product collection document

Product model



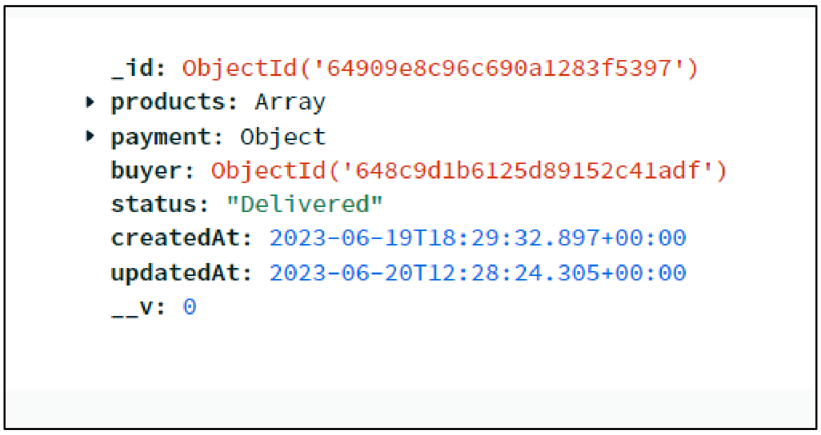
server > models > JS productModel.js > ...

```
const productSchema = new mongoose.Schema(
  {
    name: {
      type: String,
      required: true,
    },
    slug: {
      type: String,
      required: true,
    },
    description: {
      type: String,
      required: true,
    },
    price: {
      type: Number,
      required: true,
    },
    category: {
      type: mongoose.ObjectId,
      ref: "Category",
      required: true,
    },
    quantity: {
      type: Number,
      required: true,
    },
    photo: {
      data: Buffer,
      contentType: String,
    },
  },
  { timestamps: true }
);

export default mongoose.model("Products", productSchema);
```

Figure 42: product model

<u>order collection and model</u>

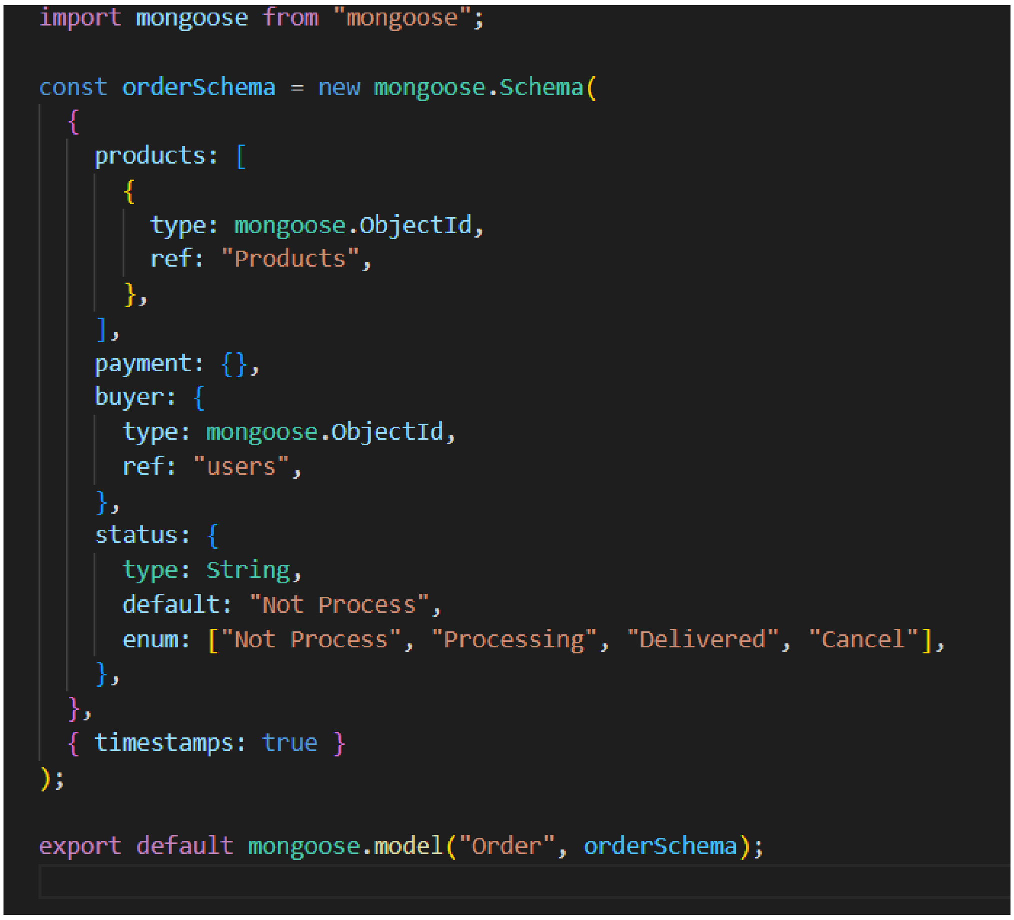
This is used to keep track of every order of users.



```
    _id: ObjectId('64909e8c96c690a1283f5397')
  ▸ products: Array
  ▸ payment: Object
    buyer: ObjectId('648c9d1b6125d89152c41adf')
    status: "Delivered"
    createdAt: 2023-06-19T18:29:32.897+00:00
    updatedAt: 2023-06-20T12:28:24.305+00:00
    __v: 0
```

Figure 43: order collection document

Order model



```
import mongoose from "mongoose";

const orderSchema = new mongoose.Schema(
  {
    products: [
      {
        type: mongoose.ObjectId,
        ref: "Products",
      },
    ],
    payment: {},
    buyer: {
      type: mongoose.ObjectId,
      ref: "users",
    },
    status: {
      type: String,
      default: "Not Process",
      enum: ["Not Process", "Processing", "Delivered", "Cancel"],
    },
  },
  { timestamps: true }
);

export default mongoose.model("Order", orderSchema);
```

Figure 44: order model

There are 4 default statuses in an array that can be used to change the status of orders of users. Products are passed as an array to include all the product items in the cart.

API implementation and respective Endpoints

For the implementation of APIs(Application Programming Interface) on the server side server.js file is used to sets up an Express server, configure middleware, establish a database connection, and define routes for different API endpoints.

The port used for server-side implementation is 8080 and the connection string for the MongoDB is stored in the .env file.

There are mainly 3 routes for different API endpoints. **authRoutes**, **categoryRoutes**, and **productRoutes** are imported from their respective files and mounted as middleware.

```
//routes
app.use("/api/v1/auth", authRoutes);
app.use("/api/v1/category", categoryRoutes);
app.use("/api/v1/product", productRoutes);
```

The APIs are handled in the respective route files.

Authorization APIs

authRoutes.js file defines the routes for authentication-related endpoints. It imports the necessary controllers and middleware functions to handle the requests.

Every auth API begins with api/v1/auth

1. Register

API endpoints: /register

Handles the registration of a new user. Accepts a POST request.

http://localhost:8080/api/v1/auth/register

In the body include the following for testing purposes:

```
{
    "name": "test",
    "email": "test@test.com",
    "password": "123456",
    "phone": "0111234567",
    "address": "Colombo",
    "answer": "spring day"
}
```

The response

```
{
    "success": true,
    "message": "User Registered Successfully",
    "user": {
        "name": "test",
        "email": "test@test.com",
        "password": "$2b$10$/pb8.b1Zo3VW2AOudihNoerb/mUURdIaXE5pNGdqeZpsjOhkCPX9i",
        "phone": "0111234567",
        "address": "Colombo",
        "answer": "spring day",
        "role": 0,
        "_id": "6491c1b357ed632cd5ff835c",
        "createdAt": "2023-06-20T15:11:47.413Z",
        "updatedAt": "2023-06-20T15:11:47.413Z",
        "__v": 0
    }
}
```

2. Login

API endpoints: /login

Handles the login process. Accepts a POST request.

http://localhost:8080/api/v1/auth/login

In the body include the following:

```
{
    "email": "user@user.com",
    "password": "user1234"
}
```

The response:

```
{
    "success": true,
    "message": "login successfully",
    "user": {
        "_id": "6491b7a957ed632cd5ff832e",
        "name": "user",
        "email": "user@user.com",
        "phone": "071509023",
        "address": "Ratnapura",
        "role": 0
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2NDkxYjdhOTU3ZWQ2MzJj
ZDVmZjgzMmUiLCJpYXQiOjE2ODcyNzQ0MzEsImV4cCI6MTY4Nzg3OTIzMX0.HWr8i3XpprWqh1N7_nCFXqv
q9Qe7iHgoflASWpmcSqE"
}
```

3. Reset Password

API endpoints: /forgot-password

Handles the password reset request. Accepts a POST request.

http://localhost:8080/api/v1/auth/forgot-password

In the body include the following:

```
{
    "answer": "The Shire",
    "email": "user@user.com",
    "newPassword": "user1234"
}
```

The Response

```
{
    "success": true,
    "message": "Password Reset Successfully"
}
```

4. Check user authorization

API endpoints: /user-auth

A protected route for authenticated users. Accepts a GET request.

http://localhost:8080/api/v1/auth/user-auth

In headers, it requires the token of the user that is generated and unique to each user to be the value to Authorization. If the token is of a user then a response message of okay is given.

5. Check admin authorization

API endpoints: /admin-auth

A protected route for authenticated admins. Accepts a GET request.

http://localhost:8080/api/v1/auth/admin-auth

In headers, it requires the token of admin as value to Authorization. If the token is of an admin then a response message of okay is given. This API checks whether the given user isAdmin or not.

6. Update Profile

API endpoints: /profile

Updates the user's profile. Accepts a PUT request.

http://localhost:8080/api/v1/auth/profile

Requires user sign-in for this method. In the body, the required fields are sent such as name, address, password, and contact number.

7. Get Orders

API endpoints: /orders

Retrieves orders for an authenticated user. Accepts a GET request.

http://localhost:8080/api/v1/auth/orders

The user should be signed in that includes sending a token as the value to Authorization key in headers.

8. Get All Orders

API endpoints: /all-orders

Retrieves all orders for an authenticated admin. Accepts a GET request.

http://localhost:8080/api/v1/auth/all-orders

9. Update Order Status

API endpoints: /order-status/:orderId

Updates the status of an order by an authenticated admin. Accepts a PUT request.

http://localhost:8080/api/v1/auth/order-status/648f6a864b3728e2c59bd00a

Along with the URL, a path parameter is sent as the id of a certain order that is needed to be updated. In the body, status should be included that needs to be changed.

Category APIs
categoryRoutes.js file defines the routes for category-related operations. Apart from getting categories, all the other requests require admin privileges to perform. Therefore, the token of admin should be sent along with the headers as Authorization.

Every category API begins with api/v1/category

1. Create a category

API endpoint: /create-category

Handles the creation of a new category. Accepts a POST request. Requires sign-in and admin privileges.

http://localhost:8080/api/v1/category/create-category

The body name field should be sent with the request.

```
{
    "name": "Dairy Products"
}
```

2. Update a category

API endpoint: /update-category/:id

Handles the update of a category by ID. Accepts a PUT request. Requires sign-in and admin privileges.

http://localhost:8080/api/v1/category//update-category/648cc76d7004a76b1f310cdf

name of the category to be updated should be sent in the body with the request.

```
{
    "name": "rice"
}
```

3. Get all categories

API endpoint: /get-category

Retrieves all categories. Accepts a GET request.

http://localhost:8080/api/v1/category/get-category

4. Get a single category

API endpoint: /single-category/:slug

Retrieves a single category by slug. Accepts a GET request.

http://localhost:8080/api/v1/category/single-category/rice

The slug is the name of the category without whitespaces and is connected with hyphens in lowercase. It should be sent along with the request as a path parameter.

5. Delete a category

API endpoint: /delete-category/:id

Deletes a category by ID. Accepts a DELETE request. Requires sign-in and admin privileges.

http://localhost:8080/api/v1/category/delete-category/648cc76d7004a76b1f310cdf

The id of the category to be removed should be sent along with the path.

Product APIs
productRoutes.js file defines the routes for product-related operations.

Every product API begins with api/v1/product

1. Create a product

API endpoint: /create-product

Handles the creation of a new product. Accepts a POST request. Requires sign-in and admin privileges. Utilizes the **formidable** middleware for parsing form data.

http://localhost:8080/api/v1/product/create-product/

The required data fields should be filled such as product name, description, quantity, and price, a photo can be uploaded and the category of the product should be sent as the id of the category.

```
"category": "6491d5a057ed632cd5ff836f",
```

2. Update a product

API endpoint: /update-product/:pid

Handles the update of a product by ID. Accepts a PUT request. Requires sign-in and admin privileges. Utilizes the **formidable** middleware for parsing form data.

http://localhost:8080/api/v1/product/update-product/648d7c98c86b8ab95ccf40eb

when updating a single product, the id of the product should be sent as a path parameter, and the body should include the fields that need to be updated.

3. Get products

API endpoint: /get-product

Retrieves all products. Accepts a GET request.

http://localhost:8080/api/v1/product/get-product

4. Get a single product

API endpoint: /get-product/:slug

Retrieves a single product by slug. Accepts a GET request.

http://localhost:8080/api/v1/product/get-product/Highland-Curd-Plastic-Container-500ml

The slug value stored in the database should be sent as the path parameter when getting details about a single product.

5. Get product photo

API endpoint: /product-photo/:pid

Retrieves the photo of a product by ID. Accepts a GET request.

http://localhost:8080/api/v1/product/product-photo/648d7c98c86b8ab95ccf40eb

The product id should be passed along with the path to get the photo of a product.

6. Delete a product

API endpoint: /delete-product/:pid

Deletes a product by ID. Accepts a DELETE request.

http://localhost:8080/api/v1/product/delete-product/648d739cb157b6cd9933fd20

The product id should be passed along with the path to delete the specific product.

7. Product Filters

API endpoint: /product-filters

Filters products based on certain criteria. Accepts a POST request.

http://localhost:8080/api/v1/product/product-filters

The body should include the following.

```
{
    "checked": ["648dbc8c4eb2f3a63e6a3347", "648dbb044eb2f3a63e6a333f"],
    "radio": [1000,1999]
}
```

Checked means filtering by category. Products can be more than one, therefore it is sent as an array and radio means the price range of the products.

8. Get a count of all products

API endpoint: /product-count

Retrieves the count of products. Accepts a GET request.

http://localhost:8080/api/v1/product/product-count

9. Limits products per page

API endpoint: /product-list/:page

Retrieves a list of products per page. Accepts a GET request.

http://localhost:8080/api/v1/product/product-list/2

The page number should be sent as a path parameter. The products on page 2 will be retrieved then.

10. Search for products

API endpoint: /search/:keyword

Searches for products based on a keyword. Accepts a GET request.

http://localhost:8080/api/v1/product/search/Rice

11. Get similar products

API endpoint: /related-product/:pid/:cid

Retrieves related products based on the product ID and category ID. Accepts a GET request.

http://localhost:8080/api/v1/product/related-product/648dda614ae43651fe31f7b5/648dbc8c4eb2f3a63e6a3347

uses both product id and category id to get similar products. The product id and category id of a product is sent and then products with the same category are displayed except the product of the product id sent along the path.

12. Get products of a certain category

API endpoint: /product-category/:slug

Retrieves products based on a category slug. Accepts a GET request.

http://localhost:8080/api/v1/product/product-category/dairy-products

When the category name is sent as the slug, all the products of the category can be retrieved.

13. Get Braintree token

API endpoint: /braintree/token

Retrieves a Braintree token for payment processing. Accepts a GET request.

http://localhost:8080/api/v1/product//braintree/token

Gets the token generated by Braintree for a specific client to be used for card payments.

14. Braintree transaction processing

API endpoint: /braintree/payment

Handles the Braintree payment processing. Accepts a POST request. Requires sign-in.

Summary

The order processing web application for a wholesale shop is designed to provide a seamless shopping experience for users. With a focus on ease of use and functionality, the application allows users to browse and order products from the wholesale shop. The target audience of this project is the small-scale retail shops around the area and all the other potential business owners who are not yet become customers of the business. The objectives of the project were to develop a robust and user-friendly application using the MERN stack (MongoDB, Express.js, React, Node.js). Key features of the application include user registration, product browsing, shopping cart functionality, and order placement. The application has been deployed on a hosting provider, allowing it to be accessed from the internet using an externally exposed IP or domain. This ensures that users can conveniently access the Wholesale Shop web app anytime, anywhere. This documentation provides a detailed explanation of the system overview, database structure, and API implementation as well.