

# **FULL STACK ENGINEERING PROJECT REPORT**

**On**

**Confectionery E-Commerce: PLUM**

Submitted in partial fulfilment of the requirement for the Course of  
Full Stack Engineering (22CS037) of

**Computer Science and Engineering**

**B. E. Batch – 2022**

**In**

**May-2025**



**Under the Guidance of**

Dr. Shaffu  
Assistant Professor  
Department of Computer Science and  
Engineering

**Submitted By**

Priyanshu  
Roll No. 2210990685  
Raghav Thakur  
Roll No. 2210990701  
Rahul  
Roll No. 2210990702

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**CHITKARA UNIVERSITY**

**PUNJAB**



## **CERIFICATE**

This is to be certified that the project entitled “Confectionery E-Commerce: PLUM” has been submitted for the Bachelor of Computer Science Engineering at Chitkara University, Punjab during the academic semester January 2025 - May2025 is a Bonafide piece of project work carried out by Priyanshu (2210990685), Raghav Thakur (2210990701) and Rahul (22109990702) towards the partial fulfillment for the award of the course Full Stack Engineering under the guidance of Dr. Shaffu and supervision.

Project Guide:

Dr. Shaffu

(Assistant Professor | DCSE)



## **CANDIDATE'S DECLARATION**

We, Priyanshu (2210990685), Raghav Thakur (2210990701) and Rahul (22109990702), B. E. CSE – 2022 of the Chitkara University, Punjab hereby declare that the Full Stack Engineering Project Report entitled “Confectionery E-Commerce: PLUM” is an original work and data provided in the study is authentic to the best of our knowledge. This report has not been submitted to any other Institute for the award of any other course.

Priyanshu  
PRIYANSHU  
2210990685

Raghav  
RAGHAV THAKUR  
2210990701

Rahul  
RAHUL  
2210990702

**Place:** Chitkara University, Punjab

**Date:** 22-05-2025



## **ACKNOWLEDGEMENT**

It is our pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and influenced our thinking, behavior and acts during the course of study.

We express our sincere gratitude to all for providing us an opportunity to undergo Full Stack Engineering Project as the part of the curriculum.

We are thankful to Dr. Shaffu for his support, co-operation, and motivation provided to us during the training for constant inspiration, presence and blessings.

We also extend our sincere appreciation Dr. Shaffu who provided his valuable suggestions and precious time in accomplishing our full stack engineering project.

Lastly, we would like to thank our fellow mates for their moral support and friends with whom we shared our day-to-day experience and received lots of suggestions that improve our quality of work.

Priyanshu  
2210990685

Raghav Thakur  
2210990701

Rahul  
2210990702



## **INDEX**

- **Abstract/Keywords**
- **Introduction to the project**
  - **Background**
  - **Problem Statement**
- **Software and Hardware Requirement Specification**
  - **Methods**
  - **Programming/Working Environment**
  - **Requirements to run the application**
- **Database Analyzing, design and implementation**
- **Program's Structure Analyzing and GUI Constructing**
- **Code-Implementation**
- **Conclusion**
- **Future Scope**
- **Bibliography**

## **ABSTRACT**

“Confectionery E-Commerce: PLUM” is an online bookstore website using the MERN stack of technologies, which includes MongoDB, Express.js, React.js, and Node.js. The aim was to make a scalable, user-friendly web platform that would enable people to browse, purchase, and manage confectionery online.

The platform features a clean, responsive user interface that allows customers to explore a wide range of sweet products such as chocolates, mithai, caramel, fudge, nougat, and lollipops. Each product includes detailed information, images, and pricing, with an intuitive cart and checkout flow that ensures a smooth purchasing experience for users across all devices.

On the backend, Node.js and Express.js handle routing, authentication, and data handling efficiently, while MongoDB provides a flexible, schema-less database structure to manage product inventories, orders, and user accounts. This architecture supports easy scalability and rapid feature updates as the platform grows with user demand.

Security and performance optimization were prioritized through token-based authentication, route protection, and lazy loading on the frontend. Making PLUM not just a store but a full-fledged, real-world e-commerce solution for the confectionery market.

## **INTRODUCTION**

### **BACKGROUND**

The growth of e-commerce has transformed industries globally, and the confectionery market is no exception. Online sweet stores like PLUM have gained popularity by offering convenience, a vast range of treats, and the flexibility to shop anytime, anywhere. Customers now expect more than just sweets—they want a memorable digital shopping experience.

Unlike physical stores limited by shelf space and local inventory, PLUM enables users to browse an extensive catalogue of chocolates, mithai, caramel, nougat, fudge, and more—all from the comfort of their homes. With just a few clicks, customers can explore unique and seasonal treats that may not be available in local stores.

A well-built online confectionery store like PLUM enhances the customer experience by making it easy to find desired sweets, discover new favourites through personalized recommendations, and manage preferences. Modern web technologies are crucial in ensuring smooth performance, responsive design, and interactive features that keep users engaged and satisfied.

### **PROBLEM STATEMENT**

Despite the growth of the online confectionery market, many existing platforms still face significant challenges that hinder user satisfaction.

For example, some sweet-selling websites are not responsive across devices, making it difficult for users to browse or shop using mobile phones or tablets. Others offer poor search and filtering options, limiting users' ability to quickly find specific sweets or categories. Many platforms also lack personalized product recommendations, leading to lower engagement and missed opportunities for upselling.

Some platforms feature tedious cart and checkout systems that frustrate buyers, resulting in abandoned purchases. Additionally, many lack robust administrative tools for managing stock, tracking orders, and analysing user behaviour. Without efficient inventory updates, order monitoring, or analytics, businesses struggle to maintain performance and customer satisfaction.

This project aims to solve these issues through PLUM, a modern online confectionery store built using the MERN stack: MongoDB, Express.js, React.js, and Node.js. The platform will be fully responsive, featuring smooth navigation, intelligent product recommendations, secure user authentication, an efficient cart and checkout experience, and a comprehensive admin dashboard for inventory control and user activity monitoring.



## **SOFTWARE & HARDWARE REQUIREMENTS**

### **METHODS**

To develop the PLUM confectionery e-commerce website using the MERN stack, a well-structured and strategic approach was essential to ensure both the front-end and back-end were optimized for performance, usability, and scalability. The project followed the Agile methodology, allowing for iterative development, regular testing, and continuous feedback integration to enhance the user experience.

The core development tasks were as follows:

**Requirement Analysis:** Understanding user needs and defining platform features such as a responsive user interface, a categorized product catalogue (e.g., chocolates, mithai, nougat, caramel), a secure login system, personalized product recommendations, and efficient cart and checkout functionalities.

**System Design:** Creating wireframes and flowcharts to design the layout and architecture of the website, as well as defining a scalable and flexible database schema to support product details, user data, and order management.

**Implementation:** Building the platform using MongoDB for database management, Express.js for the backend API, React.js for the frontend interface, and Node.js as the server-side runtime environment, ensuring seamless integration between all components.

**Testing and Validation:** Conducting unit, integration, and user acceptance testing to verify that all features worked as intended and the user experience was smooth, secure, and error-free across various devices.

**Deployment:** Deploying the final web application on a cloud-based hosting service, ensuring high availability, data security, and scalability to handle increasing user traffic and inventory expansion.

### **PROGRAMMING ENVIRONMENT**

#### **Backend:**

- **Node.js:** A powerful JavaScript runtime used for server-side logic. Its asynchronous, event-driven architecture allows PLUM to efficiently handle multiple client requests.
- **Express.js:** A minimal and flexible Node.js web application framework that simplifies the creation of APIs and server logic. It is used for routing, handling HTTP requests, and integrating with MongoDB.
- **MongoDB:** A NoSQL document-oriented database used to store user details, product information (e.g., mithai, chocolates, caramel, fudge), order records, and transaction history in a scalable JSON-like format.

#### **Frontend:**

- **React.js:** A front-end JavaScript library used to build dynamic and component-based user interfaces. It powers the interactive shopping experience and ensures responsiveness across devices.
- **HTML/CSS:** These core web technologies are used to structure and style the frontend for a visually appealing and user-friendly interface.
- **JavaScript:** Enables dynamic behaviour, form validation, event handling, and other interactive features on the client side.



### Development Tools:

- **Visual Studio Code:** The primary Integrated Development Environment (IDE) used for writing, debugging, and managing code efficiently.
- **Git:** Version control system for tracking changes and collaborating during development.
- **MongoDB Compass:** A GUI tool used to inspect, debug, and query MongoDB databases during backend development and testing.

### Requirements to Run the PLUM Application

#### Hardware Requirements:

- **Processor:** Intel Core i3 or higher
- **RAM:** Minimum 4GB (8GB recommended)
- **Storage:** At least 10GB free space to store source files, dependencies, and tools
- **Network:** Reliable internet connection for package installations, API testing, and deployment

#### Software Requirements:

- **Operating System:** Windows 10 or later, macOS, or any Linux distribution
- **Web Browser:** Google Chrome, Mozilla Firefox, or any modern browser for frontend testing
- **Node.js:** Version 12 or later
- **MongoDB:** MongoDB Atlas for cloud deployment, or local MongoDB setup for development
- **Text Editor/IDE:** Visual Studio Code or any editor supporting JavaScript, Node.js, and React development



## **DATABASE ANALYZING, DESIGN AND IMPLEMENTATION**

A database is a critical component of the PLUM confectionery e-commerce platform, responsible for securely storing and managing diverse data types such as customer profiles, product details, orders, transactions, and inventory. The primary goal of the database design is to ensure structured storage, efficient data retrieval, and smooth management of all information.

MongoDB has been chosen for this project due to its flexibility, scalability, and suitability for dynamic, heterogeneous e-commerce data. Its document-oriented, JSON-like format allows effective handling of nested and complex data structures such as user orders, product categories, and shopping carts.

The main entities in the database are:

- **Users:** Contains customer and administrator information, including name, email, and securely hashed passwords.
- **Products:** Stores detailed data about confectionery items such as chocolates, mithai, fudge, caramel, and more.
- **Orders:** Records customer order information including order ID, user ID, product list, total price, and order status.
- **Cart:** Temporarily holds user-specific selected items, quantities, and subtotal prices before checkout.

### **Database Design**

The schema is designed for optimal performance and maintainability. Key collections include:

#### **Users Collection:**

- Fields include name, email, and securely hashed password.
- Role-based access differentiates between regular customers and administrators, allowing secure access to admin-only features such as product management and order monitoring.

#### **Products Collection:**

- Contains fields like `_id`, name, description, category (e.g., Indian Sweet, Chocolate, Caramel), subCategory (e.g., Ladoo, Fudge), an array of product images, bestseller (boolean), price, and `createdAt`.
- This collection holds all confectionery products available for display, browsing, and purchase on the platform.

#### **Orders Collection:**

- Fields include buyer information (name, email, address, phone), a list of ordered product IDs (`productIds`), `totalPrice`, status (e.g., pending, shipped), and `orderDate`.
- This collection stores all confirmed customer orders along with relevant details for processing and tracking.

#### **Cart Collection:**

- Stores temporary shopping cart data with fields like `userId`, `productId`, quantity, individual price, `totalPrice`, and `createdAt`.
- Represents items currently selected by a user before completing the purchase.

## Database Implementation

MongoDB is deployed either locally during development or through MongoDB Atlas for cloud hosting in production. Collections are created and maintained using MongoDB CLI or Compass and accessed via RESTful APIs built with Express.js.

Implementation steps include:

- **MongoDB Setup:** Installed and connected via Mongoose ODM for schema modelling and data validation.
- **Collection Definitions:** Structured collections for users, products, orders, and carts with appropriate field types.
- **API Routes:** RESTful endpoints handle CRUD operations, such as:
  - Cart Routes:**
    - POST /cart/get — Retrieve the authenticated user's current cart (authUser middleware required)
    - POST /cart/add — Add an item to the cart (authUser middleware required)
    - POST /cart/update — Update quantities or remove items from the cart (authUser middleware required)
  - Order Routes:**
    - POST /order/place — Place an order (authUser middleware required)
    - POST /order/userorders — Retrieve orders placed by the authenticated user (authUser middleware required)
    - POST /order/list — Admin-only: List all orders (adminAuth middleware required)
    - POST /order/status — Admin-only: Update order status (adminAuth middleware required)
  - Product Routes:**
    - POST /product/add — Admin-only: Add a new product with image uploads (using upload.fields middleware) (adminAuth middleware required)
    - POST /product/remove — Admin-only: Remove a product (adminAuth middleware required)
    - POST /product/single — Retrieve details of a single product
    - GET /product/list — Retrieve a list of all products
  - User Routes:**
    - POST /user/register — Register a new user
    - POST /user/login — User login
    - POST /user/admin — Admin login

All inputs are validated before insertion to ensure data integrity, including unique email checks, valid pricing, and required fields verification.

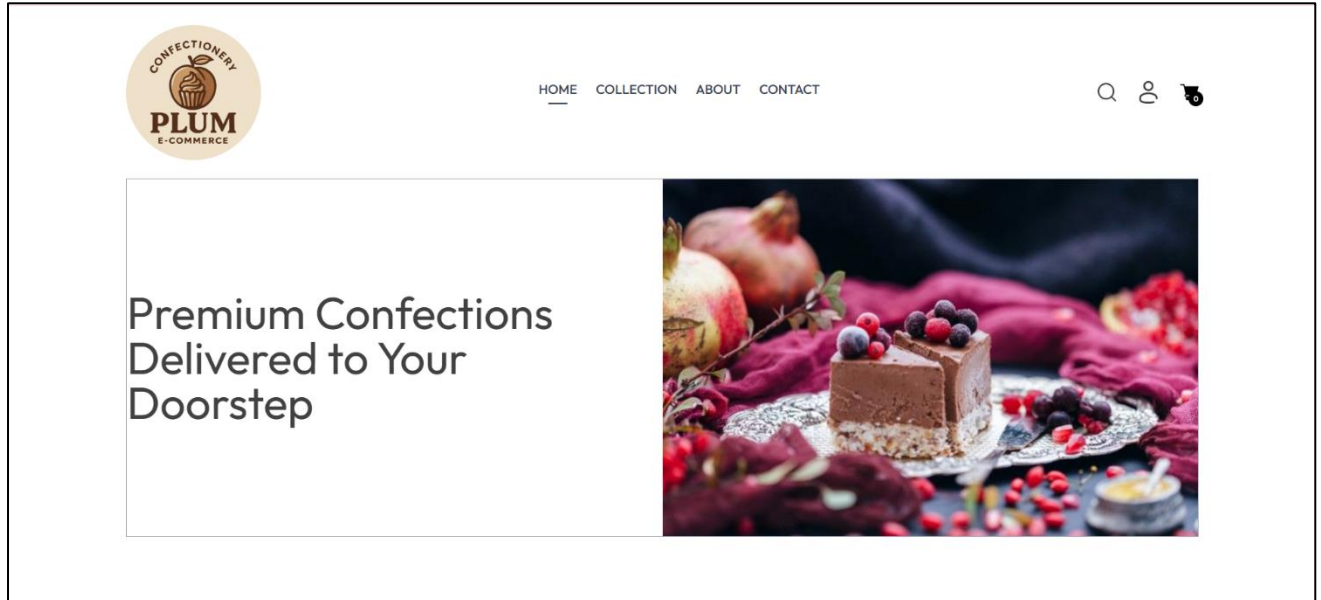
## Database Security and Integrity

To maintain data security and consistency:

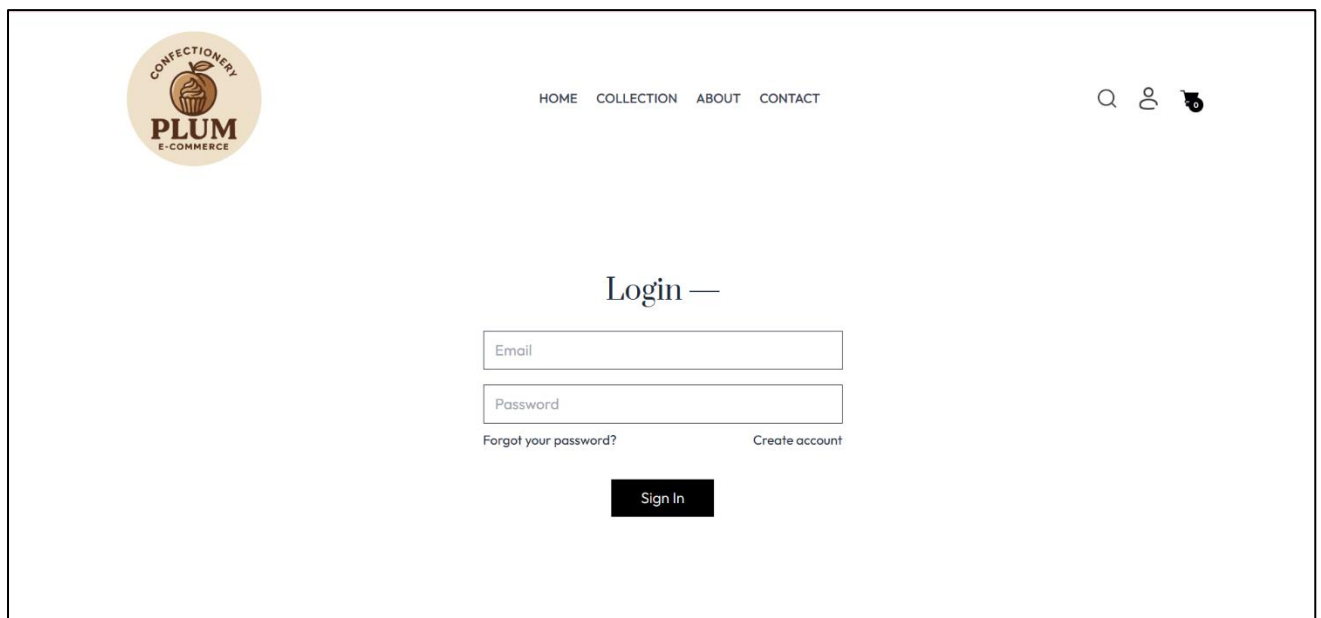
- **Encryption:** User passwords are hashed securely using bcrypt before storage.
- **Authentication:** JSON Web Tokens (JWT) are used to protect routes and manage user sessions.
- **Indexing:** Frequently queried fields such as name, category, and id are indexed to optimize query performance.

This comprehensive database design and implementation strategy provides a solid backend foundation for PLUM, ensuring seamless performance, secure operations, and scalability to support future growth and feature expansion

## Program's Structure Analyzing and GUI Constructing




**HOME PAGE**






**LOGIN PAGE**





[HOME](#)
[COLLECTION](#)
[ABOUT](#)
[CONTACT](#)






## Sign Up —




Please fill out this field.

[Forgot your password?](#)
[Login Here](#)

## REGISTER PAGE



[HOME](#)
[COLLECTION](#)
[ABOUT](#)
[CONTACT](#)

### FILTERS


#### CATEGORIES

- ☐ Chocolate
- ☐ Candy
- ☐ Caramel
- ☐ Fudge
- ☐ Chewing Gum
- ☐ Marshmallow
- ☐ Nougat
- ☐ Lollipop
- ☐ Indian Sweet


#### SUBCATEGORIES

### ALL COLLECTIONS —


Sort by: Relevant




Peanut Butter Fudge  
Rs.349



Mixed Marshmallow  
Rs.450



Mysore Pak  
Rs.899



Milk Cake  
Rs.650

## ALL COLLECTION



## LATEST COLLECTIONS —



Peanut Butter Fudge  
Rs.349



Mixed Marshmallow  
Rs.450



Mysore Pak  
Rs.899



Milk Cake  
Rs.650



Kaju Katli  
Rs.999

## LATEST COLLECTION

## BEST SELLERS —



Kaju Katli  
Rs.999



Gourmet Chocolate  
Rs.1499



Fruit Nougat  
Rs.749



Chocolate Nougat  
Rs.349



Besan Ladoo  
Rs.648

## BEST SELLER

## Subscribe Now

Enter your email

SUBSCRIBE



### COMPANY

Home  
About us  
Delivery  
Privacy policy

### GET IN TOUCH

+91 99999 8888  
plum@gmail.com

Copyright 2025@ plum.com - All Right Reserved.

## FOOTER




PLUM  
E-COMMERCE

HOME COLLECTION ABOUT CONTACT

🔍 🛒 🌙

YOUR CART —



Mysore Pak  
Rs.899

🗑

CART TOTALS —

SubtotalRs. 899.00

Shipping FeeRs. 100.00

TotalRs. 999.00

PROCEED TO CHECKOUT

## CART

PLUM  
E-COMMERCE

HOME COLLECTION ABOUT CONTACT

🔍 🛒 🌙

DELIVERY INFORMATION —

CART TOTALS —

SubtotalRs. 899.00

Shipping FeeRs. 100.00

TotalRs. 999.00


PAYMENT METHOD —

☒ CASH ON DELIVERY




PLACE ORDER

## CHECKOUT PAGE






HOME   COLLECTION   ABOUT   CONTACT


MY ORDERS



Mysore Pak  
Rs.899   Quantity: 1  
Date: Wed May 21 2025  
Payment: COD

● Order Placed

Track Order




Chocolate Fudge  
Rs.998   Quantity: 1  
Date: Wed May 21 2025  
Payment: COD

● Packing


Track Order

## USER DASHBOARD


## ADMIN PANEL




Logout



Add Items





List Items





Orders

Upload Image









Product name

Product description  

Write content here

Product category  
Chocolate

Sub category  
Chocolate

Product Price  
555


☐ Add to bestseller

## ADD ITEMS





CONFECTIONERY




PLUM

E-COMMERCE


ADMIN PANEL

+

Add Items










List Items



Orders


Logout

All Products List

Image	Name	Category	Price	Action
	Peanut Butter Fudge	Fudge	Rs. 349	X
	Mixed Marshmallow	Marshmallow	Rs. 450	X
	Mysore Pak	Indian Sweet	Rs. 899	X
	Milk Cake	Indian Sweet	Rs. 650	X
	Kaju Katli	Indian Sweet	Rs. 999	X
	Gourmet Chocolate	Chocolate	Rs. 1499	X
				

## LISTED ITEMS

CONFECTIONERY



PLUM

E-COMMERCE

ADMIN PANEL

Logout


+

Add Items

List Items

Orders

Order Page



Dry Fruit Ladoo x 1

Rahul Sharma


Alexandra Street,  
Santa Monica, California, USA, 98534  
7412589630

Items : 1

Rs. 899

Method : COD  
Payment : Pending  
Date : 5/21/2025

Order Placed



Chocolate Fudge x 1

Priyanshu fdf

Ambala,  
Ambala Cantt, Haryana, India, 133001  
7412589635

Items : 1

Rs. 1098

Method : COD  
Payment : Pending  
Date : 5/21/2025

Packing

## ORDER DETAIL

## CODE IMPLEMENTATION

### Admin Dashboard Routing and Authentication Structure in React

```
import React, { useEffect, useState } from 'react'
import Navbar from './components/Navbar'
import Sidebar from './components/Sidebar'
import { Routes, Route } from 'react-router-dom'
import Add from './pages/Add'
import List from './pages/List'
import Orders from './pages/Orders'
import Login from './components/Login'
import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

export const backendUrl = import.meta.env.VITE_BACKEND_URL
export const currency = 'Rs. '

const App = () => {

  const [token, setToken] = useState(localStorage.getItem('token')?localStorage.getItem('token'):"");

  useEffect(()=>{
    localStorage.setItem('token',token)
  },[token])

  return (
    <div className='bg-gray-50 min-h-screen'>
      <ToastContainer />
      {token === ""
        ? <Login setToken={setToken} />
        : <>
          <Navbar setToken={setToken} />
          <hr />
          <div className='flex w-full'>
            <Sidebar />
            <div className='w-[70%] mx-auto ml-[max(5vw,25px)] my-8 text-gray-600 text-base'>
              <Routes>
                <Route path='/add' element={ <Add token={token} /> } />
                <Route path='/list' element={ <List token={token} /> } />
                <Route path='/orders' element={ <Orders token={token} /> } />
              </Routes>
            </div>
          </div>
        </>
      )
    </div>
  )
}
```

## Express Server Configuration

```
import express from 'express'
import cors from 'cors'
import 'dotenv/config'
import connectDB from './config/mongodb.js'
import connectCloudinary from './config/cloudinary.js'
import userRouter from './routes/userRoute.js'
import productRouter from './routes/productRoute.js'
import cartRouter from './routes/cartRoute.js'
import orderRouter from './routes/orderRoute.js'

// App Config
const app = express()
const port = process.env.PORT || 4000
connectDB()
connectCloudinary()

// middlewares
app.use(express.json())
app.use(cors())

// api endpoints
app.use('/api/user', userRouter)
app.use('/api/product', productRouter)
app.use('/api/cart', cartRouter)
app.use('/api/order', orderRouter)

app.get('/', (req, res) => {
  res.send("API Working")
})

app.listen(port, () => console.log('Server started on PORT : ' + port))
```

## React Frontend Routing and Layout Setup

```
import React from 'react'
import { Routes, Route } from 'react-router-dom'
import Home from './pages/Home'
import Collection from './pages/Collection'
import About from './pages/About'
import Contact from './pages/Contact'
import Product from './pages/Product'
import Cart from './pages/Cart'
import Login from './pages/Login'
import PlaceOrder from './pages/PlaceOrder'
import Orders from './pages/Orders'
import Navbar from './components/Navbar'
import Footer from './components/Footer'
import SearchBar from './components/SearchBar'
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import Verify from './pages/Verify'

const App = () => {
  return (
    <div className='px-4 sm:px-[5vw] md:px-[7vw] lg:px-[9vw]'>
      <ToastContainer />
      <Navbar />
      <SearchBar />
      <Routes>
        <Route path='/' element={ <Home /> } />
        <Route path='/collection' element={ <Collection /> } />
        <Route path='/about' element={ <About /> } />
        <Route path='/contact' element={ <Contact /> } />
        <Route path='/product/:productId' element={ <Product /> } />
        <Route path='/cart' element={ <Cart /> } />
        <Route path='/login' element={ <Login /> } />
        <Route path='/place-order' element={ <PlaceOrder /> } />
        <Route path='/orders' element={ <Orders /> } />
        <Route path='/verify' element={ <Verify /> } />
      </Routes>
      <Footer />
    </div>
  )
}

export default App
```

## **CONCLUSION**

The PLUM confectionery e-commerce platform, built using the MERN stack—MongoDB, Express.js, React.js, and Node.js—offers an engaging and user-friendly interface for exploring and purchasing a wide variety of sweets. The platform is designed to enhance the online shopping experience through features like personalized recommendations, curated sweet collections, and a smooth browsing flow.

Key functionalities such as RESTful API integration for managing products and orders, JWT-based authentication for secure login sessions, and React state management contribute to seamless user interaction and efficient performance.

The inclusion of Swiper carousels, responsive layouts, and interactive button designs helps create an appealing and intuitive experience for users across all devices. These visual enhancements play a significant role in increasing user engagement and satisfaction.

An integrated admin dashboard supports administrators in managing the product catalogue, processing customer orders, and overseeing user activities. MongoDB allows for flexible and scalable storage of data such as user profiles, product listings, and transaction details. Node.js and Express.js provide a reliable backend environment to handle API requests and maintain secure user sessions.

Overall, PLUM is a complete and adaptable e-commerce solution for both customers and administrators. It effectively combines the capabilities of the MERN stack with modern development practices to deliver a secure, efficient, and enjoyable online shopping experience centered around a wide range of confectionery products.



## **FUTURE SCOPE**

While the current version of the PLUM confectionery website offers an engaging platform for browsing and adding sweets to the cart, there are several opportunities for future improvements to enhance overall functionality and user experience. One of the most crucial upgrades planned for the platform is the integration of a full-featured transaction and payment processing system.

At present, the platform allows users to explore products and add them to the cart, but it lacks a built-in payment gateway. Future developments will include the integration of popular and secure payment methods such as UPI, credit and debit cards, and other digital wallets. This will enable customers to complete their purchases directly on the site, streamlining the entire buying process.

The payment system will include secure payment forms, transaction validation, and handling of sensitive information through industry-standard practices like SSL encryption and tokenization. Additionally, features like transaction history, order confirmation, and support for refunds or failed transactions will be added to ensure transparency and build customer trust.

Integrating secure and efficient payment processing will not only improve the functionality of the PLUM platform but also prepare it for real-world commercial use. This advancement will significantly elevate the shopping experience and foster greater confidence among users when making online purchases.



## **BIBLIOGRAPHY**

### **Documentation**

- **React Documentation.** *React.js Official Documentation*. Retrieved from <https://reactjs.org/docs/getting-started.html>
- **MongoDB Documentation.** *MongoDB Official Documentation*. Retrieved from <https://www.mongodb.com/docs/>
- **Node.js Documentation.** *Node.js Official Documentation*. Retrieved from <https://nodejs.org/en/docs/>
- **Express.js Documentation.** *Express.js Official Documentation*. Retrieved from <https://expressjs.com/>
- **Tailwind CSS Documentation.** *Tailwind CSS Official Documentation*. Retrieved from <https://tailwindcss.com/docs>
- **MDN Web Docs.** *CSS Scroll Behaviour*. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/CSS/scroll-behavior>

### **References**

- **ChatGPT.** Retrieved from <https://chatgpt.com/>
- <https://www.udemy.com/course/mernstack/?couponCode=LEARNNOWPLANS>
- <https://www.udemy.com/course/web-development-zero-to-hero-complete-mern-stack-with-redux/?couponCode=LEARNNOWPLANS>