

BLOOMING BLISS FLOWER BOUQUETS

A PROJECT REPORT

Submitted by

DHARSHINI.S (920422205026)

MADHUBALA.K(920422205055)

SUDHAA SHREE.K (920422205109)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATIOTECNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution - Affiliated to Anna University, Chennai)

K.VELLAKULAM, VIRUDHUNAGAR - 625 701

NOVEMBER 2024

KAMARAJ COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution- Affiliated to Anna University, Chennai)
K.VELLAKULAM, VIRUDHUNAGAR - 625 701

BONAFIDE CERTIFICATE

Certified that the project report “**BLOOMING BLISS FLOWER BOUQUETS**“ is the bonafide work of “**Dharshini.S(920422205026), MadhuBala.K(920422205055), SudhaaShree.K(920422205109)**” who carried out the project work under my supervision.

SIGNATURE

Dr. E. VAKAIMALAR
Head of the Department,
Associate Professor,
Dept. of Information Technology,
Kamaraj College of Engg & Tech,
K.Vellakulam,
Virudhunagar - 625 701.

SIGNATURE

Dr. R. ARTHY
SUPERVISOR,
Assistant Professor,
Dept. of Information Technology,
Kamaraj College of Engg &Tech,
K.Vellakulam,
Virudhunagar - 625701.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Blooming Bliss Flower Bouquets is an online flower shop that offers a wide variety of fresh blooms, bouquets, and floral arrangements tailored for any occasion. With a user-friendly, ReactJS-powered interface, customers can easily browse and shop from a curated selection of flowers. The platform provides customization options to personalize bouquets, ensuring each arrangement is unique. Fast, same-day delivery guarantees freshness, while secure checkout options allow for a hassle-free shopping experience. Whether it's for a special event, a gift, or simply to brighten your space, Blooming Bliss helps you express emotions beautifully through flowers.

ACKNOWLEDGEMENT

I would like to thank to Dr. E. Vakaimalar, Head of the Department of Information Technology, for their encouragement and valuable insights throughout the development of the “BLOOMING BLISS FLOWER BOUQUETS” project.

Special thank to Dr. R. Arthy, my supervisor, whose guidance and expertise were instrumental in shaping the direction of this project. Their support helped me gain in-depth the knowledge of the MERN stack and its application. This project has been an invaluable learning experience, enhancing both my technical and problem-solving skills.

TABLE OF CONTENTS

| CHAPTER NO | TITLE | PAGE NO |
|------------|--|------------------------------------|
| | ABSTRACT | iii |
| | LIST OF FIGURES | vi |
| 1 | Error! Reference source not found. 1.1 Key Technologies 1.2 Features 1.3 How it Works 1.4 MERN Stack 1.5 How the MERN stack works? 1.6 How It All Comes Together | 1 2 2 2 3 4 |
| 2 | Error! Reference source not found. 2.1 Objective 2.2 Problem Statement 2.3 Block Diagram 2.4 Module Explanation | 5 5 6 7 |
| 3 | RESULTS AND DISCUSSION | 9 |
| 4 | Error! Reference source not found. | 16 |
| 5 | Error! Reference source not found. | 17 |

LIST OF FIGURES

| FIGURE NO | TITLE | PAGE NO |
|-----------|--|---------|
| 3.1 | HOME PAGE | 10 |
| 3.2 | FEATURED PRODUCTS | 10 |
| 3.3 | PRODUCT SPECIFICATOIONS | 11 |
| 3.4 | CONTACT PAGE | 11 |
| 3.5 | REGISTRATION PAGE | 12 |
| 3.6 | PLACE YOUR ORDER PAGE ORDER SUCCESSFUL PAGE | 12 |
| 3.7 | ABOUT US PAGE | 13 |
| 3.8 | DATABASE CONNECTIVITY | 14 |
| 3.9 | DATABASE CONNECTIVITY0 | 15 |

CHAPTER 1

INTRODUCTION

The Flower Shop application, a full-stack web project designed to provide a seamless shopping experience for customers looking to browse and purchase flowers. This application uses **ReactJS** for the front-end, creating an interactive and responsive user interface, while **MongoDB** serves as the database to store and manage flower inventory, user data, and order details.

1.1 Key Technologies:

- **ReactJS:** ReactJS is used to build the user interface, allowing for a smooth and responsive customer experience. It handles dynamic components such as product listings, cart updates, and real-time inventory checks.
- **Node.js :** The server-side of the application is built with Node.js and Express, providing a robust backend to handle API requests, manage business logic, and securely interact with the database.
- **MongoDB:** As a NoSQL database, MongoDB is highly scalable and capable of handling complex data structures. It stores flower details, customer information, and orders in collections, making data retrieval and updates efficient.

1.2 Features:

1. **Product Listings:** The flower inventory is dynamically fetched from the MongoDB database, allowing customers to view the latest products, prices, and availability.
2. **User Authentication:** The app includes user registration and login features, enabling customers to securely create accounts and manage their orders.

3. **Shopping Cart & Checkout:** Customers can add flowers to their cart, adjust quantities, and proceed to a secure checkout process.
4. **Admin Panel:** An admin panel allows authorized users to manage flower inventory, track orders, and update product information directly in the database.

1.3 How it Works:

- The React frontend communicates with the backend API through HTTP requests. For example, when a user browses flowers, React sends a request to the server, which then retrieves the data from MongoDB and sends it back as a JSON response.
- During the checkout process, the application saves order details to the MongoDB database, ensuring the order is tracked and managed.
- MongoDB's flexible schema makes it easy to update product details, add new items, and manage orders efficiently.

This combination of technologies results in a high-performance flower shop application that provides a pleasant user experience and an efficient platform for managing an online flower shop.

1.4 MERN STACK

MongoDB: A highly flexible, NoSQL database used to store journal entries and user data. It enables the Flower Shop App to scale efficiently, supporting large amounts of data without requiring rigid structures, which is ideal for managing diverse travel logs and user details.

Express.js: A minimal, fast, and robust web application framework built on Node.js. It manages the app's back-end logic, handling server-side routing and interactions with the database. Express simplifies connecting the front-end React components with the database, enabling smooth data flow.

React.js: A powerful front-end JavaScript library that allows for the creation of dynamic user interfaces. In the Flower Shop App, React's component-based

architecture ensures that features like writing, editing, and reading journals are rendered efficiently and updated in real time without refreshing the entire page.

Node.js: A versatile JavaScript runtime environment that runs on the server side. It processes user requests, manages routes, and interacts with the MongoDB database, allowing for a smooth, non-blocking performance. Node.js ensures that the app handles multiple user interactions and back-end processes effectively, making it highly scalable.

1.5 How the MERN stack works?

The **MERN stack**—which includes MongoDB, Express.js, React.js, and Node.js powers the Flower Shop App, letting users create, edit, and share their travel stories. Here’s how it all works together:

React.js (Front-End): Users interact with the app through React, writing, editing, and reviewing journal entries. Each part of the interface is a reusable component that updates dynamically without refreshing the page. It sends data to the server using HTTP requests.

Express.js (Back-End): Express handles incoming requests from React, routes them to the right endpoints (like saving or fetching journals), and ensures data is validated before heading to the database.

Node.js (Back-End Runtime): Running on Node.js, the server processes the requests from React and interacts with the database. It ensures multiple users can work smoothly at the same time.

MongoDB (Database): MongoDB stores all journal entries, user data, and travel info as documents, making it easy to manage, scale, and retrieve the data through CRUD operations.

1.6 How It All Comes Together:

1. **React Sends Order Data to the Server:** When a customer creates or edits an order in the flower shop (such as choosing flowers, specifying delivery details, and adding a personalized message), React gathers all the input data. This data is then sent to the backend using HTTP requests, typically through libraries like Axios or Fetch.
2. **Express Processes the Order Request:** On the server side, Express.js handles incoming requests, processes the data, and validates it to ensure accuracy (like checking stock availability for chosen flowers). It then routes the request to the appropriate endpoint, depending on whether the action is to create a new order, update an existing order, or retrieve order details.
3. **MongoDB Stores the Order Information:** After processing, Express forwards the order data to MongoDB. The order, including details like selected flowers, delivery date, customer information, and custom messages, is stored as a document within an orders collection. MongoDB's flexible schema accommodates various order details and any additional specifications.
4. **Server Responds, React Updates the UI:** Once the order is successfully saved in MongoDB, the server sends a response back to React. React then updates the UI, either by displaying a confirmation message or by updating the order list to reflect the new or edited order. This process is handled dynamically, ensuring a smooth user experience without the need to refresh the entire page.

CHAPTER 2

METHODOLOGY

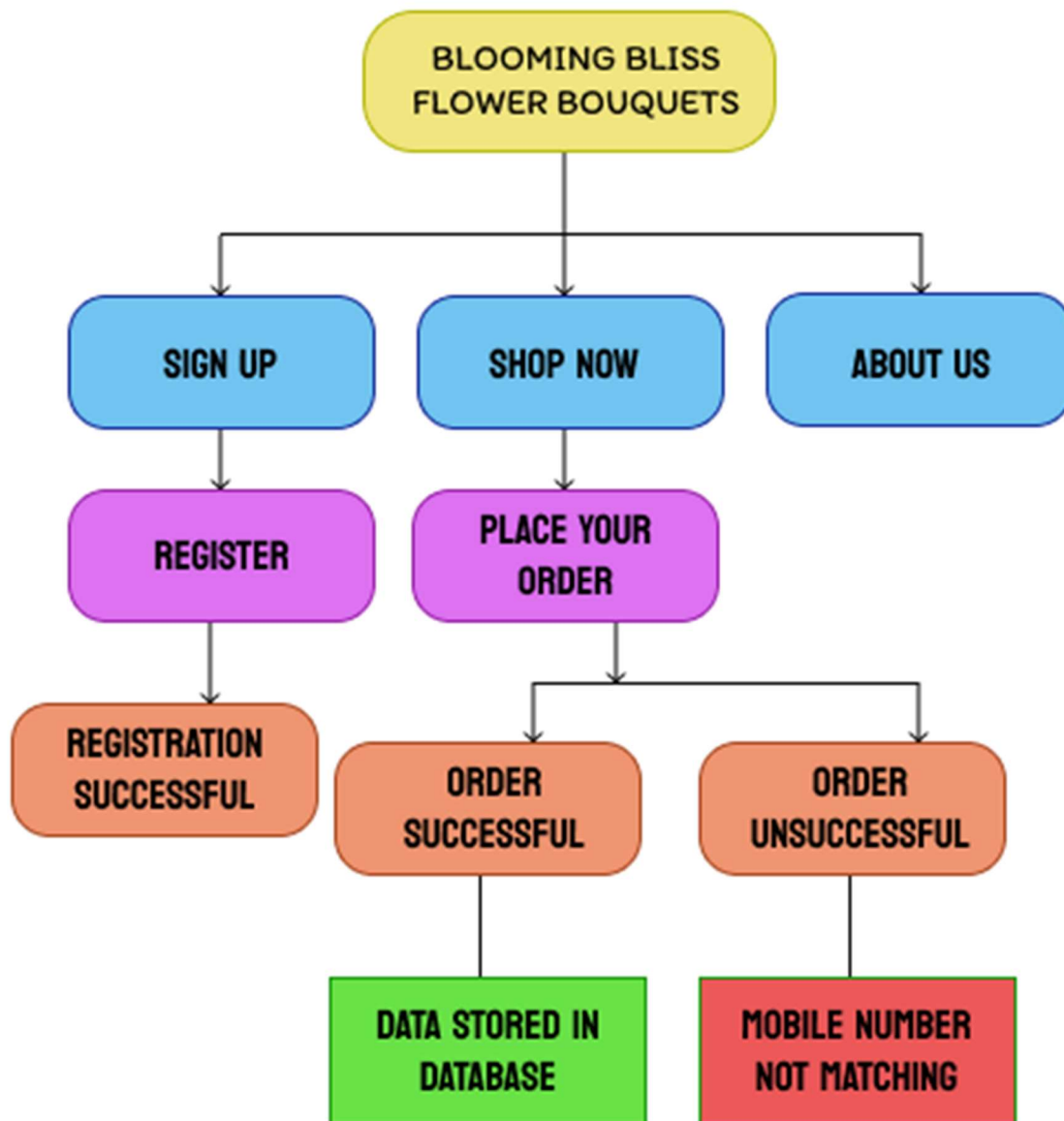
2.1 OBJECTIVE

The objective of the Flower Shop App is to offer customers a seamless platform for browsing, customizing, and purchasing beautiful floral arrangements, while also providing real-time order tracking and personalization options. This creates a personalized and interactive shopping experience, making it easy for customers to find the perfect flowers for any occasion.

2.2 PROBLEM STATEMENT

The problem addressed by the Flower Shop App is the absence of a convenient, user-friendly platform where customers can easily browse, customize, and purchase floral arrangements. Many traditional flower shopping experiences lack personalization, real-time order tracking, and efficient online ordering options, which can lead to a disconnected and time-consuming experience for customers trying to find the perfect flowers for their needs.

2.3 BLOCK DIAGRAM



2.4 MODULE EXPLANATION

User Interface (React Front-End):

- **Browse Products:** This component displays the available floral arrangements in an organized format. Customers can view flowers sorted by categories, price, or other relevant filters. Each product listing includes details like name, description, price, and available customizations.
- **Product Details:** This module shows more detailed information about a selected flower or bouquet. Users can view larger images, read a detailed description, choose custom options (such as size or add-ons), and add the item to their shopping cart.
- **Shopping Cart:** Provides a summary of selected products. Users can adjust quantities, remove items, and see the total cost. Changes made in the cart are dynamically reflected without page reloads, and users can proceed to checkout from this module.
- **Checkout Form:** Collects customer details for billing, shipping, and payment. Input fields include name, delivery address, payment information, and any special delivery instructions. Upon submission, the data is processed and saved to the back-end.
- **Order Confirmation & Tracking:** After a successful order, this component displays an order summary with tracking information. Users can check order status and expected delivery dates without leaving the page.

Back-End (Order Management Form):

- **Order Management Form:** Handles order details, allowing customers to submit their name, delivery address, payment info, and other order-related details. Once the form is completed, data is processed by the back-end server (Node.js/Express) and saved in the MongoDB database.

Back-End Logic (Express.js):

- **API Management:** Express manages API routes and processes requests from the front-end. It validates incoming data (such as payment and delivery information), handles responses, and maintains communication between React and MongoDB.

Server-Side Logic (Node.js):

- **Order Processing:** Manages back-end operations like handling incoming requests, managing database interactions, and ensuring the app runs asynchronously to handle multiple customer orders efficiently.

Database (MongoDB):

- **Order & Product Data Storage:** MongoDB stores details of orders, products, and customer information. It offers flexibility for storing text, dates, images, and customizations, allowing for efficient data retrieval and updates as customers place or modify orders.

CHAPTER 3

RESULT AND DISCUSSIONS

The Flower Shop App effectively provides users with a convenient platform to browse, customize, and purchase floral arrangements. The front-end, developed with React, offers a smooth and responsive user experience, allowing customers to navigate through products, add items to the cart, and complete purchases with real-time updates and without page reloads.

On the back-end, Node.js and Express.js manage requests efficiently, ensuring that order details, customer information, and product data are securely processed and stored in MongoDB. The app's functionality includes essential features like product browsing, order customization, shopping cart management, and checkout, creating a cohesive and interactive shopping experience.

Overall, the Flower Shop App successfully addresses the need for a streamlined online flower shopping experience, demonstrating its effectiveness in providing users with a comprehensive and personalized platform for selecting and purchasing floral arrangements.

SCREENSHOTS

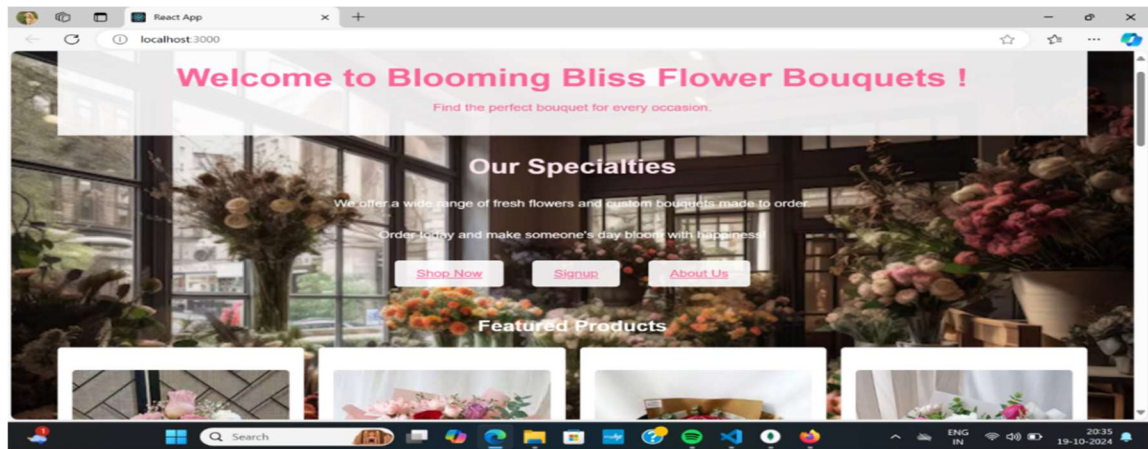


Figure 3.1 Home page

This is a homepage for "Blooming Bliss Flower Bouquets." It features a welcoming banner, highlights the store's flower and bouquet customization services, and offers options to shop, sign up, or learn more about the business.

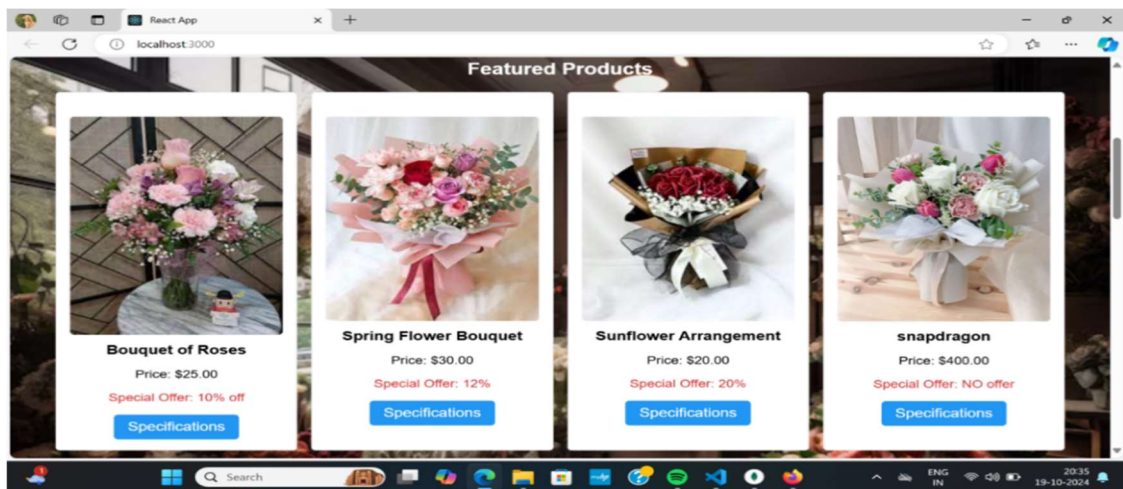


Figure 3.2 Featured products

This section showcases featured floral products with images, prices, and special discount offers for select bouquets. Each product has a "Specifications" button for further details

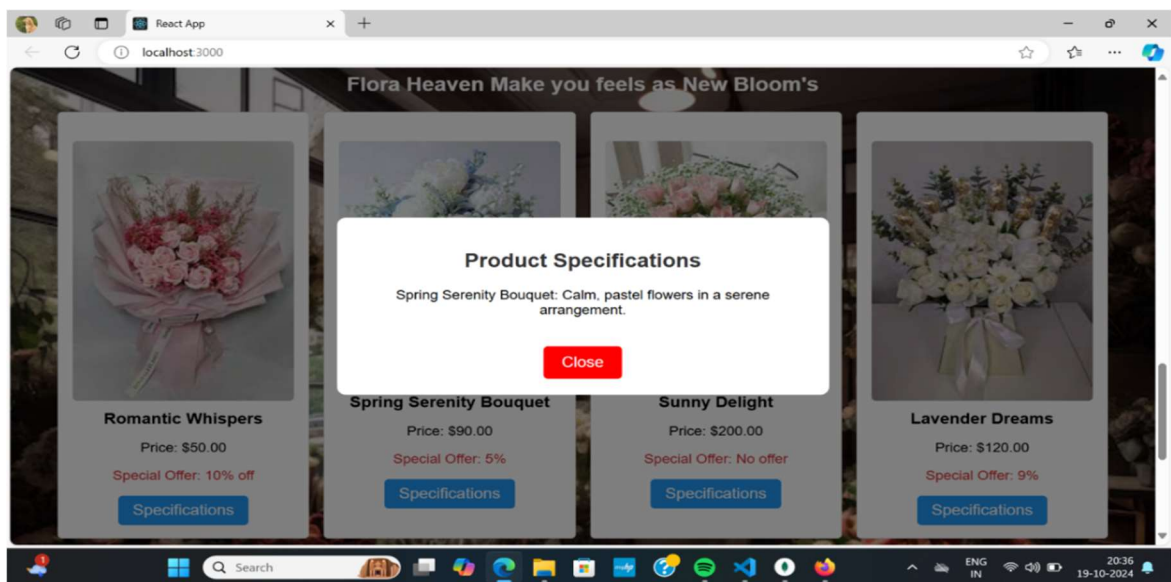


Figure 3.3 Product specifications

This image displays a pop-up window showing the product specifications

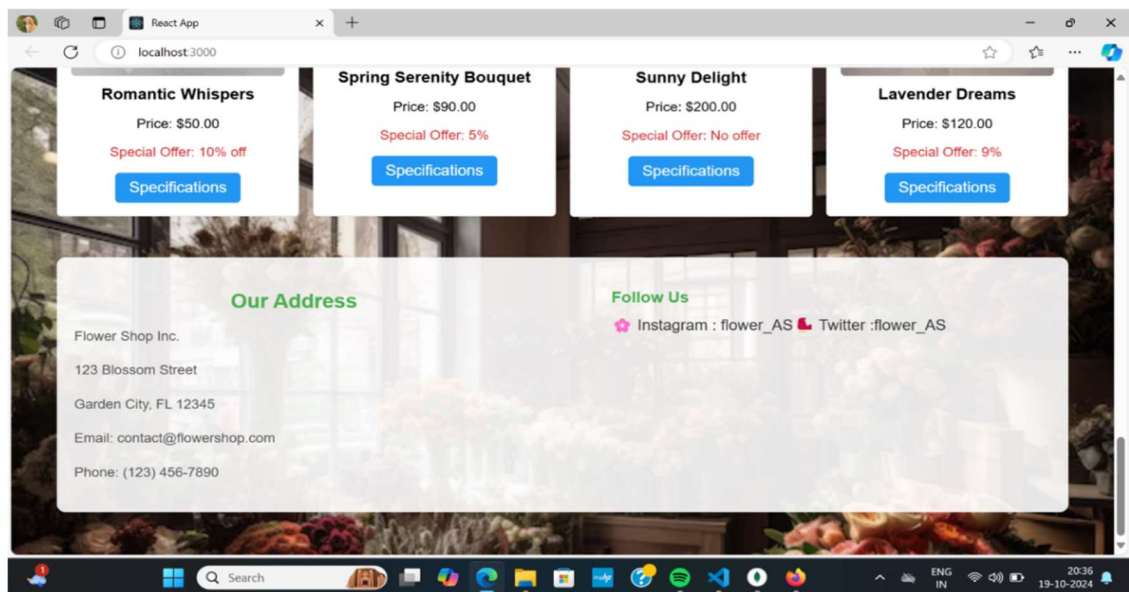


Figure 3.4 Contact page

This image displays our bouquet shop's address and our contact details

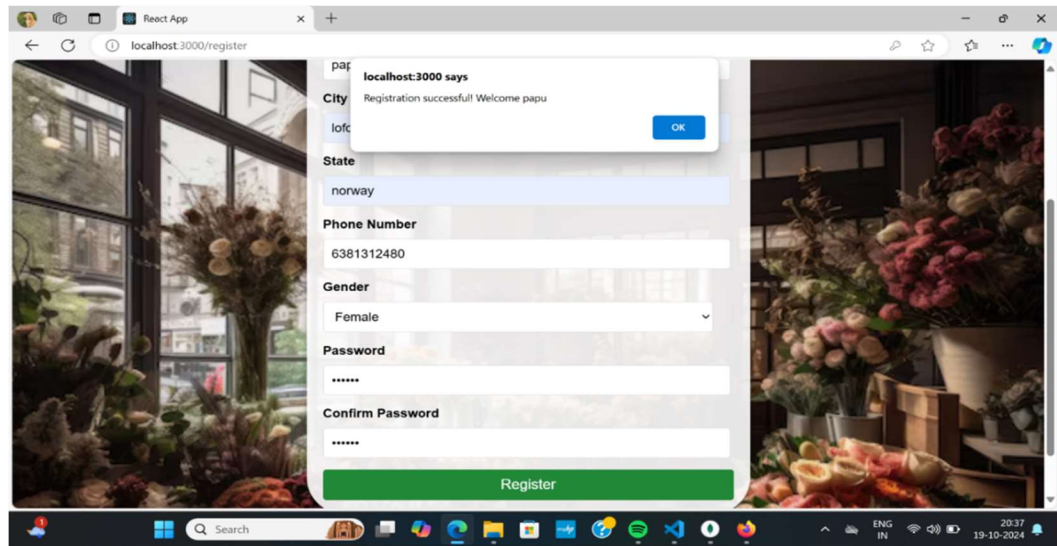


Figure 3.5 Registration page

This image shows that once all the details filled in the form , the registration completes and welcomes you to the shopping page.

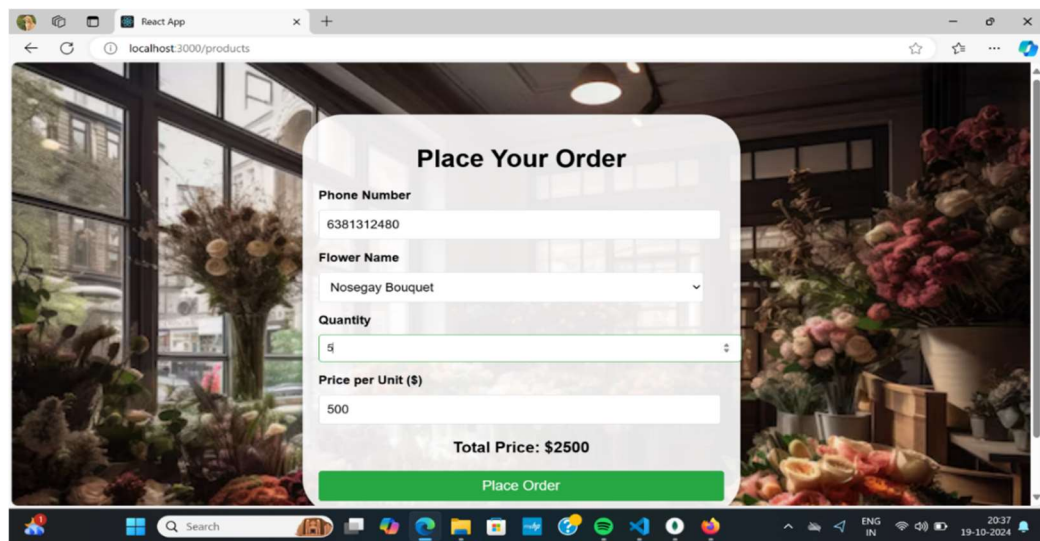


Figure 3.6 Place your order page

This image shows that to place your order and displays the total price of your orders made.

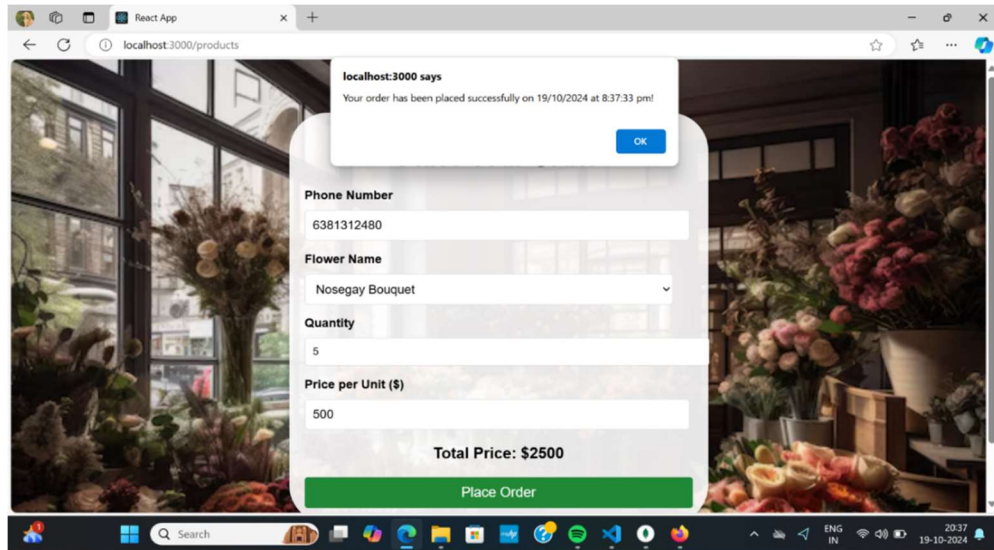


Figure 3.7 Order successful page

This image shows that your order have been placed successfully .

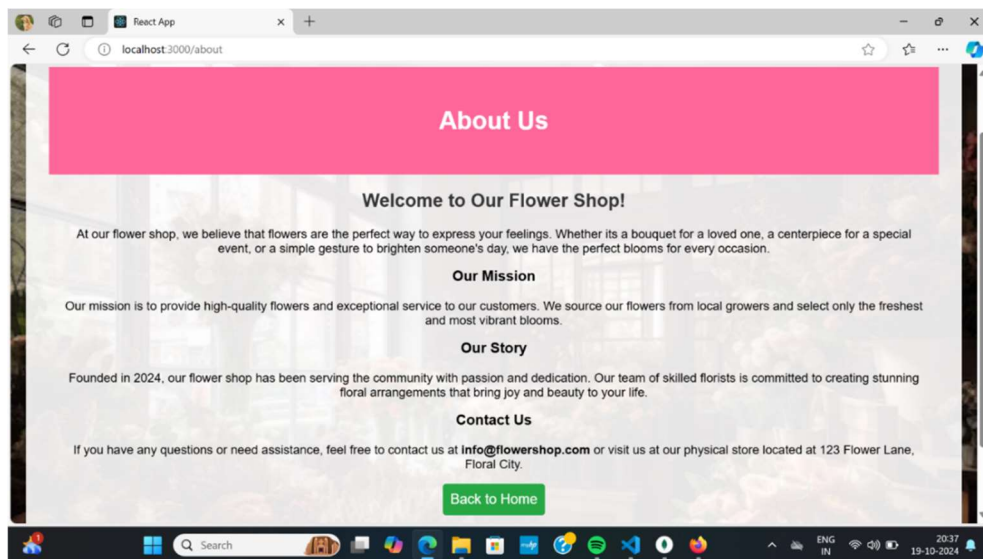


Figure 3.8 About us page

This image shows that details of our bouquet shop

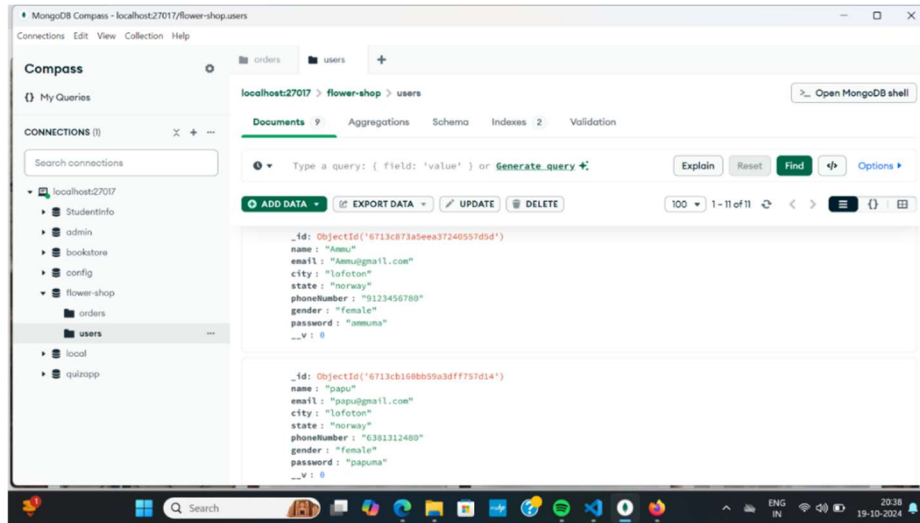


Figure 3.9 Database connectivity

This image shows that the user page have been connected to the database

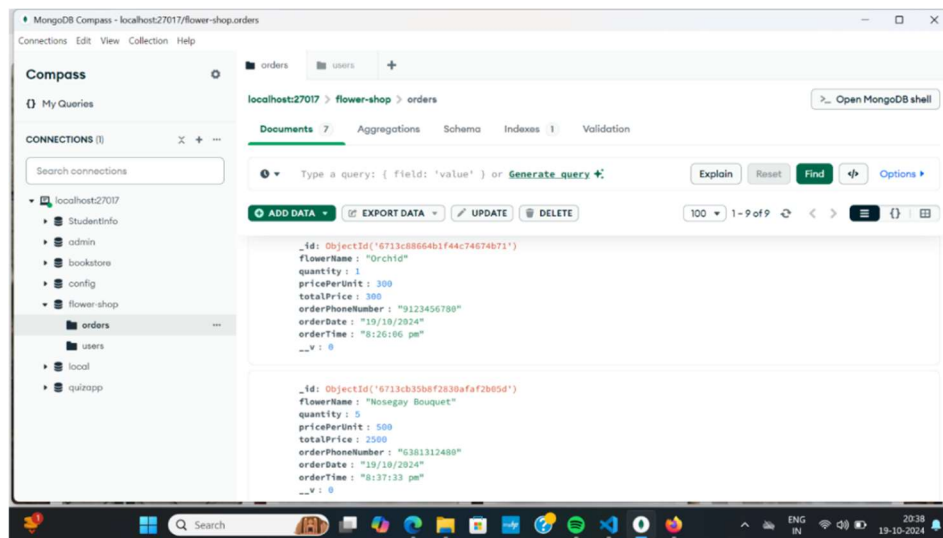


Figure 3.10 Database connectivity

This image shows that the order page have been connected to the database

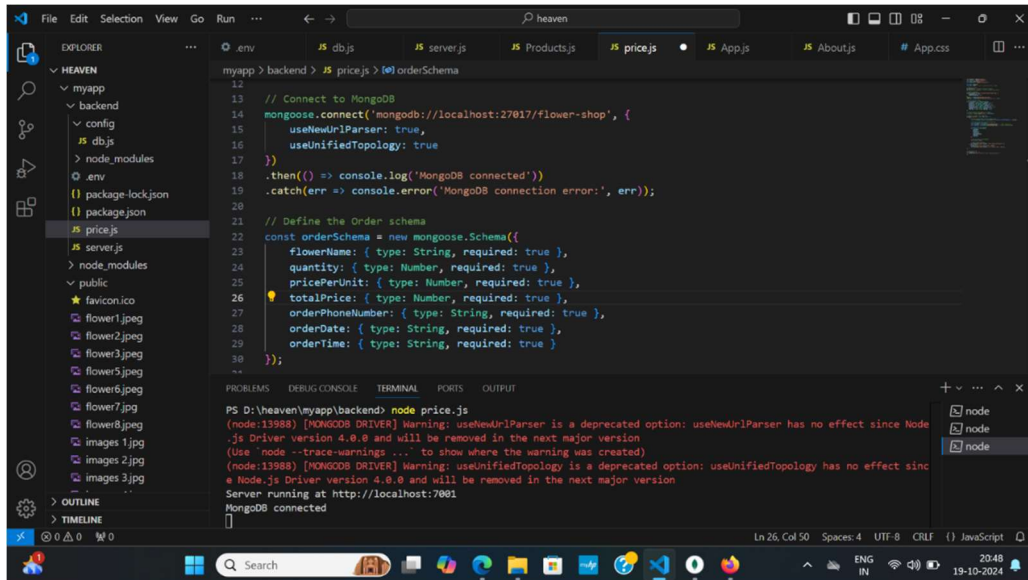


Figure 10.11 Database connectivity

This image shows that the code have been connected to the database

CHAPTER 4

CONCLUSION

In conclusion, building a flower bouquet shop website using React.js for the frontend and MongoDB as the database provides an efficient, modern, and scalable solution. React's dynamic UI capabilities allow for a smooth and responsive user experience, while MongoDB's flexibility supports efficient storage and retrieval of product and customer data. Together, they create a robust and interactive platform for managing orders, displaying products, and engaging users, making it ideal for a growing online flower business.

CHAPTER 5

REFERENCES

1.React Documentation

<https://reactjs.org/docs/getting-started.html>

2.MongoDB Documentation

<https://www.mongodb.com/docs/>

3.MERN Stack Tutorial

<https://www.freecodecamp.org/news/mern-stack-tutorial/>

4.Express.js Documentation

<https://expressjs.com/>

5.Mongoose Documentation

<https://mongoosejs.com/docs/guide.html>