**FULL STACK ENGINEERING**

**Project Report**

**Semester-VI (Batch-2022)**

**“TOMATO-A Food Delivery Website”**

**A red and white sign

Description automatically generated with low confidence**

**Supervised By: Submitted By:**

**Mr .Rahul Saloni(2210992231)**

**G-22 Saloni kumari(2210992232)**

**Samreen Kaur(2210992240)**

**Samridhi Sachdeva(2210992243)**

**Department of Computer Science and Engineering**

**Chitkara University Institute of Engineering & Technology,**

**Chitkara University, Punjab**



**Index**

**Sr. no Topic Page No**

1 Title of project 03

1. Objective & Key Learning’s 03

3 Problem Statement 03

4 Options available to execute the project 04

5 Advantages/ Disadvantages 05

6 Demonstration 07

7 References 09



**1.Title of Project :**

"Tomato: Revolutionizing Convenient and Efficient Dining Experiences"

**2. Objective & Key Learning’s :**

**Objective:**

The primary objective of this project is to design and develop a food delivery application that provides users with a seamless experience for ordering meals. The system aims to:

* Support a diverse range of restaurant options.
* Offer real-time order tracking for enhanced transparency.
* Store and manage order history for easy reference.
* Display personalized recommendations based on user preferences.

**Key Learning’s:**

* Database Management: Storing user profiles, chat history, and real-time updates.
* Scalability: Designing a scalable system architecture for handling concurrent users.
* User Interface (UI) Design: Building an intuitive and user-friendly interface.
* Security: Implementing secure messaging features like end-to-end encryption, authentication, and data protection.

**3.Problem Statement:**

In today's fast-paced world, quick and reliable access to food is a critical necessity for individuals with busy lifestyles. Existing food delivery platforms often fall short in providing an intuitive, seamless experience. Common challenges include inefficient navigation, lack of real-time order tracking, limited restaurant options, and absence of personalized features that cater to individual preferences.

* There is a growing demand for a modern food delivery application that overcomes these



limitations by offering a user-friendly interface, a wide selection of restaurants, personalized recommendations, and instant updates on order status. By addressing these needs, the application can enhance the overall dining experience, providing convenience and efficiency to users.

**4. Options Available to Execute the Project:-**

**a. Frontend Technologies:**

* React.js: A widely used JavaScript library for building user interfaces with reactive
* HTML/CSS: For basic structure and styling.

**b. Backend Technologies:**

* Express.js: A lightweight and fast Node.js framework for building robust APIs and handling server-side logic.
* APIs: Used for facilitating communication between the frontend and backend, enabling seamless data exchange.
* Axios: A popular library for making HTTP requests to APIs, ensuring efficient communication between the client and server.
* Middleware: Implemented to handle tasks like authentication, request parsing, and error handling efficiently.
* MongoDB: A NoSQL database for storing and managing user profiles, order history, and menu data.



**5. Advantages/Disadvantages**

**Advantages :**

* **Convenience:** Users can order food from their favorite restaurants without leaving their homes, saving time and effort
* **Scalability**: **:** Live order tracking and instant notifications keep users informed about the status of their orders.
* **Cross-platform**: Using frameworks like Flutter or React Native ensures the app works seamlessly on both Android and iOS devices.
* **Real-time Updates**: Live order tracking and instant notifications keep users informed about the status of their orders.

**Disadvantages:**

* **Latency Issues**: Unexpected delays in food delivery due to traffic, weather, or logistical .

issues can impact user satisfaction.

* **Latency Issues**: Unexpected delays in food delivery due to traffic, weather, or logistical issues can impact user satisfaction.
* **Complexity in Implementation**: Ensuring low latency, scalability, and security adds complexity.
* **Security Risks**: Handling sensitive user data like payment information and delivery addresses requires robust encryption and security protocols to prevent breaches..
* **Cost**: Hosting real-time applications, especially at scale, can be expensive due to server requirements and data traffic.



**6. Demonstrations :**

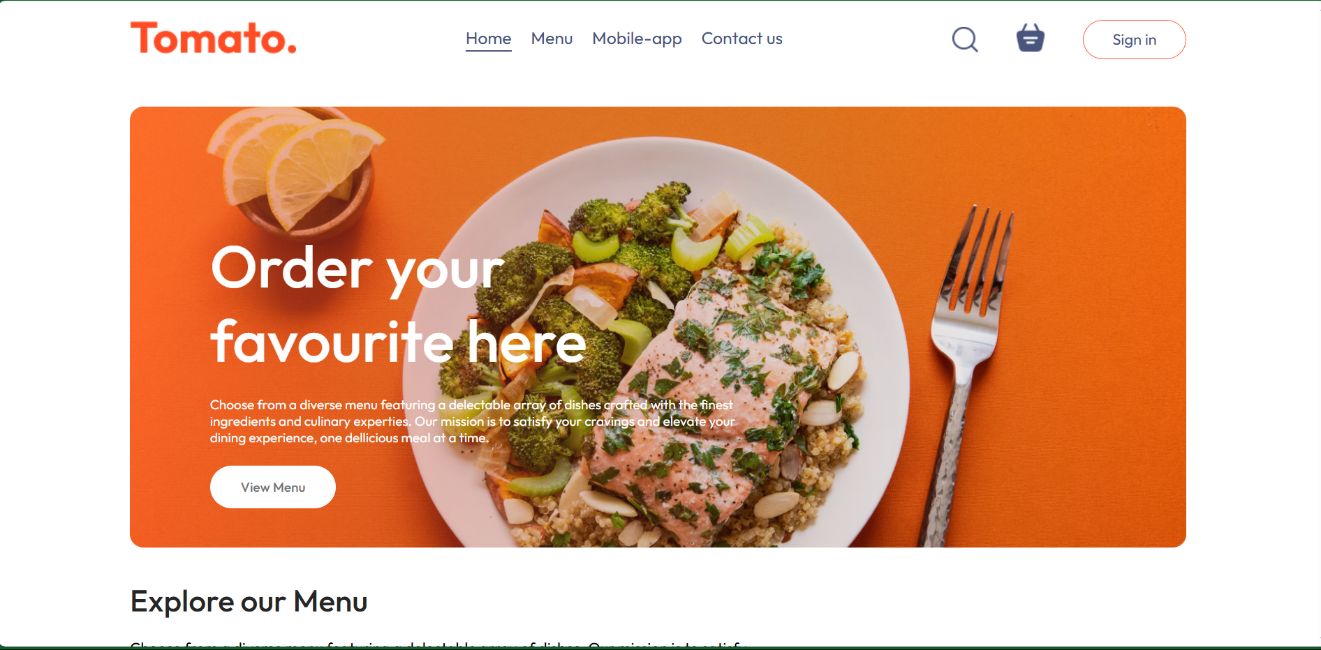
****

Figure Home page

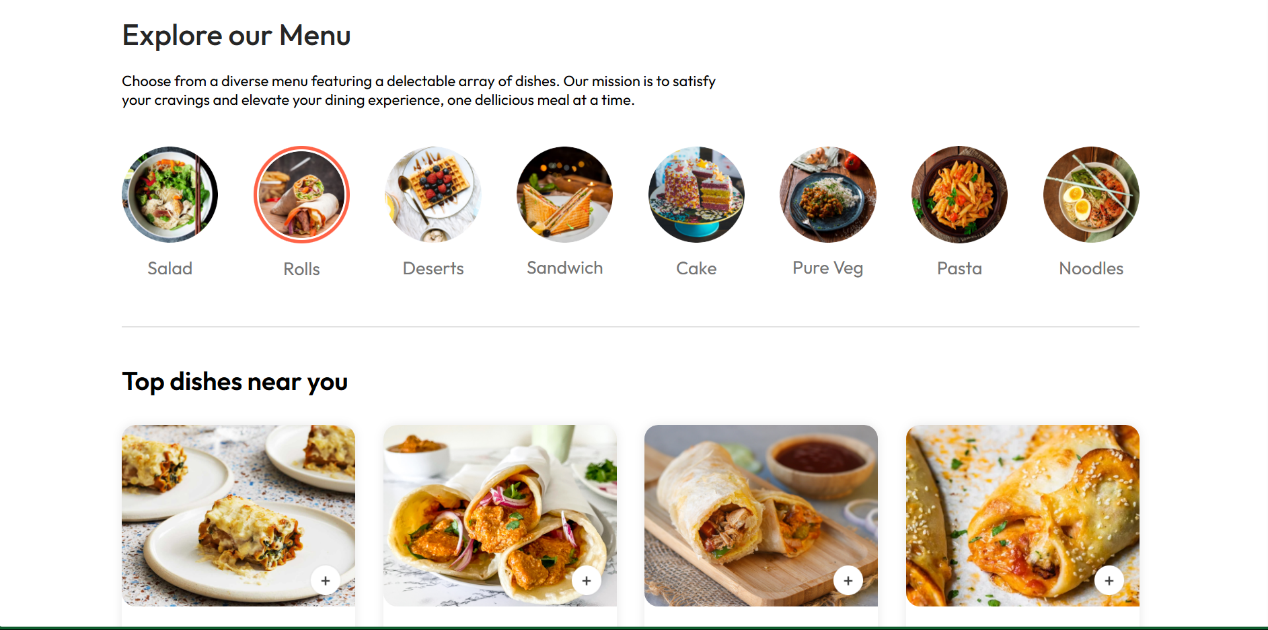
****

Figure Menu page



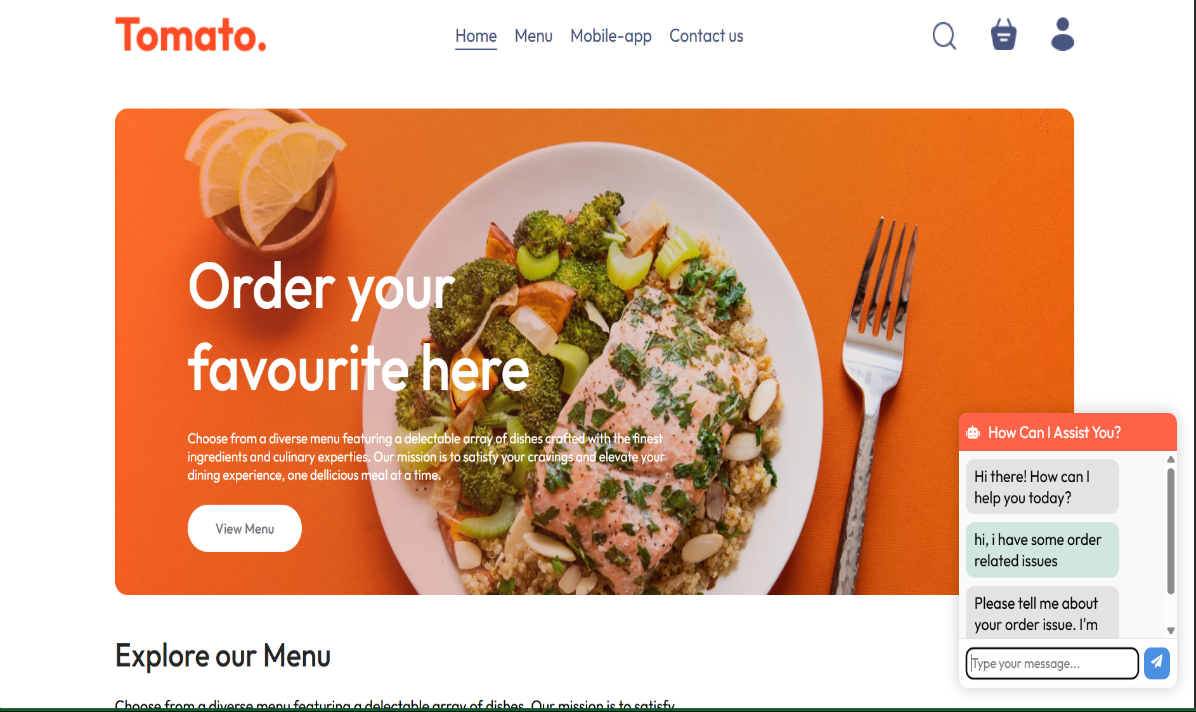
****

Figure Chatbot

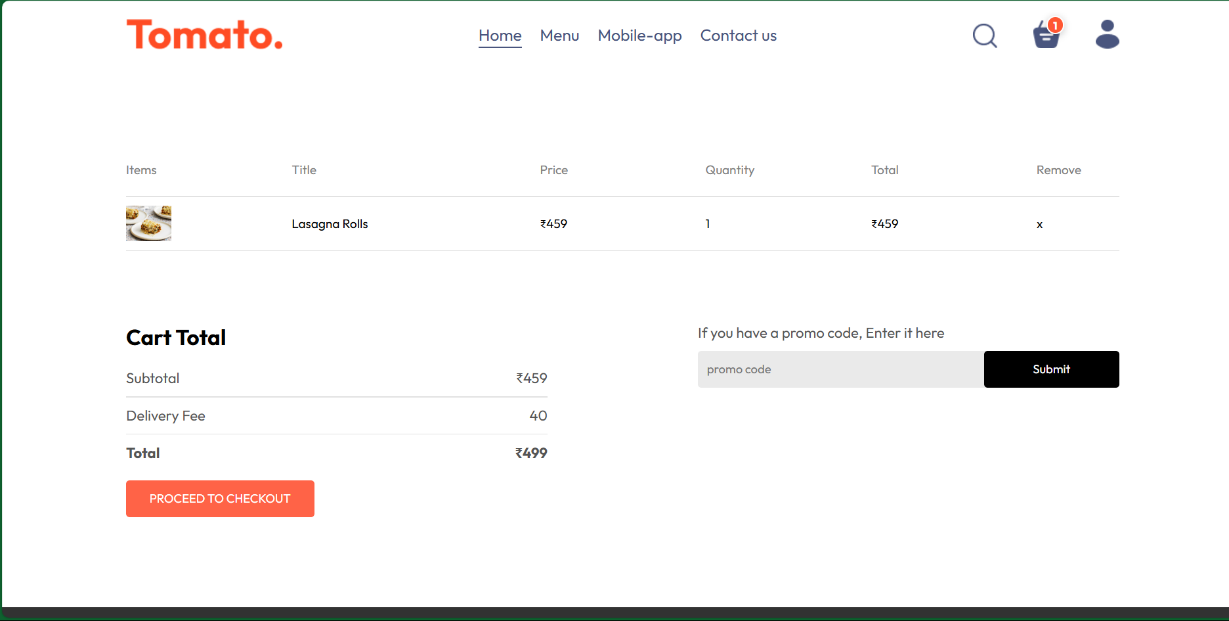
****

Figure 4 Cart page



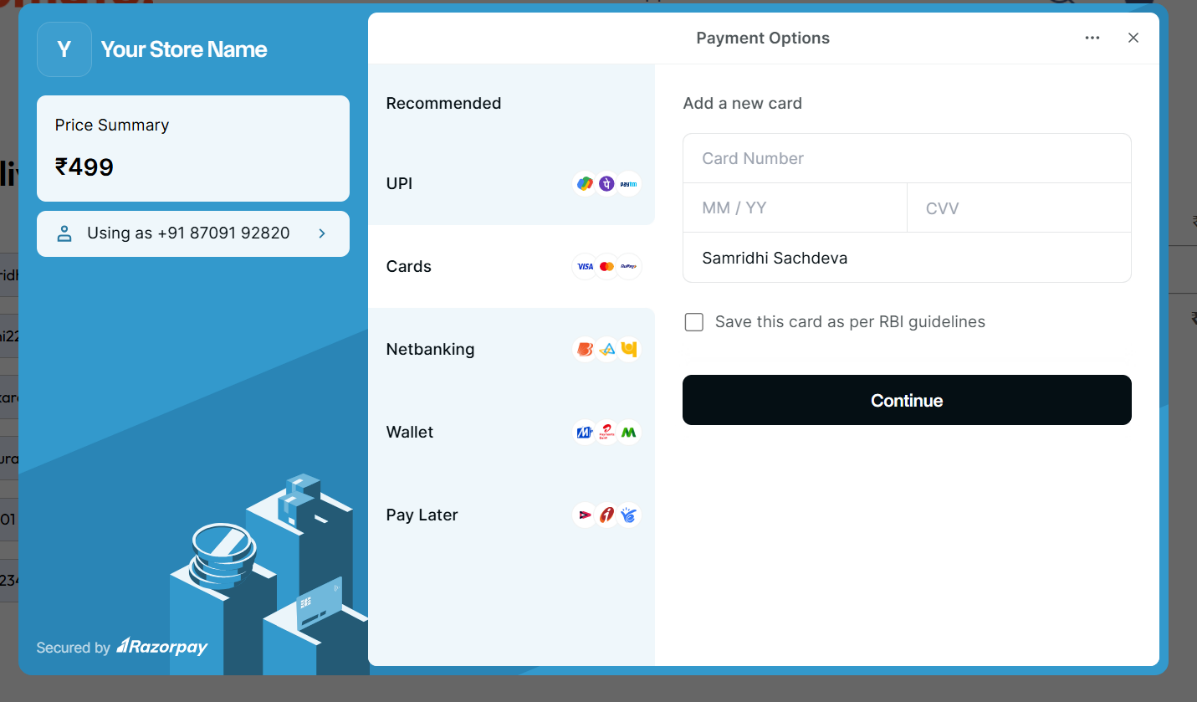
****

Figure 5 Payemnt

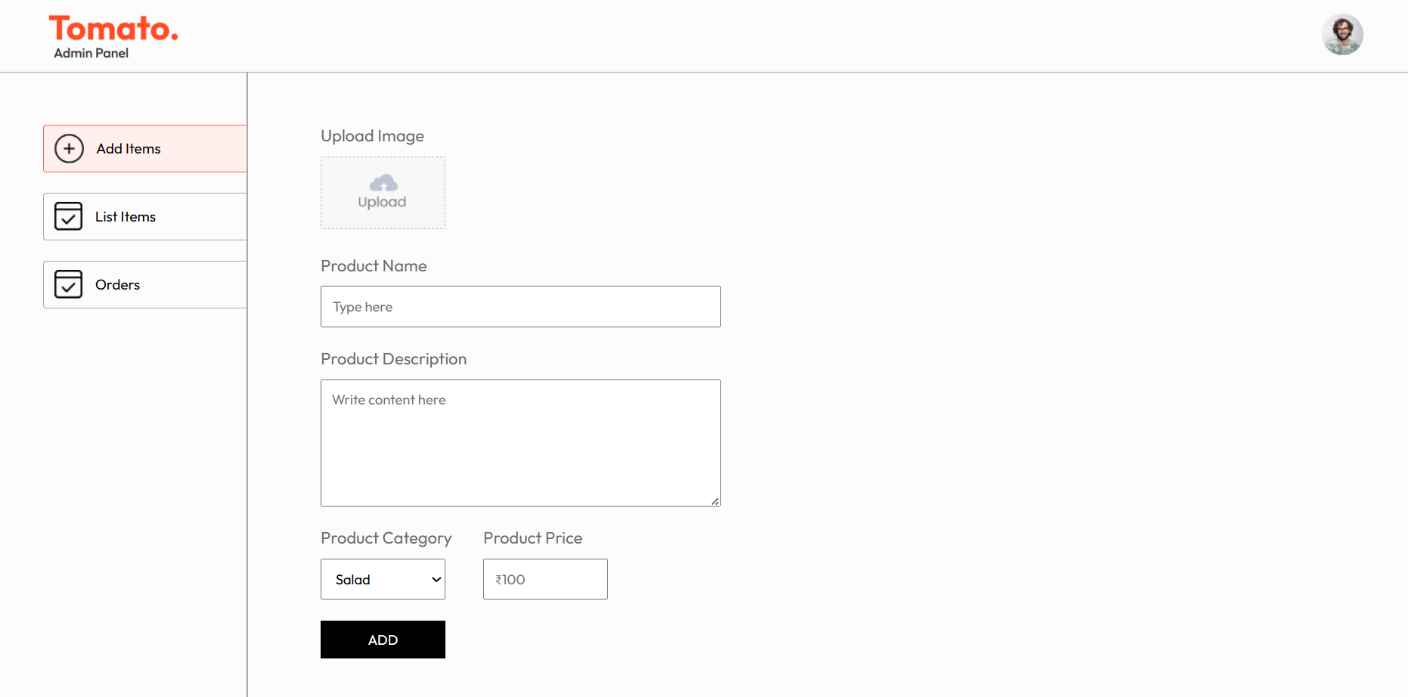
****

Figure 6 Admin Panne



**7. References**

* **Node.js Official Site:** Comprehensive documentation and guides for backend development using Node.js. ([nodejs.org](https://nodejs.org))
* **Express.js Documentation:** Official reference for using Express.js to build server-side logic and APIs. ([expressjs.com](https://expressjs.com))
* **MongoDB Documentation:** Resources for database integration and data management. ([mongodb.com/docs](https://www.mongodb.com/docs))
* **Axios GitHub Repository:** Details on Axios library for API requests. ([github.com/axios/axios](https://github.com/axios/axios))
* **React Documentation:** Guides and tutorials for building user interfaces. ([react.dev](https://react.dev))
* **W3Schools HTML/CSS Tutorials:** For foundational web development knowledge. ([w3schools.com](https://www.w3schools.com))