A SMART FOOD DELIVERY SYSTEM

BY

MD ZISHAN KABIR CHOWDHURY

ID: 182-15-11449

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

DR. MD. FOKHRAY HOSSAIN

Professor

Department of CSE

Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

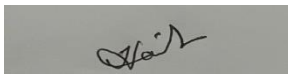
DHAKA, BANGLADESH

Fall 2024

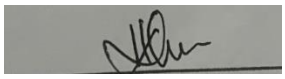
APPROVAL

This Project titled **"A SMART FOOD DELIVERY SYSTEM"**, submitted by **Md Zishan Kabir Chowdhury** to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on January 13th, 2025.

BOARD OF EXAMINERS



Dr. Sheak Rashed Haider Noori (SRH)
Professor and Head **Chairman**
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University



Most. Hasna Hena (HH)
Assistant Professor **Internal Examiner**
Department of Computer Science and Engineering
Faculty of Science & Information Technology Daffodil International University

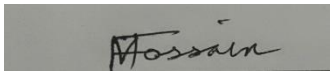
Md. Ferdouse Ahmed Foysal (FAF)
Lecturer **Internal Examiner**
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Dr. Md Arshad Ali (DAA)
Professor **External Examiner**
Department of Computer Science and Engineering
Hajee Mohammad Danesh Science and Technology University

DECLARATION

I hereby declare that, this project has been done by me under the supervision of **Professor Dr. Md. Fokhray Hossain, Professor, Department of CSE** Daffodil International University. I also declare that neither this project nor any part of this research project has been submitted elsewhere for award of any degree.

Supervised by:



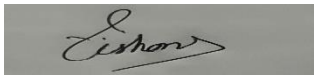
Professor Dr. Md. Fokhray Hossain

Professor

Department of CSE

Daffodil International University

Submitted by:



Md Zishan Kabir Chowdhury

ID: 182-15-11449

Department of CSE

Daffodil International University

ACKNOWLEDGEMENT

Firstly, I am express my heartiest thanks and gratefulness to almighty Allah for His divine blessing makes me possible to complete the final year research project successfully. I am really grateful and wish my profound our indebtedness to **Dr. Md Fokhary Hossain, Professor, Department of CSE, Daffodil International University, Dhaka**. Deep Knowledge & keen interest of our supervisor in the field of "Web Application" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project. I would like to express our heartiest gratitude to **Dr. Sheak Rashed Haider Noori, Professor and Head Chairman, Department of CSE**, **DIU** for his kind help to finish my project and also to other faculty member and the staff of CSE department of Daffodil International University. I would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work. Finally, I must acknowledge with due respect the constant support and patients of my parents.

ABSTRACT

This project report describes how to use the MERN stack (MongoDB, Express, React, and Node.js) to create and implement a food delivery application. Creating an intuitive and effective platform was the aim of this project that allows for easy meal ordering and delivery by connecting patrons with the pizza restaurant.

The Project's goals include creating a user interface that is easy to use, putting in place a reliable backend system, and guaranteeing data security. Because of its adaptability, scalability and capacity to manage intricate web applications, the MERN stack was selected.

The database design, which uses MongoDB to store user and order records, is the first step in the thorough development process. Node.js and Express.js were used to build the backend system, which offers a RESTful API for managing a variety of tasks like user order processing and authentication. React.js was used to complete the front-end development, with an emphasis on producing an interactive and responsive user experience. Clients are able to place orders and examine menus. The paper talks about the technical difficulties encountered during the development process and the remedies that were put in place as a result.

To sum up, the MERN stack-developed Food Delivery Application shows how to successfully create a feature-rich platform that enables smooth food ordering and delivery services. Using contemporary web technology and reliable development approaches guarantee a user-friendly experience while satisfying the needs of patrons and restaurant operators.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of the Project

This project focuses on developing an online food ordering platform using the MERN stack, which integrates MongoDB, Express.js, React.js, and Node.js to create a seamless, full-stack web application. The global food delivery industry has witnessed rapid growth driven by changing consumer preferences, urbanization, and the demand for convenience. The development of user-centric, efficient, and scalable online platforms is critical to meet this growing demand. While many existing food ordering platforms offer basic services, they often lack the seamless user experience and performance scalability required in today's market. The MERN (MongoDB, Express.js, React.js, Node.js) stack provides an integrated, full-stack JavaScript framework that can power robust web applications with real-time functionality and flexible data management.

To create an efficient, modern online system for ordering food that enhances user experience and supports restaurant operations.

1.2 Problem Statement

In the field of web development, there is a constant need for efficient and scalable technologies that enable the creation of modern, dynamic web applications. The traditional approach, which uses multiple technologies for different web development jobs often leads to integration issues, inefficiencies, and complexity.

Current Issues: Traditional food ordering processes are often slow, prone to human errors, and inconvenient for both customers and restaurant owners. Existing platforms may not cater to smaller local businesses effectively. Even while meal ordering websites are widely available, many still have serious

issues with scalability, real-time performance, and consumer happiness. These restrictions frequently lead to more complicated user interfaces, slower reaction times, and operational inefficiencies.

Need for Solution: There is a need for a comprehensive digital platform that simplifies the ordering process, reduces manual work, and provides quick service to users. By creating a comprehensive solution with the MERN stack to produce a high-performance and captivating online platform, this research project aims to address these problems.

1.3 Objectives

The primary objective of this research project is to leverage the MERN stack to create a scalable, user-friendly food ordering website that improves operational efficiency and user experience by integrating real-time order tracking and optimal data management. To build a responsive, secure, and efficient online food ordering website that connects users with restaurants seamlessly, allowing for easy menu browsing, ordering, and payment. Key goals include:

1. Real-Time Functionality: Implement live order tracking and updates to enhance user engagement.
2. Scalability: Build a system capable of handling growing user demands and high traffic efficiently.
3. Interactive Design: Develop an intuitive and responsive user interface for better navigation and usability.
4. Efficient Data Management: Use MongoDB to ensure smooth handling of orders, user data, and restaurant details.
5. Practical Learning: Apply and demonstrate intermediate-level skills in full-stack development, bridging theoretical knowledge with real-world application.

Increasing web development productivity and addressing the challenges developers have when utilizing traditional web development methodologies are the main objectives of this project. Offering a comprehensive and scalable solution that expedites the development process is our goal, increases efficiency and guarantees top-notch performance and maintainability of modern web applications with the use of the MERN stack.

1.4 Research Methodology

Techniques for Improving the MERN Stack's Efficiency in Web Development:

MERN Stack Overview:

- **MongoDB:** Used for database management due to its flexibility and scalability.
- **Express.js:** Simplifies backend development with minimal effort.
- **React.js:** Provides a fast, interactive front-end experience.
- **Node.js:** Ensures efficient server-side performance.

System Architecture:

1. Users interact with the React front-end.
2. React communicates with the Node.js server via REST APIs built on Express.
3. Data is stored and managed using MongoDB.

Key Features:

- **User Authentication:** Secure login and registration.
- **Menu Browsing:** Easy-to-navigate menus with food item details.
- **Order Management:** Real-time order processing.
- **Payment Processing:** Integration with secure payment gateways.

Project Workflow

Phases:

- **Phase 1:** Design front-end components using React.
- **Phase 2:** Develop backend APIs using Node.js and Express.js.
- **Phase 3:** Integrate MongoDB for data storage.
- **Phase 4:** Test and refine the system for a seamless user experience.

Tools Used:

- **MongoDB:** MongoDB Atlas MongoDB Compass for database hosting.
- **Node.js:** Node.js for server-side development.
- **React.js:** React Developer Tools for debugging and optimizing the front-end.

The overall objective of this technique is to offer a complete and cohesive solution that guarantees superior performance and maintainability of contemporary online applications using the MERN stack, accelerates web development, and boosts efficiency. To give a this approach to web development is sustainable and future-proof, emphasizing best practices, standard compliance, and ongoing maintenance and enhancements.

1.5 Proposed Solution

Measures of Success:

- An interface that is easy to use and intuitive.
- Secure transactions and fast loading times.
- Restaurant proprietors may easily utilize this order management system.

Impact on Users:

- Real-time updates and simple ordering have increased client satisfaction.
- Improved capacity for managing a restaurant company

1.6 Conclusion

Using the MERN stack to create an online food ordering website will show off the stack's benefits for creating cutting-edge, scalable, and fast web apps. The current shortcomings of many food delivery platforms are addressed by this research project, which also satisfies industry demands for better user experiences and operational effectiveness on a worldwide scale. Utilizing state-of-the-art technology, this platform will establish a standard for online meal delivery services in the future. By offering a platform that is user-friendly, adaptable, and reasonably priced, this project seeks to close the current gaps in the food delivery industry.

Including features like machine learning-powered tailored recommendations, delivery monitoring, and mobile application expansion to improve user engagement.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This section explores the historical evolution of food delivery platforms, current trends in the e-commerce domain, and the adoption of modern web technologies like the MERN stack. By analyzing related studies and platforms, this review identifies key challenges, opportunities, and gaps that the proposed project aims to address.

It also examines essential features of food ordering systems, such as user interfaces, payment integration, and real-time order tracking, to ensure a comprehensive understanding of industry standards. This review forms the foundation for the project's design and implementation, offering insights into how existing solutions can be enhanced to create a more efficient, user-friendly, and sustainable platform.

2.2 Literature Review

The evolution of online food ordering systems has been driven by the demand for convenience, personalization, and real-time services. These platforms have shifted from basic menu browsing to providing full-service solutions, including personalized recommendations, payment processing, and delivery tracking. User Experience (UX) has become a key focus, as consumers expect intuitive and responsive platforms. Machine learning algorithms are increasingly used to recommend meals based on user preferences, location, and past orders, enhancing personalization and driving customer engagement.

The MERN stack—comprising MongoDB, Express.js, React.js, and Node.js—has become a popular choice for building modern food ordering platforms. Its JavaScript-based architecture enables developers to build dynamic and scalable applications that handle complex data flows efficiently. MongoDB’s flexible, NoSQL database structure is ideal for managing various types of data, such as customer orders, restaurant menus, and real-time interactions. React.js powers the frontend with a responsive, component-

based user interface that ensures a smooth browsing experience, while Node.js and Express.js handle backend operations like API requests, user authentication, and payment processing.

In addition to the core technologies, integration with third-party services like Stripe for payment handling and Cloudinary for image storage helps enhance functionality. These integrations allow developers to create more comprehensive systems that manage both financial transactions and multimedia assets seamlessly. However, despite the benefits of using the MERN stack, challenges remain in scaling databases for real-time data management, particularly when dealing with large volumes of user data, delivery status updates, and geolocation information.

Real-time order management and delivery tracking are critical components of modern food ordering platforms, as consumers increasingly demand transparency and updates about their orders. Platforms like Uber Eats and DoorDash leverage geolocation to track deliveries in real-time, allowing customers to see the exact location of their food during transit. While these systems provide a valuable service to customers, they also introduce complexities in backend architecture, requiring efficient processing of real-time location data and dynamic updates. Additionally, applying machine learning to predict delivery times and optimize routes remains an area of active research and development, with promising implications for improving delivery efficiency and customer satisfaction.

Security and privacy concerns also remain paramount, particularly as payment transactions and personal data are involved. Food delivery systems commonly integrate secure payment gateways such as Stripe and PayPal, but ensuring data protection throughout the entire user journey, from ordering to delivery, remains a significant challenge. Using secure authentication systems like JWT (JSON Web Tokens) ensures that only authorized users can access personal accounts and place orders, while data encryption protocols protect sensitive information during payment transactions.

In summary, the combination of emerging technologies, particularly within the MERN stack, offers a promising foundation for building smart and scalable food ordering systems. However, ongoing research into areas such as data scalability, machine learning-based route optimization, and enhanced security measures is essential to improving the efficiency, security, and overall user experience of these platforms.

2.3 Research Gap

- Limited features for regional cuisines or local restaurants.
- Poor offline support or lack of accessibility for certain demographics.
- Despite their success, these platforms sometimes include expensive service costs for eateries and little alternatives for smaller enterprises to customize their designs.

2.4 Conclusion

The literature review highlights the rapid growth of online food ordering platforms and the role of emerging technologies in shaping their functionality. By analyzing existing solutions, it becomes evident that while current systems excel in usability and convenience, there are opportunities to improve scalability, accessibility, and sustainability.

The findings emphasize the importance of incorporating user-friendly interfaces, secure payment systems, and efficient delivery mechanisms, which are essential to meeting user expectations. Moreover, the review identifies gaps, such as limited support for small businesses and a lack of eco-friendly initiatives, which the proposed project aims to address.

Overall, the insights gained from this review provide a solid foundation for designing and implementing a food ordering system that aligns with industry standards while introducing innovative features to enhance user experience and societal impact.

CHAPTER 3

Initial Study and Feasibility Analysis

3.1 Introduction

The aim of this study is to assess the practicality of developing a smart food ordering system that combines user-friendly features with advanced functionalities like personalized recommendations, real-time order tracking, and seamless payment integration.

The idea arises from the increasing demand for efficient food delivery solutions, as current systems often face challenges such as order delays, poor personalization, and lack of accessibility for smaller businesses.

The proposed system aims to:

- Enhance the user experience with intelligent and interactive features.
- Simplify operations for restaurants and streamline delivery processes.
- Address gaps in existing platforms by offering unique functionalities.

3.2 Problem Identification

Current food ordering platforms face the following challenges:

Customer Challenges:

- Lack of tailored suggestions based on preferences or dietary restrictions.
- Poor user interface or navigation on some platforms.
- Limited payment options and occasional security concerns.

Restaurant Challenges:

- Difficulty in managing peak-hour orders.
- Limited insights into user behavior and preferences for improving menus.
- Dependence on third-party platforms that reduce profit margins.

Delivery Challenges:

- Inefficient route management leading to delays.
- Lack of real-time updates for users about their orders.

The proposed system will address these issues using:

- AI-powered recommendations based on user history.
- Efficient order management dashboards for restaurants.
- Real-time tracking and optimized delivery routes.

3.3 Scope of The Project

This smart food ordering system will focus on the following areas:

a. User Perspective

- Simple, intuitive UI for browsing menus and placing orders.
- Personalized suggestions based on preferences, location, and purchase history.
- Multiple payment options, including cards, digital wallets, and cash on delivery.
- Real-time notifications for order status and delivery tracking.

b. Restaurant Perspective

- Easy-to-use dashboard for managing orders, menus, and inventory.
- Analytics for understanding customer behavior and improving services.
- Notifications for order updates and stock alerts.

c. Delivery Perspective

- Real-time assignment of delivery personnel to orders.
- Optimized routes using map integrations to reduce delivery time.
- Delivery updates for both restaurants and customers.

d. Limitations

The initial version will focus on:

- Limited geographic coverage (a pilot region).
- Core features such as ordering, delivery, and restaurant management.
- Excluding advanced logistics (e.g., drone delivery) and offline support.

3.4 Feasibility Study

3.4.1 Technical Feasibility

The technical implementation will rely on the following technologies:

- Frontend: React.js for creating a responsive and dynamic user interface.
- Backend: Node.js and Express for handling API calls and server-side logic.
- Database: MongoDB for storing user profiles, order data, and restaurant details.
- APIs: Payment gateways like Stripe or PayPal for secure transactions. Google Maps API for delivery route optimization.
- AI/ML: Personalized recommendations based on user preferences and order history.

3.4.2 Operational Feasibility

This system will align with the needs of users, restaurants, and delivery personnel by:

- Offering an intuitive design that simplifies order placement and tracking.
- Ensuring that restaurants can manage orders efficiently with minimal training.
- Providing delivery personnel with tools for route optimization and timely updates.
- To ensure smooth adoption, the system will include:
- Onboarding tutorials for restaurants and users.
- A feedback mechanism to continuously improve user satisfaction.

3.4.3 Financial Feasibility

A cost-benefit analysis reveals:

Estimated Costs:

1. **Development Costs**:
    - Salaries for developers and designers.
    - Hosting, domain, and infrastructure costs.
2. **Operational Costs**:
    - Maintenance and updates post-launch.
    - Marketing and promotional expenses.

Expected Benefits:

- Revenue from restaurant subscriptions and delivery commissions.
- Increased user base through innovative features.
- Competitive edge in the market with advanced personalization tools.

The initial investment is projected to be recovered within 12-18 months after launch, assuming steady growth.

3.5 Assumptions and Constraints

Assumptions:

- Users will have access to stable internet connections.
- Restaurants will actively participate in the platform.
- Delivery personnel will have smartphones for route tracking.

Constraints:

- Limited funding for extensive marketing in the initial stages.
- Competition from established players like Uber Eats or Foodpanda.
- Reliance on third-party APIs for payment and tracking, which may have downtime.

3.6 Conclusion

The feasibility study demonstrates that the smart food ordering system is technically, operationally, and economically viable. With careful planning, the project can address the gaps in existing solutions and provide a scalable platform that benefits users, restaurants, and delivery personnel alike.

CHAPTER 4

System Analysis and Design

4.1 Introduction

An effective, scalable, and user-friendly application is built on the basis of system analysis and design, two essential stages in the creation of any software system. These procedures aid in defining the architecture, identifying requirements, and making sure the system satisfies user and business goals in the context of a smart food ordering system that uses the MERN stack. While the system design phase concentrates on developing a comprehensive blueprint that directs the system's development, the system analysis phase entails evaluating the current system, obtaining requirements, and finding any issues or inefficiencies. Using contemporary web technologies, namely the MERN stack (MongoDB, Express.js, React.js, and Node.js), this chapter examines the analytical and design stages of a smart food ordering system.

4.2 System Analysis

The act of comprehending and outlining a software application&#39’s requirements is known as system analysis. The objective is to determine the functional and non-functional requirements for a smart food ordering system by analyzing the demands of different stakeholders, including patrons, restaurant owners, and administrators. Information collection from stakeholders is the initial stage of system analysis. Usually, surveys, interviews, and an analysis of current systems are used to do this. Users may need to be able to browse menus, add products to their carts, track orders in real time, and make secure payments while using a meal ordering platform. In order to see orders, track delivery, and manage menus, restaurant owners and administrators require certain administrative functions.

4.2.1 Functional Requirements

A smart food ordering system must include the following functionalities:

- **User Authentication and Registration:** Users may create and maintain their accounts through secure login and registration.
- **Menu management:** Present a list of dishes together with costs, photos, and descriptions.
- **Order management:** Give users the ability to place orders, change amounts, and monitor the status of their orders in real time.
- **Payment Gateway Integration:** Use third-party services like PayPal or Stripe to enable safe payments.
- **Delivery tracking:** Display the predicted time of arrival and the current delivery status.

4.2.2 Non-Functional Requirements

Among the non-functional criteria might be:

- **Performance:** A lot of people should be able to use the system at once.
- **Scalability:** As demand rises, the system should be able to handle additional users and transactions by scaling horizontally.
- **Security:** By using encryption and secure protocols, the system should safeguard user data and payment information.
- **Usability:** Both administrators and users should find the interface simple to use and intuitive. Finding possible problems and bottlenecks in current systems is the next stage of system analysis once requirements have been established. For example, real-time updates and effective database management are issues for many traditional restaurant ordering services. The MERN stack, which provides speed, scalability, and flexibility for creating contemporary web applications, can help overcome these difficulties.

4.3 System Design

Following the system analysis phase, system design focuses on describing the architecture, parts, and interactions that will allow the system to meet the previously determined criteria. The system's high- level structure is conceived and documented at this phase. Both high-level design, also referred to as architectural design, and detailed design, which includes low-level component details, are commonly included in system design.

4.3.1 Class Diagram:

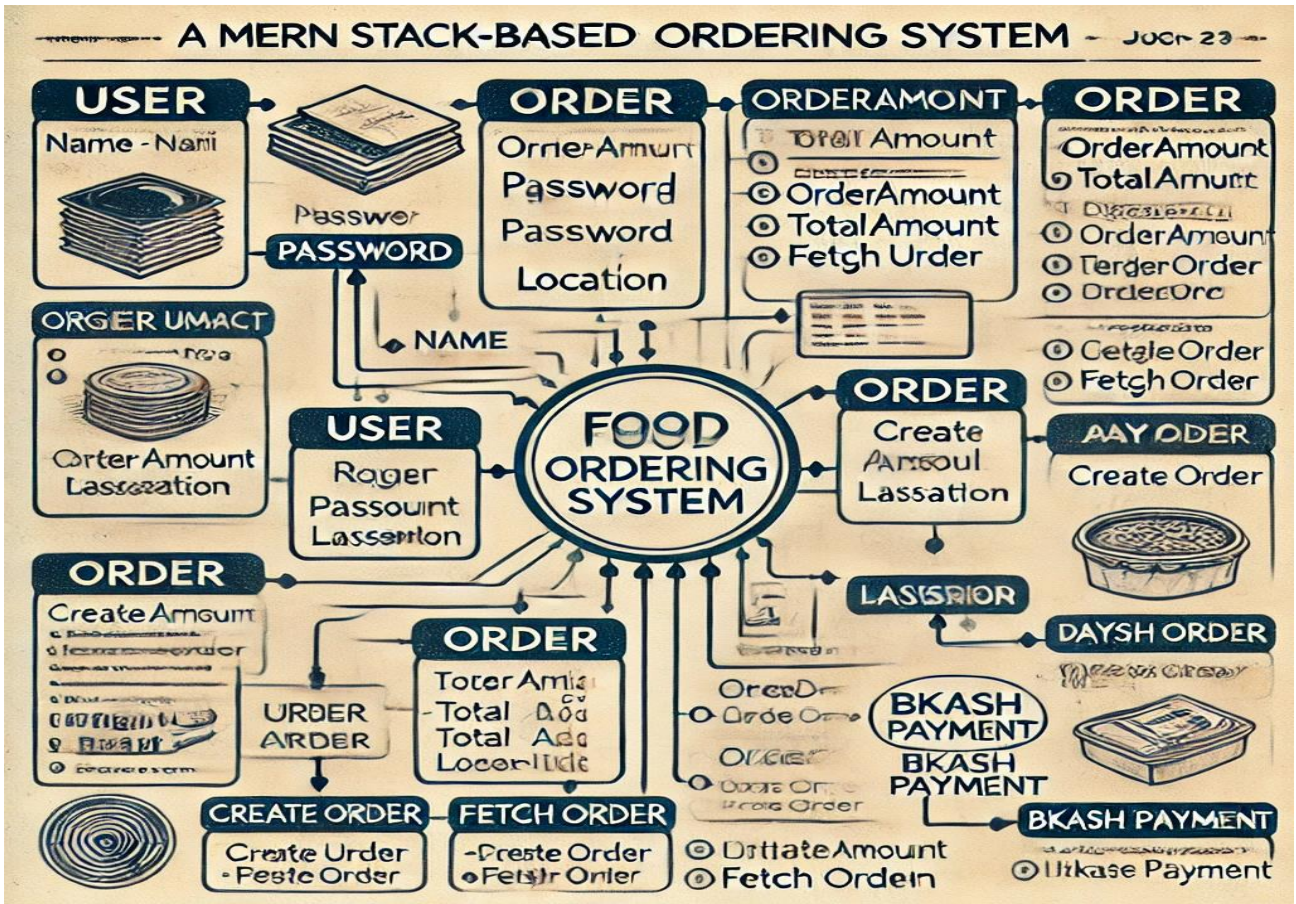


Figure 1: Class Diagram

4.3.2 UML Design:

Figure 2: UML Diagram

4.3.3 High-Level Design (Architecture): Three primary layers make up the system architecture for a smart food ordering system that uses the MERN stack:

The user interface that communicates with administrators and customers is part of the frontend layer (React.js). React.js will be used in its construction to offer a dynamic, component-based user interface. The menu, shopping cart, order status, and payment gateway will all be shown by this layer. Layer of Backend (Node.js & Express.js): Business logic, user authentication, order administration, and payment processing are all handled by the backend. To control communication between the frontend and the database, RESTful APIs will be created using Express.js running on Node.js.

4.3.4 Database Layer (MongoDB): Orders, transaction data, food products, and user information will all be stored in MongoDB. The NoSQL structure of MongoDB is perfect for managing dynamic data that may change over time, such as menus and orders.

Third-party connectors like Cloudinary for image storage and Stripe for payment processing will also be incorporated into the system. These outside firms will guarantee safe payment processing and effective media administration.

Database Design: Collections such as Users, FoodItems, Orders, and Payments will make up the MongoDB database. Relevant information including user profiles, food item details, order histories, and payment records will be stored in each collection. References and embedded documents will be used to describe relationships between collections in order to minimize redundancy and guarantee data integrity.

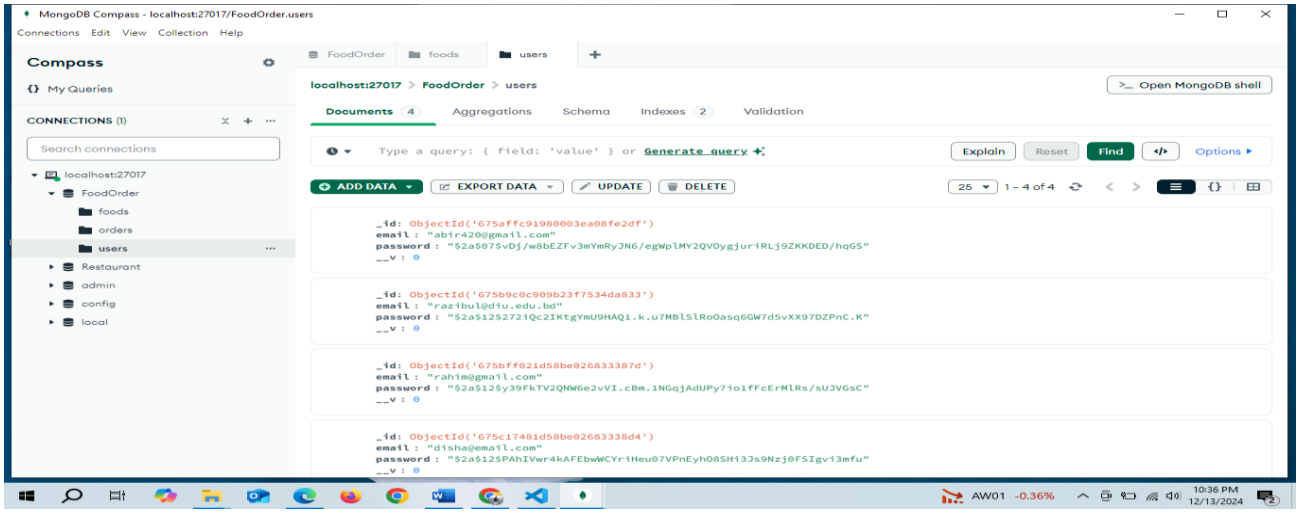


Figure 3: MongoDB Design

API Endpoints: The backend will make available RESTful APIs for order creation (/order/create), menu management (/food), user authentication (/auth/register, /auth/login), and payment processing (/stripe/checkout). Every API endpoint will respond appropriately to certain queries.

4.3.4 User Interface Design: Users will be able to explore the menu, add goods to their basket, track their purchases, and make payments using the frontend's simple, user-friendly interface. To guarantee usability on PCs, tablets, and smartphones, the design will be responsive.

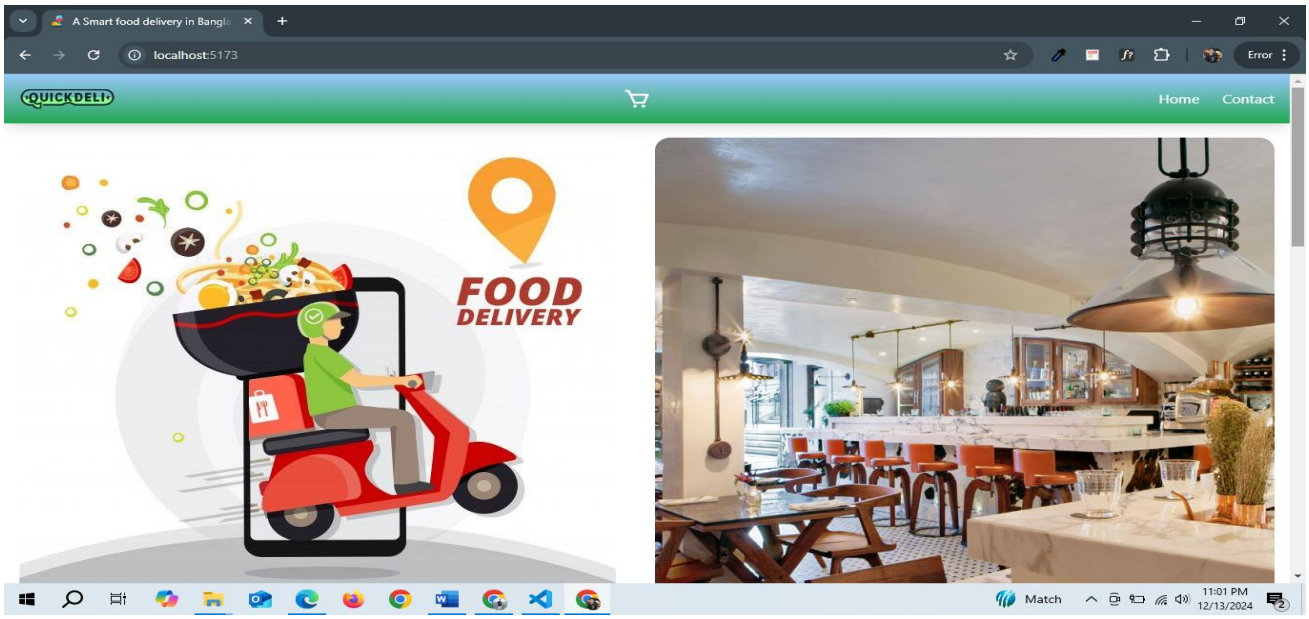


Figure 4: Home Page Design

4.4 Conclusion

The creation of a successful smart food ordering system is based on system analysis and design. Developers may produce a platform that is not only effective but also safe and easy to use by comprehending the demands of administrators and users, determining functional and non-functional requirements, and resolving any potential system issues.

In order to achieve these objectives, the design phase makes sure that the system is constructed with the appropriate architecture, parts, and technology. With its adaptability and scalability, the MERN stack provides the perfect answer for a variety of applications, such as secure payment processing, real-time order tracking, and user authentication.

To sum up, a thorough system analysis and design approach guarantees that the Smart Food Ordering System will be reliable, safe, and able to manage obstacles in the actual world.

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 Introduction

Users can order meals from any restaurant and have it delivered straight to their homes via a meal delivery application, which is an internet platform that connects consumers and companies. Three main components will make up the program: an API server, a database, and a web-based software. The framework will be built using the MERN (MongoDB, Express, React, Node.js) stack, which provides a robust and flexible environment for creating scalable online applications. MongoDB will be the database used to store consumer and restaurant information. Express will be used in the construction of the backend API server, which controls queries and responses between the database and front-end. React will be used to create a responsive and user-friendly frontend that will allow customers to browse and order food from the restaurant. Node.js will power the backend server.

5.2 Development

The application will have the following features:

1. **User Authentication -** Customers will be required to register and log in to access the application.
2. **Menu Management -** Customers will be able to browse through the menu.
3. **Cart and Checkout -** Customers will be able to add items to the cart and checkout using different payment methods.
4. **Order Management -** Restaurants will be able to view and manage orders and their status.

5.2.1 Frontend

The Frontend is the part of the Smart Food Ordering System that users interact with. The development of the frontend focuses on creating an intuitive, responsive, and aesthetically pleasing user interface that offers a seamless experience across various devices. For this system, React.js is used to build a dynamic, component-based frontend.

Key components of the frontend include:

User Interface (UI):

- **Menu Display:** A clean layout for displaying available food items, categorized by type (e.g., appetizers, main course, drinks). Each item will have an image, description, and price.
  Shopping Cart: A dynamic shopping cart where users can add, update, or remove items. It shows the total cost, the list of selected items, and their quantities.
  Order Tracking: A real-time order tracking page where users can view the status of their order, from preparation to delivery.
- **Payment Gateway:** Integration with a secure payment system (e.g., Stripe) for handling transactions.

- **Authentication Pages:** Login and registration pages that allow users to create accounts and log in securely.

- **Responsive Design:**
  The frontend is developed to be fully responsive, ensuring that the system works seamlessly on both desktop and mobile devices. This is achieved through the use of CSS frameworks like Tailwind CSS for styling and Media Queries for handling different screen sizes.

- **State Management:**
  React Context and useState hooks are used to manage application states like cart contents, user data, and order status. This ensures that data is consistent and updated across various components of the application.

5.2.2 Backend

The Backend of the Smart Food Ordering System is responsible for handling business logic, user management, order processing, and database interaction. The backend is built using Node.js with Express.js to create RESTful APIs that communicate with the frontend and the database. The system also integrates with external services such as Stripe for payments.

Key components of the backend include:

API Development:

- **User Authentication:** The backend includes endpoints for user registration and login, using JWT (JSON Web Tokens) for secure session management.

- **Menu and Order Management:** APIs are provided for managing food menus, placing orders, viewing past orders, and managing the order status.

- **Payment Integration:** Stripe's API is used to handle secure payments. The backend processes payment details and updates the order status once payment is completed.

- **Database Integration:**
  The system uses MongoDB as a NoSQL database to store data. Collections include:
- **Users:** Stores user profiles, including login credentials and order history.
- **Food Items:** Stores information about each food item, including name, price, image URL, and category.
- **Orders:** Stores customer orders, including food items, quantities, total price, and delivery details.
- **Payments:** Stores payment records associated with orders.

Server-Side Logic:

Order Processing: After a user places an order, the backend processes the order, calculates the total cost, and initiates the payment process.

Real-Time Order Tracking: The backend handles updates to the order status, allowing customers to track their orders in real time.

Security: Security measures like password hashing with bcrypt.js, input validation, and authorization using JWT are implemented to protect user data and ensure secure transactions.

5.3 Algorithm Design

The Algorithm Design section outlines the critical algorithms that power the functionality of the Smart Food Ordering System. These include the logic for order placement, payment processing, and real-time order tracking.

Order Management Algorithm:

The system must handle multiple orders concurrently, ensuring that each order is processed accurately. The algorithm processes user inputs (food items, quantities, and payment), updates the order status, and triggers appropriate notifications (e.g., "order received," "order shipped").

Payment Processing Algorithm:

The payment algorithm handles interactions with Stripe's API to securely process payments. This includes validating payment information, calculating the total cost (including taxes and fees), and confirming the transaction. Upon successful payment, the order status is updated to "paid.

Order Tracking Algorithm:

The algorithm responsible for real-time order tracking calculates the estimated delivery time, monitors the status of the order (whether it's being prepared, in transit, or delivered), and updates the customer's interface.

Recommendation Algorithm:

A basic recommendation algorithm may be included to suggest meals based on previous user orders or similar users' preferences. This could involve collaborative filtering or content-based filtering techniques.

5.4 Model Development

5.4.1 MongoDB

MongoDB is a popular NoSQL database management system used in many modern web applications, including those that provide food service. Among MongoDB's primary advantages include its document-based data storage capabilities, which make it the ideal choice for applications that need dynamic schema structures.

A meal delivery application may use MongoDB to store a range of data types, such as user profiles, orders, menus, and reviews. The representation of each data item is a document, which is a JSON-like data structure that accommodates numerous fields with different kinds and values. Because of this, developers may store and retrieve complex data objects without worrying about complex SQL queries or strict schema structures.

Figure 5: MongoDB

Furthermore, MongoDB offers robust indexing and querying capabilities that enable developers to create complex searches with a straightforward syntax and retrieve data with ease. For example, a meal delivery service may utilize MongoDB to find all eateries offering a certain cuisine or to obtain each and every order that a single user has made. As data volume increases, developers can effortlessly scale their applications thanks to MongoDB's flexibility, querying power, and automatic shading and replication features. MongoDB has several security features, including authentication, access control, and encryption, to assist safeguard the data kept in the database.

Overall, MongoDB is a robust and flexible database management system that works well with cutting-edge web apps like those that serve meals. Developers that must produce dependable and scalable apps frequently use it due to its ability to its robust indexing and searching features allow it to store and retrieve complicated data items in a scalable and adaptable way.

5.4.2 React.js

React.js is a popular JavaScript library for designing user interfaces. Facebook established it, and a large developer community is currently in charge of keeping it up to date. Respond was employed to create the application's front end, or the part of the software that users interact with, in the case of the food delivery application.

React provides a set of elements and functionalities that facilitate the quick and easy creation of complex user experiences. Utilizing a component-based design, it suggests that the user interface is made up of basic, interchangeable components that may be combined to create more complex interfaces.

Figure 6: React.js

To control the UI's state, React additionally makes use of a virtual DOM (Document Object Model). Because of this, React can refresh the UI quickly when the state changes without needing to reload the website.

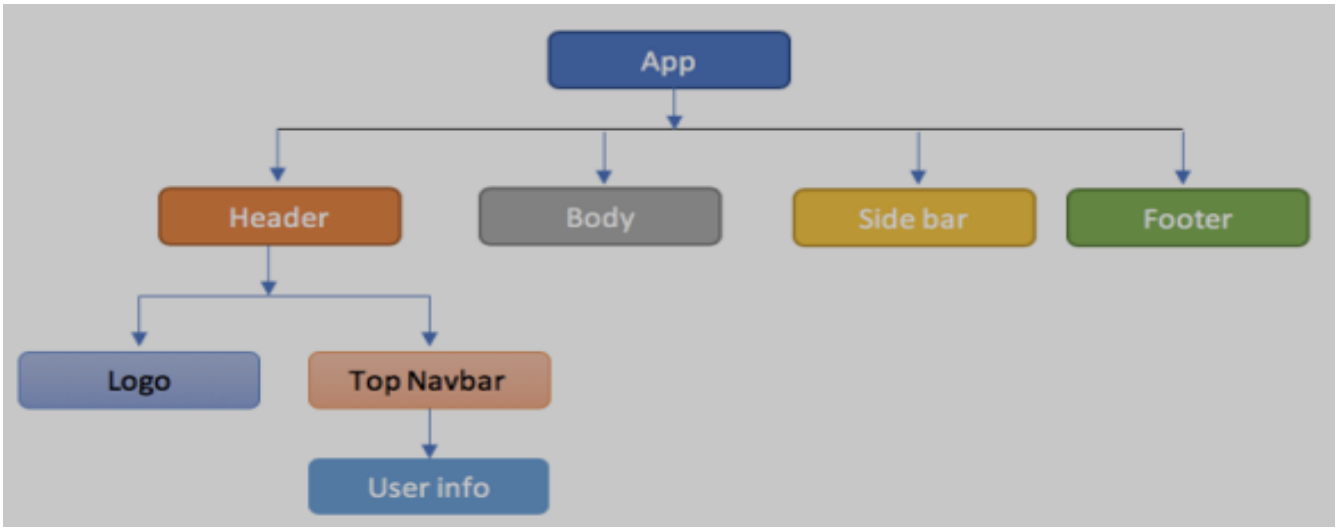


Figure 7: React Component-based Development

React is one of the best technologies to utilize in a food delivery business because of its smooth user experience. Building and reusing UI elements is made easy by React.js component-based architecture, and its virtual DOM allows for rapid and efficient UI changes. This could result in an application that provides a better user experience and is faster and more responsive.

A seamless user experience in a food delivery service that uses the MERN stack depends on a number of factors. The main page, menu, checkout, navigation, pizza, and sign-up are some of these elements.

- When the user launches the application, the home screen appears as their initial screen. A description of any highlighted products and ongoing deals, along with a rundown of the app's features, may be included.

- A summary of the user's order is displayed, and the checkout component controls the payment process. It might include auxiliary components such as a confirmation page, a form for selecting an address, and a form for entering payment information.

- The menu component displays the options for the application. Sorting features, filter options, and search bars can all be included in this element.

- The navigation function, providing links to other websites and facilitating easy. An important part of the app's user experience is navigation. In addition to a logo, there might be links to the menu, checkout, and home page. A pizza HTML div with the size, quantity, and price of the pizza, along with options to add it to the basket, is rendered by the reusable pizza component. It allows for easy ordering and pizza personalization and may be applied to the menu and checkout processes.

- Additionally, users can create an account and log in using the registration tool. Additional supporting components could include a confirmation screen, a form for entering personal information, and a form for inputting payment details.

React hooks such as useState, useEffect, and useContext allow developers to construct functional and class components in React. For example, the quantity and dimensions of the pizza can be controlled using the useState hook and a functional component. The component can use provided props, like as the pizza's name, price, and toppings, to render the pizza div with the relevant data. It is possible to implement the add-to-cart feature by using a useContext hook to update the user's cart and access the app's global state. When constructing the signup component, which can also be built using a class component, form validation and submission may be managed using the componentDidMount and componentDidUpdate lifecycle methods.

It may use a library such as Axios to store user data in a database and send HTTP requests to the app's backend. To sum up, using the MERN stack to create food delivery service components requires a combination of technical expertise, design skill, and user experience knowledge. These features—checkout, home, menu, navigation, pizza, and registration—are essential to a perfect and enjoyable user.

The functional and class components of React, along with hooks and frameworks like Axios, allow developers to construct these components fast and efficiently.

5.4.2.1 React Hooks

The React Hooks are a crucial component of contemporary React development, allowing programmers to control state and lifecycle events in functional components.

We will go over how to use React hooks in a food delivery app developed with the MERN stack in this section:

- **The useState Hook:** Functional components can incorporate the state by using the useState hook. A function to update the state value and an array with the current state value are returned after it receives an argument representing the original state value. The food delivery app's useState hook allows it to manage the user's cart, menu items, and personal data.

Example:

```
Import React, {useState} from “react”;

Function Home(){
   Const [count, setCount] = useState(0);

    Return (

      <div>
        <p> you clicked{count} times<p>
        <button onClick={() =>setCount(count+1)}>
           Click me
        </button>
```

```
        <div/>

    );
}
```

In this instance, a functional element called Home is being defined. In order to change the state variable count within the component, we're constructing a method called setCount and using the useState hook. Utilizing the provided input (in this case, 0). The count is initialized to 0 in order to useState. Subsequently, we return some JSX that shows the current count along with a button that, when pressed, updates the count by running setCount 20 with a new value (in this case, one). Invoking setCount will cause React to re-render the component with the updated state. We could therefore design dynamic user interfaces that adapt to user input.

- **useLocation Hook:** This hook returns the current location object comprising information about the current URL. The current route, query arguments, and other information can be extracted from the URL using this method. The one that The food delivery app may utilize a useLocation hook to show the pertinent elements according to the current path.

  Example:

```
import { useLocation } from 'react-router-dom';
function MyLocation() {

        const location = useLocation();
        return (
        <div>
                <h2>Current Path: {location.pathname}</h2>
        </div>
        );
}
```

In the example above, the react-router-dom package imports the useLocation hook. It is then used to determine the current position of the component. The current path, which is then displayed, is obtained using the location path name in the part. A change in location will cause the component to re-render with the updated path.

- **useEffect Hook:** This hook is used in functional components to control side effects like data retrieval and DOM modification. After every render, it runs a callback function that is sent in as an input. Additionally, we might offer dependencies, so that the effect only manifests when those dependencies are altered. The food delivery app's useEffect hook allows us to retrieve menu items from the server and update the DOM as a user adds items to their basket.

Example:

```
import React, { useState, useEffect } from 'react';
const ExampleUseEffect = () => {
    const [count, setCount] = useState(0);
    useEffect(() => {
      document.title = `Count is: ${count}`;
   }, [count]);
  const handleClick = () => {
     setCount(count + 1);
  };
  return (
        <div>
              <h1>Count: {count}</h1>
              <button onClick={handleClick}>Click me</button>
        </div>
  );
};
export default ExampleUseEffect;
```

- **useNavigate Hook:** The useNavigate hook allows us to programmatically explore different app pages. It accepts no parameters and generates a navigate function that we may call with the desired path. Following the customer's addition of items to their The useNavigate hook can be used to navigate to the checkout page in the food delivery app's basket.

  Example:

```
import { useNavigate } from 'react-router-dom';
function MyComponent() {
        const navigate = useNavigate();
        function handleClick() {
                navigate('/some-route');
        }
        return (
                <div>
                        <button onClick={handleClick}>Go to some route</button>
                </div>
        );
}
```

  This example uses react-router-dom to import the useNavigate hook. It is then utilized to provide a navigation feature that allows users to explore various application pathways. Pressing the button causes the navigate method to be executed inside the function of MyComponent. The '/some-route' option is passed to the navigate function, indicating the course to follow. Upon pressing the button, the handleClick function is triggered, which subsequently calls the traverse method using the selected path. The user is then directed to the route that was chosen by the program. The useNavigate hook, in general, simplifies route switching in React applications.

In conclusion, the powerful JavaScript package React.js was used to develop the food delivery service's front end. The ability to provide a smooth user experience is a critical benefit for any program that needs dynamic UI changes. It's Virtual DOM and component-based architecture make it easy to quickly and efficiently create complex user interfaces.

5.4.3 Express.js

A robust Node.js web application framework called Express.js is used to create the backbone of online applications such as food delivery services. It is a well-liked option for API design because of its ease of use and flexibility. With the help of the straightforward Express.js API, developers can provide middleware, process HTTP requests and answers, and create routes.

Express.js is used in the food delivery industry to offer a RESTful API that connects to the React.js front-end application. The API interacts with the database, responds to client-side queries, and provides data to the front end. It makes the program easily maintainable, scalable, and efficient.

Figure 8: Express Js

Middleware plays a major role in Express.js. The functions that run in the middle of an application's request-response cycle are known as middleware functions. They have several uses, including error management, authentication, and logging. Throughout the Applications for food delivery employ middleware to handle issues, validate input data, and manage authentication.

Its capability to carry out HTTP requests is another crucial feature of Express.js. With its support for all HTTP methods—GET, POST, PUT, and DELETE—it offers a wide range of capabilities. The food

delivery application uses GET queries to pull information from the database, POST requests to create new information, PUT requests to update information, and destruct requests to remove information.

All things considered, Express.js is a powerful and adaptable framework for web applications that works well for creating the backend of websites such as food delivery services. Its flexibility, scalability, and ease of use make it a popular choice among developers.

5.4.4 Node.js

Node.js is a well-known server-side JavaScript runtime environment for creating quick and scalable online apps. The V8 JavaScript engine serves as the foundation for this open-source platform within Google Chrome. Node.js is a popular option for developing web apps because of its lightweight architecture and fast speed.

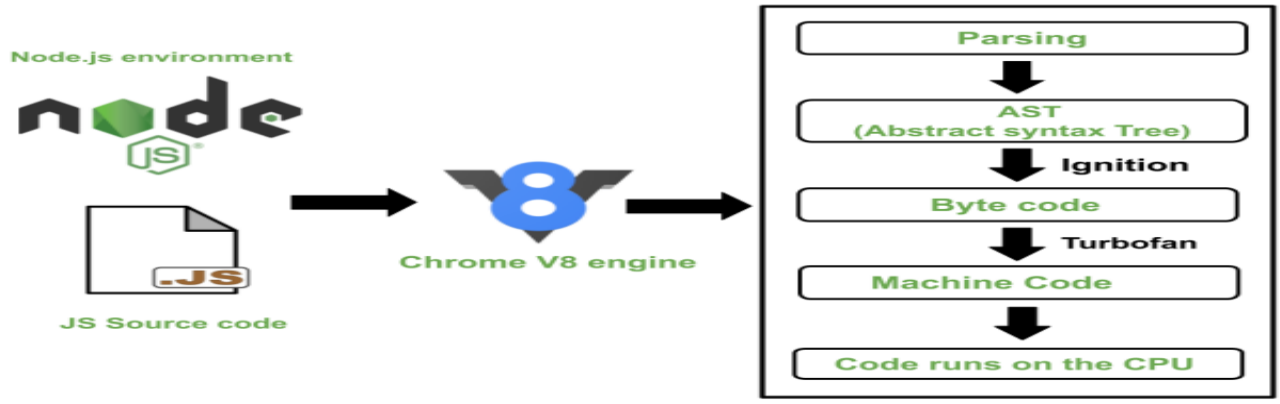


Figure 9: Node.js and Google's V8 Engine

Node.js is used as the server-side runtime environment by the food delivery application to manage the whole service workflow, accept and reply to HTTP requests, and communicate with the database. Node.js is the ideal option for real-time applications requiring high concurrency and scalability.

Express.js, one of the numerous modules and frameworks in the Node.js ecosystem, is used by the meal delivery service. An easy-to-use and adaptable API for developing online applications and APIs is offered

by the Express.js web framework for Node.js. Routing, middleware, and template engines are some of its features that facilitate the development of scalable and modular systems.

The capacity of Node.js to manage several concurrent connections is one of its primary benefits for the food delivery industry. Node.js can manage multiple requests at once without overtaxing the main event loop because it uses event-driven, non-blocking I/O. Because of this, it is appropriate for real-time applications that require low latency and high concurrency.

To sum up, Node.js is a popular framework for building scalable and effective web applications. Its speed, lightweight design, and extensive ecosystem of libraries and frameworks make it an excellent option for food delivery systems. Node.js is appropriate for real-time applications because of its event-driven, non-blocking I/O design, which allows it to accommodate many concurrent connections.

5.4.5 Hyper Text Transfer Protocol

Web servers connect with clients via an application protocol called HTTP, or Hypertext Transfer Protocol. The foundation of the World Wide Web's data transfer is this. Web servers and browsers must respond to different requests in accordance with HTTP, which also governs the format and transport of messages.

A client request and a server response usually make up an HTTP request. The client sends an HTTP request to the server, which replies with an HTTP response. Request and answer messages consist of both a message body and headers. Among the headers are information about the request or response, such as the type and quantity of the content being sent. The message body contains the actual information being conveyed.

The server does not retain a record of the client's previous queries due to HTTP's stateless client-server architecture. Each request is processed independently, and the server responds to each one using only the information provided in that specific request.

HTTP comes in a number of variants, the most widely used being HTTP/1.1 and HTTP/2. HTTP/1.1 is a commonly used web communication protocol. It is a text-based protocol that uses a request/response architecture. HTTP/2, the most recent version of HTTP, is designed to improve online apps' functionality.

It allows for the simultaneous transmission and receiving of many requests since it supports multiplexed streams and uses binary communication instead of text-based communication.

The development of complex web applications has been made easier by the extensive use of HTTP, a vital component of the modern internet. Among its various applications are web browsing, social media, video streaming, and online shopping. It is expected that HTTP will remain an essential component of data transfer and online communication as the web evolves.

5.4.5.1 HTTP & HTTPS

Data is moved over the internet using two protocols: HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure). The primary difference is the level of security each provides amongst themselves.

HTTP is a popular application protocol for data transmission over the internet. On top of the TCP/IP protocol, it specifies the message format and delivery. Because HTTP is a dangerous protocol, hackers can intercept data sent over it because it isn't encrypted.

HTTPS, on the other hand, is a more secure version of HTTP. SSL/TLS (Secure Sockets Layer/Transport Layer Security) encrypts data sent via HTTPS between a web server and a client. The SSL/TLS protocol is used to encrypt data while it is in transit making it harder for hackers to intercept and steal private information.

The use of encryption in HTTPS enables the secure and private transfer of data between the client and the server. When communicating sensitive information, such as credit card numbers, login credentials, or other private data, this is essential.

Another difference between HTTP and HTTPS is the port that they use. HTTP utilizes port 80, whereas HTTPS uses port 443. Your browser will automatically add the appropriate protocol and port number when you enter a website's URL, depending on whether it uses HTTP or HTTPS.

In addition to its security benefits, HTTPS can improve a website's search engine rankings. Google claims that HTTPS is a ranking factor, and websites that use it may experience a slight increase in their search engine rankings.

To sum up, HTTPS is a secure variant of HTTP that protects data sent between the client and the server using encryption. HTTPS is becoming more and more common as websites look to improve security and protect personal information, even if HTTP is still widely used.

5.4.5.2 HTTP Body & Header

A web server and a client, like a web browser, exchange information according to the HTTP (Hypertext Transfer Protocol) protocol. The HTTP protocol is used to send and receive data whenever a client requests a resource from a server.

The HTTP protocol's two essential parts are the header and the body. Metadata describing the request or answer is included in the header, whereas the body contains the actual data being transmitted.

- **HTTP Header:** The HTTP header, the first part of an HTTP request or response, contains a number of fields that describe the message being sent. Headers can be divided into two categories: request headers and response headers.

Request headers, which provide information about the submitted request, are a part of an HTTP request. Common request headers include:

- ✓ **Host:** Indicates the server's domain name that is being accessed.
- ✓ **User-Agent:** Indicates which client program or browser is submitting the request.
- ✓ **Accept:** Indicates which MIME data types the client is capable of handling.
- ✓ **Authorization:** Provides the request's authentication information.

Response headers, which provide information about the delivered response, are included in an HTTP response. Typical response headers include the following:

- ✓ **Content-Type:** Indicates the response's MIME type for the material being sent.
- ✓ **Content-Length**: Shows how many bytes the response body is.
- ✓ **Cache-Control:** Indicates if and how long the response may be cached.
- ✓ **Set-cookies:** Sending cookies to the client in order to store client-side state is known as set-cookie.

HTTP Body:

The second part of an HTTP request or response, the HTTP body contains the actual data being sent. The format and structure of the body are determined by the MIME type specified in the Content-Type header.

For example, if the Content-Type header specifies that the body contains JSON data, the body will be organized as a JSON object. If the Content-Type header specifies that the body contains HTML data, the body will be formatted as an HTML page.

Typically, an HTTP request's body contains data that is sent to the server, while an HTTP response contains data that is returned to the client. When the server has processed the data, it sends a response to the client, which could be a message or a link to another website.

In summary, the HTTP protocol's header and body are both essential components. The header contains 31 pieces of metadata pertaining to the request or response, while the body contains the actual data being transmitted. For web applications to be created and maintained, these components must be used appropriately.

5.4.5.3 Status Codes for HTTP

An HTTP status code is the three-digit response that a web server provides to a client's resource request. These codes reflect whether the client's request was successful or unsuccessful, as well as the condition of the resource that was sought.

Five categories of HTTP status codes exist:

1. **Informational 1xx:** These numbers show that the request has been received by the server and is still being processed.
2. **Success 2xx:** These numbers show that the request was successfully handled by the server and that the desired data is being returned.
3. **Redirection 3xx:** These codes are used to tell the client that they need to use an alternative URL to access the resource or that the resource they requested has been relocated.
4. **Client Error 4xx:** This code denotes that the client's request had an error, such as an invalid or missing argument.
5. **Server problem 5xx:** These codes are used to show that the server side experienced a problem.

Here are a few HTTP status codes that are frequently seen:

- **200 OK:** This code signifies that the request was fulfilled and the desired data is being returned by the server.
- **301 Moved Permanently:** This signifies that the resource that was requested has been permanently relocated to a different URL.
- **404 Not Found:** This result means that the server was unable to locate the requested resource.
- **500 Internal Server issue:** This code signifies that the request could not be fulfilled due to a server-side issue.

Web developers and server administrators should be aware of HTTP status codes since they can be used to diagnose and fix problems with web apps and services.

5.4.5.4 Methods for HTTP

As seen in the Food Delivery Application, HTTP methods are a crucial part of web development. The kind of request that is sent to a server is specified by HTTP methods.

The program makes use of the HTTP methods listed below:

- **GET:** To retrieve data from a server, using the GET approach. The Food Delivery Application uses the GET method to retrieve order information, a list of restaurants, and food items.

- **POST:** To send data to the server, use the POST method. The POST technique is used in the Food Delivery Application to create new orders, add new eateries, and add new food items.
- **PUT:** Data on an existing server can be updated using the PUT method. Current orders, restaurants, and food items are updated in the Food Delivery Application using the PUT technique.
- **ERASE:** To remove data from the server, use the DELETE command. The erase option in the food delivery application is used to remove food items, restaurants, and orders.
- **PATCH:** This technique only modifies a portion of an already-existing data collection. The PATCH approach is limited to altering an order's status in the food delivery application.

By using these HTTP techniques, the Food Delivery Application complies with RESTful web service guidelines, which promote the adoption of a standardized client-server interface.

5.4.5.5 Request/Response Cycle for HTTP

The client and server communicate using HTTP methods, requests, and responses. Hypertext transfer Protocol, or HTTP, is an internet-based data transfer protocol.

The client specifies the kind of request it want to submit using HTTP methods. GET, POST, PUT, DELETE, PATCH, and OPTIONS are the most often used methods. Information is obtained from the server using the GET method, added to the server using the POST method, changed on the server using the PUT method, and deleted from the server using the DELETE method.

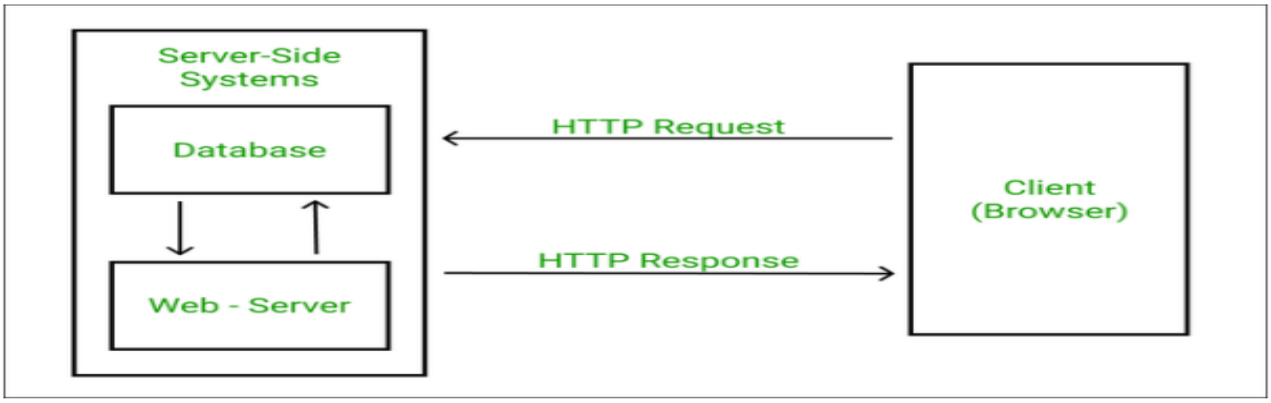


Figure 10: HTTP Request/Response Cycle

The client sends requests to the server. The URL requested, the HTTP method used, and any additional data delivered with the request are all included in the request. To add, edit, and retrieve data, requests are made to the server.

The client receives a response from the server. The status of the request, any data provided to the client, and any extra metadata pertaining to the response are all included in the response. The meal delivery app utilizes answers to give the customer information when a request is made.

The communication between the client and server is defined by REST APIs (Representational State Transfer). A RESTful API employs URLs to identify the objects being acted on and HTTP methods to specify the actions that can be taken on a resource. RESTful APIs are utilized in the food delivery application to express the actions that may be taken on food items, orders, and clients.

In conclusion, the food delivery program uses HTTP methods, requests, and responses to facilitate client-server communication. The activities that can be performed on the application's resources are specified by RESTful APIs.

5.4.6 RESTful APIs

The food delivery application uses REST APIs to make it easier for client-side and server-side components to communicate with one another. The HTTP protocol is used by a web architecture known as REST (Representational State Transfer) to provide user-accessible online services. There are RESTful APIs that are stateless and easily scalable.

Rest API Model

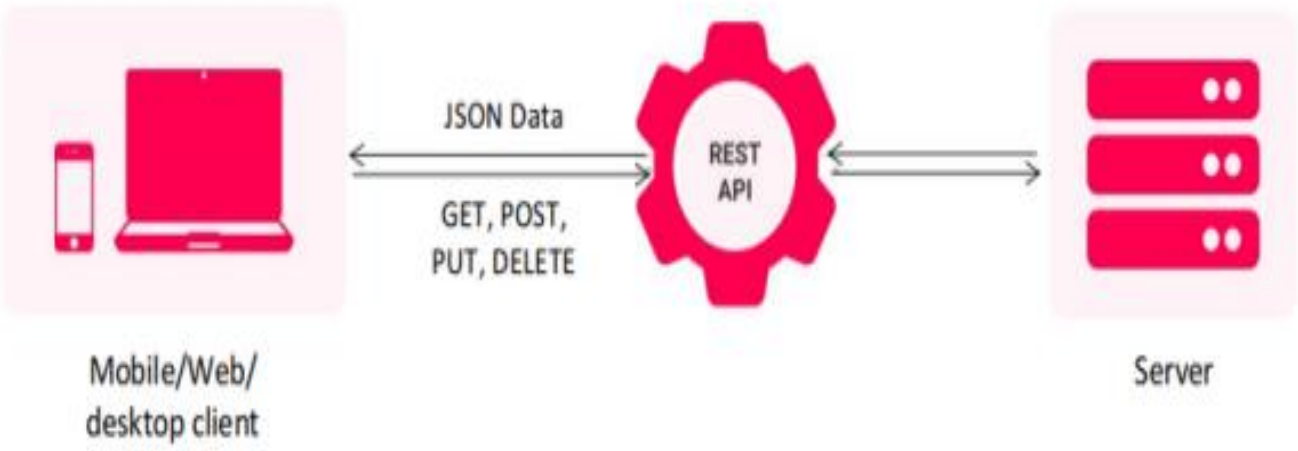


Figure 11: REST API

The RESTful APIs in the food delivery program are designed to process HTTP requests and responses reliably. After receiving a request from the client with the required parameters, the server processes it and responds to the client. Among other formats, the response could be in HTML, XML, or JSON.

The RESTful APIs in the food delivery application are designed to carry out a number of functions, such as data generation, retrieval, updating, and deletion. These APIs provide a variety of HTTP methods, including GET, POST, PUT, and DELETE.

5.4.7 Packages and Modules

5.4.7.1 Front-end Modules

- **React Router DOM :** React Router Dom v6 is a routing module for React apps that provides a declarative method for switching between different components in response to a URL. It provides an API that is easier to use and more straightforward than previous iterations.In our application for food delivery, we used React Router Dom v6 to control the client-side application's routing. We can map specific URLs to specific components using the package's Routes and BrowserRouter components. One of the main distinctions with v6 is the use of the Routes component instead of the Switch component. We can establish routes using the Routes component by utilizing the Route

component, which takes a path prop and a component prop to map the path to a certain component. This allows for greater flexibility and route nesting. Another additional new feature in version 6 is the UseNavigate hook. This hook provides a programmatic way to select a different course of action. We utilized this hook to guide users to different components based on their actions.
Furthermore, v6 adds the useSearchParameters hook, which makes it simple to retrieve and modify URL query parameters.

- **Axios:** A browser or Node.js can send HTTP requests using the popular JavaScript package Axios. An easy-to-use user interface is provided by this promise-based framework for sending and receiving asynchronous HTTP requests to servers. Axios is widely used in contemporary web applications, particularly those developed with the MERN (MongoDB, Express.js, React, and Node.js) stack. Axios provides a wide range of HTTP request methods, including GET, POST, PUT, DELETE, and more. It also makes it possible to submit requests with distinct headers, parameters, and payloads. Axios can handle a variety of response formats, including plain text, XML, and JSON. Furthermore, native functionality is provided for processing HTTP error messages and response codes. One of the main characteristics of Axios is its support for interceptors. By employing interceptors, you can modify the request or response before it is sent or received. This could be useful for altering the request payload, adding or removing headers, handling custom errors, etc. Axios also provides the option to cancel requests that are no longer needed or are taking too long to complete. Axios is designed to function flawlessly with the latest JavaScript frameworks and modules. For example, it can be used with React to change a component's state depending on the output and receive data from a server. Axios can also be used to make HTTP queries from server-side code.

5.4.7.2 Middleware Modules

- **Body-parser:** This middleware element is utilized in the Express.js and Node.js applications for food delivery. It is available through the'req.body' property and is utilized in middleware prior to your handlers to parse incoming request bodies.

The Food Delivery service uses body-parser to parse JSON payloads sent in the body of REST API requests. Because it enables the server to parse, interpret, and respond effectively to incoming JSON requests from clients, this module is essential to the functionality of the application.

Body-parser is easy to install and use using NPM. It may be added to the project with the necessary line, and after installation, it can be sent to the 'app.use()' function to be used as middleware in the Express.js application. By default, body-parser can handle URL-encoded form input, raw text, and JSON.

All things considered, the Food Delivery application's body-parser is a crucial part since it enables the server to interpret incoming requests and provide the information required to respond correctly.

- **CORS:** In web development, Cross-Origin Resource Sharing (CORS) is an essential security feature that allows web browsers to access resources from several domains.Because of browser security restrictions, web applications hosted on one domain are frequently not allowed to request resources from another domain. With the help of CORS, servers can add additional HTTP headers that inform web browsers that sending requests to specific domains is acceptable, allowing web apps to circumvent this restriction.

  To allow cross-origin requests in our food delivery business, we used the CORS middleware package in our Node.js/Express.js server. As a result, our backend server and React.js frontend could securely communicate via HTTP.

  By using CORS, we were able to develop a more dependable and secure application that could handle requests from multiple domains and enhance our clients' user experience.

5.4.7.3 Other Frameworks and Extensions

- **JavaScript XML:** JavaScript XML, or JSX for short, is a syntactic extension for JavaScript that enables programmers to create HTML-like language inside JavaScript files. Usually, this feature is linked to the React library, which uses JSX to design and manage user interfaces.

The use of JSX in web development has become more widespread in recent years due to its adaptability and ease of use. By allowing developers to design 40 reusable components that can be used throughout an application, it reduces the amount of code required and facilitates maintenance.

JSX's capacity to streamline the development and presentation of HTML components inside web applications is among its most noteworthy benefits. JSX may be used by developers to create bespoke components that encapsulate HTML elements and their related JavaScript functionality, making code reuse easier across different parts of the application.

Another benefit of adopting JSX is that it enables developers to fully harness the power of JavaScript when interacting with HTML components. This means that developers may utilise JavaScript logic to dynamically adjust the content and properties of HTML elements, enabling them to create very dynamic and interactive user experiences.

In addition to these advantages, JSX offers a few capabilities that facilitate code writing and maintenance. For instance, JSX gives developers the ability to use inline styling, which lets them apply CSS styles directly within JavaScript code. This can facilitate style control across an application and help to minimize the amount of code needed.

All things considered, JSX is a vital tool for web developers, particularly those who deal with the React framework. It is a strong and versatile tool for building contemporary, dynamic web interfaces because of its capacity to streamline the generation and presentation of HTML components, as well as its support for JavaScript logic and inline style.

5.5 Project Management

The project flow will start with the Home component, which will be shown to the user when they first launch the software. A list of the items that are available can then be viewed by the user by navigating the Menu component. The Pizza component will be used to render each pizza item on the menu.

The user can proceed to the Checkout section, which displays the order summary and total cost, after choosing the items to be ordered. After then, the user has the option to change or cancel the order before paying.

The user can create an account by going to the Signup section if they don't already have one. The user can use it to log in and examine their order history or place new orders after creating an account. The flowchart will include hooks like useState, useLocation, useEffect, and useNavigate to handle routing, manage the state of the application, and initiate changes in response to user actions. All things considered, the MERN stack-based food delivery service's flowchart will be clear and easy to use, enabling users to navigate between the app's many sections and place orders with minimal effort

5.6 Conclusion

The System Development phase for the Smart Food Ordering System successfully brings together frontend, backend, algorithm design, and project management to create a robust, scalable, and user-friendly application. By using the MERN stack and modern web development practices, the system offers an intuitive interface, efficient order management, and secure payment processing.

Through careful project management, iterative development, and testing, the system is developed to meet user needs and provide a seamless food ordering experience. The algorithms designed for order management, payment processing, and real-time tracking ensure smooth operation, while ongoing optimization and security measures guarantee that the system remains efficient and secure as it grows.
Ultimately, the Smart Food Ordering System represents a significant step forward in the digital food service industry, providing a comprehensive and reliable solution for both customers and restaurant owner.

CHAPTER 6

Implementation and Testing

6.1 Introduction

The process of converting the design and specifications into a practical smart food ordering system and making sure it satisfies the system&#39’s desired functionality, performance, and security criteria is described in the Implementation and Testing chapter. This chapter describes the methodical process for putting the system into practice using the MERN stack, which is followed by extensive testing to confirm the system&#39’s accuracy and efficacy. Good testing guarantees that both administrators and users will have a flawless and error-free experience with the platform. In order to make sure the system functions properly in real-world scenarios, the chapter also discusses optimization strategies, problem fixes, and deployment testing.

6.2 Implementation

Implementation is the phase where the system is actually built according to the designs and specifications determined during the system design phase. In the case of the Smart Food Ordering System, the implementation involves setting up the backend with Node.js and Express.js, developing the frontend using React.js, and setting up the database with MongoDB. Here’s an overview of the implementation steps:

6.2.1 Backend Implementation:

- **API Development:** Express.js is used to develop the backend API endpoints, which handle requests for payment integration (/stripe/create-checkout-session), food item management (/food), order management (/order), and user authentication (/auth/register, /auth/login).

- **Authentication:** Secure user authentication and authorization are accomplished through the usage of JWT (JSONWeb Token).
- **Integration of Databases:** To store information about users, food products, orders, and payments, MongoDB is incorporated. These collections are all made to deal with particular types of data.

- **Payment Gateway:** To manage safe order transactions, Stripe is implemented.

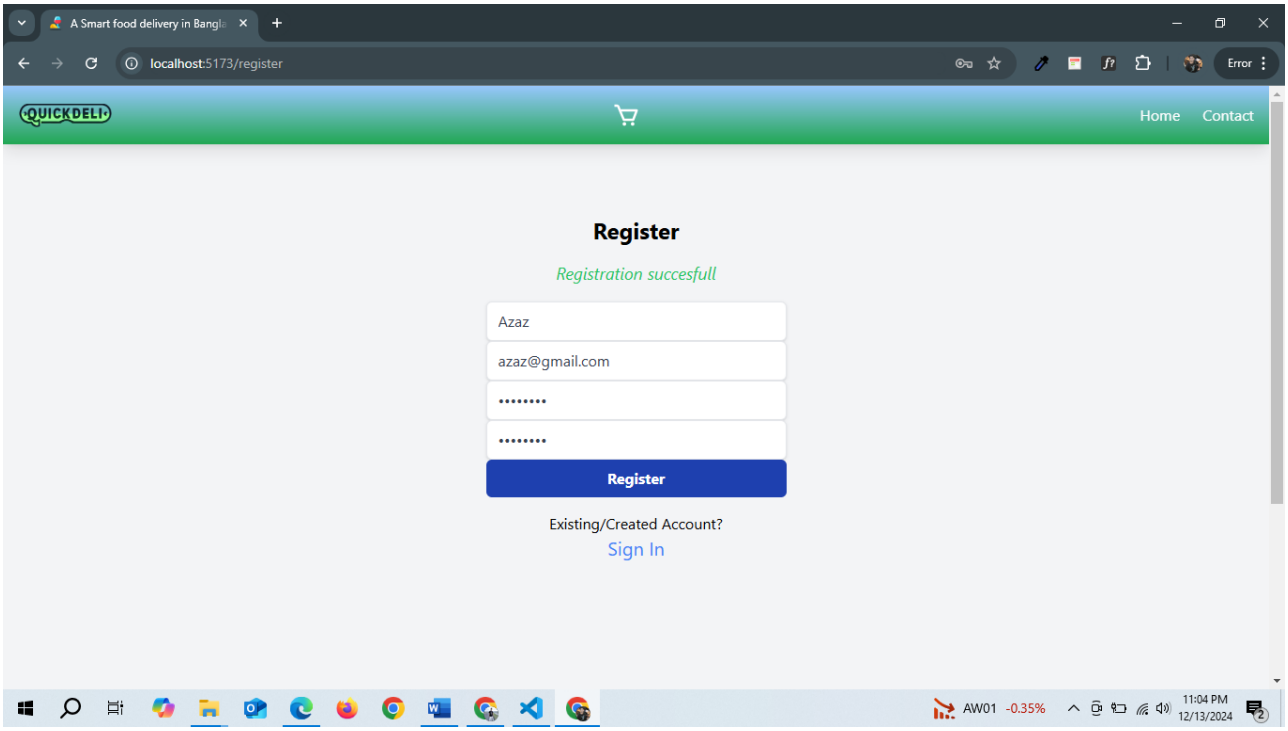


Figure 12: Registration Form

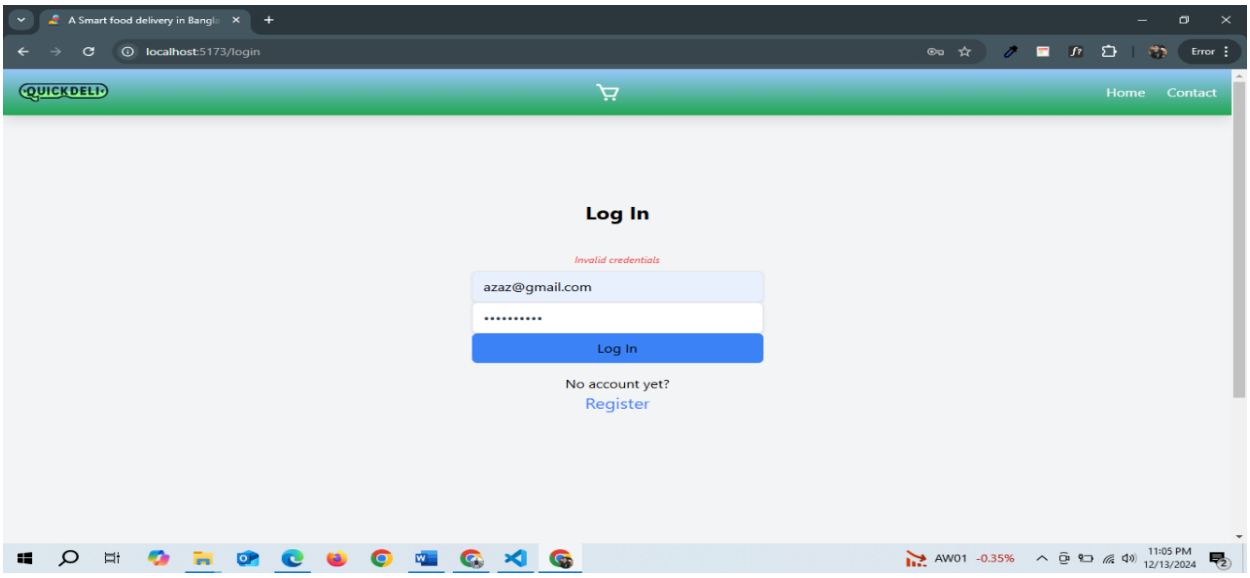


Figure 13: Login Form

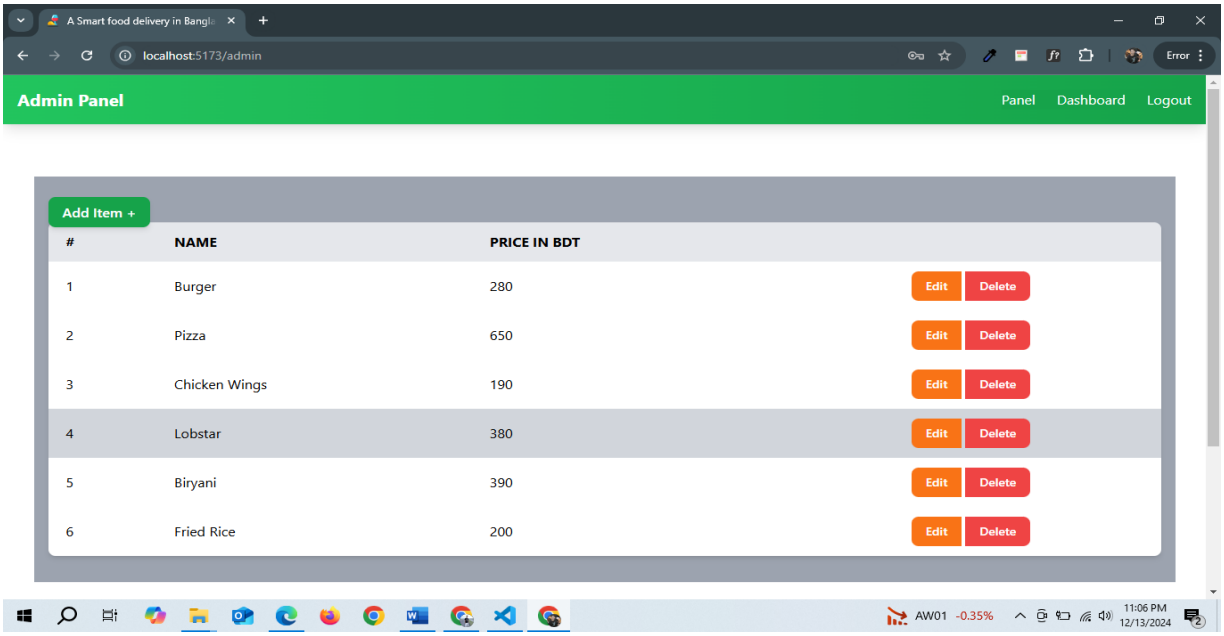


Figure 14: Admin Panel

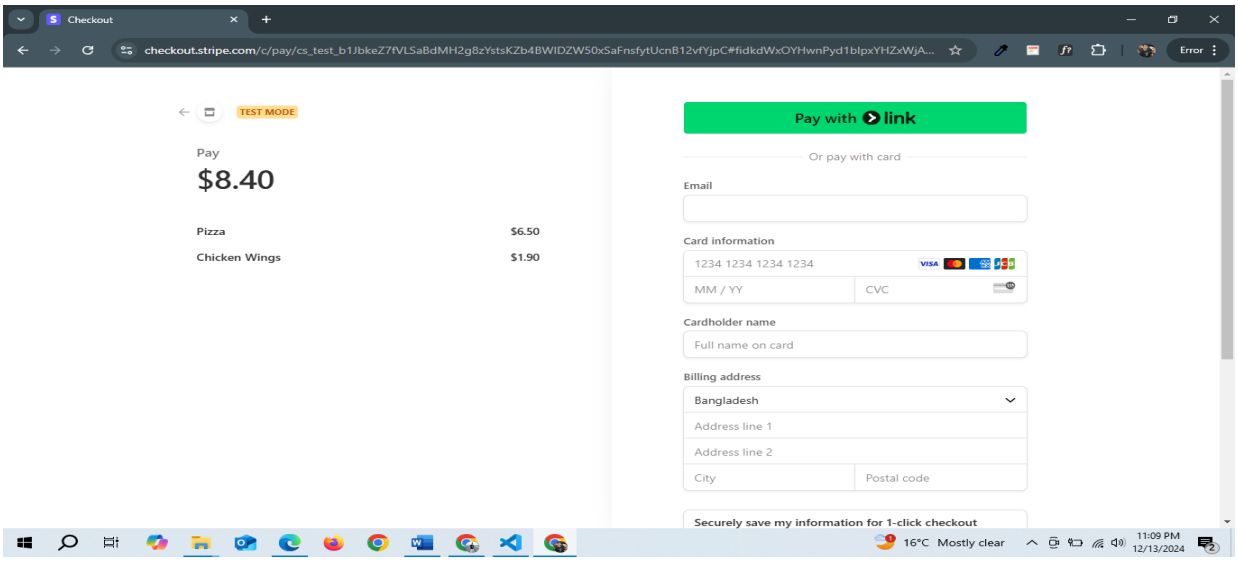


Figure 15: Payment Gateway

6.2.2 Frontend Implementation:

- **Interface:** React.js was used to build the frontend, which includes elements for perusing food products, adding them to the cart, monitoring orders, and completing payments.

- **State Management:** To manage application state (such as cart contents and user information), the application makes use of React's useState and useContext hooks.

- **Responsive design:** It ensures a seamless experience on desktop and mobile devices by making the user interface(UI) responsive.

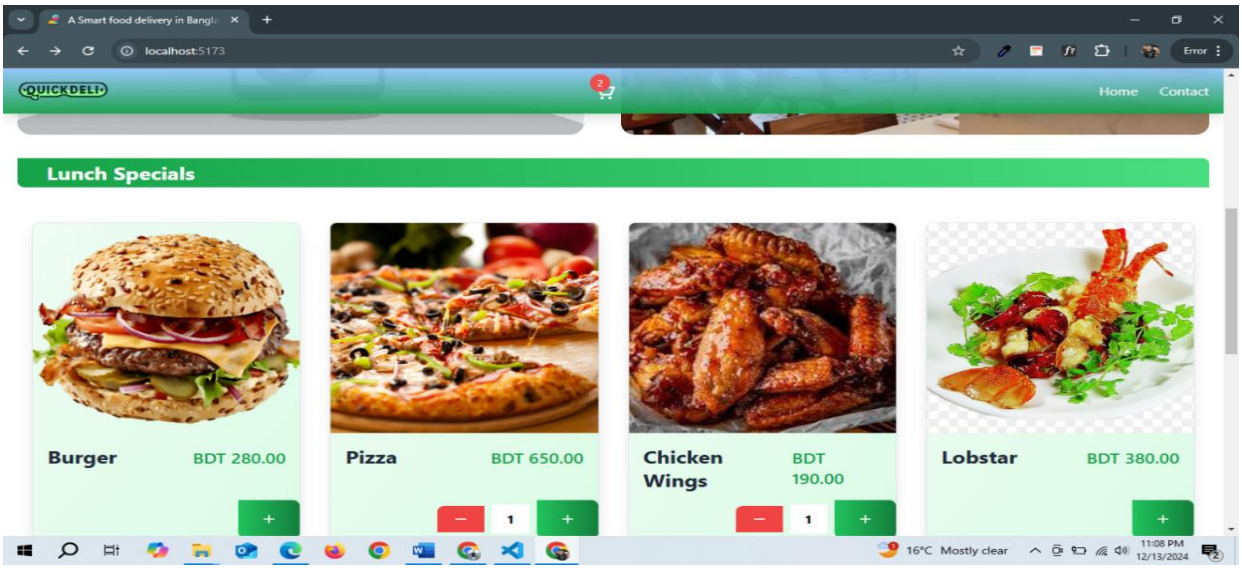


Figure 16: Menu Page

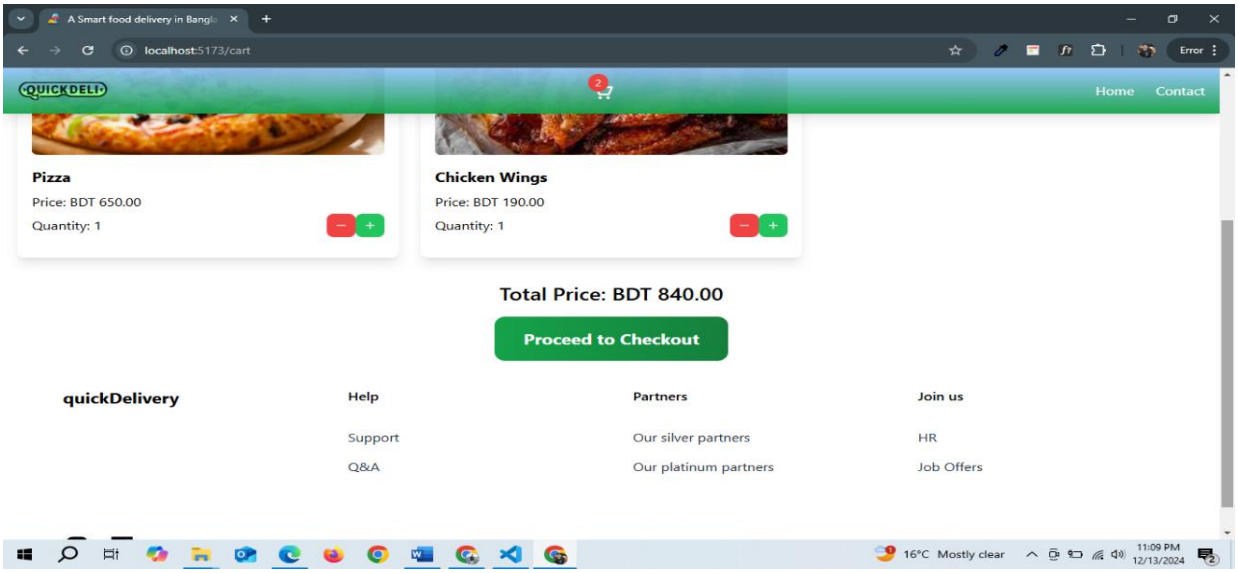


Figure 17: Cart Page

6.2.3 Integrations with Third Parties:

Restaurants may upload food photos that are shown in the frontend thanks to Cloudinary&#39’s integration for picture storage and retrieval. Customers may safely pay for their orders using Stripe&#39’s integration for secure online payments Testing the system to make sure it works as intended is an essential next step after implementation.

6.3 Testing

An essential component of the software development lifecycle is testing. It guarantees that the system functions as planned and aids in the early detection of any errors or performance problems. There are several steps involved in testing:

6.3.1 Prior to Testing

Input: It is crucial to provide the required inputs for every function prior to starting the testing process.

For instance:

User registration: Email, password, and password confirmation are entered. Prior to processing, these inputs should have their format and length verified.

Order placement: A list of the chosen products, the quantity, and the user's details (name, address, and payment method) are entered.

Expected Outcome: Every action should have a clearly defined result.

Successful Registration: A JWT token and a 200 OK response should be returned by the system after a successful registration.

A successful order placement should result in the user receiving an order confirmation with real-time tracking information and the order being recorded in the database.

6.3.2 After implement to testing

User Registration: Following testing, the system ought to be able to manage various situations, including short passwords, mismatched passwords, and incorrect email forms. An error response with a pertinent message should be displayed for every incorrect input.

Order Processing: Various inputs, like adding food items to the cart, updating the cart, and processing payments, should be handled appropriately by the system. The system ought to react with the proper error messages in the event that any mistakes arise (such as an incorrect payment method or insufficient cash). The system should produce the anticipated outcomes following the testing, such as the production of new orders, the updating of food products, and the confirmation of payments.

4. Bug Fixing and Optimization

Various flaws and difficulties are usually found during the testing phase. Finding the cause of theseproblems and making the required adjustments are the steps involved in bug repair. Typical problems that might occur include:

UI bugs: It includes things like inaccurate food item presentation, missing links, and layout problems. Usually, these vulnerabilities are found during the frontend testing stage. Inaccurate cart calculations, unsuccessful user authentication, or payment issues are examples of functionality bugs. Usually, these issues are found during the backend testing stage.

Performance bugs: Slow page loads or server crashes brought on by heavy traffic are examples of performance problems that may surface when the system is tested under load. The system is optimized to increase performance once flaws are found and fixed. This includes:

Code Refactoring: Code refactoring is the process of simplifying code to increase its readability and efficiency. Database optimization is the process of managing heavy traffic and lowering latency by using appropriate indexing, query optimization, and database connection pooling.

Caching: To decrease database load and speed up response times, frequently requested data, like meal menus, may be stored using caching technologies (like Redis).

6.5 Testing for Deployment

Deployment testing comes after the system has been put into use and tested locally. This guarantees that the system functions properly in a production setting and meets real-world requirements.

Staging Environment: The system is set up in a staging environment that closely resembles the production environment prior to being deployed to the live server. Here, additional testing is done to confirm database connectivity, server performance, and deployment configurations.

Load testing: To assess the system's capacity to manage several concurrent users and orders, it is put through simulated heavy traffic. To conduct load testing and gauge the system's reaction time under extreme demand, utilize programs like JMeter or LoadRunner.

End-to-End Testing: In order to make sure that users can register, place orders, pay, and track delivery without any problems, the complete process is tested in the deployed environment. Another name for this is acceptability testing.

Security Testing: To find any possible flaws in the system, the deployment is also put through security testing, which includes vulnerability scanning and penetration testing. Verified security features include SQL injection protection, XSS (Cross-Site Scripting) prevention, and SSL/TLS encryption.

6.6 Conclusion

For the Smart Food Ordering System to be successful, the implementation and testing phase is essential. The system is created in accordance with design specifications throughout implementation, guaranteeing that it satisfies the needs of administrators, patrons, and eateries. Testing verifies that the application can withstand real-world situations, finds defects, and makes that the system operates as intended.

Deployment testing guarantees that the platform can function flawlessly in a live production environment, while bug fixes and optimization contribute to improving the system's speed, security, and user experience. The system will offer a dependable, effective, and user-friendly online meal ordering and delivery solution with successful deployment, extensive testing, and appropriate optimization.

CHAPTER 7

Societal Impact and Sustainability

7.1 Introduction

The creation of smart food ordering systems is becoming a more crucial area of research as the need for online meal ordering platforms keeps rising. These systems, which were constructed with the help of contemporary web technologies like the MERN stack, have important ramifications for enterprises, society, and the environment in addition to enhancing operational effectiveness and customer experience. To make sure that these systems benefit the community, lessen adverse impacts, and uphold moral principles, it is essential to comprehend their sustainability and social influence. The societal effect, sustainability, ethical issues, possible difficulties, and general ramifications of putting in place a smart food ordering system are all examined in this chapter.

7.2 Societal Impact and Sustainability

A smart food ordering system has an influence on society in a number of ways, such as service accessibility, employment growth, consumer behavior, and urbanization.

- **Access to Services:** Smart Food Ordering Systems increase accessibility to food services, especially in metropolitan areas, by utilizing technology. By making ordering and tracking delivery more convenient, these systems aid in bridging the gap between customers and food providers. Consequently, more inclusive food service alternatives help consumers from a variety of demographics, such as those who live in underprivileged regions, have mobility challenges, or have time limits.
- **Job Creation:** By creating work possibilities across several industries, these platforms aid in the production of jobs. Customer service agents, restaurant employees, delivery drivers, and software engineers are all included in this. People looking for temporary or part-time work are drawn to the flexible work arrangements offered by the gig economy, which is fueled by food delivery businesses.

- **Consumer Behavior:** Consumer behavior is changing as a result of the growth of meal delivery services. Customers are now more inclined to try new cuisines and eating alternatives since they have more access to a greater range of food selections and may enjoy the convenience of having food delivered right to their home. This change in behavior could have a good impact on food trends, especially when it comes to supporting regional companies and a variety of culinary alternatives.

- **Environmental Impacts:** Although these systems are very convenient, they may also have unforeseen drawbacks, particularly when it comes to carbon emissions and food waste. Fast delivery frequently leads to packing waste, and making several journeys for separate deliveries can increase carbon footprints. The detrimental effects on the environment can be lessened by supporting sustainable practices including promoting eco-friendly packaging and streamlining delivery routes.

7.3 Sustainability

Any contemporary system must take sustainability into account while being developed, but this is especially true for systems like smart food ordering systems that communicate with a large number of stakeholders. In this sense, sustainability includes social, environmental, and economic aspects.

- **Economic Sustainability:** By creating a competitive and effective food delivery environment, a well-designed food ordering system may support economic sustainability. These platforms may increase the profitability of food enterprises by giving restaurants access to a larger client base and simplifying the order management process. Furthermore, the system may accommodate a wide variety of restaurant types, from small neighborhood restaurants to major chains, in order to foster a vibrant and diversified food service industry.

- **Environmental Sustainability:** Green logistics must be given top priority in order to guarantee that the system functions in an ecologically responsible manner. This include promoting environmentally friendly packing techniques and streamlining transportation routes to use less

fuel. To further reduce the environmental effect of food delivery, systems might collaborate with environmentally concerned businesses or put carbon offset schemes into place.

- **Social Sustainability:** Ensuring fair access for all users is a key component of the platform's social sustainability.

7.4 Ethical Considerations

Strict ethical standards must be followed during the development and deployment of smart food ordering systems to guarantee equity, privacy, and security for all parties—customers, restaurant owners, delivery drivers, and platform developers. Data security and privacy are crucial ethical considerations since these platforms gather sensitive data, including user personal information, payment information, and order preferences. To secure user data, the General Data Protection Regulation (GDPR) and other privacy rules must be adhered to.

Furthermore, it is crucial to put in place secure payment mechanisms that guard against fraud and illegal access to private data.

- **Fair Employment Practices:** The treatment of gig economy workers, including delivery drivers, is also a matter of ethics. A lot of food delivery services depend on independent contractors or freelancers, which raises concerns about worker protections, job security, and fair pay. For delivery workers, businesses must guarantee fair pay, sufficient assistance, and a secure workplace.

- **Fair pricing and transparency:** Platforms must guarantee that all prices, including delivery, service, and tipping costs, are transparent. Price gouging should be avoided to preserve equity and customer confidence, particularly during times of high demand (holidays, for example). Customers will be better able to make judgments if fees and charges are communicated clearly.

7.5 Potential Challenges

Despite the obvious advantages of smart food ordering systems, there are a few issues that must be resolved:

- **Operational Difficulties:** Coordinating food delivery logistics can be difficult, particularly in densely populated locations where bad weather or traffic jams impact delivery schedules. Restaurant kitchens must also have the necessary equipment to efficiently process a high volume of orders without sacrificing the quality of the cuisine.

- **Competition and Market Saturation:** There are several companies fighting for market share in the very competitive meal delivery sector, including DoorDash, Grubhub, and Uber Eats. Without a clear value proposition or unique features, newcomers could struggle to stand out from the competition and get clients.

- **Scalability:** It might be difficult to make sure the system can accommodate more traffic, data storage, and transactions as the user base expands. Robust backend design, including the usage of cloud computing services and load balancing techniques, is necessary to guarantee that the platform stays responsive and effective under high load.

- **Cultural and Regional Barriers:** It may take some time to educate customers and restaurant owners about the advantages of online ordering platforms, and food delivery culture may not be well-established in some areas. It could also be necessary to get beyond cultural obstacles pertaining to payment methods, cuisine preferences, and trust in online transactions.

7.6 Conclusion

Smart food ordering systems have a big influence on society and are sustainable, presenting both opportunities and difficulties. These methods might make food services more accessible, provide employment, and help the meal delivery industry expand. To guarantee that these technologies function

justly and responsibly, ethical issues including privacy, worker treatment, and price transparency must be properly controlled.

Smart food ordering systems may contribute significantly to the development of a more sustainable and inclusive future by emphasizing sustainable behaviors, such as social equality and green logistics. Collaboration between developers, companies, and regulators is essential to overcoming the challenges and guarantee that these systems keep helping the environment and society.

In conclusion, even if there are a number of obstacles to overcome, a smart food ordering system may have a significant beneficial influence on the environment, society, and economy by promoting a more ethical, sustainable, and effective food delivery sector.

CHAPTER 8

CONCLUSION

8.1 Conclusion

This project's goal was to use the MERN stack to create an online food ordering platform. In response to the growing need for quick and easy online meal ordering platforms, the platform enables smooth communication between customers, eateries, and delivery staff.

The front end of the system was developed and built with React.js, the back end with Node.js and Express.js, and the database was managed with MongoDB. Third-party APIs enabled safe payment processing and real-time delivery tracking, while RESTful APIs allowed for smooth communication between the components. In order to ensure an intuitive user experience, the development process included best practices in UI/UX design.

In addition to offering end users comfort, this platform helps local companies by increasing their online presence. By implementing eco-friendly procedures and scalable technology, it promotes sustainability and inclusion. Additionally, the project contributes to the larger field of e-commerce solutions by showcasing the useful application of contemporary web development technology.

8.2 Further Suggested Work

1. **Mobile App Development**

   Build a mobile app to provide better accessibility and enhance user convenience.

2. **Advanced Features**

   Add features like AI-based food recommendations and loyalty programs for users.

3. **Improved Analytics**

Develop analytics dashboards for restaurants to track sales and customer data.

4. **Delivery Optimization**

   Introduce algorithms to optimize delivery routes and reduce delivery time.

5. **Accessibility and Sustainability**

   Enhance accessibility features for users with disabilities and promote eco-friendly practices, like minimal packaging options.

6. **Scalability**

   Optimize the system to handle higher traffic and larger user bases in the future.

REFERENCES

[1] A. S. Jaiswal, C. R. Kulkarni, Y. Patil, S. Ponde, and R. B. Vaidya, "Smart food ordering system for restaurants," Department of Information Technology, Sinhgad College of Engineering, Pune, India, 411041, pp. [2051 - 2055]. Available:https://www.researchgate.net/publication/370939827_Smart_Food_Ordering_System_For_Restaurants.

[2] H. Pathak, N. Gupta, D. Premaker, and P. Garg, "Design of Web App for Online Food Services," International Journal for Science Technology and Engineering, vol. 10, no. 5, pp. 4316-4322, 2022, doi: 10.22214/ijraset.2022.43136.
Available: https://www.ijraset.com/research-paper/design-of-web-app-for-online-food-services.

[3] Gujar, S. Rao, K. Panpatil, S. Badhe, and D. D. Patil, "Food-delivery web application with MERN," International Research Journal of Modernization in Engineering, Technology and Science, June 2024, doi: 10.56726/IRJMETS59268.
Available:https://www.irjmets.com/uploadedfiles/paper//issue_6_june_2024/59268/final/fin_irjmets1719323307.pdf.

[4] V. Agrahari, K. S. Singh, P. Chauhan, and H. Gupta, "Online burger delivery system," International Research Journal of Modernization in Engineering, Technology and Science, May 2023, doi: 10.56726/IRJMETS40463.
Available:https://www.irjmets.com/uploadedfiles/paper//issue_5_may_2023/40463/final/fin_irjmets1685299820.pdf.

[5] Shweta S. T., Priyanka R. S., Madhura M. J. (2013). Automated Food Ordering System with Real-Time Customer Feedback. International Journal of Advanced Research in Computer Science and Software Engineering, 3(2): 220-225. Available:https://www.academia.edu/31643193/Automated_Food_Ordering_System_with_Real_Time_Customer_Feedback.

[6] S. S. T., P. R. S., and M. M. J., "Automated Food Ordering System with Real-Time Customer Feedback," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 2, pp. 220-225, 2013. [Online].
Available: https://www.academia.edu/31643193.

[7] L. Garcia, "Digital platforms in emerging markets: The rise of online food ordering in South Sudan," International Journal of Technological Advances, vol. 7, no. 1, pp. 45-61, 2023. [Online].
Available: https://www.researchgate.net/publication/378293405_Online_Food_Ordering_System.

[8] Johnson, R. (2020). The disruptive power of online food ordering systems. Journal of Culinary Technology, 8(4), 201-218.
Available: https://www.researchgate.net/publication/321844341_Online_Food_Ordering_System

[9] Hatim, N. A. Mohamad Zamani, L. M. Abdul Latif, N. Kamaruddin, N. Ahmad, and M. Kardri, "E-FoodCart: An Online Food Ordering Service," International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2019. [Online].
Available:
https://www.researchgate.net/publication/334558836_EFoodCart_An_Online_Food_Ordering_Service.

[10] K. Bhandge, T. Shinde, D. Ingale, N. Solanki, and R. Totare, "A Proposed System for Touchpad Based Food Ordering System Using Android Application," International Journal of Advanced Research in Computer Science Technology (IJARCST), 2015. [Online].
Available:
https://www.academia.edu/13345797/A_Proposed_System_for_Touchpad_Based_Food_Ordering_System_Using_Android_Application.

[11] K. Khairunnisa, A. Johari, M. Wahab, M. Erdi, M. Ayob, and A. Ayob, "The Application of Wireless Food Ordering System," MASAUM Journal of Computing, vol. 1, no. 2, pp. 178-184, 2009. [Online].
Available:
https://static.aminer.org/pdf/PDF/000/329/435/using_wireless_personal_digital_assistants_in_a_restaurant_impact_and.pdf.

[12] E. Sarmento, S. Loureiro, and R. Martins, "Foodservice tendencies and tourists' lifestyle: new trends in tourism," Revista Turismo & Desenvolvimento, vol. 1, no. 27/28, pp. 2265-2277, 2018. [Online].
Available:
https://www.researchgate.net/publication/322716238_Foodservice_tendencies_and_tourists'_lifestyle_new_trends_in_tourism_Journal_of_tourism_and_development.

[13] Acharya, Kamal, Online Food Order System (May 2, 2024). Available at SSRN: https://ssrn.com/abstract=4814732 or http://dx.doi.org/10.2139/ssrn.4814732. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4814732

[14] K. Acharya, "Online Food Order System," May 2, 2024. [Online]. Available: https://ssrn.com/abstract=4814732 or http://dx.doi.org/10.2139/ssrn.4814732.

[15] M. Roh and K. Park, "Adoption of O2O food delivery services in South Korea: The moderating role of moral obligation in meal preparation," International Journal of Information Management, vol. 47, pp. 262–273, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0268401217306941.

[16] H. S. Sethu and B. Saini, "Customer Perception and Satisfaction on Ordering Food via Internet, a Case on Foodzoned.Com, in Manipal," in Proceedings of the Seventh Asia-Pacific Conference on Global Business, Economics, Finance and Social Sciences (AP16Malaysia Conference), Kuala Lumpur, Malaysia, 15–17 July 2016, ISBN: 978-1-943579-81-5, Paper ID: KL631. [Online]. Available: https://www.scribd.com/document/429617703/Food-Zoned.

[17] S. E. Kimes, "Customer Perceptions of Electronic Food Ordering," Cornell Hospitality Report, vol. 11, no. 10, pp. 6-15, 2011. [Online]. Available: https://www.academia.edu/19814479/Customer_Perceptions_of_Electronic_Food_Ordering.

[18] L. W. Hong, "Food Ordering System Using Mobile Phone," Bachelor's thesis, Faculty of Information and Communication Technology (Perak Campus), UTAR, 2016. [Online]. Available: https://www.studocu.com/row/document/babcock-university/human-resource-management/ia-2016-1203135-1-rthhh/37276718.

[19] V. Chavan, P. Jadhav, S. Korade, and P. Teli, "Implementing Customizable Online Food Ordering System Using Web-Based Application," International Journal of Innovative Science, Engineering &

Technology, vol. 2, no. 4, pp. 231–235, Apr. 2015. [Online]. Available: https://ijiset.com/vol2/v2s4/IJISET_V2_I4_112.pdf.

[20] J. Stephens, H. Miller, and L. Militello, "Food delivery apps and the negative health impacts for Americans," Frontiers in Nutrition, vol. 7, p. 14, 2020. [Online]. Available: https://www.frontiersin.org/journals/nutrition/articles/10.3389/fnut.2020.00014/full.

[21] J. Wohl, "Domino's Unseats Pizza Hut as Biggest Pizza Chain," AdAge, Feb. 20, 2018. [Online]. Available: https://www.dominos.com.my/.

[22] K. Khairunnisa, J. Ayob, A. Mohd. Helmy, M. Erdi Ayob, M. Izwan Ayob, and M. Afif Ayob, "The Application of Wireless Food Ordering System," [Online]. Available: http://eprints.uthm.edu.my/5726/1/Wireless_Food_Ordering_System.PDF.

[23] S. M. Alagoz and H. Hekimoglu, "A study on TAM: Analysis of customer attitudes in online food ordering system," Procedia - Social and Behavioral Sciences, vol. 62, pp. 1138–1143, 2012. [Online]. Available: https://www.researchgate.net/publication/271565172_A_Study_on_Tam_Analysis_of_Customer_Attitudes_in_Online_Food_Ordering_System.

[24] W. H. Leong, "Food Ordering System Using Mobile Phone," Doctoral dissertation, UTAR, 2016. [Online]. Available: https://www.researchgate.net/publication/343555464_E-Food_Ordering_and_Diet_Monitoring_System.

Appendix

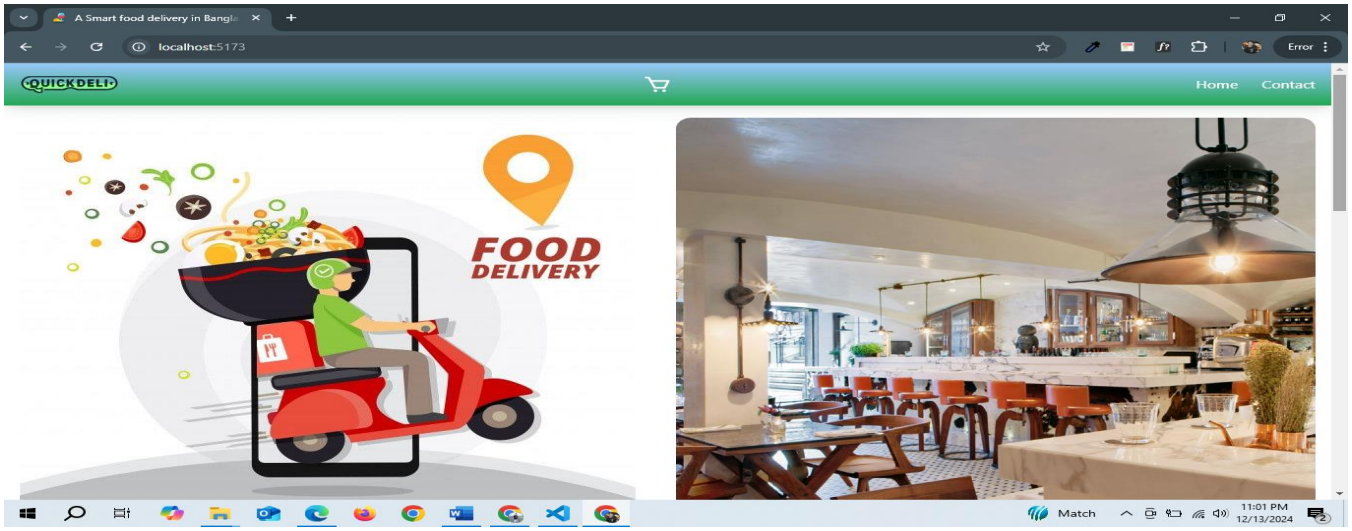


Figure: A1

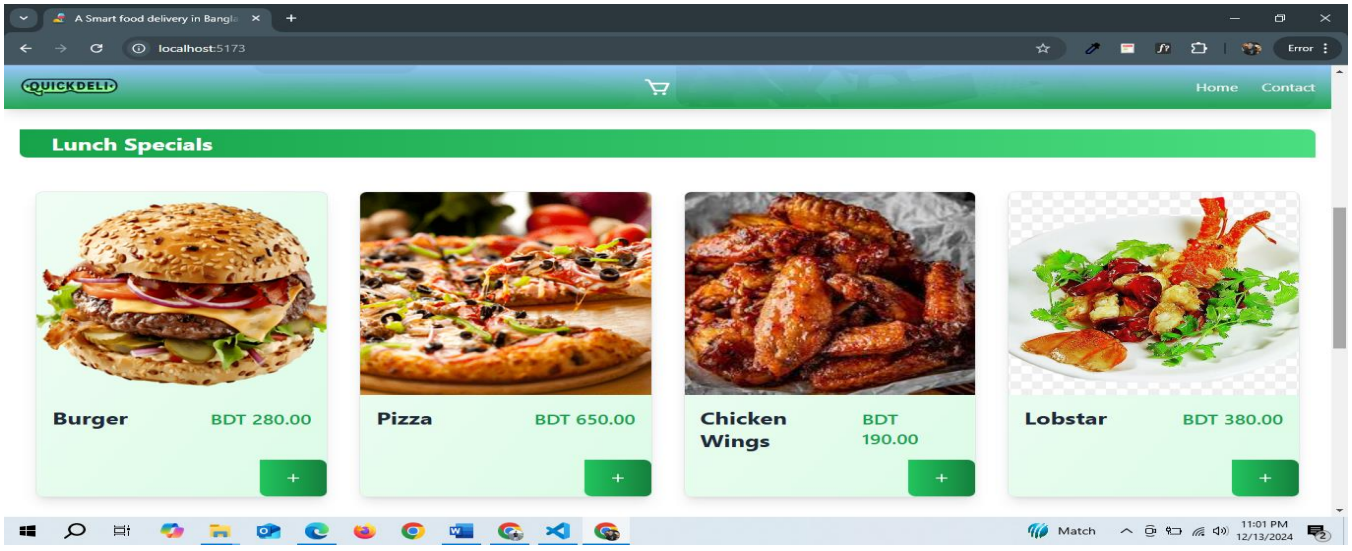


Figure: A2

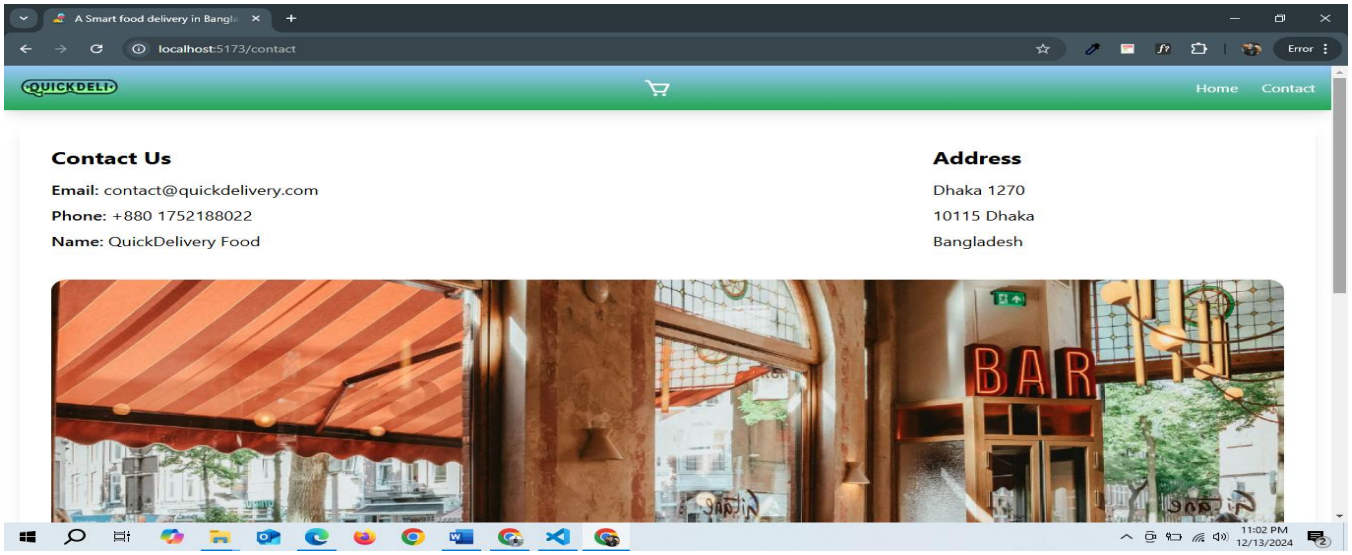


Figure: A3

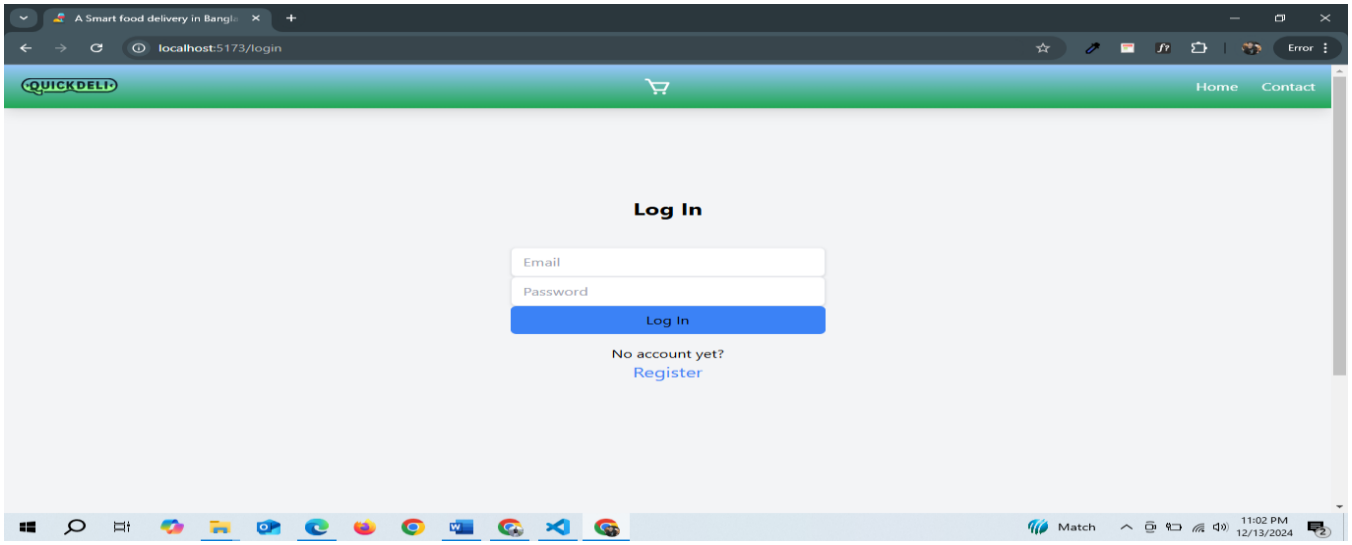


Figure: A4

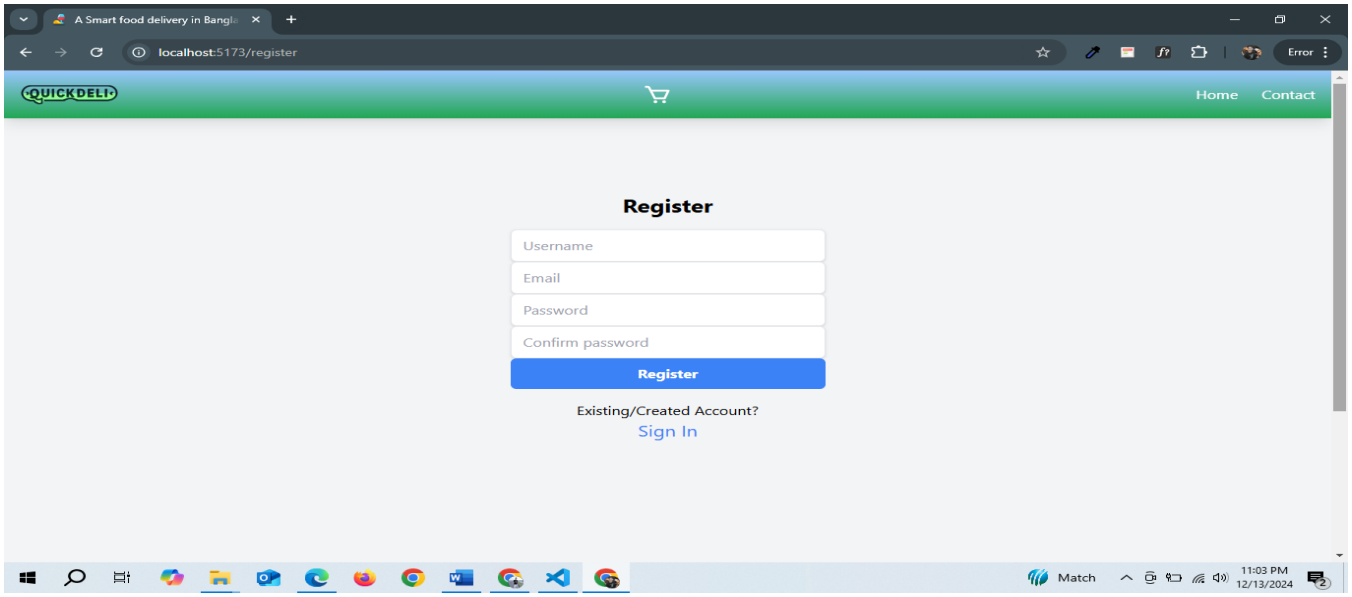


Figure: A5

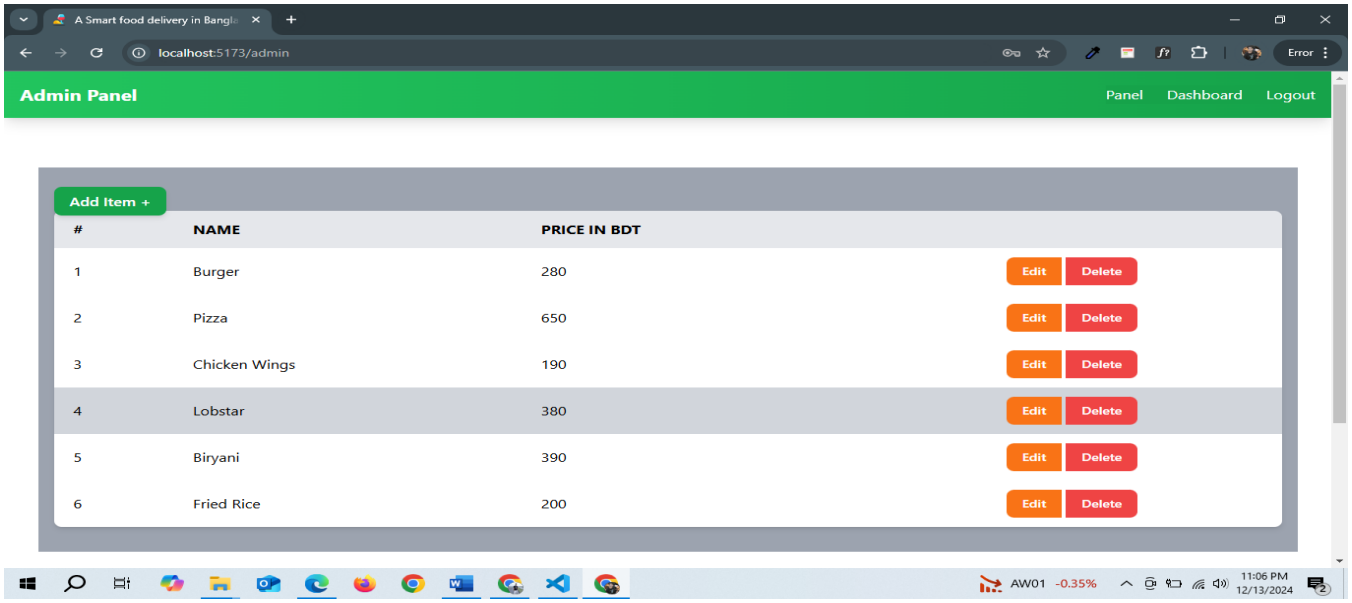


Figure: A6

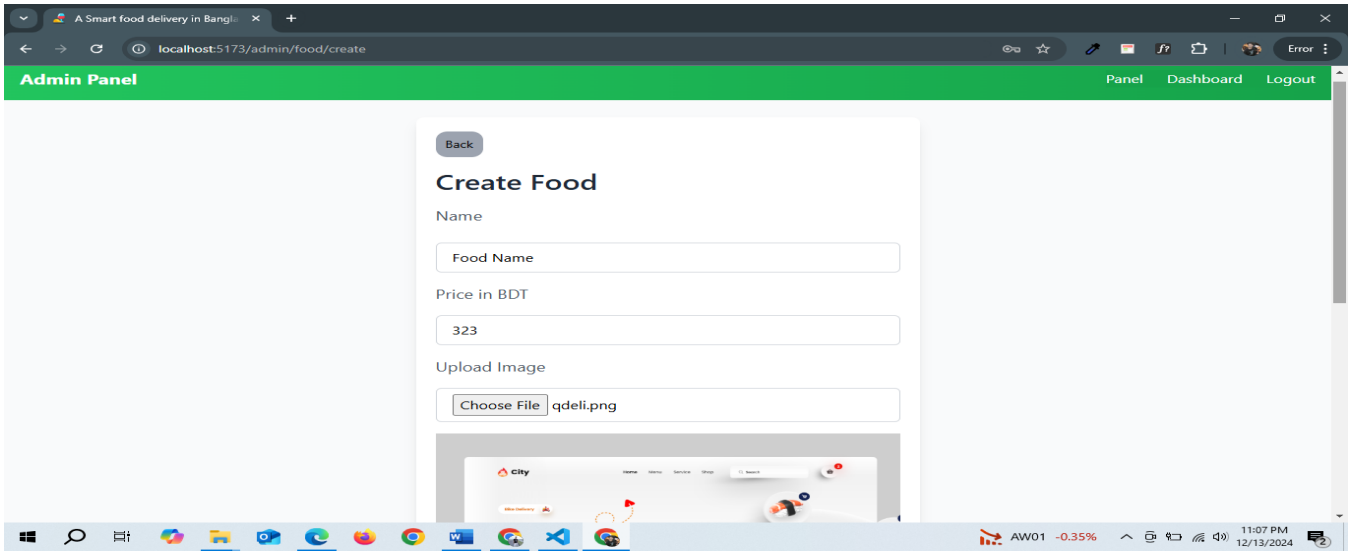


Figure: A7

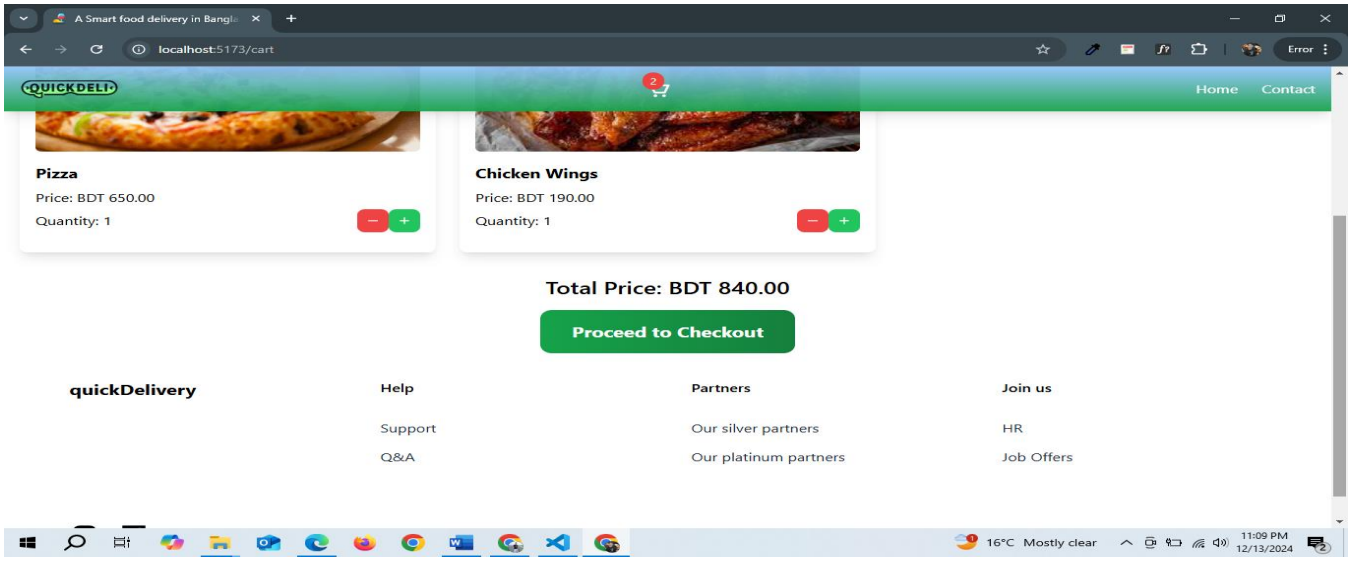


Figure: A8

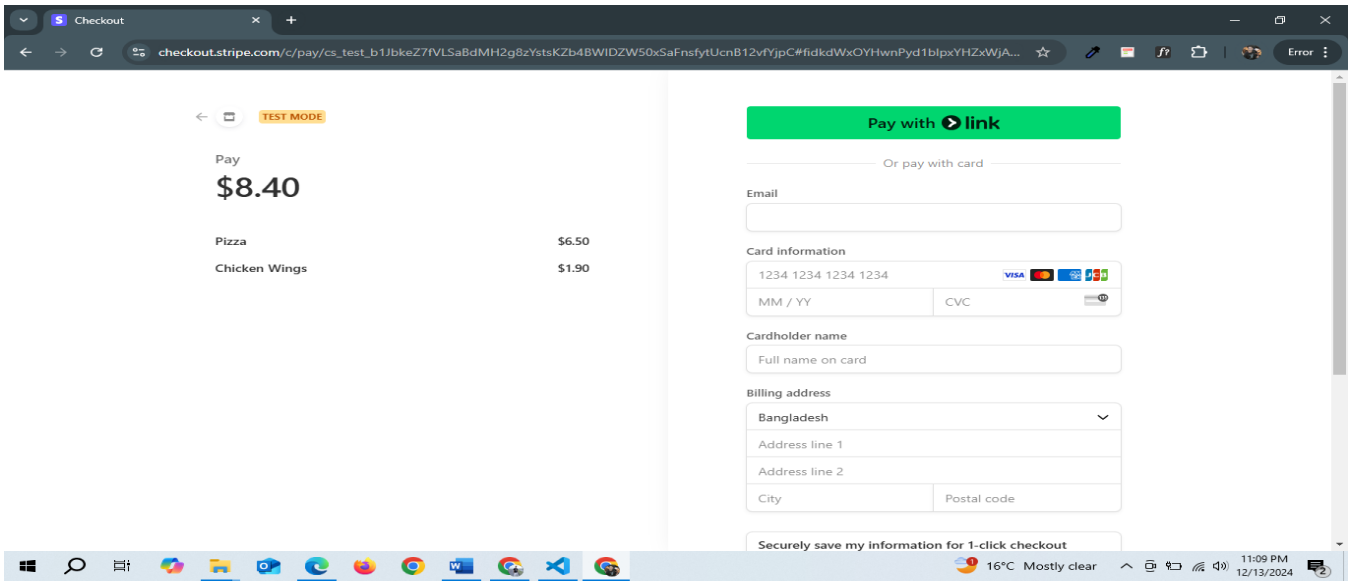


Figure: A9

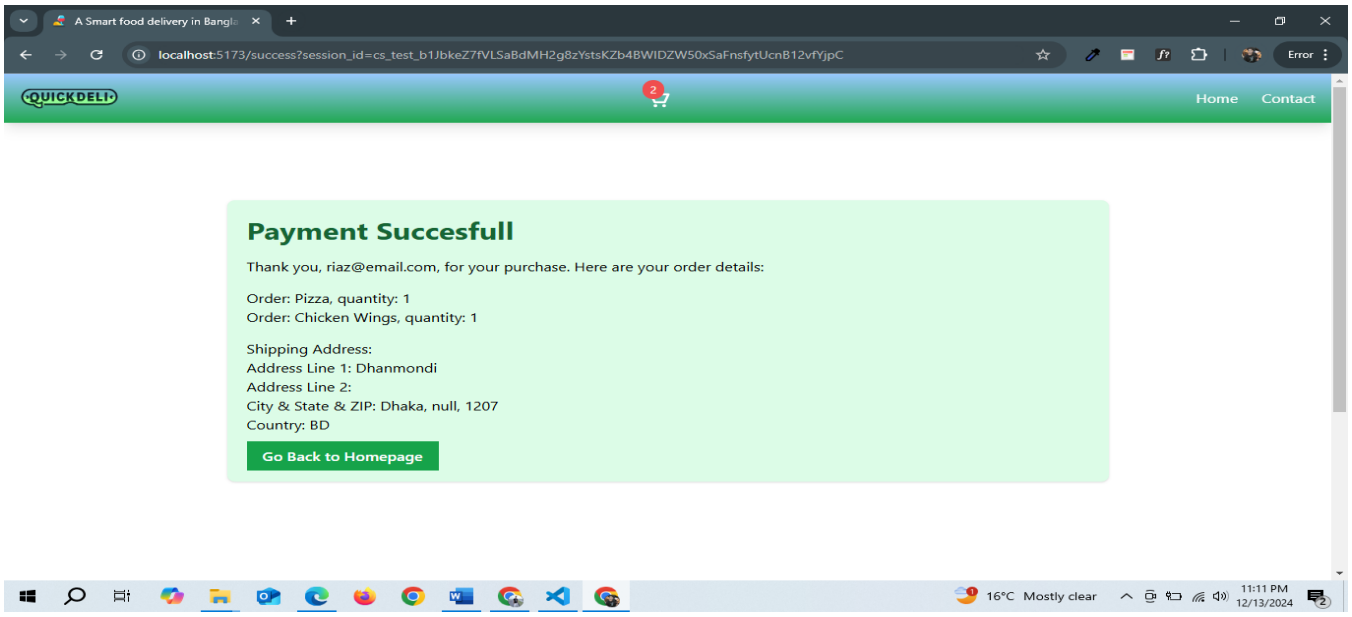
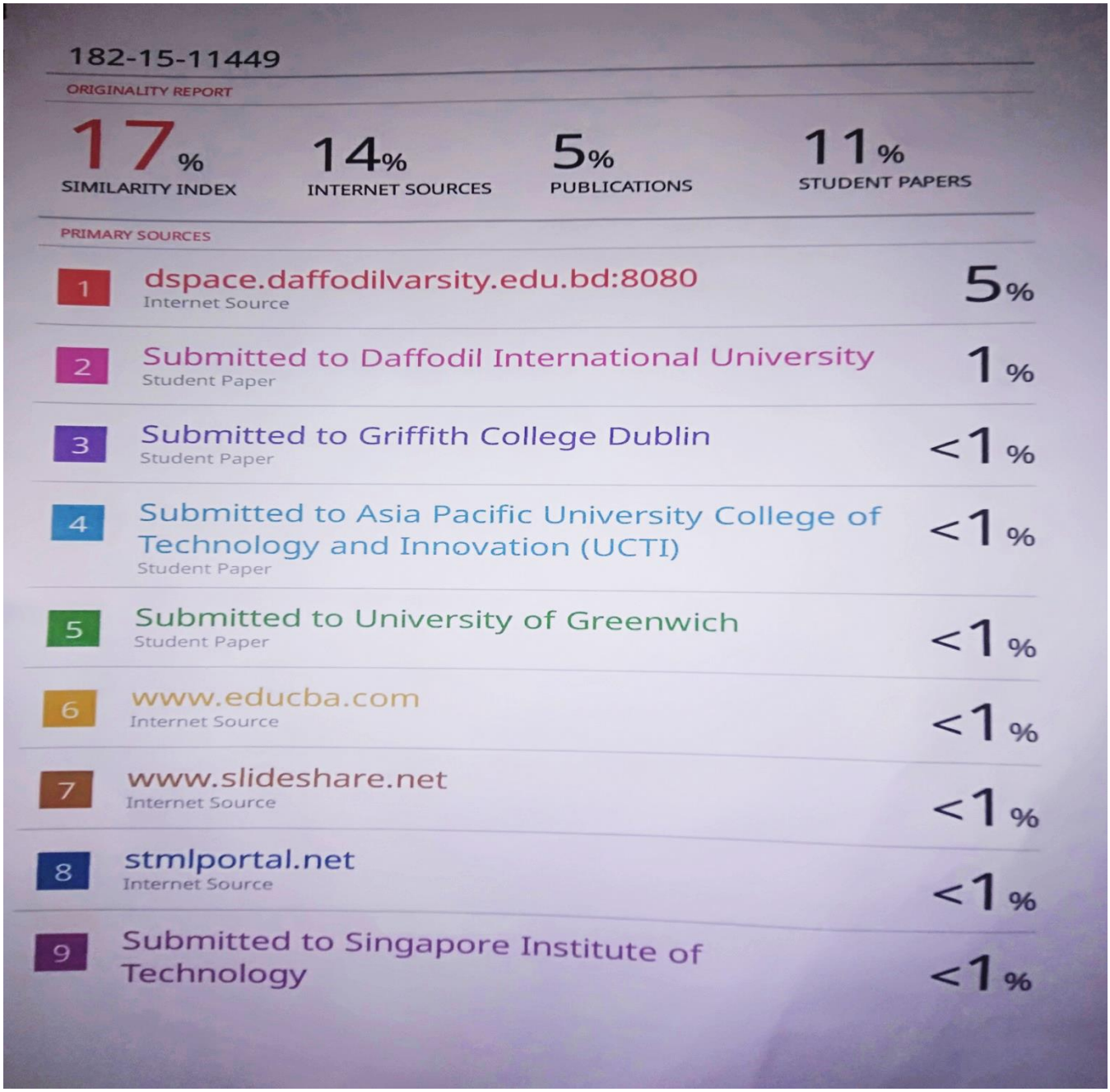


Figure: A10

PLAGIARISM REPORT



182-15-11449

ORIGINALITY REPORT

17%	14%	5%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	5%
2	Submitted to Daffodil International University Student Paper	1%
3	Submitted to Griffith College Dublin Student Paper	<1%
4	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
5	Submitted to University of Greenwich Student Paper	<1%
6	www.educba.com Internet Source	<1%
7	www.slideshare.net Internet Source	<1%
8	stmlportal.net Internet Source	<1%
9	Submitted to Singapore Institute of Technology	<1%