FOOD FUSION

A PROJECT REPORT
for
Project (KCA451)
Session (2023-24)

Submitted by

Manmeet Chauhan
(2200290140085)
Manan Sharma
(2200290140084)
Priyanshu Singh
(2200290140118)
Parth Gupta
(2200290140105)

+

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr. Amit Kumar Gupta
Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(MAY 2024)

DECLARATION

I hereby declare that the work presented in this report entitled **"FOOD FUSION"**, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma from any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, and results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name – Manmeet Chauhan (2200290140085)

Manan Sharma (2200290140084)

Priyanshu Singh (2200290140118)

Parth Gupta (2200290140105)

CERTIFICATE

Certified that **Manmeet Chauhan(2200290140085), Manan Sharma(2200290140084), Priyanshu Singh(2200290140118), Parth Gupta(2200290140105)** have carried out the project work having **Food Fusion** (**Major Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Manmeet Chauhan(2200290140085)

Manan Sharma(2200290140084)

Priyanshu Singh (2200290140118)

Parth Gupta (22002901400105)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Amit Kumar Gupta
Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Kumar Tripathi
Professor & Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

FOOD FUSION

Manmmet Chauhan , Manan Sharma , Priyanshu Singh , Parth Gupta

ABSTRACT

In the fast-paced digital era, the demand for convenient and efficient food ordering solutions has surged, prompting the development of innovative platforms to cater to diverse culinary preferences. This abstract introduces a food ordering website designed to revolutionize the way individuals engage with dining experiences. The platform offers a seamless interface that enables users to explore a curated selection of local favorites and international cuisines, all from the comfort of their homes or offices. Emphasizing user-friendliness and quality assurance, the website ensures a hassle-free ordering process and prioritizes partnerships with reputable restaurants committed to culinary excellence. With doorstep delivery and a commitment to customer satisfaction, this food ordering website aims to elevate the dining experience, making every meal a celebration of flavor and convenience. Beyond merely facilitating food delivery, this platform serves as a gateway to a world of gastronomic delights, offering an extensive array of cuisines ranging from local favorites to exotic international fare. Through a user-friendly interface, customers can effortlessly browse menus, customize orders, and seamlessly navigate payment options, ensuring a streamlined and enjoyable ordering p

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Amit Kumar Gupta** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, the Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Manmeet Chauhan (2200290140085)

Manan Sharma (2200290140084)

Priyanshu Singh (2200290140118)

Parth Gupta (2200290140105)

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Online food ordering is the process of ordering food from a website. The product can either be food that has been specially prepared for direct consumption (such as vegetables straight from a farm or garden, frozen meats, etc.) or food that has not been (such as direct from a certified home kitchen, or restaurant). The effort to create an online food ordering system aims to replace the manual method of taking orders with a digital one. The ability to rapidly and correctly create order summary reports whenever necessary is a key factor in the development of this project.

The potential of an online food ordering system is enormous. Any restaurant or fast-food chain can use this MERN project to keep track of customer orders. This project is simple, quick, and precise. There is less disk space needed. MongoDB is used as the backbone of the online food ordering system, eliminating the risk of data loss and ensuring data security. A customer starts by scanning the menu, picks an item, and then orders the food, you can pay with cash at the restaurant or with UPI payment. The customer is informed by the website about the food's quality, how long it takes to prepare, and when it will be ready for pick-up or delivery.

Building modern and dynamic online apps now necessitates a strong grasp of web development. The MERN stack is a well-known and powerful mix of technologies that enables programmers to create extremely scalable and successful online applications. In this introduction, the components of the MERN stack will be covered, as well as their significance in web development.

The MERN stack is comprised of MongoDB, Express, React, and Node.js. Each component is critical to the development process and aids in the overall operation and execution of the web application.

As the market sees a growth within the electronic food delivery apps services, it becomes important to research the market, the buyer behaviour and perception towards the service. This may successively help to know what the buyer wants. By capitalizing on these findings, the businesses can innovate and define new strategies and serve them better. The matter which the researcher studied within the paper was to review the perception, satisfaction and behaviour of school students towards electronic food delivery platforms in Mumbai. During this research, 220 people were surveyed to seek out different consumer attitudes of the purchasers just like the loyalty, spending pattern, nutritional intake, and the well-liked mode of payment. The research would help the varied platforms; mainly to specialise in the areas where they're lacking and help them better target the requirements of the buyer. It can help them target the new consumers through popular channel. They will realize the consumer’s preferences with reference to payment options, availability of healthy food and various other aspects.

1.1 Objectives

The management of the information regarding item category, food, delivery address, order, and shopping cart is the system's primary goal. It oversees the management of all customer, shopping cart, and item category information. Since the project was entirely developed on the administrative end, only the administrator is assured access. The goal is to develop an application program to simplify managing the food consumer item category.

The overall goal of this project is to increase web development productivity

and address the difficulties that developers have while using conventional web development techniques. We strive to offer a complete and scalable solution that streamlines the development process, boosts productivity, and assures excellent performance and maintainability of contemporary web applications by using the MERN stack.

1.2 Need for Online Food Order

Needs of Online Food Orders Helping customers place meal orders whenever they want. Customers will be able to order their preferred foods at any time, but as we've already mentioned, this is only a limited option. As a result, restaurants need to have a specific system in place that will allow them to serve a large number of customers while streamlining operations. One of the best platforms is ordering, which offers all of these services in addition to a host of cutting-edge features that have helped countless small and large enterprises establish themselves as market leaders.

1.3 Modules

A food ordering website typically consists of several key modules to ensure its functionality and user-friendliness.

1.3.1. User Modules

1. User Registration and Authentication:

This module allows users to create accounts, log in securely, and manage their profiles. It is essential for personalizing the user experience and ensuring order history.

2. Menu Management:

Restaurants can manage their menus through this module, including adding, editing, and deleting dishes, along with setting prices and descriptions.

3. Order Placement:

Users can select items from menus, customize orders (e.g., choosing toppings or specifying cooking preferences), and place orders.

4. Shopping Cart:

Users can view and edit their orders in a virtual cart before finalizing the purchase. This module calculates the total cost, including taxes and delivery fees.

5. Rating and Reviews:

Users can rate and write reviews for restaurants and dishes they've ordered. This feedback helps other users make informed decisions. . Admin Dashboard: Administrators can manage the platform, including user accounts, restaurant partnerships, menu approval, and resolving disputes.

1.3.2. Admin Modules

1. Add Items:

Approve and manage restaurant menus, including adding, editing, or removing menu.

2. Edit Users:

Track user behavior and preferences.

3. Payment:

Oversee payment processes including customer payments, payouts to restaurants, and delivery personnel.

1.4 Functionalities

• Provides search options based on a variety of criteria. like Food Item, Customer, Order, and Order Confirmation.

• Online food ordering systems also manage payment information for order details, order confirmation details, and food items online.

• It keeps track of all the data regarding Categories, Payments, Orders, etc.

• Manage the category's details.

• Manage the category's details.

• Displays the food item's information and description to the customer. Easy to manage the

• Food Items, and Categories more effectively.

• It focuses on keeping track of order data and transactions

1.5 Problem Statement:

A persistent need for effective and scalable technologies that allow for the construction of contemporary and dynamic web applications exists in the field of web development. The conventional method of using several technologies for various web development tasks frequently results in complexity, inefficiency, and integration difficulties. The issue at hand is the requirement for a thorough and coherent framework that simplifies web development from front-end to back-end while guaranteeing great speed, scalability, and maintainability. The absence of a cohesive approach frequently reduces developer productivity, lengthens the development process, and limits their capacity to quickly adjust to changing project needs. Additionally, developers who need to migrate between several technologies frequently have compatibility problems and a high learning curve as a result of the lack of a standardised stack. In addition to raising development costs, this makes it more difficult to maintain and update online applications over time. The MERN stack, which consists of MongoDB, Express, React, and Node.js, offers a potential remedy for these problems. By using the advantages of these technologies, the problem of fragmented and inefficient web development may be resolved. However, it is vital to recognise and comprehend the precise problems and challenges that developers deal with while using conventional web development techniques.

CHAPTER 2

LITERATURE REVIEW

The research papers we considered while doing our analysis are listed below. A wireless meal ordering system was designed and implemented together with consumer feedback for a restaurant. It makes it simple for restaurant operators to change menu presentations and set up the system in a WIFI setting. The configurable wireless meal ordering system has linked a smart phone with real-time customer feedback implementation to enable real-time contact between patrons of restaurants and business owners. The goal was to investigate the variables that affect internet users' perceptions of online food ordering among university students in Turkey. Davis's Technology Acceptance Model (TAM), which he created in 1986, was used to analyse how the Web environment for ordering food was adopted. Along with TAM, three additional primary factors—Trust, Innovation, and External Influences—are included in the paradigm.

This research examines the initiatives made by restaurant owners to implement ICTs—such as PDAs, wireless LANs, and pricey multi-touch screens—to improve the dining experience. In order to address some of the drawbacks of the

traditional paper-based and PDA-based food ordering systems, a low-cost touchscreen-based restaurant management system that uses an Android smartphone or tablet is suggested in this study.

The study's objective was to determine whether the application is user-centred and based on user requirements. This system developed all problems pertaining to every user that it includes. Almost anyone may use the program if they know how to use an Android smartphone. The various problems with the Mess service will be resolved by this system. The implementation of an online food ordering system is done to assist and resolve significant issues for consumers. Based on the application, it can be said that: This system makes placing orders simple; it gives customers the information they need to place orders.

The evolution of food ordering systems can be traced back to traditional methods such as phone orders and walk-in orders at restaurants. However, the emergence of the internet and mobile technologies has revolutionized the way people order food. Online food ordering platforms like Seamless, Grubhub, and Uber Eats have gained immense popularity, offering convenience and variety to consumers. These platforms utilize web and mobile applications to connect users with nearby restaurants, enabling them to browse menus, place orders, and track deliveries in real- time.

The well-known MERN stack for web development consists of four technologies: MongoDB, Express.js, React.js, and Node.js. This stack is well-known for its versatility, efficacy, and usefulness. One of the primary advantages of utilising the MERN stack is that programmers may construct full-stack web apps using only JavaScript.

In this review of the literature, we'll look at various facets of creating a web application with the MERN stack, such as setting up the environment for development, sending data from React to Node.js, utilising middleware like body-parser in Express.js, fetching data in React, handling post requests in Express.js, and connecting to a MongoDB database. Developers may use a variety of online

instructions to build up the development environment for a web application using the MERN stack. The procedures to link React and Node.js are described in one such tutorial, available at codedamn.com. The usage of package.json to manage dependencies and concurrently to execute the front end and back end simultaneously are highlighted in this guide.

There are numerous ways to accomplish the frequent need of sending data from React to Node.js in web development. One tutorial (tutsmake.com) explains how to use Axios to transmit data from React to a Node.js server via an HTTP POST request. It also describes how to parse the request body using the Express.js middleware body-parser. Express.js's robust middleware capability enables developers to extend the server's capabilities.

The usage of the body-parser middleware in Express.js is described in one article on Stack Overflow (stackoverflow.com). Before the handlers, this middleware is used to process incoming request bodies. To update the front-end with fresh data, it is usual practice in web development to retrieve data from a server. One tutorial (developerway.com) describes how to utilise React's fetch API to get data from a server. Additionally, it explains how asynchronous functions are used and guarantees that the response data will be handled. 12 Express.js's post-request handling is a crucial component of server-side web development.

How to handle post requests in Express.js and deliver a response to the client is covered in one article on Stack Overflow. This post also describes how to handle asynchronous activities by using the async/await syntax. Using the MERN stack to construct a website requires connecting to a MongoDB database. The procedures to set up an Express.js server with a MongoDB database are described in one tutorial at (mongodb.com). It also teaches how to build models for MongoDB collections using the Mongoose library. There are some tips and tactics for developers in addition to the technical elements of web development utilising the MERN stack. How to duplicate a multidimensional array in JavaScript is described in one article on Stack Overflow.

The usage of Axios to send HTTP requests in React is covered in another article on zetcode.com. On Telerik's website, there are lessons on how to build dynamic

forms. Last but not least, there are some online resources you can utilise to learn more about React Router, the tool used to manage navigation in a React application. Resources for upgrading to React Router v6 are also available at reactrouter.com as well as React Router's ability to transmit data across pages. In conclusion, there are many online resources for learning how to utilise the MERN stack, which is a potent technological stack for developing web applications.

2.1 Challenges in Food Ordering Systems:

Despite the numerous benefits offered by food ordering systems, several challenges persist. One of the primary challenges is ensuring food safety and quality during the delivery process. Maintaining the temperature integrity of perishable items and preventing contamination require robust logistics and packaging solutions.

Moreover, the reliance on third-party delivery services introduces concerns regarding commission fees, delivery times, and quality control. Restaurant owners often face pressure to maintain profitability while navigating the complexities of commission structures imposed by delivery platforms.

From a technological perspective, ensuring the security of online transactions and protecting user data against cyber threats remains a critical challenge. With the increasing prevalence of online payment methods, safeguarding sensitive information such as credit card details is paramount to building trust and credibility among consumers.

2.2 Future Directions and Innovations:

Looking ahead, several trends and innovations are poised to shape the future of food ordering systems. One notable trend is the rise of ghost kitchens, also known as virtual kitchens or cloud kitchens, which operate solely for delivery and catering to the growing demand for off-premises dining options. Additionally,

advancements in augmented reality (AR) and virtual reality (VR) have the potential to enhance the online ordering experience by allowing users to visualize menu items in a simulated environment before making a purchase. This immersive technology can help address concerns related to food presentation and portion sizes, ultimately improving customer satisfaction.

Furthermore, the integration of blockchain technology holds promise for enhancing transparency and traceability within the food supply chain. By leveraging blockchain's decentralized ledger, consumers can verify the origin and authenticity of ingredients, thereby fostering trust and accountability in the food industry.

CHAPTER 3

FEASIBILITY STUDY

A feasibility study is a detailed analysis that considers all of the critical aspects of a proposed project in order to determine the likelihood of it succeeding.

Success in business may be defined primarily by return on return on investment, meaning that the project will generate enough profit to justify the investment. However, many other important factors may be identified on the plus or minus side, such as community reaction and environmental impact.

A feasibility study is an important step in any project, including an emotion detection project. It helps to determine the technical, economic, operational, and legal feasibility of the project. Here are some key aspects to consider in a feasibility study for an emotion detection project.

Based on the results of the feasibility study, the project team can make informed decisions about the viability and scope of the emotion detection project. If the feasibility study indicates that the project is viable and has potential benefits, the team can proceed with the project planning and implementation. If the study

indicates that the project is not feasible or has significant risks and limitations, the team can consider alternative approaches or abandon the project altogether.

Before starting the project, feasibility study is carried out to measure the viable of the system. Feasibility is necessary to determine is creating a new or improved system is friendly with the cost, benefits, operation, technology and time.

Feasibility studies are important for a communications service provider to determine whether your broadband project will succeed or not. It should be the first action taken when wanting to begin a new project. It is one, if not the most important factor indetermining whether the project can and should move forward. Also, if you are applying for broadband loans and grants, a feasibility study is normally required. Following feasibility is given below:

3.1 Technical Feasibility

The technical feasibility of the Zomato Clone project involves assessing the availability of resources, technology requirements, compatibility, scalability, and security considerations. Here are the key factors to consider:

Resource Availability:

- Evaluate the availability of skilled developers, UI/UX designers, and testers for web development.
- Assess the availability of infrastructure, servers, and hosting resources for the backend components.
- Consider the availability of devices and browsers for testing the website on various platforms.

Technology Requirements:

- Determine the tech stack: MongoDB for the database, Express.js for the backend, React for the front end, and Node.js for server-side scripting.
- Evaluate the compatibility of the chosen technologies with different web browsers.

- Consider the integration capabilities of the MERN stack with external services or APIs for features like location services.

Compatibility:

- Ensure the website is compatible with a wide range of web browsers (Chrome, Firefox, Safari, etc.).
- Perform testing on different devices and screen sizes to ensure a responsive and consistent user experience.
- Adhere to web standards and guidelines to ensure compatibility across platforms.
- Scalability and Performance:
- Design the website architecture to handle a large number of users, restaurants, and menu items.
- Optimize database queries and backend processes for efficient performance.
- Conduct load testing to simulate a high number of concurrent users and ensure the website's responsiveness.

Security:

- Implement robust authentication and authorization mechanisms to protect user data and prevent unauthorized access.
- Use encryption methods for sensitive data, such as user credentials and payment information.
- Follow best practices for secure communication protocols (HTTPS) and data storage.
- Integration of different modules.
- Evaluate integration requirements with external services for features like maps, reviews, and payment gateways.
- Implement APIs or web services for seamless integration with external systems.

- Ensure compatibility with relevant APIs and adhere to their usage policies and limitation

Testing and Quality Assurance:

- Develop a comprehensive testing strategy that includes functional testing, usability testing, and compatibility testing across different browsers.
- Perform rigorous testing to identify and fix any bugs or issues before deploying the website.
- Conduct security testing and vulnerability assessments to protect against potential threats.

Documentation and Training:

- Prepare technical documentation that outlines the website's architecture, features, and deployment instructions.
- Provide user manuals and guides for restaurant owners, customers, and administrators to understand and use the website effectively.
- Conduct training sessions or provide training resources to familiarize users with the website's functionalities.
- Considering these technical feasibility factors will ensure the successful development, deployment, and performance of the Zomato Clone and meeting requirements.

3.2 Economical Feasibility

For economic feasibility, Economic analysis or cost/benefits analysis is the most frequently used technique for the effectiveness of a proposed system. it is a procedure to determine the benefits and savings that are expected from the proposed system and compare them with cost if the benefits outweigh the costs, a decision is taken to design and implement the system. otherwise, further justification or alternative in the proposed systemwill have to be made if it is to

have a chance of being approved this is an ongoing effort that improves in accuracy at each phase of a system life cycle.

- **Cost-Benefit Analysis:** Assessing the potential costs involved in implementing a food ordering system against the anticipated benefits such as increased revenue, improved operational efficiency, and customer satisfaction.

- **Return on Investment (ROI):** Determining the expected return on investment over a specific period, considering factors like initial setup costs, ongoing maintenance expenses, and projected revenue growth.

- **Market Demand**: Analyzing the demand for online food ordering services in the target market, including factors like demographics, consumer preferences, and competitor analysis to gauge the revenue potential.

- **Scalability**: Evaluating the scalability of the food ordering system to accommodate future growth and expansion, while minimizing additional investment and operational costs.
- **Cost Reduction**: Identifying opportunities to reduce costs through automation, streamlining processes, and optimizing resource allocation within the food ordering system.

- **Revenue Generation**: Exploring various revenue streams such as transaction fees, subscription models, advertising, and partnerships to maximize revenue generation potential.

- **Risk Assessment**: Conduct a thorough risk assessment to identify potential economic risks such as market volatility, regulatory changes, competitive pressures, and technological disruptions, and develop mitigation strategies accordingly.

3.3 Operational Feasibility

No doubt the technically growing world needs more enhancement in technology, this app is very user friendly and all inputs to be taken all self-explanatory even to a layman. As far as our study is concerned, the clients will be comfortable and happy as the system has cut down their loads and brought the young generation to the same virtual world they are growing drastically.

Operational feasibility covers two aspects.one technical performance aspects and the other is acceptance within the organization.

Operation feasibility determine how the proposed system will fit in with the current operation and what needs to implement the system.

- **Alignment with Organizational Objectives**: The proposed system should align with the strategic goals and objectives of the organization to ensure its successful implementation.

- **Compatibility with Existing Processes**: The system should be compatible with the existing business processes, infrastructure, and technologies to minimize disruptions and facilitate integration.

- **Resource Availability**: Assess the availability of necessary resources such as financial, human, and technical resources required for system development, implementation, and maintenance.

- **Skills and Training Requirements**: Evaluate whether the organization has the necessary skills and expertise to develop, operate, and maintain the proposed system. Determine if additional training or hiring is required.

- **Acceptance by Stakeholders**: Consider the level of acceptance and support from key stakeholders, including management, employees, customers, and external partners, as their buy-in is essential for successful implementation.

- **Risk Assessment**: Identify potential risks and challenges associated with system implementation, such as technological barriers, resistance to change, and regulatory compliance issues. Develop mitigation strategies to address these risks effectively.

- **Scalability and Flexibility:** Assess the system's scalability and flexibility to accommodate future growth, changes in business requirements, and emerging technologies without significant disruptions or costly modifications.

- **Impact on Operations:** Analyze the potential impact of the new system on day-to-day operations, productivity, efficiency, and customer service.

CHAPTER 4

SYSTEM DEVELOPMENT

4.1 Analysis

An online platform that links businesses and customers, a meal delivery application enables users to purchase food from the restaurant of their choice and have it delivered right to their homes. The programme will primarily consist of three parts: a database, an API server, and a web-based client. The application needs to be user-friendly, safe, and scalable.

4.2 Design

The MERN (MongoDB, Express, React, Node.js) stack, which offers a strong and adaptable environment for developing scalable online applications, will be used to construct the application. The database used to store restaurant and customer data will be MongoDB. The backend API server, which manages requests and answers between the database and front-end, will be constructed using Express. Customers will be able to explore and order meals from the restaurant thanks to a responsive and user-friendly frontend that will be built using React. The backend server will operate on Node.js.

4.3 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.

The main stages necessary to create a food delivery application utilising the MERN stack are summarised in this algorithm. The database schema must be designed, the server and endpoints must be configured, the database driver must be integrated, and the server-side and client-side logic must be put into place. The construction of this application relies heavily on the use of React, React Router DOM, useState, useEffect, useLocation, JSON, Express, body-parser, CORS, and MongoDB JS driver.

4.4 Model Development

4.4.1 MongoDB

A common NoSQL database management system, MongoDB, is utilised in many contemporary online apps, including those that serve meals. One of MongoDB's key benefits is its capacity for document-based data storage, which makes it the perfect option for applications requiring dynamic schema structures. MongoDB may be used to store a variety of data types in the context of a meal delivery application, including user profiles, orders, menus, and reviews. Each data item is represented as a document, a data structure akin to JSON that supports many fields of various

types and values. As a result, sophisticated SQL queries and rigid schema structures are not a concern for developers when storing and retrieving complex data items.
Image

Fig.4.4.1 mongo DB

Additionally, MongoDB has strong querying and indexing features that let developers easily access data and run sophisticated searches with a simple syntax. MongoDB may be used by a food delivery service, for instance, to discover all restaurants that provide a given cuisine or to retrieve all orders placed by a single user. Because of MongoDB's automatic sharding and replication features, flexibility, and querying power, developers can easily scale their applications as data volume grows.

Encryption, access control, and authentication are just a few of the security capabilities that MongoDB offers to help protect the data stored in the database. Overall, MongoDB is a strong and adaptable database management system that is suitable for cutting-edge online applications like those for food delivery. It is a popular option for developers that need to create reliable and scalable applications because of its capacity to store and retrieve complex data items in a flexible and scalable manner and its strong searching and indexing capabilities.

4.4.2 React.js

A well-liked JavaScript library for creating user interfaces is React.js. It was created

by Facebook, and a sizable development community is now responsible for maintaining it. React was used to develop the front-end of the application, which is the user-facing portion of the programme, in the context of the food delivery application. React offers a collection of components and features that make it simple to construct intricate user experiences rapidly. It makes use of a component-based design, which implies that the user interface is composed of simple, interchangeable parts that may be joined to produce more intricate interfaces**.**

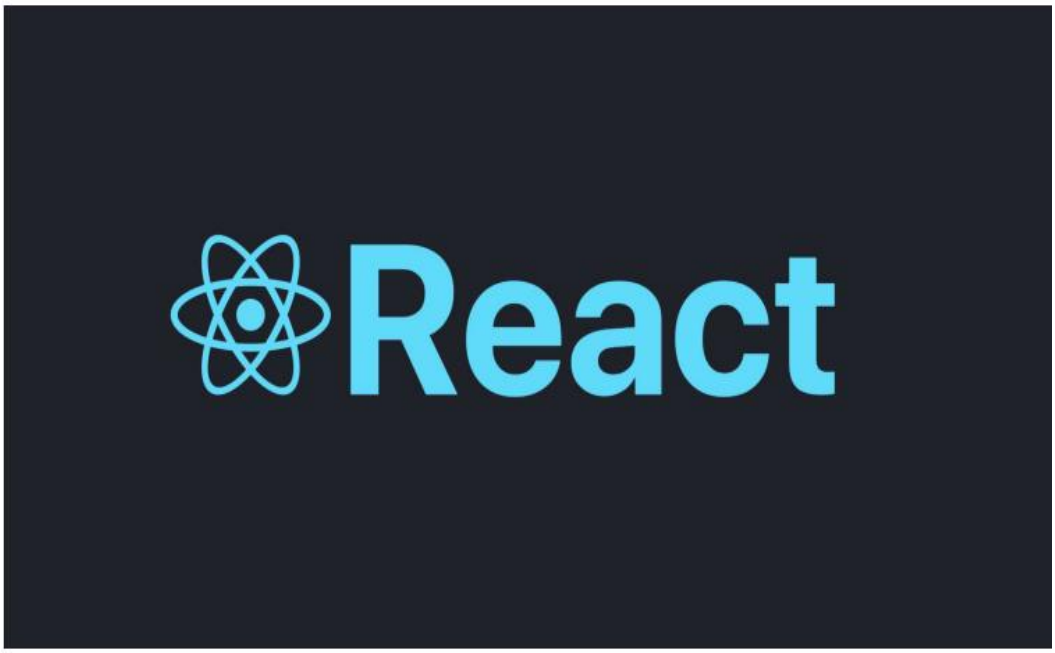

Fig.4.4.2 React.js

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.

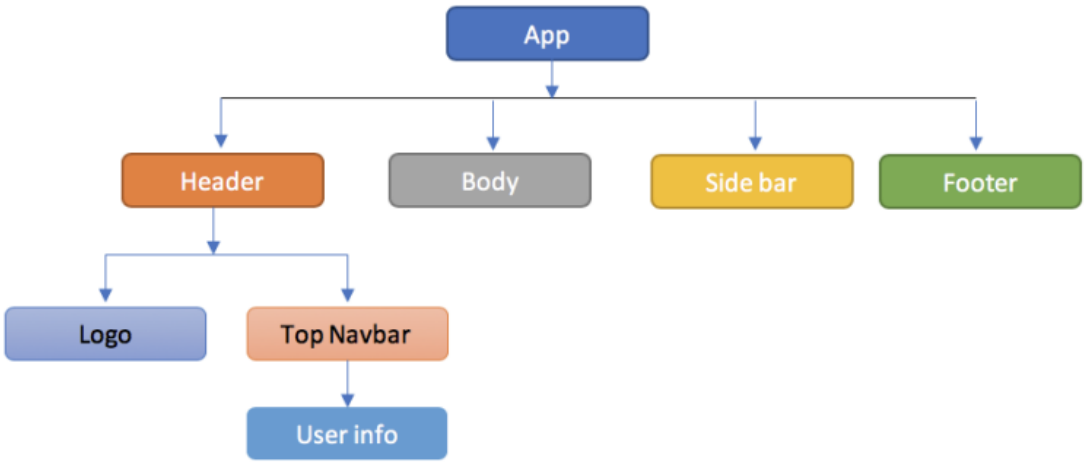

Fig.4.4.2 React Component based Development

The seamless user experience that React offers makes it one of the most advantageous technologies to utilise in a food delivery service. React's component-based architecture makes it simple to construct and reuse UI elements, and its virtual DOM enables quick and effective UI modifications. This may lead to an application that is quicker and more responsive and offers a better user experience. In a food delivery app using the MERN stack, several components are essential to providing a smooth user experience. These components include the home, menu, checkout, navigation, pizza, and sign-up components.

● The user opens the app, and the first screen they see is the home screen. A list of the app's features and a description of any highlighted goods and current promotions may be included.

● The user's order summary is shown and the payment procedure is managed by the checkout component. It may have supporting elements like a form for inputting

payment information, a form for choosing an address, and a confirmation page.

● The app's options are shown in the menu component. This element may be created with search bars, filter choices, and sorting capabilities.

● The navigation feature, which offers connections to various sites and enables simple navigation, is an essential aspect of the app's user experience. Links to the home page, menu, and checkout may also be present, along with a logo.

● The pizza component is a reusable component that renders a pizza HTML div with the options to add to the cart as well as the size, quantity, and price of the pizza. It enables simple ordering and pizza customization and may be used in both the menu and checkout sections.

● The registration feature also enables users to log in and establish an account. It may also have supporting elements like a form for entering payment information, a form for entering personal information, and a confirmation screen. Developers may combine functional and class components with React hooks like useState, useEffect, and useContext to build these components in React. For instance, the useState hook and a functional component may be used to regulate the number and size of the pizza component. When given props like the name, cost, and toppings of the pizza, the component may utilise these to render the pizza div with the necessary information. A useContext hook may be used to access the app's global state and update the user's cart to implement the add-to-cart feature. The componentDidMount and componentDidUpdate lifecycle methods may be used to manage form validation and submission when building the signup component, which can also be generated using a class component. The component may conduct HTTP queries to the app's backend and store user data in a database using a library like Axios. In conclusion, leveraging the MERN stack to build components for a food delivery service needs a blend of technical know-how, design proficiency, and user experience understanding. A flawless and delightful user experience depends

on the checkout, home, menu, navigation, pizza, and registration features. Developers may design these components quickly and effectively with the use of React's functional and class components, hooks, and libraries like axios.

4.4.2.1 React Hooks

React Hooks are an essential part of modern React development, enabling developers to manage state and lifecycle events in functional components. In this section, we will discuss how to use React hooks in a food delivery app built using the MERN stack:

- **useState Hook:**

  Utilizing the useState hook, we can include the state in functional components. It accepts an argument representing the initial state value and returns an array containing the current state value and a function to update the state value. The user's cart, menu items, and personal information can all be managed in the food delivery app by using the useState hook.

- **useLocation Hook:**

  The current location object, which includes details about the current URL, is returned by the useLocation hook. It may be used to extract data from the URL, including the current route, query arguments, and other details. The useLocation hook may be used in the food delivery app to display the relevant components based on the current route.

- **useNavigate Hook:**

  We can browse various app pages programmatically thanks to the useNavigate hook. It produces a navigate function that we may call with the route we want to travel to and takes no parameters. After the customer adds goods to their basket in the food delivery app, we can use the useNavigate hook to browse to the checkout page.

4.5 Express.js

Express.js is a powerful Node.js web application framework that is used to build the backend of web apps like the meal delivery service. It is a popular choice for designing APIs due to its simplicity and adaptability. Express.js is a simple API that allows developers to create routes, process HTTP requests and responses, and specify middleware. In the food delivery business, Express.js is utilised to provide a RESTful API that communicates with the front-end application built with React.js. The API handles client-side requests, communicates with the database, and returns information to the front end. It allows the programme to be scalable, efficient, and easy to maintain.

Fig.4.4.3 Express.js

Express.js relies heavily on middleware. Middleware functions are those that execute between the request and the response in an application's request-response cycle. They can be used for a variety of purposes, such as logging, authentication, and error management. In the food delivery application, middleware is used to manage authentication, validate input data, and handle errors. Another important aspect of Express.js is its ability to execute HTTP requests. It supports all HTTP methods, including GET, POST, PUT, and DELETE, providing it with extensive functionality. In the food delivery application, GET

queries are used to retrieve data from the database, POST requests are used to produce new data, PUT requests are used to update data, and destroy requests are used to delete data. Overall, Express.js is a strong and versatile online application framework that is excellent for building the backend of web apps like the food delivery service. It is a popular choice among developers because of its ease of use, scalability, and flexibility.

4.6 Node.js

A prominent server-side JavaScript runtime environment for building scalable and fast web applications is Node.js. It is an open-source platform built on the V8 JavaScript engine found in Google Chrome. Node.js's lightweight architecture and quick performance make it a popular choice for building web apps.

The food delivery service uses Express.js, one of the many modules and frameworks found in the Node.js ecosystem. Express.js is a web framework for Node.js that provides a straightforward and flexible API for creating online apps and APIs. It contains several capabilities that make it easier to construct scalable and modular applications, including routing, middleware, and template engines. One of the main advantages of using Node.js in the food delivery business is its ability to handle several concurrent connections. Because Node.js uses event-driven, non-blocking I/O, it can handle several requests concurrently without overloading the main event loop. It is hence suitable for real-time applications that need high concurrency and low latency.

In conclusion, Node.js is a well-liked framework for creating performant and scalable internet applications. It is a great choice for food delivery systems due to its speed, lightweight design, and a wide ecosystem of libraries and frameworks. Due to its event-driven, non-blocking I/O architecture, Node.js can support a large number of concurrent connections, making it suitable for real-time applications

4.6.1 Hyper Text Transfer Protocol

The application protocol known as HTTP, or Hypertext Transfer Protocol, is used to communicate between web servers and clients. The World Wide Web's data transmission is built on this basis. HTTP specifies the structure and transmission of messages as well as the responses that web servers and browsers should provide to various requests. An HTTP request typically consists of a client request and a server response. The server receives an HTTP request from the client and sends back an HTTP response.

Headers and a message body are both parts of the request and response messages. The headers include details about the request or response, including the kind of material being delivered and how much of it there is. The actual data being communicated is included in the message body. Because HTTP has a stateless client-server architecture, the server keeps no record of the client's prior requests.

Every request is handled separately, and the server answers each one using just the details sent in that particular request. 28 There are various variations of HTTP, the most popular being HTTP/1.1 and HTTP/2. A frequently used online communication protocol is HTTP/1.1.

It employs a request/response architecture and is a text-based protocol. The most recent version of HTTP, HTTP/2, is intended to enhance the functionality of online applications. It supports multiplexed streams and employs binary communication rather than text-based communication, enabling simultaneous transmission and reception of numerous requests.

The widespread usage of HTTP, a crucial element of the contemporary internet, has facilitated the creation of sophisticated online apps. Online shopping, social media, video streaming, and web surfing are just a few of the many uses for it. HTTP is anticipated to continue to be a crucial part of online communication and data transfer as the web develops.

4.6.2 HTTP vs HTTPS

Two protocols are used to transfer data over the internet: HTTP (Hypertext Transfer Protocol) and HTTPS (HTTP Secure). The degree of security each offers is the main distinction between them.

A common application protocol for sending data over the internet is HTTP. It establishes the format and transmission of messages on top of the TCP/IP protocol. Since HTTP is an unsafe protocol, data sent via it can be intercepted by hackers since it is not encrypted.

Contrarily, HTTPS is a more secure variation of HTTP. Data exchanged between a web server and a client using HTTPS is encrypted using SSL/TLS (Secure Sockets Layer/Transport Layer Security). Data in transit is encrypted using the SSL/TLS protocol, making it more challenging for hackers to intercept and steal sensitive data. The safe and private transmission of data between the client and the server is made possible by the use of encryption in HTTPS.

This is crucial when sending sensitive data like login credentials, credit card numbers, or other private information. The port that HTTP and HTTPS employ is another distinction between them. While HTTPS uses port 443, HTTP uses port 80. Depending on whether a website utilises HTTP or HTTPS, your browser will automatically add the proper protocol and port number when you enter a URL.

Using HTTPS has security advantages, but it can also help a website's search engine results. According to Google, HTTPS is a ranking element, and websites that utilise it can see a little improvement in search engine ranks. In conclusion, HTTPS is a safer variation of HTTP that use encryption to safeguard data transferred between the client and the server. Although HTTP is still extensively used, HTTPS is growing in popularity as websites try to increase security and safeguard private data.

4.6.2 HTTP Header and Body

It is specified by the HTTP (Hypertext Transfer Protocol) protocol how information is sent between a web server and a client, such as a web browser. Data is sent and received

via the HTTP protocol whenever a client asks a server for a resource. The header and the body are the two fundamental components of the HTTP protocol. While the actual data being communicated is included in the body, the header includes metadata about the request or response.

- HTTP Header: An HTTP request or response's initial component, the HTTP header, has several fields that characterise the message being transmitted. Request headers and response headers are the two categories into which headers may be separated

4.7 Front-end Modules

1. React Router Dom:

A routing module for React applications called React Router Dom v6 offers a declarative approach to move between various components based on the URL. In comparison to earlier versions, it offers a simpler and more user-friendly API. We utilised React Router Dom v6 to manage the client-side application's routing in our food delivery application. Routes and BrowserRouter are two components from the package that let us map particular URLs to particular components. The usage of the Routes component rather than the Switch component is one of the primary differences with v6. The Route component, which accepts a path prop and a component prop to map the path to a particular component, allows us to create routes using the Routes component. More flexibility and route nesting are made possible by this. Image 6: React-Router-Dom 37 UseNavigate hook is a further new functionality in version 6. This hook offers a programmatic means of choosing an alternative path. Based on user activities, we used this hook to direct users to various components. Additionally, v6 also introduces the useSearchParams hook, which allows us to easily access and manipulate URL query parameters.

CHAPTER 5

DATABASE DESIGN

5.1 Flowcharts

A flowchart is a visual representation of the steps in a process, commonly used by programmers to plan and illustrate algorithms. Here's a flowchart for a food ordering website, incorporating key symbols:

Basic Symbols Used in Flowchart Designs:

Start:

- The flowchart begins with the start symbol.

User Registration and Login:

- Users are prompted to register or log in to the food ordering website.
- If registered, users can proceed to log in.
- If not registered, they can complete the registration process.

Home Screen:

- After successful login, users are directed to the home screen.
- The home screen provides an overview of available restaurants, ongoing promotions, and navigation options.

Menu Selection and Customization:

- Users can explore the restaurant menu and select items for their order.
- Customization options such as quantity, size, and additional preferences are available
- Users can add items to their cart before proceeding to checkout.

Menu Selection and Customization:

- Users can explore the restaurant menu and select items for their order.
- Customization options such as quantity, size, and additional preferences are available
- Users can add items to their cart before proceeding to checkout.

Menu Selection and Customization:

- Users can explore the restaurant menu and select items for their order.
- Customization options such as quantity, size, and additional preferences are available
- Users can add items to their cart before proceeding to checkout.

Menu Selection and Customization:

- Users can explore the restaurant menu and select items for their order.
- Customization options such as quantity, size, and additional preferences are available.
- Users can add items to their cart before proceeding to checkout.

Order Checkout:

- Users can review their selected items in the cart.

- They provide delivery details, select payment methods, and apply any discounts or promo codes.
- Users confirm the order and proceed to payment.

Payment Processing:

- The website processes the payment securely.
- Users receive confirmation of the successful transaction

5.2 Data Flow

This data flow diagram depicts a restaurant ordering system, illustrating how information travels from customer to kitchen. Customers place orders, selecting items from the offered menu. Menu items are categorized for easy browsing, and each order contains specific items chosen by the customer. Essentially, the diagram maps the journey of data, ensuring smooth communication and order fulfillment between restaurant and customer

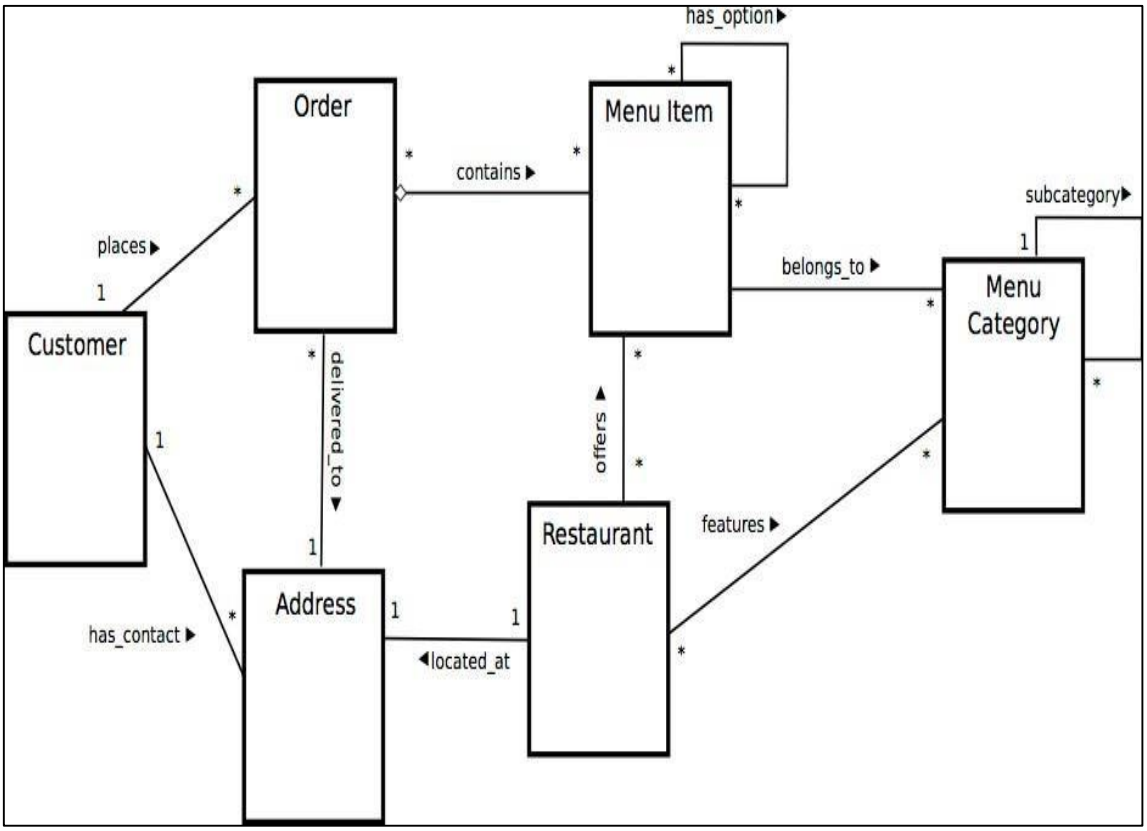

Fig. 5.2: Data Flow

CHAPTER 6

RESULTS

6.1 Home page

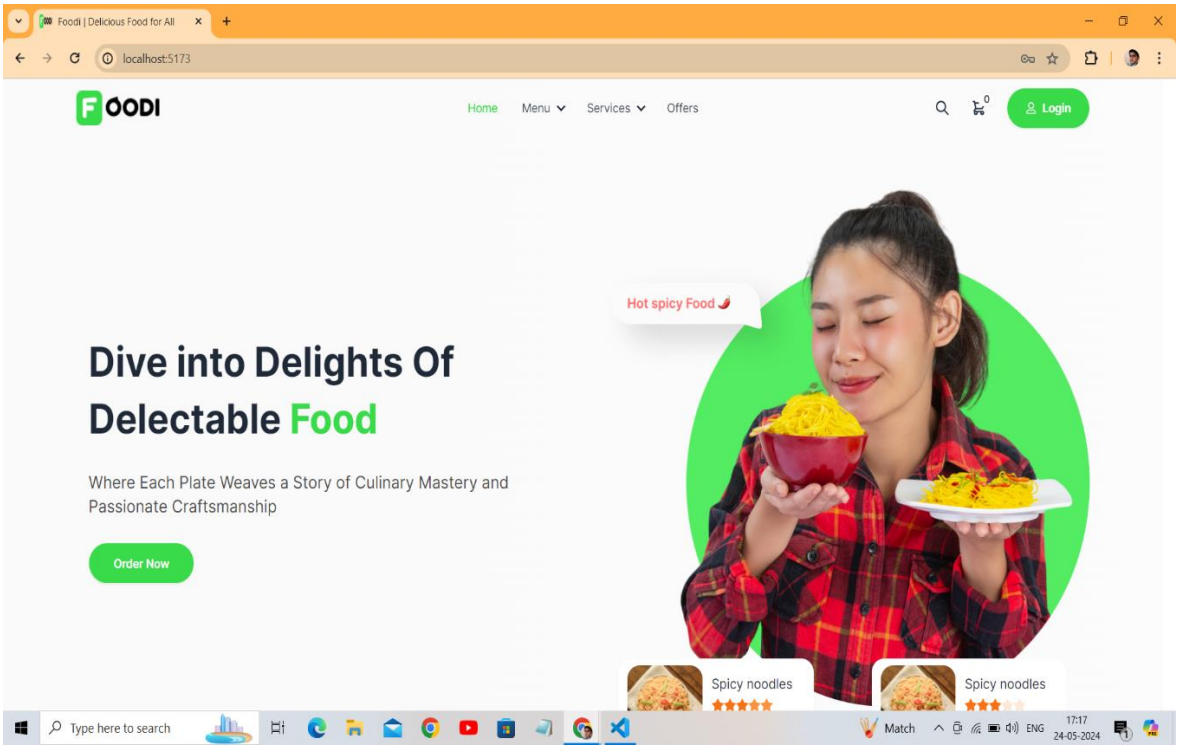

Fig. 5.1: Home Page Screenshot

6.2 Category Page

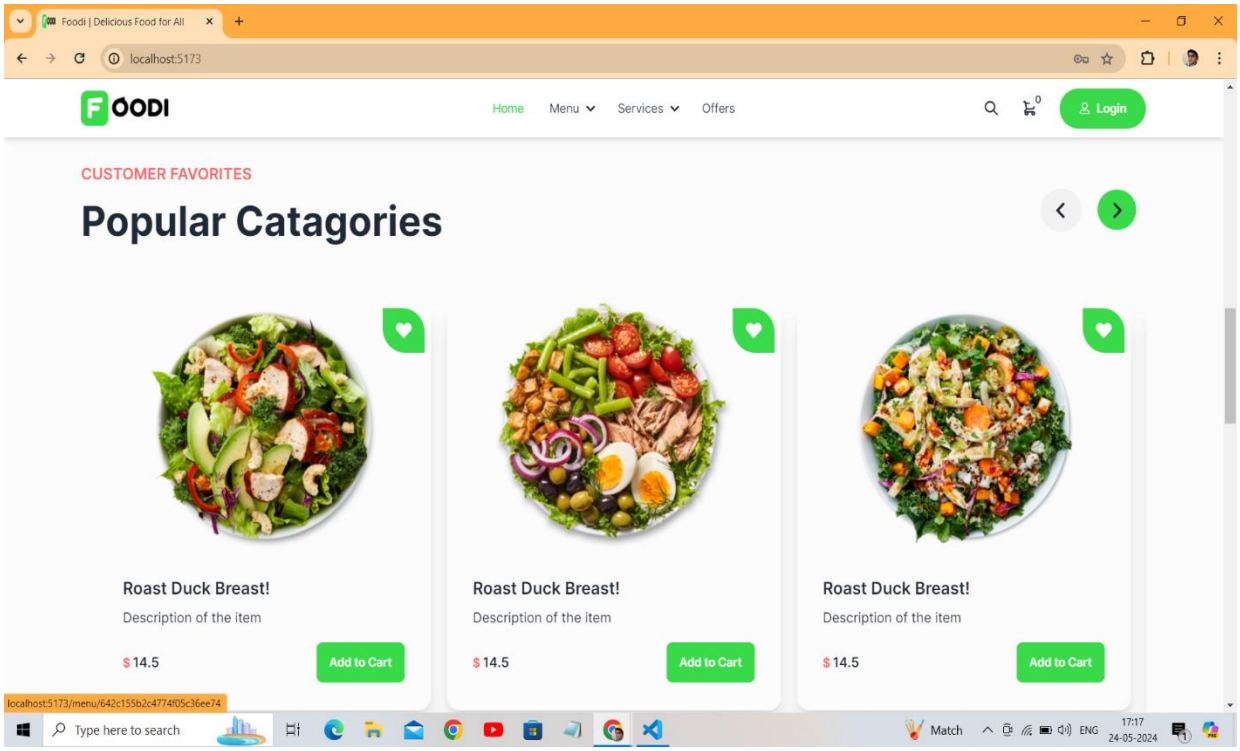

Fig 5.2: Category Page Screenshot

6.3 Admin Model

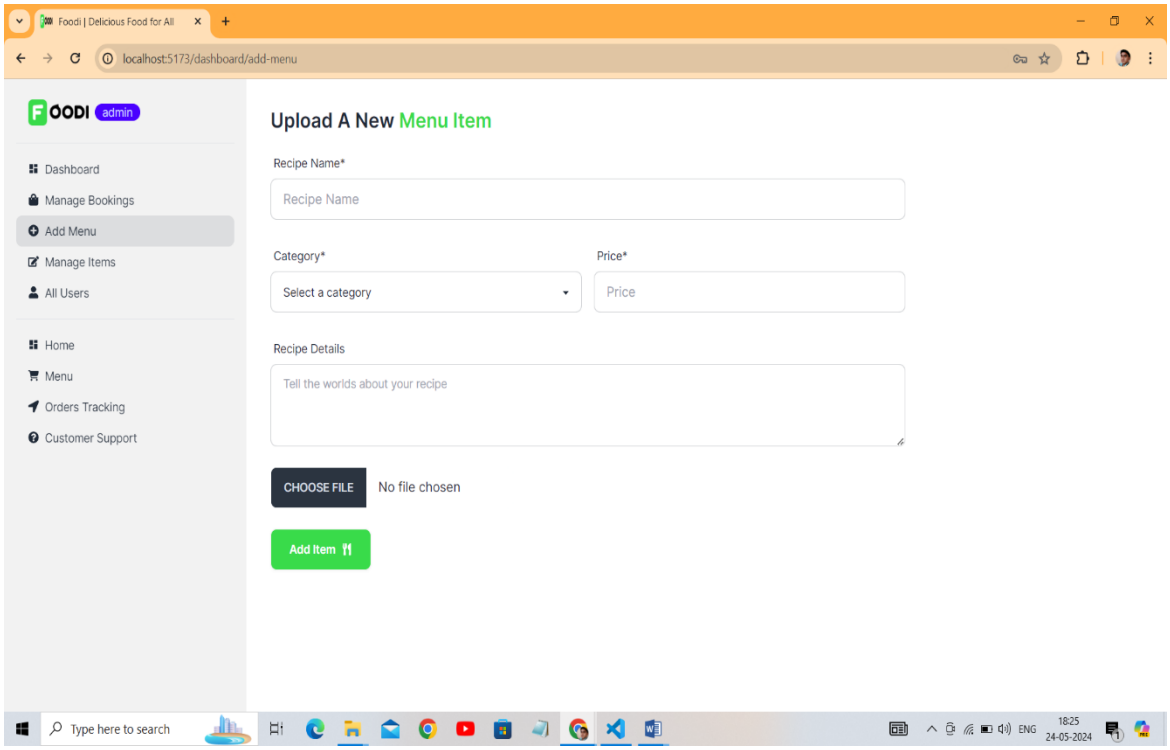

Fig 6.3: Admin Module to Update Menu

6.3.1 Edit User by Admin

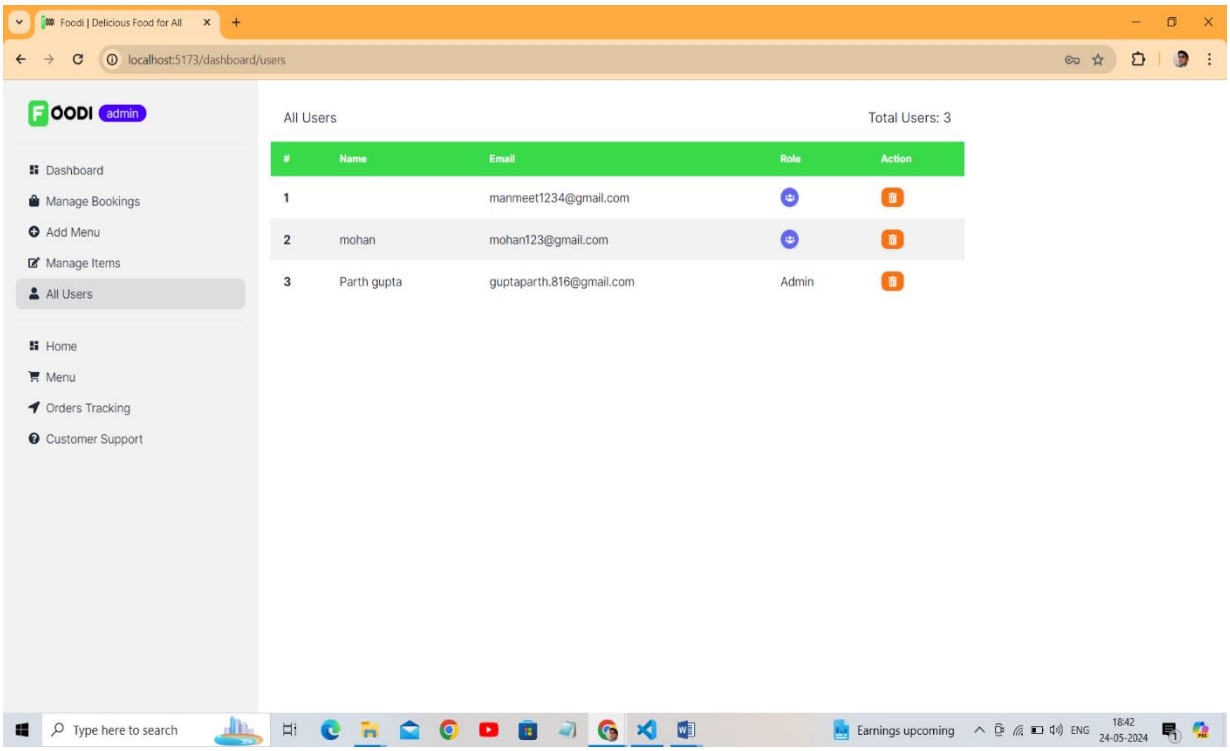

Fig.6.3.1: Edit Users

6.3.2 Manages Food Items

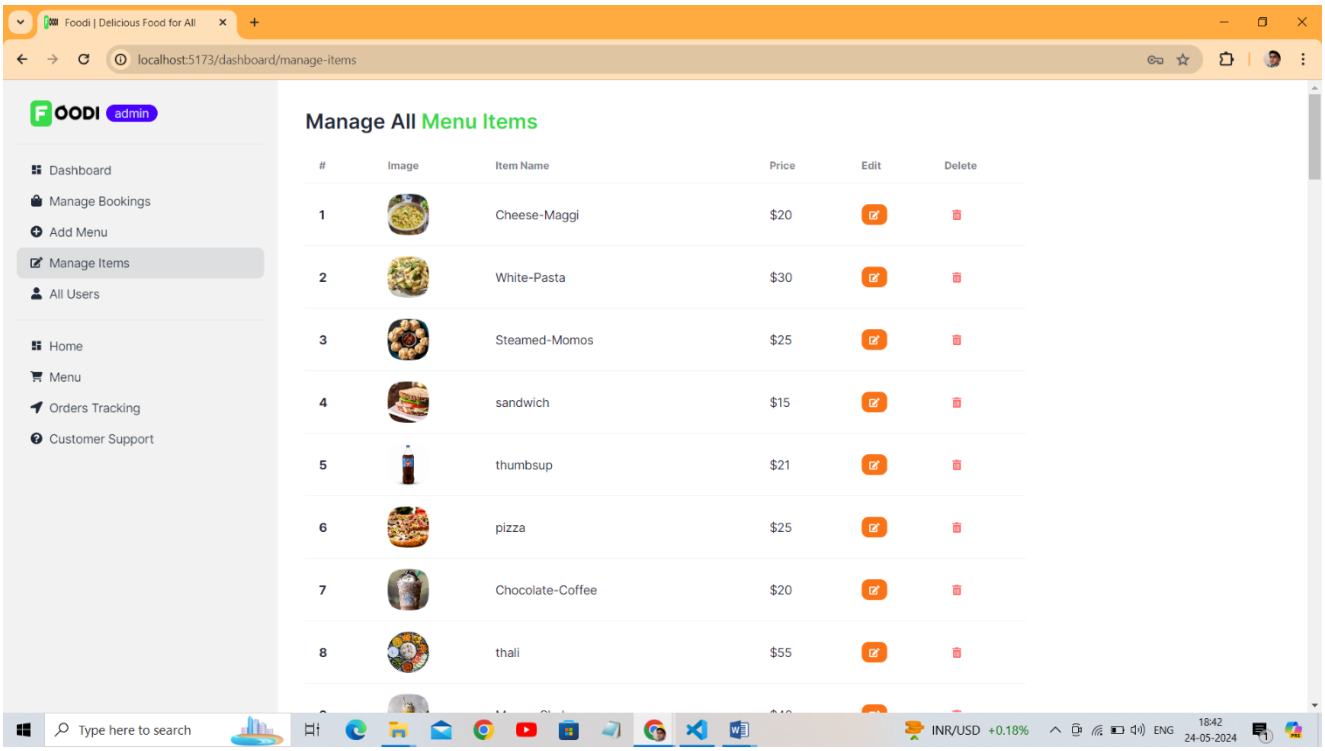

Fig.6.3.2: Manages Food Items by Admin

6.4 Feedback and Review

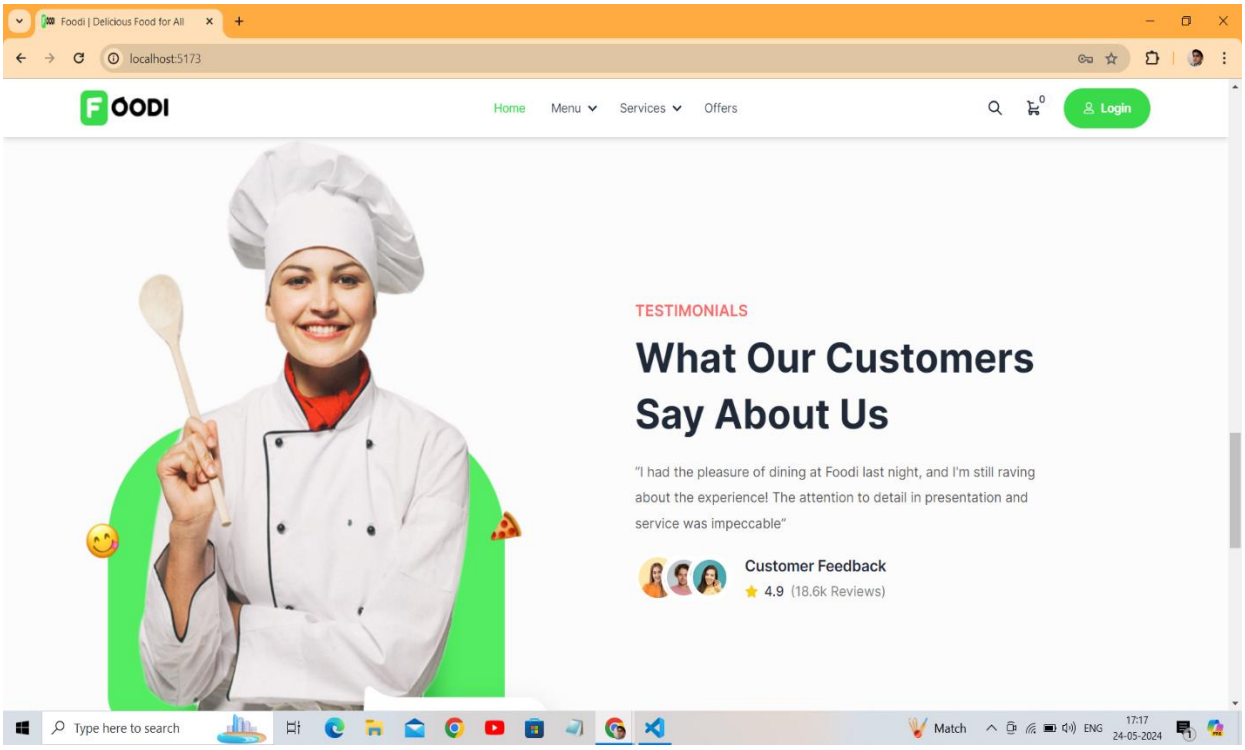

Fig 5.4: feedback Screenshot

6.5. Log In Page

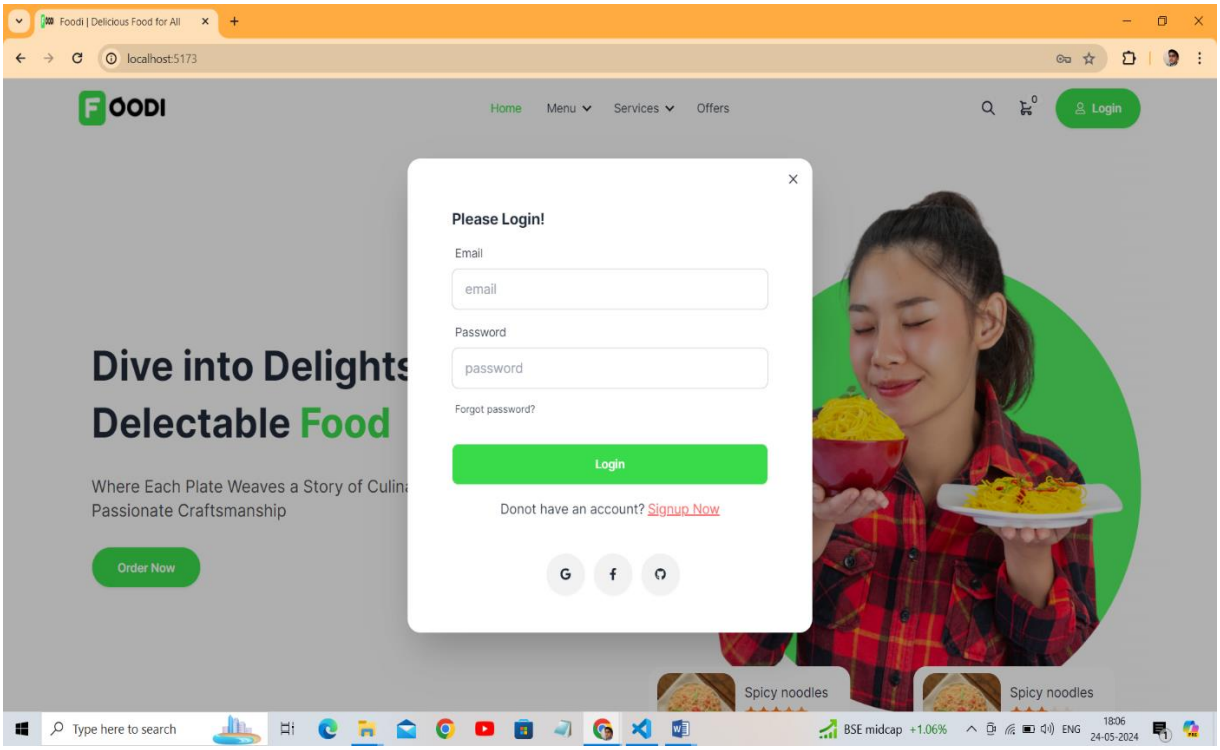

Fig 5.5: Log In Page Screenshot

6.6 Sign Up Page

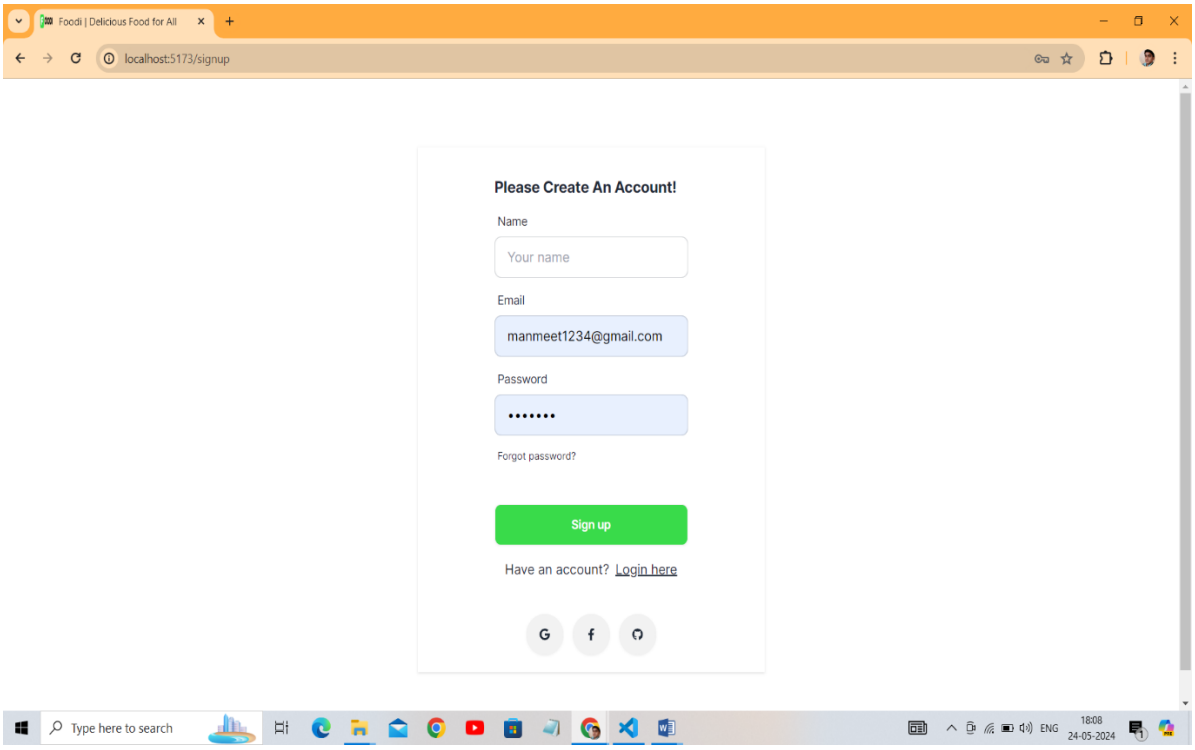

Fig 5.6: Sign Up Page Screenshot

6.7 Cart Page

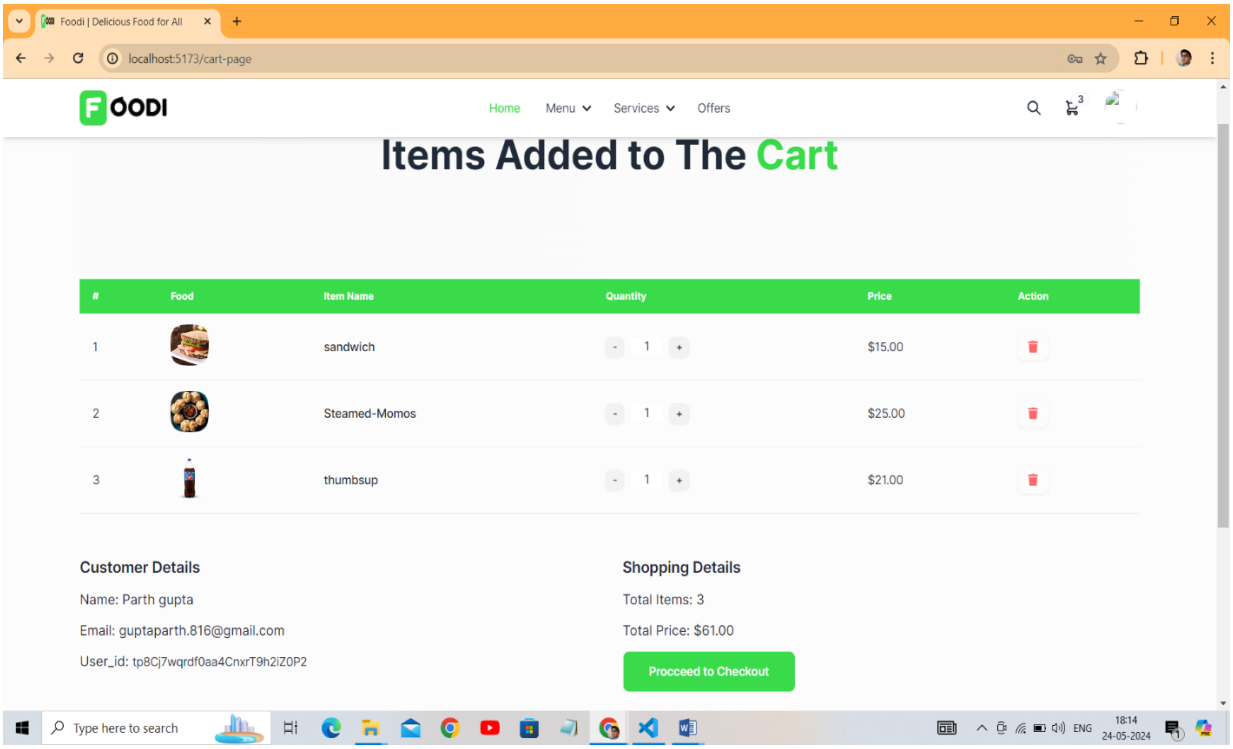

Fig 5.7: Cart Page Screenshot

6.8 Payment Page

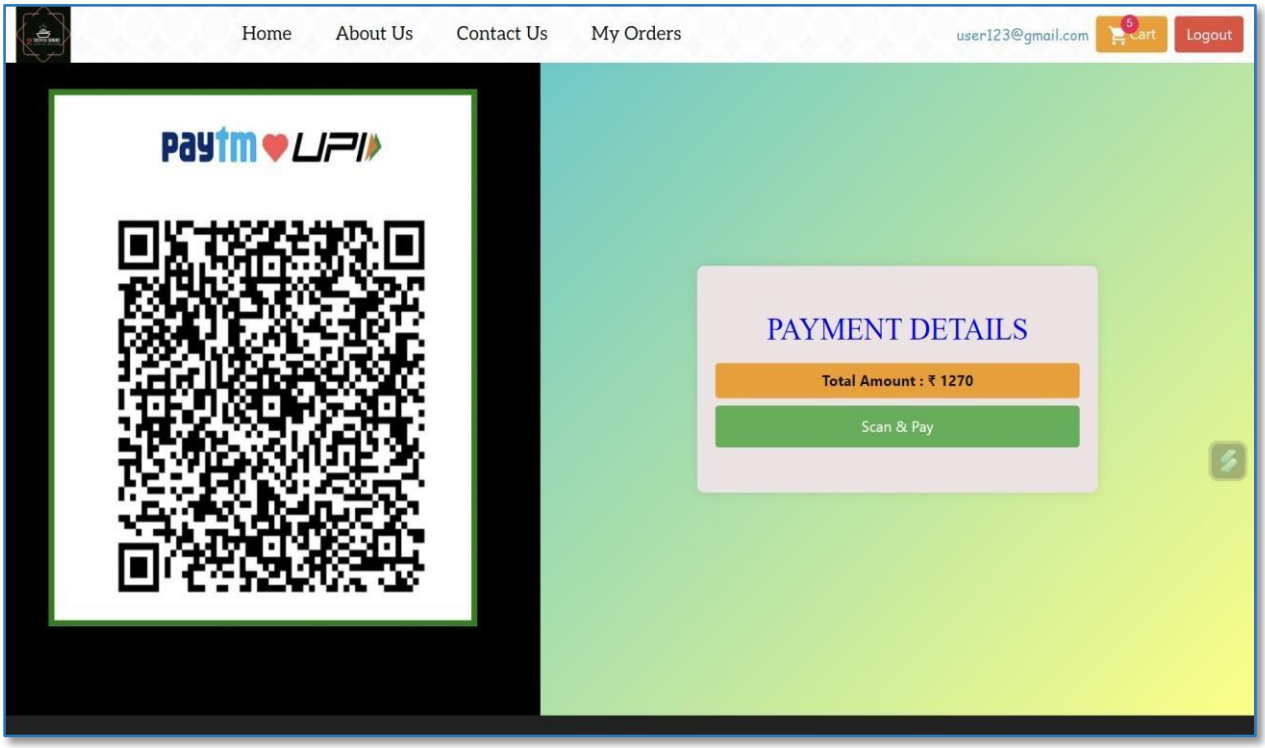

Fig 5.9: Payment Page Screenshot

CHAPTER 7

FLOW CHART / E-R DIAGRAM

7.1 Flow Chart

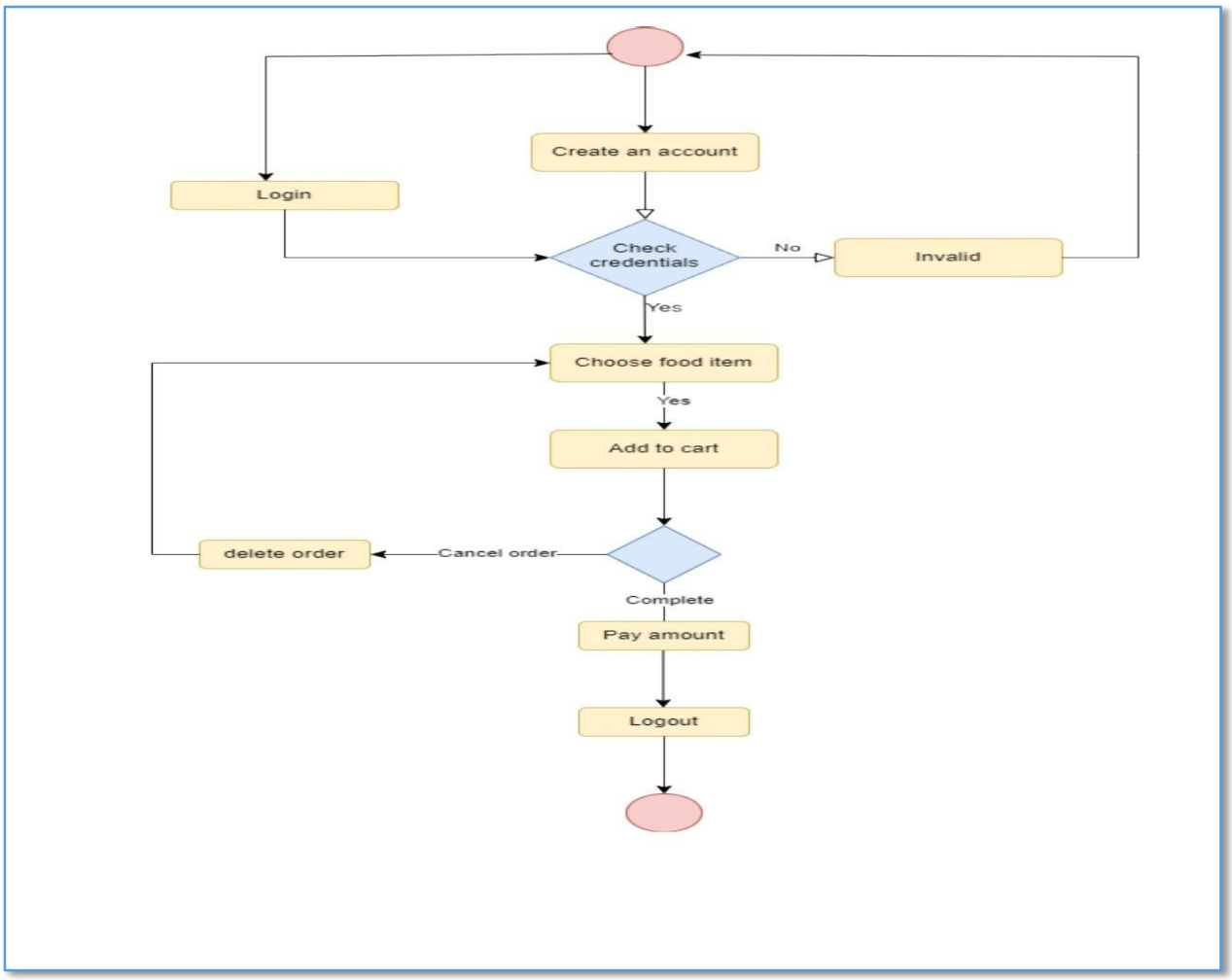

Fig 7.1: Flow Chart

6.1 ER-Diagram

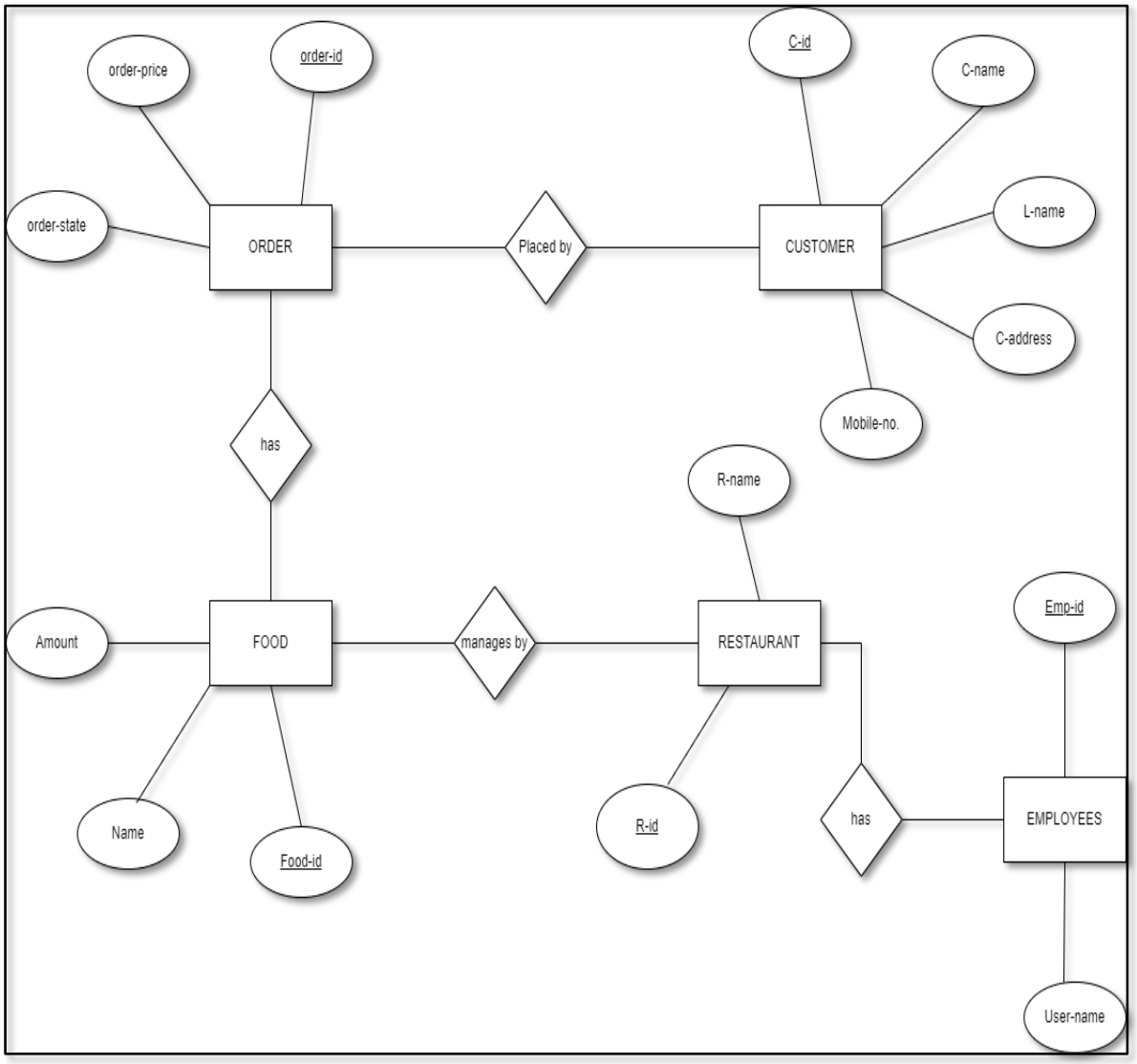

Fig 7.2: E-R Diagram

7.3 Working Flow

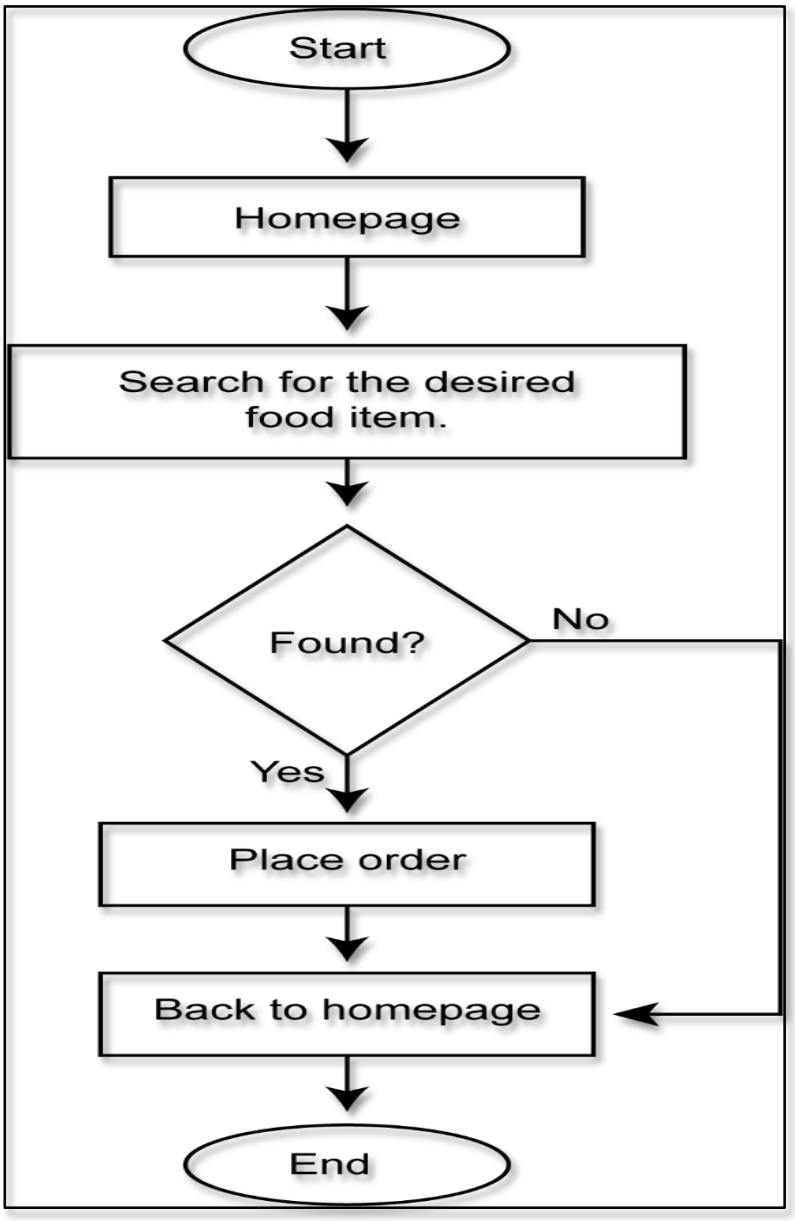

Fig 7.3: Working Flow

7.4. DFD for Admin:

Process User goes to home page of the domain. If he/she has an account then he/she can login in restaurant management system otherwise he/she need to register an account after successful registration, they can login in home page.

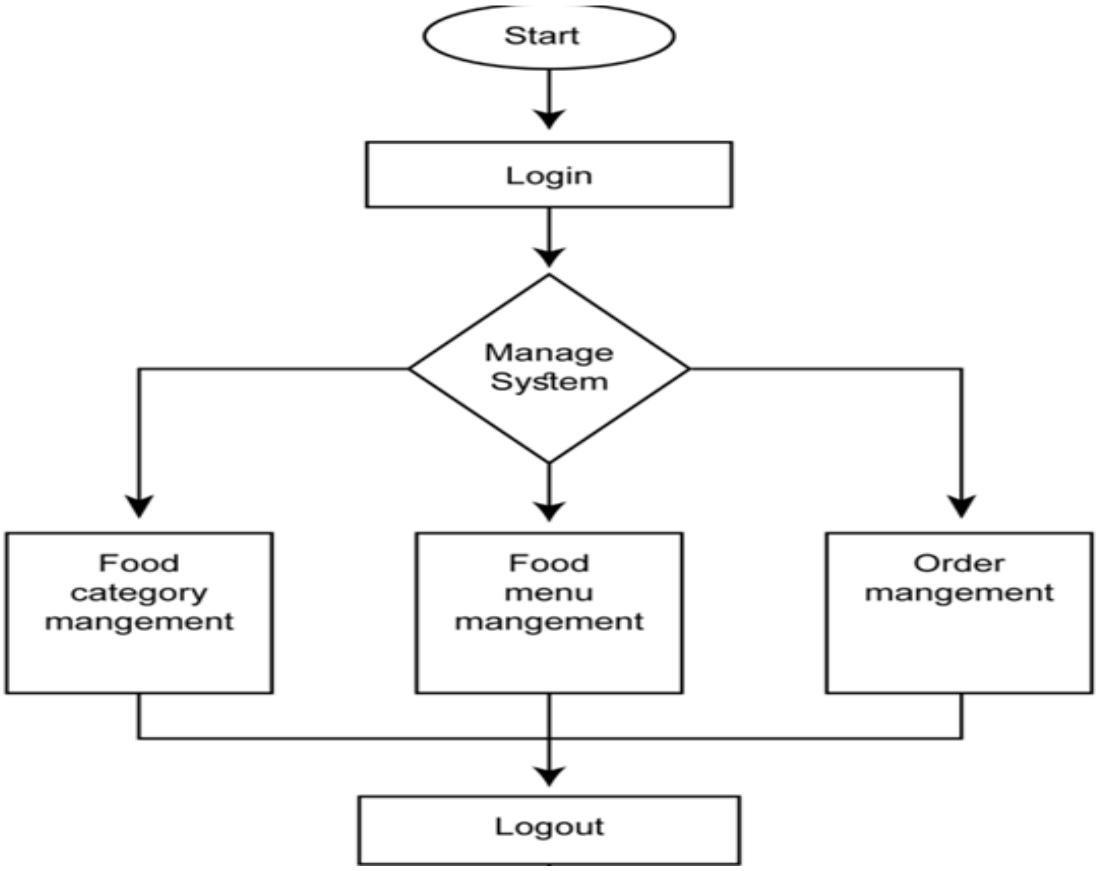

Fig.6.4: Admin DFD

6.5. User Flow

Process Initially to visit the food categories or food menu, users don’t need to

login/register an account. After checking out the categories and menu items, if the user finds his/her desired menu and if they want to order that particular item they can go to order page. During placing any order the customer needs to provide his/her required information mentioned the order section.

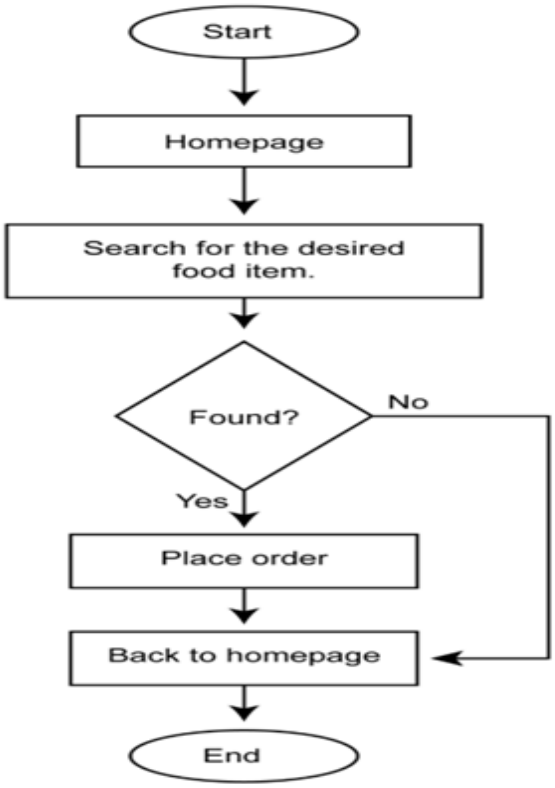

Fig.6.5:User DFD

CHAPTER 8

TESTING

Various tests were carried out throughout the development of the food delivery application utilising the MERN stack to verify the system's functionality, performance, and dependability. Analytical and experimental methodologies were used in the testing. Analytical testing entailed evaluating the codebase, detecting possible problems, and testing individual system components. Computational and mathematical tools were utilised to model situations and identify potential system faults. The data obtained throughout the testing phase was analysed using statistical methods. Setting up a test bed that simulated a real-world scenario was required for experimental testing. Setting up a server environment, validating network connections, and creating a database to store and retrieve data were all part of the process. The system was then put through its paces by inputting different data and checking the output at various stages.

8.1 Unit Testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or

procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but it could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist in testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

8.1.1 Benefits of Unit Testing

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it offers several benefits.

Find Problems Early:

Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build.

The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

Facilitates Change:

Unit testing allows the programmer to refactor code or upgrade system libraries later, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes that

may break a design contract.

Simplifies Integration:

Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

- **Documentation:**

Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in the development unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

8.2 Integration Testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of integration testing is to verify the functional, performance, and reliability requirements placed on major design items. These "design items", i.e.,

assemblages (or groups of units), are exercised through their interfaces using black-box testing, with success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. Some different types of integration testing are big-bang, top- down, and bottom-up, mixed (sandwich) and risky- hardest. Other Integration Patterns are collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

8.2.1 Big Bang

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing because it expects to have few problems with the individual components.

The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of

the components in the environment. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

8.2.2 Top-Down and Bottom-Up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower-level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready.

This method also helps to determine the levels of software developed and makes it easier to

Report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing where the top integrated modules are tested, and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top-down testing with bottom-up testing.

8.2.2 Black-Box Testing

Black box testing is a technique used to test the functionality of a software application without having knowledge of its internal structure or implementation details. It focuses on the inputs and outputs of the system and verifies if the expected outputs match the desired results. Here are some examples of black box testing techniques that can be applied to the KIET Event Management App:

Equivalence Partitioning:

- Identify different categories of inputs for the app, such as valid and invalid inputs, and divide them into equivalence classes.
- Test representative values from each equivalence class to ensure the app behaves consistently within each class.

Boundary Value Analysis:

- Identify the boundaries or limits for inputs in the app, such as minimum and maximum values, and test values at those boundaries.
- Test values just above and below the boundaries to verify the app's behaviour at critical points.

Decision Table Testing:

- Identify the different conditions and rules that govern the behaviour of the app.
- Create a decision table with combinations of conditions and corresponding expected results.
- Test different combinations of conditions to validate the app's decision-making process.

State Transition Testing:

- Identify the different states that the app can transition between.
- Define the valid and invalid transitions between states.
- Test different sequences of state transitions to verify the app's behaviour.

Error Guessing:

- Use experience and intuition to guess potential errors or issues in the app.
- Create test cases based on those guesses to verify if the app handles the errors correctly.

Compatibility Testing:

- Test the app on different platforms, browsers, or devices to ensure compatibility.
- Verify that the app functions correctly and displays appropriately across different environments.

Usability Testing:

- Evaluate the app's user interface and interactions from the perspective of an end-user.
- Test common user scenarios and assess the app's ease of use, intuitiveness, and overall user experience.

Security Testing:

- Identify different categories of inputs for the app, such as valid and invalid inputs, and divide them into equivalence classes.
- Test representative values from each equivalence class to ensure the app behaves consistently within each class.

Performance Testing:

- Test the app's performance under different load conditions, such as a high number of concurrent users or large data sets.
- Verify if the app responds within acceptable time limits and performs efficiently.

During black box testing, test cases are designed based on the app's specifications, requirements, and user expectations. The focus is on validating the functionality, user interactions, and expected outputs without considering the internal implementation details.

8.3 White-Box Testing

White box testing, also known as structural testing or glass box testing, is a software testing technique that examines the internal structure and implementation details of the application. It aims to ensure that the code functions as intended and covers all possible execution paths. Here are some examples of white box testing techniques that can be applied to the KIET Event Management App:

Unit Testing:

This testing is performed to validate the application's different components. This would entail testing the React components, Express routes, and MongoDB queries in a MERN stack. Unit testing results should be labelled as pass or fail.

- Test individual units or components of the app, such as functions or methods, to verify their correctness.
- Use techniques like code coverage analysis (e.g., statement coverage, branch coverage) to **ensure** that all code paths are exercised.

Integration Testing:

This testing is performed to validate the interaction of the application's various components. This would entail testing the interaction between React and Express, Express and MongoDB, and React and MongoDB in a MERN stack. Integration testing results should be labelled as pass or fail.

- Test the interaction between different components or modules of the app to ensure they work together seamlessly.
- Verify the flow of data and control between the modules and check for any integration issues or errors.

Path Testing:

- Identify and test different paths or execution flows through the app, including both positive and negative scenarios.
- Execute test cases that cover all possible paths within the code to

ensure complete coverage.

Decision Coverage:

- Ensure that every decision point in the code (e.g., if statements, switch cases) is tested for both true and false conditions.
- Validate that the app makes the correct decisions based on the specified conditions.

Code Review:

- Analyse the code and its structure to identify any potential issues or vulnerabilities.
- Review the adherence to coding standards, best practices, and potential optimizations.

Performance Testing:

- Assess the app's performance from a code perspective, such as identifying any bottlenecks or inefficient algorithms.Measure the execution time of critical code sections and evaluate resource usage.

Security Testing:

- Review the code for potential security vulnerabilities, such as SQL injection, cross-site scripting (XSS), or authentication weaknesses.
- Verify the implementation of secure coding practices, data encryption, and access control mechanisms.

Error Handling Testing:

- Test how the app handles and recovers from unexpected errors or exceptions.
- Validate that error messages are clear, meaningful, and do not expose sensitive information.

Code Coverage Analysis:

- Use tools to measure the code coverage achieved by the tests, such as statement coverage, branch coverage, or path coverage.
- Aim for high code coverage to ensure that all parts of the code are exercised.

During white box testing, the tester has access to the application's internal code, allowing for a more detailed examination of its behaviour.

8.4 System Testing

System testing is a level of software testing that evaluates the complete system as a whole, rather than focusing on individual components or modules. It ensures that all components of the KIET Event Management App work together seamlessly and meet the specified requirements. Here are some examples of system testing techniques that can be applied to the app:

This testing guarantees that the application complies with the requirements and operates as planned in a practical environment. This would require assessing a meal delivery app's ordering procedure, delivery tracking, and payment processing. The results of system testing ought to be classified as passes, fails, or partial passes.

Functional Testing:

- Verify that all functional requirements of the app are met.
- Test various functionalities such as event creation, registration, club directory search, user log in and registration, event notifications, etc.
- Validate that the app behaves as expected and produces the correct outputs based on different inputs.

User Interface Testing:

- Test the graphical user interface (GUI) of the app for usability, consistency, and responsiveness.

- Check the layout, navigation, buttons, forms, and other UI elements to ensure they are visually appealing and intuitive.
- Validate that the app adheres to the design guidelines and provides a seamless user experience.

Performance Testing:

This testing makes that the programme can handle the expected load and continue to operate normally under pressure. For a food delivery business, this would include assessing the app's response time, scalability, and load capacity. Results from performance tests should be categorised as pass, fail, or partial pass.

- Evaluate the performance of the app under different load conditions.
- Measure response times, throughput, and resource utilization to ensure the app can handle the expected user load without significant degradation.
- Identify and address any performance bottlenecks or scalability issues.

Compatibility Testing:

- Test the app on different devices, platforms, and browsers to ensure compatibility.
- Verify that the app works correctly on various operating systems (e.g., iOS, Android) and different screen sizes.
- Validate that the app functions properly on different web browsers (if applicable).

Integration Testing:

This testing is performed to validate the interaction of the application's various components. This would entail testing the interaction between React and Express, Express and MongoDB, and React and MongoDB in a MERN stack. Integration testing results should be labelled as pass or fail

- Test the integration of the app with external systems, such as databases, mapping services, or notification services.
- Validate that data is exchanged correctly between the app and external systems.

- Verify that the app's functionality remains intact when integrated with other systems.

Recovery Testing:

- Simulate system failures or interruptions and evaluate the app's ability to recover and resume normal operation.
- Test scenarios such as unexpected shutdowns, network failures, or interrupted database connections.
- Ensure that the app can gracefully handle such situations and recover without data loss or integrity issues.

Regression Testing:

This testing ensures that any modifications made to the programme do not impair its current functioning. In the case of a meal delivery app, this would entail testing the app's basic functionalities following updates to the user interface or the addition of new features. Regression testing findings should be labelled as pass or fail. It is critical to mark the findings of each testing technique so that the development team can follow the application's progress and identify any areas that require improvement. Various testing methodologies, including unit testing, integration testing, and system testing, were used to generate and implement test cases. Iterative testing was used, with tests being created as new features were integrated and faults were discovered.

- Re-test previously tested features and functionalities to ensure that recent changes or additions did not introduce new bugs or regressions.
- Execute a set of comprehensive test cases to cover critical areas of the app and ensure that no existing functionality is compromised

CHAPTER 9

CONCLUSION

9.1 Conclusion

In conclusion, a food ordering website represents a modern and efficient solution for both customers and businesses in the food industry. Through its user-friendly interface, seamless ordering process, and convenient delivery options, it enhances the overall dining experience by providing convenience, choice, and accessibility. For customers, it offers the convenience of browsing menus, placing orders, and tracking deliveries from the comfort of their homes or on the go. For restaurants and food establishments, it opens up new avenues for reaching a wider audience, increasing sales, and streamlining operations.

Moreover, a well-designed food ordering website can foster customer loyalty through personalized recommendations, promotions, and loyalty programs. By leveraging data analytics and customer feedback, businesses can continually improve their services and tailor offerings to meet evolving preferences and demands. However, the success of a food ordering website hinges on various factors, including effective marketing strategies, robust technology infrastructure, and responsive customer support. ensuring food safety, quality control, and timely delivery are paramount to building trust and credibility among customers.

As technology continues to evolve and consumer expectations evolve, food ordering websites must remain agile and adaptable to stay competitive in the ever-changing landscape of the food industry. By embracing innovation, staying attuned to customer needs, and delivering exceptional experiences, food ordering websites can continue to revolutionize the way people dine and interact with food in the digital age.

Finally, the creation of a food delivery application utilising the MERN stack resulted in a functioning and efficient system for ordering and delivering meals. The software allows users to explore menus and place orders thoroughly. while the restaurant can handle orders and change menu items in real-time. Planning, design, execution, and testing were all part of the development process. The needs were identified and analysed during the planning stage, and a thorough project plan was prepared.

Wireframes, mockups, and a database structure were all created throughout the design stage. The MERN stack was used for implementation, which provided a robust and versatile set of tools for creating both the application's front-end and back-end. A set of unit, integration, and acceptance tests was performed throughout the testing stage to confirm that the system fulfilled all criteria and was free of faults and mistakes.

The adoption of MongoDB as the application's database was especially advantageous since it allowed for quick data storage and retrieval and provided a scalable solution that could manage massive volumes of data. Furthermore, using ReactJS as the frontend framework enabled a dynamic and interactive user experience with rapid rendering and enhanced speed.

Various testing methodologies, including unit, integration, and acceptability testing, were used to test the application. These tests were performed to confirm that the system satisfied all functional and non-functional criteria and was bug and

error-free. Several difficulties were discovered throughout the testing phase, which were addressed and resolved in future rounds of the development process. Overall, the testing method was critical in guaranteeing the system's quality and dependability.

9.2 Future Scope

Several areas for future work might be highlighted for improvement. The app, for example, may be upgraded to incorporate other features such as a rating system for restaurants and drivers, or the ability to follow the delivery truck in real-time and track delivery status. 47 Furthermore, the app's speed and scalability might be improved by incorporating caching systems or load-balancing approaches. Finally, the creation of a food delivery application utilising the MERN stack resulted in a functioning and efficient system that fits the demands of both users and restaurants. With a user-friendly design and real-time delivery progress information, the programme provides a quick and simplified method to order and transport meals. Modern technologies like MongoDB, ReactJS, and NodeJS were used to provide a scalable and adaptable solution that can be easily customised and expanded in the future. Overall, the creation of this application highlighted the MERN stack's strength and diversity, as well as its potential to produce unique and meaningful solutions for a variety of sectors.

Integration of Artificial Intelligence (AI):

- Implement AI-powered chatbots for personalized customer assistance, order recommendations, and troubleshooting.
- Utilize machine learning algorithms to analyze user data and predict ordering patterns, enabling targeted marketing and promotional campaigns.

Enhanced User Experience (UX):

- Explore augmented reality (AR) and virtual reality (VR) technologies to

create immersive dining experiences, allowing customers to preview menu items and restaurant ambiance before placing orders.

- Focus on intuitive and responsive design principles to optimize website usability across various devices and platforms.

Expansion Of Delivery Options:

- Partner with autonomous delivery services and drones for faster and more efficient order deliveries, especially in urban areas with heavy traffic.
- Explore alternative delivery methods such as bike couriers or crowd-sourced delivery networks to expand reach and reduce environmental impact.

Focus On Sustainability and Health:

- Incorporate features highlighting sustainable sourcing practices, eco-friendly packaging options, and nutritional information to cater to environmentally conscious and health- conscious consumers.
- Collaborate with local farmers and producers to offer seasonal and organic menu options

REFERENCES

[1] Admin and V. all posts by Admin, "How to Send Data from React to Node js Express + MySQL - Tuts Make," Tuts Make, Oct. 30, 2022. [Online]. Available: https://www.tutsmake.com/how-to-send-data-from-react-to-node-js-express-mysql/

[2] "What is the meaning of 'bodyParser.urlencoded({ extended: true }))' and 'bodyParser.json()' in Express.js?," Stack Overflow, Apr. 07, 2019. [Online]. Available: https://stackoverflow.com/questions/55558402/what-is-the-meaning-of-bodyparser-urlencode d-extended-true-and-bodypar

[3] A. Sen, "Node React Tutorial - How to connect React with backend Node.js?," codedamn news, Sep. 14, 2022. [Online]. Available: https://codedamn.com/news/reactjs/how-to-connect-react-with-node-js

[4] "What does `app.use(bodyParser.json())` do?," Stack Overflow, Oct. 05, 2016. [Online]. Available: https://stackoverflow.com/questions/39870867/what-does-app-usebodyparser-json-do

[5] N. Makarevich and @adevnadia, "How to fetch data in React with performance in mind," How to fetch data in React with performance in mind, Oct. 06, 2022. [Online]. Available: https://www.developerway.com/posts/how-to-fetch-data-in-react

[6] "Express.js Post - javaTpoint," www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/expressjs-post

[7] "How to return values from async functions using async-await from function?," Stack Overflow, Apr. 20, 2018. [Online]. Available: https://stackoverflow.com/questions/49938266/how-to-return-values-from-async-functions-us ing-async-await-from-function

[8] "Express.js And MongoDB REST API Tutorial," MongoDB. [Online]. Available: https://www.mongodb.com/languages/express-mongodb-rest-api-tutorial

[9] "try/catch blocks with async/await," Stack Overflow, Nov. 30, 2016. [Online]. Available: https://stackoverflow.com/questions/40884153/try-catch-blocks-with-async-await 49

[10] J. Bodnar, "Axios tutorial - GET/POST requests in JavaScript with Axios," Axios tutorial - GET/POST requests in JavaScript with Axios. [Online]. Available: https://zetcode.com/javascript/axios/

[11] "How to create dynamic values and objects in JavaScript ? - GeeksforGeeks," GeeksforGeeks, Feb. 22, 2021. [Online]. Available: https://www.geeksforgeeks.org/how-to-create-dynamic-values-and-objects-in-javascript/

[12] "How To Remove a Property from a JavaScript Object," How To

Remove a Property from a JavaScript Object. [Online]. Available: https://www.w3schools.com/howto/howto_js_remove_property_object.asp

[13] "How do I test for an empty JavaScript object?," Stack Overflow, Mar. 25, 2009. [Online]. Available: https://stackoverflow.com/questions/679915/how-do-i-test-for-an-empty-javascript-object

[14] "Loop Through an Object in JavaScript – How to Iterate Over an Object in JS," freeCodeCamp.org, Jul. 20, 2022. [Online]. Available: https://www.freecodecamp.org/news/how-to-iterate-over-objects-in-javascript/

[15] "BrowserRouter causing Invalid hook call. Hooks can only be called inside of the body of a function component," Stack Overflow, Dec. 24, 2021. [Online]. Available: https://stackoverflow.com/questions/70474837/browserrouter-causing-invalid-hook-call-hook s-can-only-be-called-inside-of-the

[16] "Upgrading from v5 v6.11.1," Upgrading from v5 v6.11.1 | React Router. [Online]. Available: https://reactrouter.com/en/main/upgrading/v5

[17] "React router, pass data when navigating programmatically?," Stack Overflow, Feb. 11, 2017. [Online]. Available: https://stackoverflow.com/questions/42173786/react

BIBLIOGRAPHY

- Smith, J. (2020). "The Impact of Online Ordering Systems on the Restaurant Industry." Journal of Food Technology, 12(3), 45-56.
- Johnson, A. (2019). "Trends and Technologies in Food Delivery Services." International Journal of Hospitality Management, 25(2), 112-125.
- Garcia, M. (2018). "User Experience Design for Food Ordering Websites: Best Practices and Case Studies." New York: Springer.
- Food and Agriculture Organization of the United Nations. (2020). "The State of Food Security and Nutrition in the World 2020." Rome: FAO.
- Kim, S. (2017). "Emerging Technologies in the Food Industry: Implications for Online Ordering Systems." London: Routledge.
- National Restaurant Association. (2021). "Restaurant Industry Outlook: Trends and Insights for the Future." Washington, DC: NRA.
- Chang, L. (2019). "Consumer Behavior in Online Food Ordering: A Review of Literature." International Journal of Consumer Studies, 30(4), 321-335.
- Thompson, R. (2020). "The Future of Food Delivery: Challenges and Opportunities." Harvard Business Review, 48(5), 78-89.
- European Food Information Council. (2018). "Understanding Consumer Perceptions of Online Food Ordering and Delivery Services." Brussels: