

SMS FOOD DELIVERY SYSTEM

SUBMITTED BY:

Maliha Akhtar 2021-GCUF-058598

Sana Kouser 2021-GCUF-058595

Samreen Akhtar 2021-GCUF-058630

SUPERVISED BY:

PROF. Yasir Arfat

BACHELORS OF SCIENCE IN COMPUTER SCIENCE



**DEPARTMENT OF COMPUTER SCIENCE
GOVERNMENT COLLEGE UNIVERSITY FAISLABAD
2021-2025**



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

In the name of Allah, the most gracious, the most merciful

DECLARATION

We, the undersigned, solemnly declare that the Final Year Project titled “**SMS Food Ordering System**” is an original work carried out by **Maliha Akhtar, Samreen Akhter, and Sana Kouser** under the supervision of **Mr. Dur-subhani** (Head of Department, Computer Science), at Government Graduate College Chowk Azam, GC University Faisalabad, Pakistan.

We affirm that this project is the result of our sincere efforts, study, and research, and that no part of it has been copied from any published source. The contents of this project have not been submitted, in part or full, for any other degree, diploma, or certificate at any educational institution.

All external sources of information, ideas, or contributions have been properly acknowledged through appropriate references and citations. We take full responsibility for the authenticity and originality of the project, including its code, documentation, and other submitted materials.

We understand that any false declaration or misrepresentation of facts may result in disciplinary action from the university.

Signature of Student:

Maliha Akhtar _____

Registration No. 2021-GCUF-058598

Samreen Akhter _____

Registration No. 2021-GCUF-058630

Sana kouser _____

Registration No. 2021-GCUF-058595

Dedication

I dedicate this project to God Almighty, my Creator, my strong pillar, and my source of inspiration, wisdom, knowledge, and understanding. He has been the source of my strength throughout this program, and it is on His wings alone that I have soared. I also dedicate this work to my friends, who have encouraged me along the way and whose support ensured that I gave my all to complete what I started. This project, titled "**SMS food ordering system**" is dedicated to the individuals who have been a source of inspiration and support throughout this journey. To my family, for their unwavering love, encouragement, and understanding. Your belief in me has been my driving force, and I am deeply grateful for your constant support. To my project supervisor, **Mr. Yasir arfat**, for his guidance, expertise, and valuable insights. His mentorship has been invaluable in shaping this project and my growth as a developer. To my friends and classmates, for their camaraderie, motivation, and the countless hours spent brainstorming ideas and discussing challenges. Your presence made this project more enjoyable and fulfilling. To the staff and faculty members at **Govt Graduate College Chowk Azam**, for providing a conducive learning environment and resources that contributed to my education and the successful completion of this project. To all the users and stakeholders who participated in the testing and evaluation of the app, providing valuable feedback and helping me improve its functionality and user experience. Lastly, I would like to express my gratitude to everyone who played a role, big or small, in this project. Your support and belief in my abilities have been instrumental in its completion. This project is dedicated to each and every one of you. Thank you for being a part of my journey and making it a truly enriching experience.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who contributed to the successful completion of our Final Year Project titled “**SMS Food Ordering System.**”

First and foremost, we are deeply thankful to our project supervisor **Mr. Yasir Arfat** for his invaluable guidance, support, and encouragement throughout the duration of this project. His mentorship provided clarity and direction to our efforts.

We also extend our thanks to the faculty and staff of the Computer Science Department, Govt Graduate College Chowk Azam, for providing a conducive learning environment and access to necessary resources.

Our heartfelt thanks go to our families and friends for their continuous motivation, patience, and emotional support, especially during challenging phases of the project.

We are also grateful to the participants, users, and stakeholders who provided valuable feedback during the testing and evaluation phases. Their insights greatly helped improve the usability and functionality of the application.

Special thanks to the developers and contributors of the open-source tools, libraries, and frameworks we utilized. Their work significantly accelerated our development process.

Lastly, to anyone who played even a small role in our success—thank you. Your contribution is genuinely appreciated.

Maliha Akhtar (RollNo.730614)

Samreen Akhter (RollNo.730646)

Sana kouser (Roll No730611)

To

The Controller of Examinations,

Government Graduate College, Chowk Azam

GC University, Faisalabad

This is to certify that the Final Year Project titled “**SMS Food Ordering System**” submitted by:

Maliha akhtar (Roll No. 730614)

Samreen Akhter (Roll No. 730646)

Sana kouser (Roll No. 730611)

has been examined and found satisfactory for the partial fulfillment of the requirements for the award of the Bachelor of Science in Computer Science degree.

Internal Examiner

Name: _____

Signature: _____

Lecturer: Govt Graduate College Chowk Azam

Department of Computer Science

External Examiner

Name: _____

Signature: _____

Content	
Chapter 01:	3
Introduction to the Problem	4
1.1 Introduction:	4
1.2 Background:	4
1.3 Purpose:	5
1.4 Scope:	5
1.5 Objective:	6
1.6 Intended Audience and Reading Suggestions:	6
1.7 Process Model (Rapid Application Development - RAD):	7
1.8 Document Convention:	7
1.9 Existing System:	8
1.10 Proposed System:	8
Chapter 2:	10
Literature Review	10
2.1 Overview of SMS Food Ordering Systems:	10
2.2 Major SMS Food Ordering Platforms:	10
2.3 Comparison of Uber Eats, DoorDash, Grubhub, and Foodpanda:	13
2.4 Differences in User Experience:	14
2.5 Impact on the Food Service Industry:	14
Chapter 3:	15
Software Requirement Specification (SRS)	15
3.1 Overall Description:	15
3.1.1 Product Perspective:	15
3.2 System Features:	16
3.3 External Interface Requirements:	24
Chapter 4:	27
Analysis	27
4.1 Identifying Actors and Use Cases:	27
4.2 Forming Use Case Diagram with Candidate and Use Cases:	27
4.3 Describe the Events Flow for Use Case:	29
Chapter 5:	33
System Design	33
5.1 Architecture Diagram	33
5.2 ERD (Entity Relationship Diagram) with Data Dictionary	33

5.3 Class Diagram	34
5.4 Object Diagram	35
5.5 Sequence Diagram	36
5.6 Activity Diagram.....	36
5.7 Collaboration Diagram.....	37
Chapter 06 :	39
Development	39
6.1 Operating System:.....	39
6.2 Development Approach:	39
6.3 Programming Language:.....	40
6.4 Platform:	42
6.5 Database:	43
Chapter 7:	44
Testing	44
7.1 Test Case Specification:	44
7.2 Black Box Testing:.....	44
7.3 White Box Testing:	45
7.4 Test Cases:.....	46
Chapter 08:	49
Implementation	49
8.1 Component Diagram	49
8.2 Deployment Diagram.....	49
8.3 Database Architecture:	50
Chapter09:	52
Tools and Technologies	52
9.1 Frontend Technologies:.....	52
9.2 Backend Technologies:	52
9.3 Development and Testing Tools:.....	53
APPENDIX A:	54
USER DOCUMENTATION	54
User Documentation for Restaurant Owner.....	54
User Documentation for Customer	55
APPENDIX B:	57
References	57

Abstract:

The evolution of SMS food ordering has revolutionized how customers interact with restaurants, offering convenience, speed, and accessibility. This project, **SMS Food Ordering System**, aims to provide a seamless digital platform that connects customers with restaurants through a responsive web-based interface. The system comprises two main interfaces: a customer-facing web application and an admin panel for restaurant owners.

The frontend of the application is developed using **React, HTML, CSS, and JavaScript**, ensuring an interactive and intuitive user experience. Customers can browse menus, filter cuisines, add items to their cart, place orders, make online payments, and track their order history. Restaurant owners can log in through a secure admin dashboard to manage menu items, view incoming orders, update order statuses, and create special deals.

The backend is built using **Node.js** with **Express.js**, providing a fast, scalable, and secure server-side environment. A **MySQL** database is used to manage and store data related to users, restaurants, menus, and orders efficiently. **RESTful APIs** facilitate smooth communication between the frontend and backend layers, ensuring a seamless user experience.

The system follows the **Rapid Application Development (RAD)** model to ensure quick iteration and continuous user feedback. The platform is designed to handle multiple users simultaneously while maintaining data integrity and offering secure transactions. This project addresses the need for a modern, efficient, and user-friendly food ordering system that improves customer satisfaction and restaurant operations.

Keywords:

SMS Food Ordering, Web Application, Restaurant Management, ReactJS, Node.js, Express.js, MySQL, RESTful APIs, RAD Model

Chapter 01:

Introduction to the Problem

1.1 Introduction:

In today's technology-driven world, online systems have become essential in providing convenience and improving efficiency across various sectors, including the food industry. The rapid increase in internet usage and mobile device adoption has transformed traditional food ordering into a more digital and automated process.

The SMS Food Ordering System is a web-based application designed to allow users to browse restaurant menus, place orders, and make payments online, all from the comfort of their home or workplace. It also offers restaurant administrators the tools to manage orders, update menus, and monitor user activity through an admin dashboard.

This project aims to bridge the gap between customers and restaurants by offering a reliable, user-friendly, and efficient platform that enhances the overall dining experience. By implementing real-time order tracking and secure payment options, the system benefits both the customers and the restaurant management.

The system will be developed using modern web technologies including HTML, CSS, JavaScript for the frontend, Node.js and Express.js for the backend, and MySQL for the database. The design process will be initiated using Figma, ensuring an intuitive and user-focused interface. The development methodology used is Rapid Application Development (RAD), which allows for iterative and flexible development with regular feedback and improvements.

1.2 Background:

The evolution of technology has significantly transformed how services are delivered, and the food industry is no exception. Traditionally, food ordering was limited to in-person visits or phone calls, which often resulted in communication errors, delays, and long waiting times. These traditional methods lacked efficiency and were unable to meet the growing demands of modern consumers who prefer quick and seamless services.

With the rise of internet technologies and e-commerce platforms, online food ordering systems have emerged as a practical solution to overcome these limitations. They offer a convenient way for customers to view menus, place orders, and make payments without physical interaction. At the same time, they provide restaurant staff with a systematic approach to managing orders, reducing human error, and improving service delivery.

However, many small to medium-sized restaurants still face challenges in adopting such systems due to high costs, lack of customization, or technical complexity. This project aims to develop an affordable, customizable, and user-friendly web-based application that addresses these challenges. The Online Food Ordering System is designed to help restaurants improve their efficiency while enhancing the customer experience through digital transformation.

1.3 Purpose:

The purpose of this project is to design and develop a web-based SMS Food Ordering System that enhances the way restaurants interact with their customers. The system aims to provide a convenient, efficient, and secure platform for users to browse menus, place orders, and make payments online. Simultaneously, it empowers restaurant owners and staff to manage orders, update menu items, and monitor customer interactions through an integrated admin dashboard.

This project is intended to reduce the dependency on traditional food ordering methods, minimize human error, and increase operational efficiency. By automating the food ordering process, the system not only improves customer satisfaction but also supports restaurants—particularly small and medium-sized businesses—in adopting modern technology without requiring significant financial or technical investment.

Overall, the system is developed to serve as a scalable and customizable solution that addresses the needs of both end-users and restaurant administrators, promoting digital innovation in the food service industry.

1.4 Scope:

The SMS Food Ordering System is designed as a web-based application that facilitates food ordering between customers and restaurants. The scope of the project defines the boundaries and features that will be included and excluded in the final system. It focuses on delivering essential functionality for both users and administrators while keeping the system lightweight, scalable, and easy to use.

Included in Scope:

- **User Registration and Login:** Secure account creation and login functionality for customers and administrators.
- **Menu Browsing:** Display of food items with options for filtering and searching.
- **Online Ordering:** cart, place Users can add items to their orders, and receive order confirmations.
- **Payment Integration:** Secure payment options for customers to complete transactions online.
- **Order Tracking:** Real-time status updates for users to track the progress of their orders.
- **Admin Dashboard:** A back-end interface for restaurant staff to manage menu items, view and update orders, and monitor system activity.
- **Order History:** Users can view their previous orders and re-order items if desired.

Excluded from Scope:

- **Multi-language Support:** The system will be developed in a single language (English) for simplicity.
- **Third-Party Delivery Integration:** The system does not include APIs for integration with third-party delivery services like UberEats or Foodpanda.
- **Advanced Analytics and Reporting:** In-depth business intelligence tools and analytics dashboards are beyond the scope of this version.

This scope ensures that the project remains focused, feasible within the given timeline, and aligned with the needs of the target users.

1.5 Objective:

The primary objective of this project is to design and implement a web-based Online Food Ordering System that enhances the food ordering process for both customers and restaurant administrators. The system aims to provide a reliable, user-friendly platform that simplifies the ordering experience while improving operational efficiency for restaurants.

- **The specific objectives of the project are as follows:**
- **To develop a responsive and interactive web application** for customers to browse menus and place food orders online.
- **To implement secure user authentication**, allowing users and administrators to log in and manage their respective dashboards.
- **To provide real-time order tracking**, enabling customers to monitor the status of their orders from preparation to delivery.
- **To integrate a secure online payment gateway** to allow users to make payments through the platform.
- **To build an admin panel**, where restaurant staff can manage menu items, monitor incoming orders, and update order statuses.
- **To maintain a user-friendly interface** that ensures a smooth experience for users with varying levels of technical skills.
- **To ensure data security and privacy**, especially for user information and transaction details.

These objectives align with the overall goal of improving the food ordering process through digital innovation while keeping the system scalable and customizable for future enhancements.

1.6 Intended Audience and Reading Suggestions:

This documentation is intended for the following audiences:

- **Developers** – To understand the technical aspects, design principles, and system architecture of the **Online Food Ordering System**.
- **Restaurant Owners** – To explore how the application assists in managing menus, tracking orders, and handling customer interactions effectively.
- **Users (Customers)** – To learn about the platform's features such as menu browsing, order placement, real-time tracking, and secure payments.
- **Project Managers and Stakeholders** – To gain insights into the project's objectives, scope, functionality, and overall development approach.

Reading Suggestions:

- Begin with **Chapter 1** for an overview of the system and its objectives.
- Refer to **Chapter 2** for background research and system analysis.
- Continue to **Chapters 3 to 5** for design, development, and testing phases.
- Conclude with **Chapter 6** for results, conclusions, and future work.

1.7 Process Model (Rapid Application Development - RAD):

For the development of the SMS Food Ordering System, the Rapid Application Development (RAD) model has been adopted. RAD is a type of incremental software development process model that emphasizes rapid prototyping and quick feedback over long drawn-out development and testing cycles.

Key Reasons for Choosing RAD:

- It supports fast delivery with the help of reusable components.
- It encourages early user involvement, which helps in refining the application based on real feedback.
- It is ideal for projects like this one, where requirements are well understood and time is a constraint.

Phases of RAD in This Project:

- **Requirements Planning:**
 - Gather initial requirements from stakeholders (users, restaurant owners).
 - Define scope and prioritize core functionalities.
- **User Design:**
 - Design wireframes and UI/UX using Figma.
 - Conduct feedback sessions and refine the interface before development.
- **Construction:**
 - Develop the frontend using HTML, CSS, JavaScript, and React.
 - Implement the backend using Node.js and Express.js.
 - Integrate the MySQL database for data storage.
 - Ensure modular coding to speed up development and testing.
- **Cutover (Finalization):**
 - Conduct system testing, user acceptance testing, and performance evaluations.
 - Deploy the final application.
 - Prepare documentation and deliverables.

This iterative model allows flexibility and faster development while involving the user at every key stage, ensuring that the final product meets the expectations and requirements of all stakeholders.

1.8 Document Convention:

The documentation follows the following conventions to maintain clarity and consistency:

- **Title:** The document sections are clearly titled to reflect their respective content.
- **Introduction:** Each section begins with an introduction that provides an overview of the topics covered.

- **Getting Started:** This section provides step-by-step instructions for downloading, installing, and setting up the system, along with any initial setup steps required.
- **User Interface:** The document includes screenshots or illustrations of the application's user interface to assist users in navigating the system.
- **Features:** This section provides a detailed description of the system's features, including how each feature works and how users can interact with it.
- **Troubleshooting:** The document includes a troubleshooting section to address common issues users may encounter, along with solutions or workarounds.

1.9 Existing System:

In the current food service landscape, most small to medium-sized restaurants either rely on manual processes or third-party food delivery platforms to manage their orders. These existing systems come with various limitations that affect both the restaurant's operations and the customer's experience.

Limitations of the Existing System:

- **Manual Order Management:** Traditional restaurants take orders via phone calls or walk-in, which often results in human errors, miscommunication, and slower service.
- **No Real-Time Tracking:** Customers have no visibility into the status of their orders, leading to frustration and uncertainty.
- **Dependence on Third-Party Apps:** Many restaurants use platforms like FoodPanda or Uber Eats, which charge high commission fees and offer limited control over branding, menu customization, or customer data.
- **Limited Payment Options:** Some existing systems only support cash on delivery, which is inconvenient for users who prefer digital payments.
- **Lack of Integration:** There is no central system to manage orders, update the menu, track order history, or generate useful analytics.

Due to these challenges, the current systems fail to provide a seamless and efficient experience for both restaurant staff and customers. This creates the need for a dedicated, customizable, and efficient online food ordering system tailored to the specific needs of restaurants.

1.10 Proposed System:

The proposed **SMS Food Ordering System** is designed to solve the problems faced by traditional restaurant order management systems and third-party platforms. This system aims to provide an efficient, user-friendly, and secure platform for both restaurant owners and customers. The key features and solutions offered by this system are as follows:

Key Features of the Proposed System:

- **User-Friendly Interface:**
The system will provide an intuitive and easy-to-navigate interface for both customers and restaurant owners. Users can quickly browse menus, customize their orders, and make payments seamlessly. The platform will be responsive, ensuring an optimal experience across different devices (desktop, tablet, and mobile).

- **Real-Time Order Tracking:**
Customers will be able to track their orders in real-time, from preparation to delivery, improving transparency and reducing customer uncertainty. This feature also enables restaurant owners to update the order status, ensuring better communication with customers.
- **Secure Payment Gateway Integration:**
The system will integrate a secure and reliable payment gateway to support various payment methods, including credit/debit cards, e-wallets, and other online payment systems. This will ensure the safety of sensitive customer data during transactions.
- **Admin Dashboard for Restaurant Management:**
The admin panel will allow restaurant owners to manage their orders, update menus, set prices, and create special offers. Owners can also monitor real-time analytics, view sales data, and track customer preferences to make data-driven decisions.
- **Order History and Personalized Recommendations:**
The system will store customer order history, enabling them to quickly reorder their favorite meals. Personalized recommendations will be provided based on previous orders, allowing for an enhanced customer experience.
- **Customer Registration and Login:**
Users will be able to register and create personal accounts to store their preferences, order history, and payment information. This feature will make future orders faster and more convenient.
- **Multilingual Support (Future Expansion):**
Although not a part of the initial system, multilingual support can be implemented in the future to cater to customers from different regions, improving the system's accessibility.
- **Push Notifications and Alerts:**
The system will notify customers about their order status, delivery updates, or special promotions through real-time notifications. This helps in keeping the customers engaged and informed.

Technological Framework:

- **Frontend:** The frontend will be developed using **HTML**, **CSS**, **JavaScript**, and **React**, ensuring a dynamic and responsive user interface.
- **Backend:** The backend will be powered by **Node.js** and **Express.js**, providing a scalable and efficient server-side environment.
- **Database:** **MySQL** will be used for data storage, ensuring secure and reliable management of user data, orders, and payment information.

The **Proposed System** addresses the limitations of existing solutions by offering a streamlined, efficient, and secure platform for both customers and restaurant owners.

Chapter 2:

Literature Review

2.1 Overview of SMS Food Ordering Systems:

SMS food ordering systems have revolutionized the food delivery industry, offering customers convenience and variety at the touch of a button. These platforms allow users to browse through different restaurants and food items, place an order, and track its delivery status. In addition, these systems facilitate a smooth payment process, often including various payment methods like credit/debit cards, digital wallets, and cash on delivery.

The growing popularity of online food ordering is driven by several factors, including busy lifestyles, changing consumer habits, and the increasing use of smartphones. Major platforms in this space, such as **Uber Eats**, **DoorDash**, **Grubhub** and **foodapanda** have established themselves as leaders in the market.

2.2 Major SMS Food Ordering Platforms:

In this section, we will explore three major players in the SMS food ordering industry, comparing their technical features and business strategies.

2.2.1 Uber Eats:

Uber Eats, an extension of the Uber brand, is one of the leading online food delivery services globally. The platform connects users with local restaurants through an easy-to-use mobile app, allowing customers to browse menus, place orders, and track deliveries.

Technical Features:

- **API Integration:** Uber Eats integrates with various restaurant POS (Point of Sale) systems, providing real-time updates to customers about the availability of menu items, order status, and estimated delivery time.
- **Payment Systems:** The platform offers secure payment methods through Uber's existing payment infrastructure, supporting options like credit/debit cards, digital wallets, and Uber credits.
- **Order Tracking:** Uber Eats uses GPS technology to offer real-time tracking of food delivery, allowing users to see the delivery status and route in real-time.

Business Insights:

- **Global Reach:** Uber Eats operates in over 45 countries and continues to expand.
- **Revenue Model:** Uber Eats generates revenue by charging restaurants a commission fee per order, typically around 15-30%. They also charge customers delivery fees and service charges.
- **Partnerships:** Uber Eats partners with both large chains and small local restaurants, providing a wide selection of food choices to users.

2.2.2 DoorDash:

DoorDash is another significant player in the food delivery industry, particularly in North America. It focuses on offering a broad range of restaurants and food delivery services, including groceries and alcohol.

Technical Features:

- **Platform Flexibility:** DoorDash provides a robust platform for both restaurants and customers, with options to manage orders, track deliveries, and process payments.
- **Logistics and AI:** DoorDash uses artificial intelligence and machine learning to optimize delivery routes, ensuring that food arrives quickly and efficiently.
- **Delivery and Order Management:** DoorDash employs its own delivery drivers, known as "Dashers," who are assigned orders based on proximity and availability.

Business Insights:

- **Revenue Model:** DoorDash charges a commission fee from restaurants, ranging from 15-30%. Customers also pay a service fee and delivery fee depending on the order size and delivery distance.
- **Market Penetration:** DoorDash operates in the U.S., Canada, and Australia, and continues to increase its market share through strategic partnerships with restaurants and expansion into new cities.
- **Focus on Delivery Logistics:** DoorDash differentiates itself with its focus on logistics and using advanced algorithms to optimize delivery efficiency.

2.2.3 Grubhub:

Grubhub is one of the oldest and most well-established online food ordering platforms in the U.S. It connects customers with local restaurants, offering an easy ordering process and multiple payment options.

Technical Features:

- **Comprehensive Menu System:** Grubhub provides users with a comprehensive menu of food items, complete with pricing, photos, and restaurant reviews.
- **Customer Reviews and Ratings:** The platform allows customers to leave reviews and rate restaurants, helping future users make informed decisions.
- **Delivery Options:** Grubhub partners with both in-house drivers and third-party delivery services to provide food delivery.

Business Insights: Revenue Model:

- Grubhub generates revenue primarily through restaurant commissions (20-30% per order). It also charges customers a delivery fee, which varies based on distance and order size.

- **Acquisitions and Growth:** Grubhub merged with Seamless in 2013 and later acquired Eat24 and other smaller platforms to expand its market presence.
- **Strategic Partnerships:** Grubhub focuses on building relationships with major restaurant chains as well as independent outlets, offering a diverse selection of food.

2.2.4 Foodpanda:

Foodpanda is a prominent online food delivery platform that operates in several countries across Asia, including Pakistan, Bangladesh, Thailand, and others. It connects users to a wide variety of restaurants and food outlets through its mobile app and website, enabling quick and convenient food ordering and delivery.

Technical Features:

- **User-Friendly App Interface:** Foodpanda offers an intuitive and responsive mobile app that allows users to easily browse restaurants, filter cuisines, and place orders.
- **Real-Time Tracking:** The platform offers real-time order tracking via GPS, enabling customers to monitor their food from kitchen to doorstep.
- **Multi-Language Support:** In regions like Pakistan and Bangladesh, the app provides support for local languages, improving accessibility and user experience.

Business Insights:

- **Operational Model:** Foodpanda partners with restaurants and employs its own delivery riders, ensuring full control over the delivery chain.
- **Revenue Generation:** It earns revenue through commission from restaurants (typically 15–25%) and delivery fees charged to customers.
- **Market Positioning:** In countries like Pakistan, Foodpanda is the leading food delivery service with high market penetration in both metropolitan and tier-2 cities.
- **Expansion Strategy:** Apart from food delivery, Foodpanda has expanded into grocery and pharmacy delivery services in certain regions under the “Pandamart” brand.
- **Marketing Tactics:** Foodpanda frequently uses promotional discounts, loyalty rewards, and targeted ads to retain and attract users.

2.2.5 Proposed Online Food Ordering System:

Technical Features:

- **Frontend:** Built using **HTML, CSS, JavaScript, and React** to ensure a responsive and interactive user experience. The UI mirrors real-world platforms like Foodpanda and Uber Eats.
- **Backend:** Developed with **Node.js** and **Express.js**, providing a scalable and fast RESTful API.
- **Database:** Uses **MySQL** for storing user data, orders, menu items, and admin functions.
- **Payment Gateway Integration:** Allows users to choose between **credit card, digital wallet,**

or cash on delivery.

- **Order Tracking:** Basic real-time tracking through order status updates and timestamp logging.
- **Admin Dashboard:** Enables restaurant owners or admins to manage menus, view order analytics, and update inventory.
- **Login & Registration:** Secure login for customers and admins using password encryption and session management.

Business Insights:

- **Target Market:** Local restaurants and customers in a specific region (e.g., university campus, city, or community).
- **Revenue Model (Proposed):**
 - Commission-based revenue from restaurant partners.
 - Optional delivery fee from users.
 - Potential for premium membership or advertising inside the platform.
- **Scalability Plan:**
 - Easy addition of new restaurants and menu items.
 - Modular design for future features like coupons, ratings, and delivery partner tracking.

This system offers a lightweight but modern alternative to large-scale platforms and demonstrates how a custom solution can serve both technical learning and real-world needs.

2.3 Comparison of Uber Eats, DoorDash, Grubhub, and Foodpanda:

Feature	Uber Eats	DoorDash	Grubhub	Foodpanda	Proposed System
Global Reach	45+ countries	North America, Australia	Primarily U.S.	Asia, Pakistan, etc.	Local/Regional
Revenue Model	Commission + Fees	Commission + Fees	Commission + Fees	Commission + Fees	Commission + Delivery Fee
Technology	GPS, Real-time Tracking	AI Routing & Management	Menus & Ratings	Lightweight app, dispatch	React + Node.js + MySQL
Delivery Model	Uber Drivers	Dashers	In-house + Third-party	Foodpanda riders	Custom/local delivery
UI/UX	Sleek Interface	Personalized App	Review-based	Lightweight, simple	Custom Figma-to-React UI

2.4 Differences in User Experience:

- **User Interface (UI):** All three platforms offer an easy-to-use mobile app, but **Uber Eats** is renowned for its sleek and minimalistic design. **DoorDash** provides a more personalized experience, especially with its AI-driven delivery management. **Grubhub** has a more traditional UI, focusing heavily on restaurant listings and reviews.
- **Customization:** **DoorDash** stands out with its extensive customization options for both customers and restaurants, including the ability to reorder previous meals easily. **Uber Eats** and **Grubhub** also offer customization but have fewer options for personalized user experiences.

2.5 Impact on the Food Service Industry:

The SMS food ordering system has significantly transformed the food service industry:

- **Increased Competition:** Restaurants now compete with not only local eateries but also international delivery giants like Uber Eats, DoorDash, and Grubhub.
- **Consumer Behavior:** Online food ordering has become a norm for many customers, especially in urban areas. The convenience and time-saving nature of these platforms have shifted consumer behavior towards online ordering rather than traditional dine-in experiences.
- **Opportunities for Small Businesses:** Smaller and local restaurants now have an opportunity to compete with larger chains by leveraging these platforms to reach more customers without investing heavily in their own delivery infrastructure.

Chapter 3:

Software Requirement Specification (SRS)

3.1 Overall Description:

3.1.1 Product Perspective:

The **SMS Food Ordering System** is a web application designed for customers to browse menus, place food orders, and manage their cart. It allows users to interact with a dynamic menu, view daily specials, add items to their cart, and make payments. The system is implemented using **React.js** for the frontend, providing a smooth, user-friendly interface.

It aims to provide a simplified ordering experience comparable to apps like **Uber Eats** or **DoorDash**, with a focus on local restaurant offerings. The backend (currently not implemented) would be responsible for managing user data, orders, and restaurant menus.

- **System Context:** The application interfaces with a hypothetical backend, although currently only the frontend components are developed, including the **landing page, product menu, order summary, payment gateway, and user authentication**.
- **Scope:** The project is limited to the frontend and designed to be integrated with a backend in the future.

3.1.2 Product Features:

The primary features of the SMS Food Ordering System include:

- **User Registration and Login** - Customers can sign up and log in to the system to place orders.
- **Menu Browsing** - Users can view the food menu categorized by restaurant.
- **Order Placement** - Users can place orders by selecting food items, adding them to the cart, and proceeding to checkout.
- **Order Tracking** - Users can track the status of their orders.
- **Payment Integration** - Payment can be processed securely through integrated payment gateways.
- **Restaurant Menu Management** - Admins can add, update, or remove menu items and set prices.
- **Admin Dashboard** - Admins can view, manage, and update orders and track customer activities.

3.1.3 Design and Implementation Constraints:

- The system should work across all modern browsers (Chrome, Firefox, Safari, Edge).
- The user interface should be responsive and support both desktop and mobile views.
- The payment gateway should be integrated securely using HTTPS, and sensitive information should be encrypted.

3.1.4 Assumptions and Dependencies:

- The system will require an active internet connection for users to place orders and access the system.
- The system will depend on third-party payment gateways like **Stripe** or **PayPal** for processing payments.
- The platform will rely on **MySQL** for data storage, which will be managed by an administrator.

3.2 System Features:

3.2.1 Login/Sign Up:

- **Description:** Users can create an account or log in to the system using their email and password.
- **Functional Requirements:**
 - Users must provide a valid email address and password for registration.
 - Users must verify their credentials before accessing the platform.
 - On successful login, users are redirected to the home page or menu page.
- **UI Design:**

The login/signup form will include fields for the email, password, and a submit button.

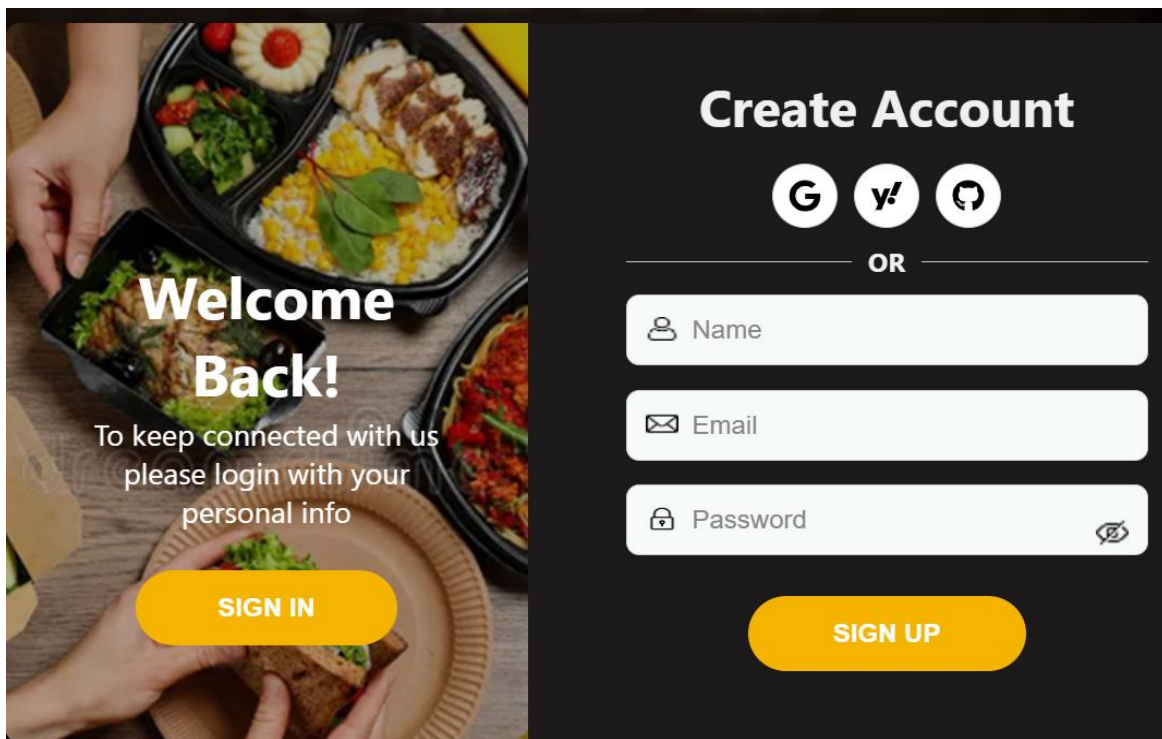


Fig.1:login

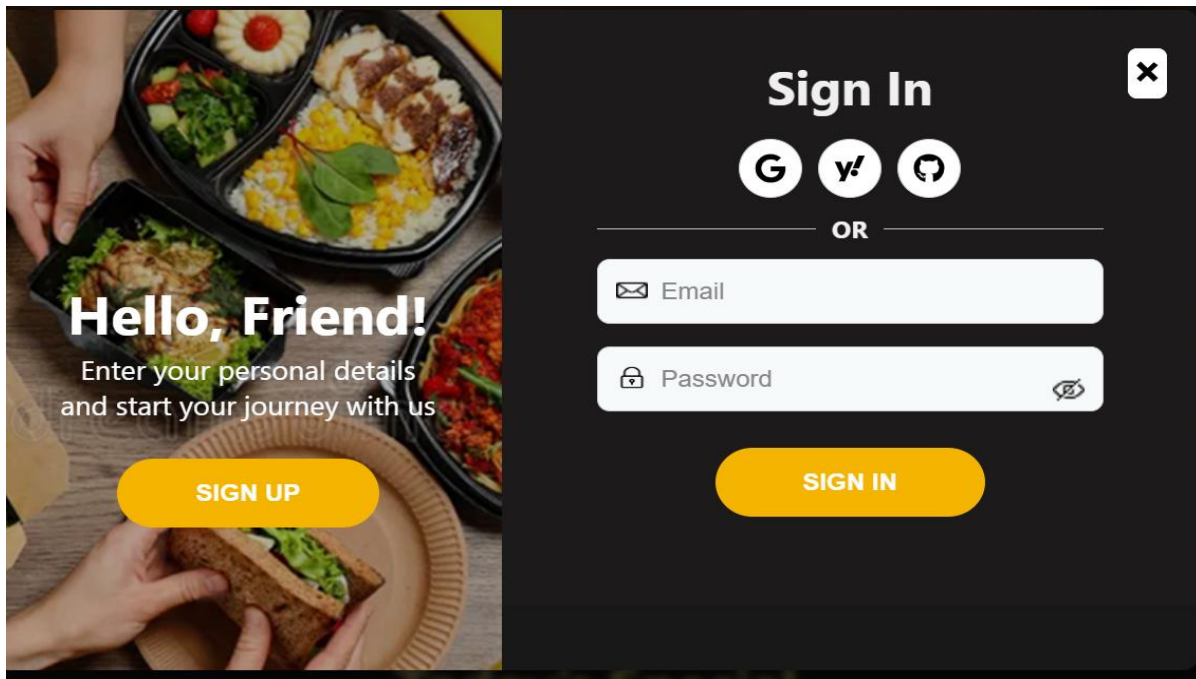


Fig.2: sign-in

3.2.2 Manage Restaurant Dishes and Deals:

- **Description:** Admin users can manage the food menu, including adding, updating, or removing dishes and setting prices.
- **Functional Requirements:**
 - Admins can upload images, descriptions, and prices for new dishes.
 - Admins can categorize dishes by type (e.g., vegetarian, non-vegetarian, desserts).
 - Admins can set promotions and discount deals.
- **UI Design:**

A dashboard interface where admins can add or edit menu items and view current items.

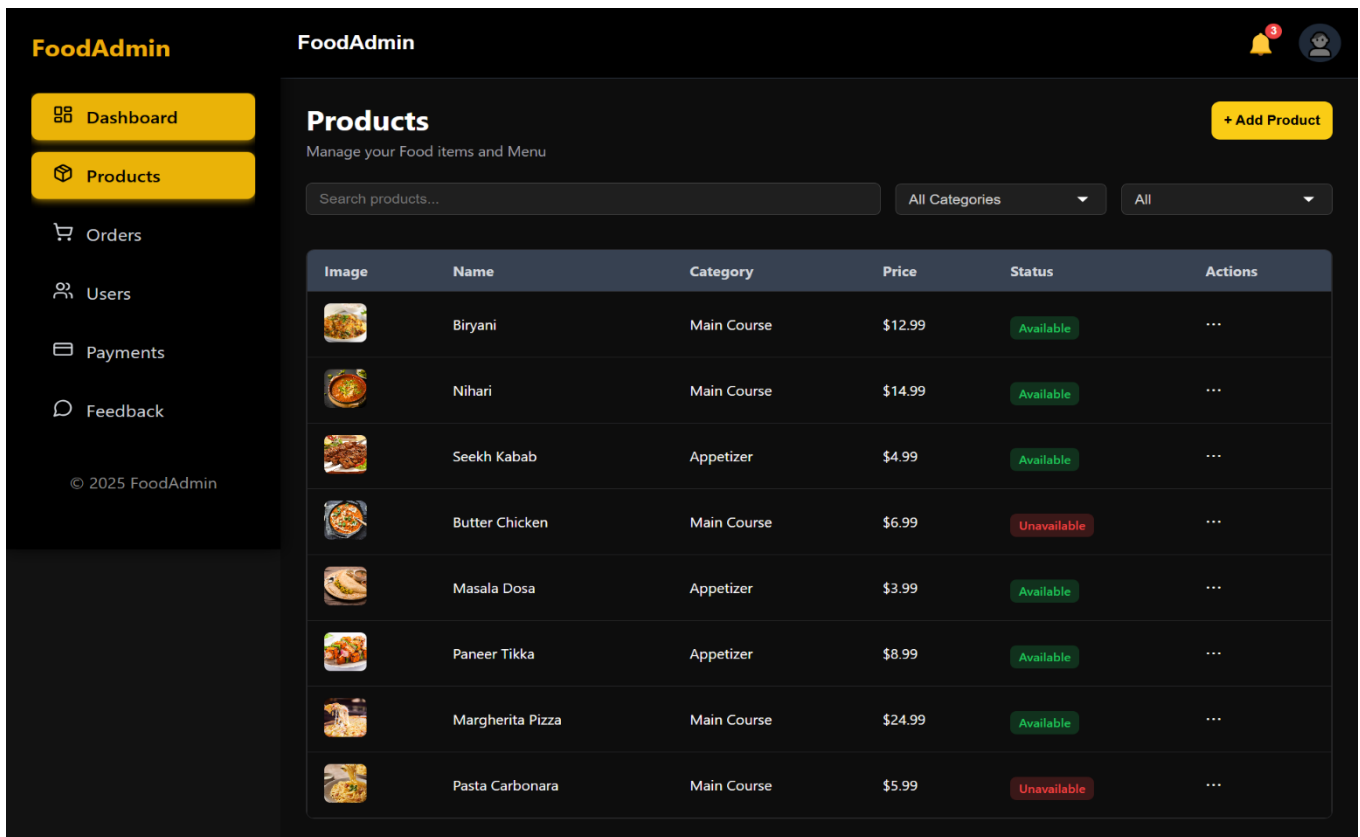


Fig.3: Manage Restaurant Dishes and Deals:

3.2.3 Place Orders:

In an SMS food delivery system project report, the "Place Orders" topic explains how customers interact with the system to submit their orders. It covers the entire process, from selecting items and adding them to a cart, to providing delivery details and making payment. This section also describes how the system manages the order queue, confirms orders, and communicates with the restaurant for preparation.

- **Description:** Users can place orders by adding items to their cart and proceeding to checkout.

- **Order Placement Process:**

- **Menu Display and Search:**

Explain how the system presents menus (categories, subcategories, descriptions, images) and how customers can search for specific items.

- **Cart Management:**

Detail how customers can add items to their cart, view the total cost, remove items, and edit quantities.

- **Delivery Address Input:**

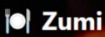
Explain how customers provide their delivery address, including options for using saved addresses or entering a new one.

- **Payment Options:**

Describe the available payment methods (credit card, mobile payments, etc.) and how the system securely processes payments.

- **UI Design:**

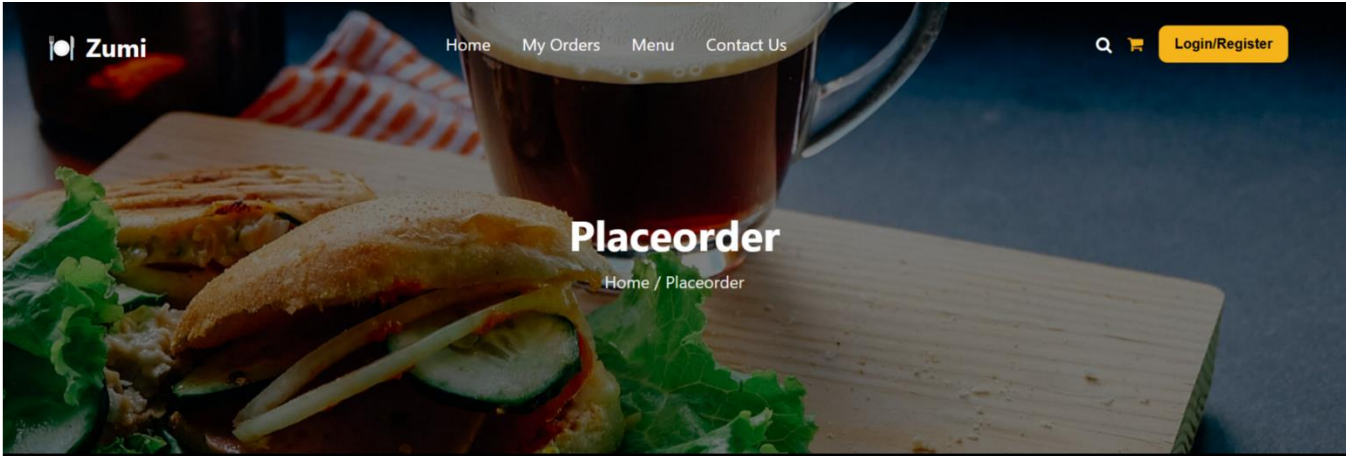
A cart page where users can view and modify the items they have selected. The report would detail how the UI allows customers to browse menus, select items, add them to a cart, and adjust quantities. This might include screenshots or mockups of the UI.



[Home](#)[My Orders](#)[Menu](#)[Contact Us](#)


Q

Login/Register



Placeorder

Home / Placeorder



Share

Estimated Delivery: 30-45 minutes

Chicken Biryani

Spicy basmati rice with marinated chicken

Price: **5.99 PKR**

Type:

Cuisine:

Meat:

Quantity:

- 1 +

Add to Cart

Add to Wishlist

Buy Now

Description

Important Yummi Products:

Ratings and Reviews

4.4

★ ★ ★ ★ ☆

1.43M reviews

5

4

3

2

1

Ashar Hasnain

★ ★ ★ ★ ☆

August 14, 2024

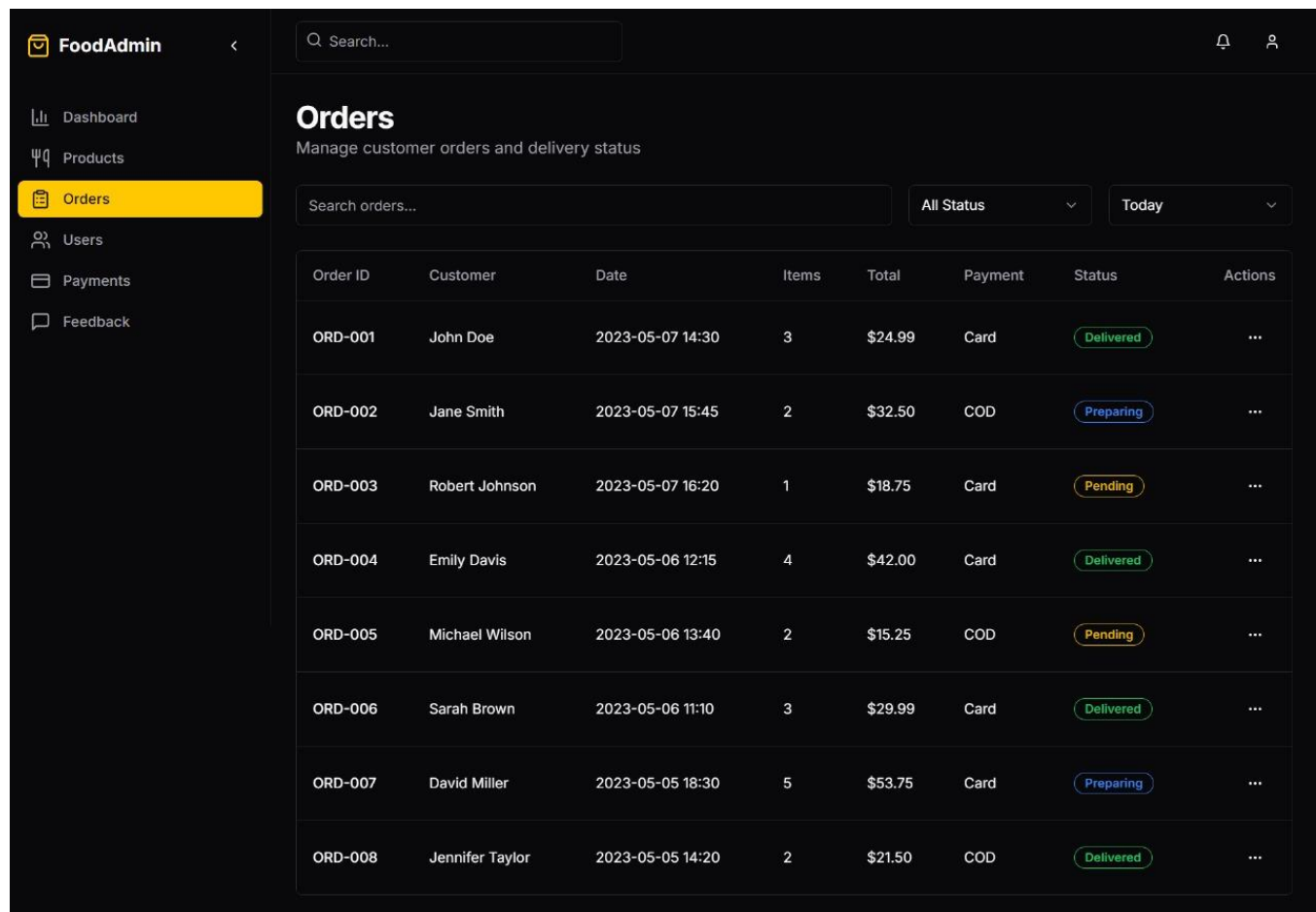
The food was absolutely delicious! The chicken burger was juicy, and the fries were perfectly crispy. Delivered hot and on time. Will definitely order again

Write a Review

Fig.4: Place Orders:

3.2.4 Manage Orders:

- **Description:** Admins and users can manage orders. Admins can update order statuses, and users can track the status of their orders.
- **Functional Requirements:**
 - Admins can view, update, and change the status of any order (e.g., Preparing, Delivered, Cancelled).
 - Users can track their order's status through the interface.
- **UI Design:**
 - A simple, yet detailed order management page for both admins and users to track orders. For users, the **Order Status** will be displayed after they log in.



Order ID	Customer	Date	Items	Total	Payment	Status	Actions
ORD-001	John Doe	2023-05-07 14:30	3	\$24.99	Card	Delivered	...
ORD-002	Jane Smith	2023-05-07 15:45	2	\$32.50	COD	Preparing	...
ORD-003	Robert Johnson	2023-05-07 16:20	1	\$18.75	Card	Pending	...
ORD-004	Emily Davis	2023-05-06 12:15	4	\$42.00	Card	Delivered	...
ORD-005	Michael Wilson	2023-05-06 13:40	2	\$15.25	COD	Pending	...
ORD-006	Sarah Brown	2023-05-06 11:10	3	\$29.99	Card	Delivered	...
ORD-007	David Miller	2023-05-05 18:30	5	\$53.75	Card	Preparing	...
ORD-008	Jennifer Taylor	2023-05-05 14:20	2	\$21.50	COD	Delivered	...

Fig .5: mange order:

3.2.5 Payment :

In an SMS food delivery system project report, the payment topic should explain how customers can make secure and convenient online payments for their food orders. This includes detailing the payment methods accepted (credit/debit cards, mobile wallets, cash on delivery), the integration with payment gateways, and the security measures in place to protect customer data. The report should also address

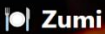
any fees or charges associated with payment processing, and explain how refunds are handled in case of issues.

- **Functional Requirements:**

- The **Payment Page** will have fields for credit card numbers, expiry dates, and a "**Submit Payment**" button.
- **Validation** is done to ensure that all fields are filled in correctly, including proper formatting for credit card numbers and expiry dates.

- **UI Design:**

- **Payment Page** showing the layout with form fields (Credit Card Number, Expiry Date, etc.) and the "**Submit Payment**" button.
- Explain how customers can use their credit or debit cards to make payments, including the security measures used to protect card information.



HomeMy OrdersMenuContact Us

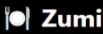
Q

🛒

Login/Register

Payment

Home / Payment



Contact

☐ Email me with news and offers

Delivery

Pakistan

▼

First name (optional)

Last name

Address

Apartment, suite, etc. (optional)

City

Postal code (optional)

☐ Save this information for next time

Shipping method

Standard

\$17.45

Payment

All transactions are secure and encrypted.

Credit card

Card number

🔒


Expiration date (MM/YY)

Security code

Name on card

☐ Use shipping address as billing address

Pay now



Spicy basmati rice with marinated chicken

1

5.99 PKR

Subtotal

5.99 PKR

Shipping

17.45 PKR

Total

23.44 PKR

Fig .5: place order

23

3.3 External Interface Requirements:

3.3.1 User Interfaces:

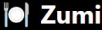
The user interface of the Online Food Ordering System should be designed to be intuitive, easy to use, and responsive on both desktop and mobile devices.

- **Menu Page:** Displays food items categorized by restaurant and cuisine. The menu is categorized by cuisine type (e.g., Italian, Pakistani). Each food item displays an image, title, description, and price. Users can click on items to get more details.



Fig.6: menu page:

- **Order page:**
 - Displays all items in the user's cart, allows modifications, and shows the total amount. Once the user is ready, they can proceed to checkout.
 - showing the layout with form fields (Credit Card Number, Expiry Date, etc.) and the **"Submit Payment"**



Contact

Email or mobile number

☐ Email me with news and offers

Delivery

Pakistan

First name (optional)

Last name

Address

Apartment, suite, etc. (optional)

City

Postal code (optional)

☐ Save this information for next time

Shipping method

Standard

\$17.45

Payment

All transactions are secure and encrypted.

Credit card

Card number


Expiration date (MM/YY)

Security code

Name on card

☐ Use shipping address as billing address

Pay now



Spicy basmati rice with marinated chicken

1

5.99 PKR

Subtotal

5.99 PKR

Shipping

17.45 PKR

Total

23.44 PKR

Fig.7: order page:

3.3.2 Hardware Interfaces:

- **User Devices:** Desktop computers, laptops, and mobile devices.
- **Server:** The backend of the system will run on a cloud server (e.g., AWS) to ensure scalability and uptime.

3.3.3 Software Interfaces:

- **Frontend:** ReactJS
- **Backend:** Node.js, Express.js

- **Database:** MySQL
- **Payment Gateway:** Integrated APIs like Stripe or PayPal.

3.3.4 Communications Interfaces:

- **REST APIs** will be used for communication between the frontend and backend of the system.
- **Secure HTTPS** for all transactions and sensitive data handling.

3.4 Other Nonfunctional Requirements:

3.4.1 Performance Requirements:

- The system should be able to handle up to 500 concurrent users without performance degradation.
- The response time for page loading should not exceed 3 seconds.

3.4.2 Usability Requirements:

- The system should have an intuitive UI/UX design that requires minimal training for both end users and administrators.
- Mobile responsiveness should be prioritized for users accessing the platform on mobile devices.

3.4.3 Security Requirements:

- The system should implement **SSL/TLS** encryption for secure communication.
- Payment information must be processed via third-party payment gateways that comply with PCI-DSS.
- **JWT** (JSON Web Tokens) will be used for secure authentication and session management.

Chapter 4:

Analysis

4.1 Identifying Actors and Use Cases:

In the SMS Food Ordering System, various actors interact with the system to perform different functions. The two primary actors are:

4.1.1 Use Case Name: Restaurant Owner

Actor: Restaurant Owner

Description: The restaurant owner manages the backend of the application, including menu items, offers, and order status.

Use Cases:

- Add, edit, or delete food items.
- Create and manage discount deals.
- View and manage customer orders.
- Update order status (e.g., pending, in-progress, delivered).

4.1.2 Use Case Name: User (Customer):

Actor: Customer (User)

Description: The end-user accesses the platform to browse the menu, place orders, and make payments.

Use Cases:

- Register and log into the system.
- Browse restaurant menu with filters.
- Add items to cart and place orders.
- Make online payments.
- Track order status.
- View order history.
- Add items to wishlist.
- Contact support.

4.2 Forming Use Case Diagram with Candidate and Use Cases:

Use case diagrams visually represent the interactions between users (actors) and system functionalities.

4.2.1 Use Case Diagram: Restaurants:

Actors: Restaurant Owner

Use Cases:

- Manage Menu
- Manage Deals
- Manage Orders
- View Reports

Diagram:

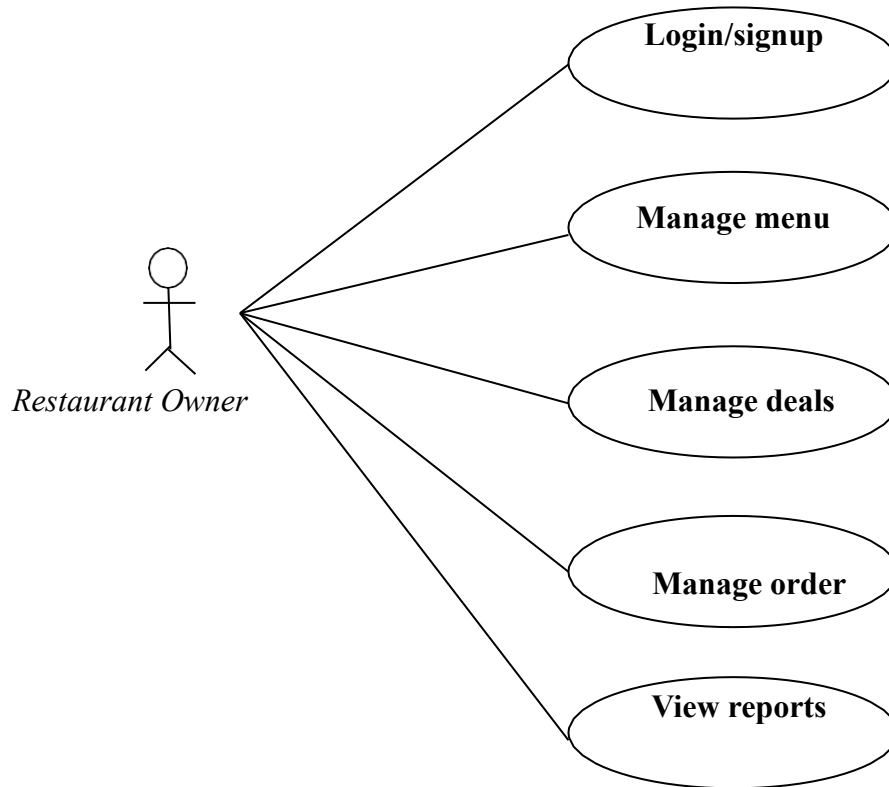


Fig.8: Use case Restaurant Owner

4.2.2 Use Case Diagram: Users:

Actors: Users (Customers)

Use Cases:

- Register/Login
- Browse Menu
- Add to Cart
- Place Order
- Make Payment
- Track Order
- Add to Wishlist

- Contact Support

Diagram:

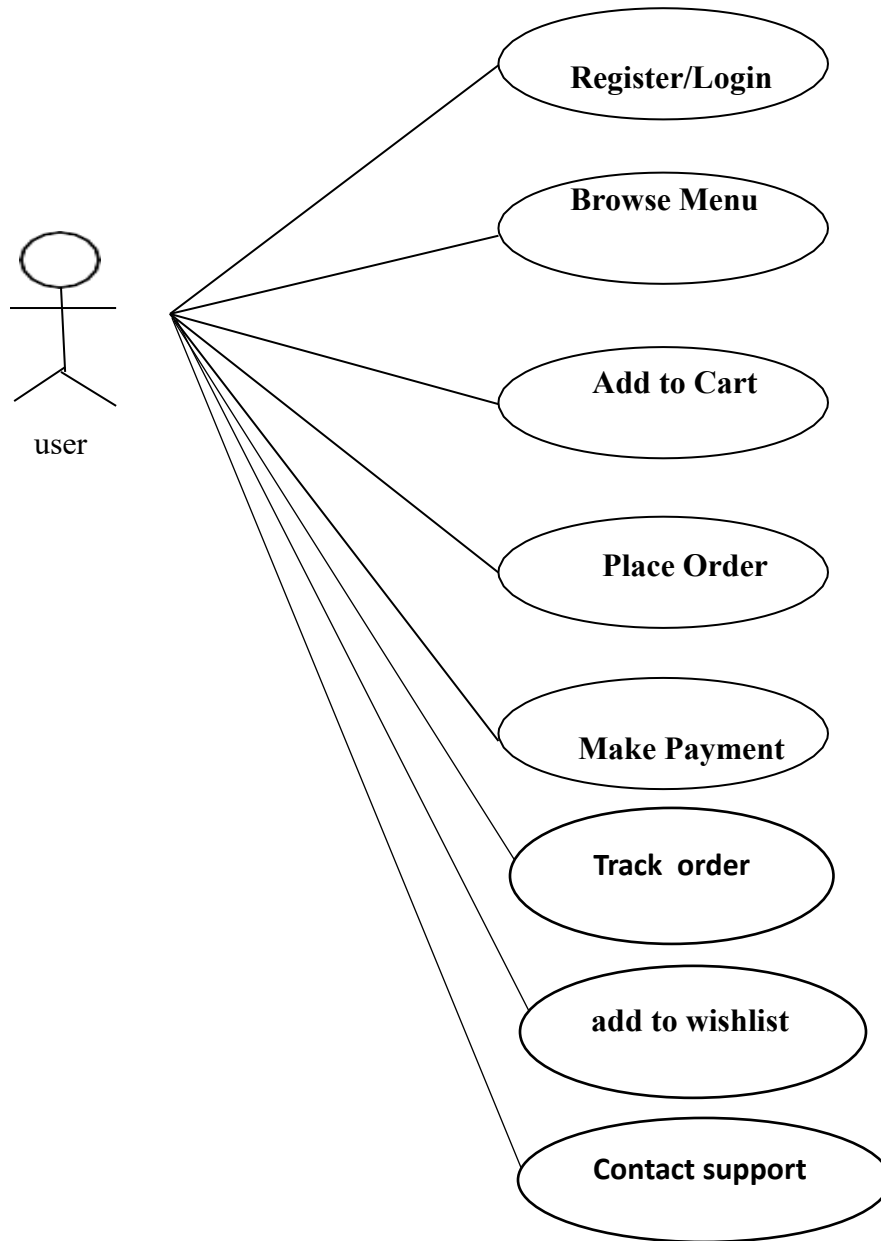


Fig.9: Use case user

4.3 Describe the Events Flow for Use Case:

This section provides the event flows (step-by-step interactions) for key operations in the system.

4.3.1 Event Flow: Managing Dishes and Deals (Restaurant Owner):

Primary Actor: Restaurant Owner

Flow:

- Restaurant owner logs into the admin panel.
- Navigates to the “Manage Menu” section.

- Adds new dishes with name, price, image, and description.
- Updates existing dishes.
- Deletes dishes if required.
- Navigates to “Manage Deals”.
- Creates or modifies deal packages (e.g., Buy 1 Get 1).

Daigram:

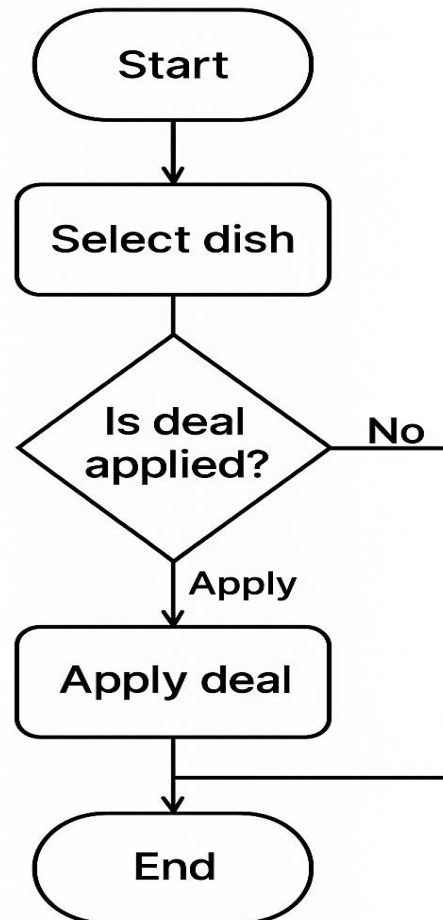


Fig.10: Activity Flow for Managing Dishes and Deals

4.3.2 Event Flow: Placing Orders (User):

Primary Actor: User

Flow:

- User logs into their account.
- Browses the menu or uses filters.
- Adds items to cart.

- Reviews cart and applies promo code (optional).
- Confirms delivery address.
- Selects payment method (Online/Cash).
- Places the order.
- Receives order confirmation and tracking link.

Diagram:

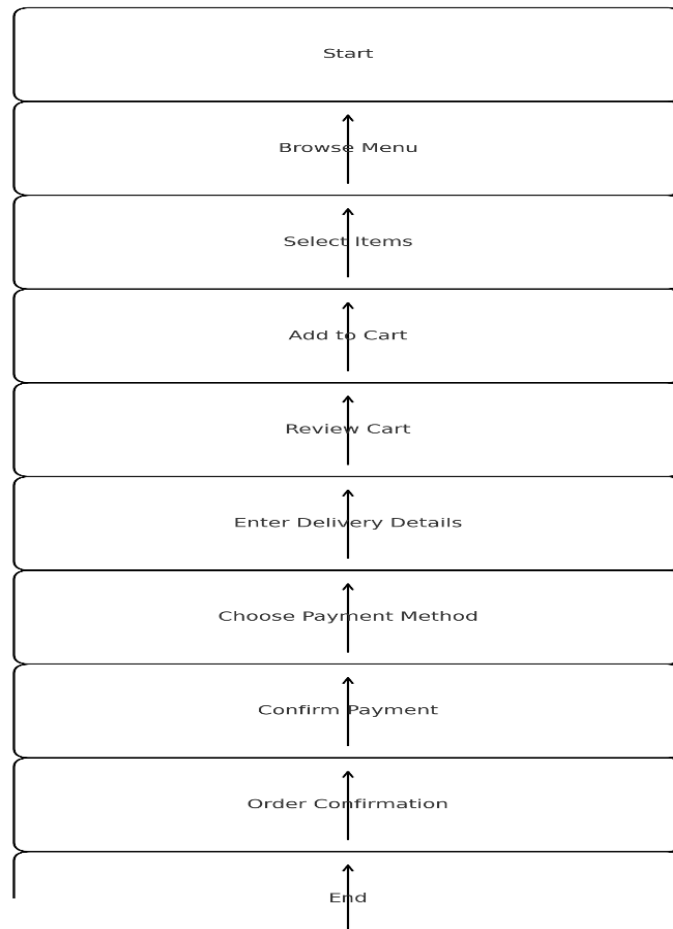


Fig.11: Activity Diagram for Placing Orders

4.3.3 Event Flow: Managing Orders (Restaurant Owner):

Primary Actor: Restaurant Owner

Flow:

- Restaurant owner views all incoming orders.
- Changes status to "Accepted", "Preparing", or "Delivered".
- Tracks delivery if integrated with delivery partner.

- Marks order as completed after successful delivery.

Diagram:

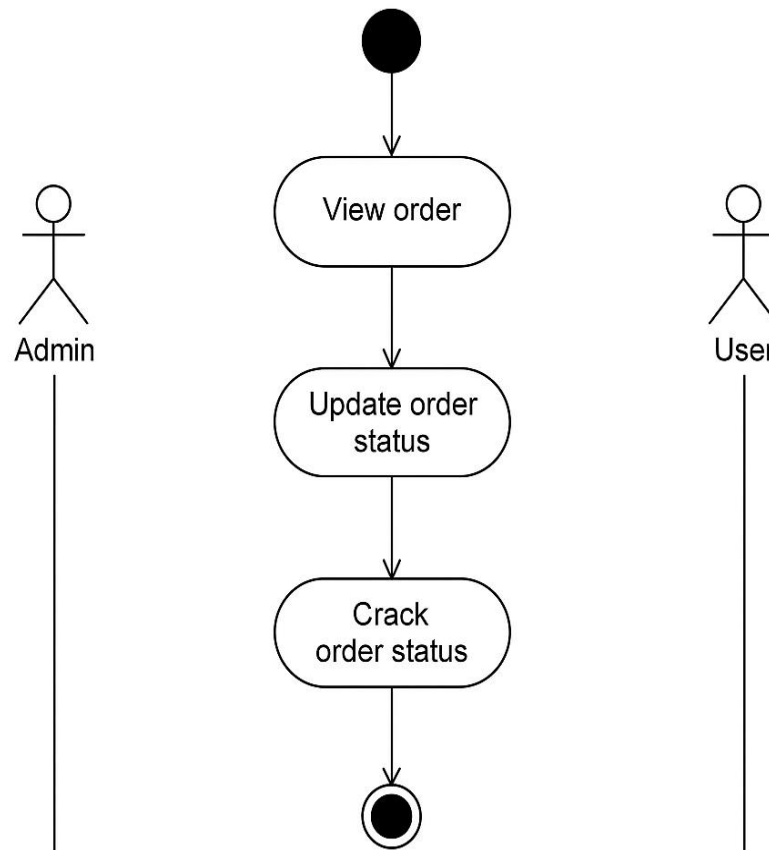


Fig.12: Sequence Diagram or Activity Diagram for Managing Orders

Chapter 5:

System Design

5.1 Architecture Diagram:

The architecture of the SMS Food Ordering System follows a **three-tier architecture**, ensuring modularity and scalability:

- **Presentation Layer:** The front end developed in React.js handles user interactions. Customers can browse menus, place orders, and track delivery, while restaurant owners can manage menus and view orders via the admin dashboard.
- **Application Layer:** This middle tier is powered by Node.js and Express.js, which handles all business logic such as processing orders, authenticating users, and applying offers or discounts.
- **Data Layer:** MySQL database stores all persistent data including users, orders, menu items, payments, and restaurants.

This layered structure helps separate concerns and makes the system easy to maintain and upgrade.

5.2 ERD (Entity Relationship Diagram) with Data Dictionary:

Entity Relationship Diagram (ERD) shows the relationship between major entities of the system like Customer, Restaurant, Order, MenuItem, and Payment.

Key Entities:

- **Customer:** Stores user details like name, email, contact.
- **Restaurant:** Holds data about each restaurant (name, address, status).
- **MenuItem:** Stores food items with attributes like title, price, description.
- **Order:** Captures order-related info (order_id, customer_id, total, status).
- **Payment:** Records payment details linked with orders.

Sample Data Dictionary:

Table	Attribute	Type	Description
Customer	customer_id	INT (PK)	Unique ID for each customer
	Name	VARCHAR	Customer's full name
	Email	VARCHAR	Email for login
Order	order_id	INT (PK)	Unique order ID
	customer_id	INT (FK)	Linked customer
	total_price	DECIMAL	Total price of the order
MenuItem	item_id	INT (PK)	Unique item ID
	restaurant_id	INT (FK)	Linked restaurant
	Price	DECIMAL	Price of the item

Daigarm:

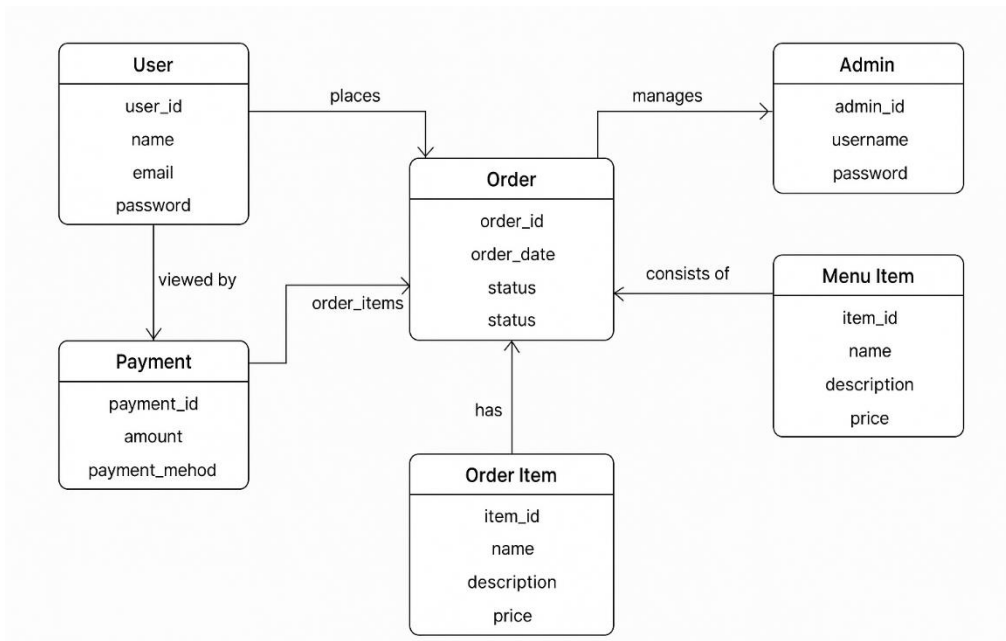


Fig.13: ER daigram

5.3 Class Diagram:

The class diagram reflects the object-oriented structure of the system.

Major Classes:

- **Customer**
 - Attributes: id, name, email, password
 - Methods: register(), login(), placeOrder()
- **Restaurant**
 - Attributes: id, name, location
 - Methods: addMenuItem(), updateMenu(), viewOrders()
- **Order**
 - Attributes: id, customerId, status, total
 - Methods: calculateTotal(), updateStatus()
- **MenuItem**
 - Attributes: id, name, description, price
 - Methods: getDetails()

This shows the system's components and their interrelationships, helping in both design and maintenance.

Diagram:

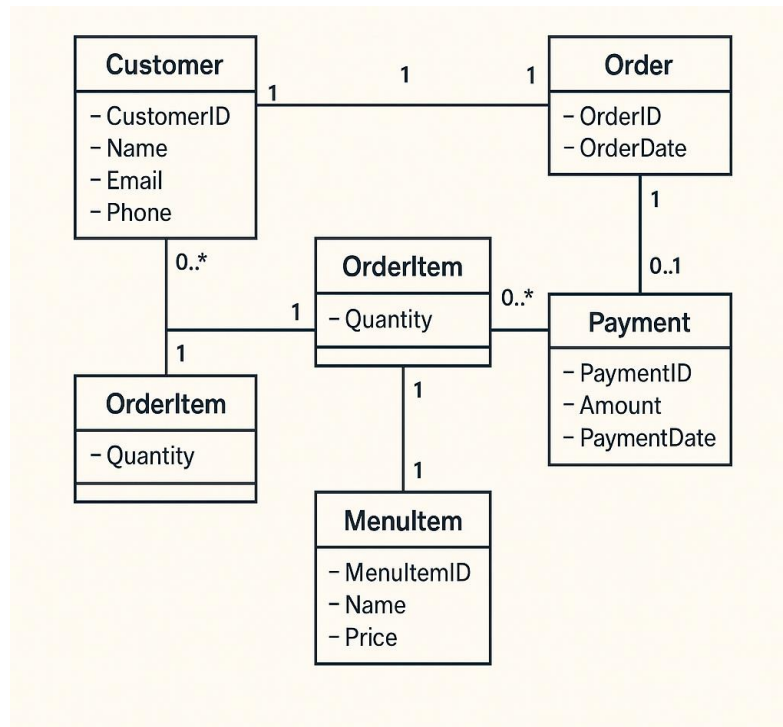


Fig.14: class daigram

5.4 Object Diagram:

An object diagram is a snapshot of the objects and their relationships at a specific point in time.

Example Scenario: A customer places an order.

- Objects: Customer1, RestaurantA, Order102, PizzaItem, BurgerItem
- Order102 links to Customer1 and includes references to PizzaItem and BurgerItem.

This diagram helps in understanding runtime object interactions:

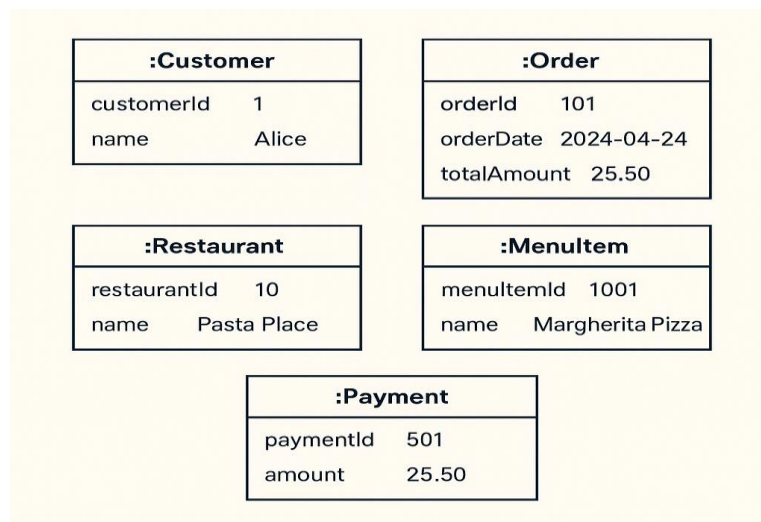


Fig.15: object daigram

5.5 Sequence Diagram:

The sequence diagram shows how different components interact in a time-ordered manner during the **order placement** process:

Actors: Customer, Frontend (React), Backend (Node.js), Database

- Customer logs in
- Selects menu item and adds to cart
- Places the order
- Backend stores the order in the database
- Confirmation is sent back to the frontend

This visualizes the flow of data and control across the system.

Diagram:

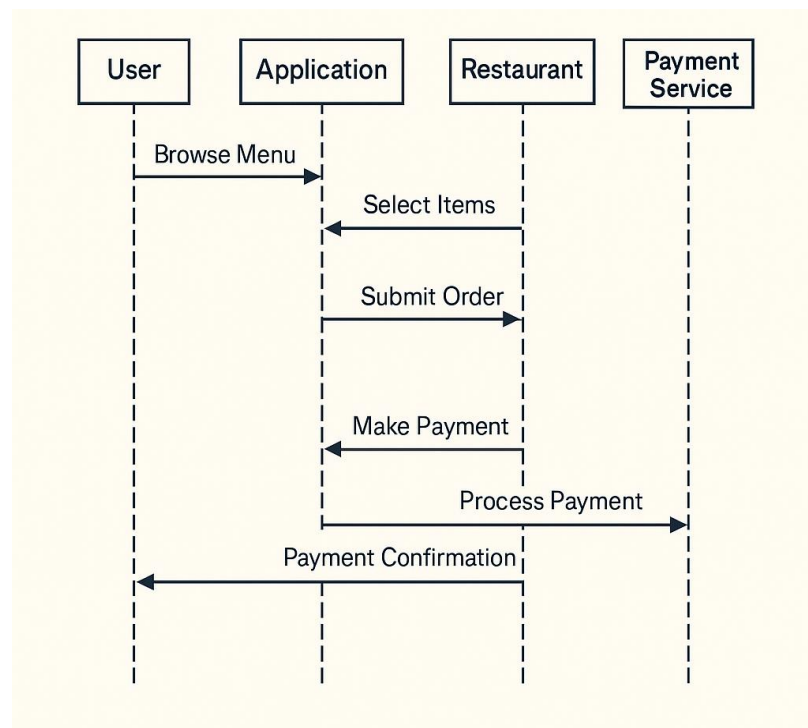


Fig.16: sequence daigram

5.6 Activity Diagram:

The activity diagram outlines the workflow of **managing an order**:

- Login
- Browse Menu
- Add items to cart
- Place order

- Confirm payment
- Order status: pending → in preparation → out for delivery → delivered

This shows the conditional logic and flow between actions.

Diagram:

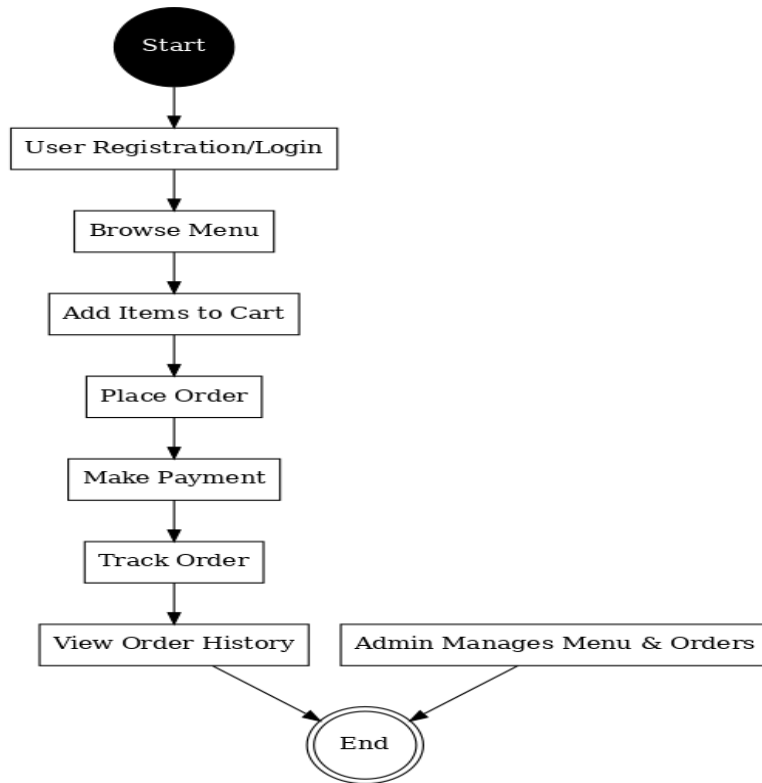


Fig.17: collobration diagram

5.7 Collaboration Diagram:

The collaboration diagram is similar to the sequence diagram but emphasizes **object relationships**:

- Customer interacts with the OrderHandler
- OrderHandler interacts with MenuManager and PaymentProcessor
- PaymentProcessor updates the order status
- This highlights how modules communicate with one another during tasks like ordering.

Diagram:

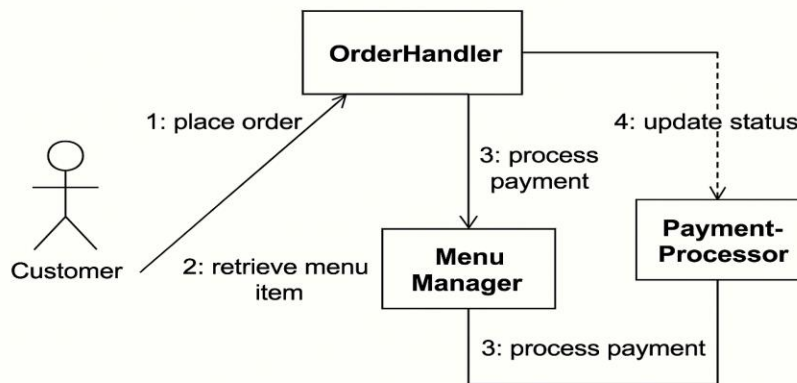


Fig.18: collobration diagram

4.8 State Transition Diagram:

This diagram shows the **state changes of an order**:

States:

- New
- Confirmed
- Preparing
- Dispatched
- Delivered
- Cancelled

Events/Triggers: Customer places order, payment processed, restaurant confirms, rider picks up.

Daigram:

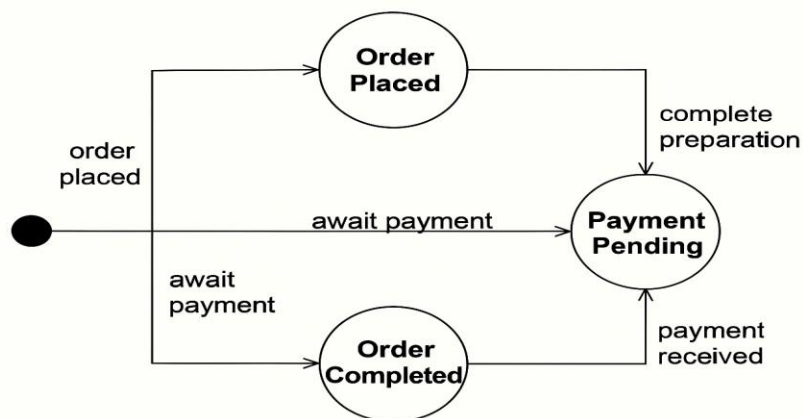


Fig.19: state transition diagram

Chapter 06 :

Development

6.1 Operating System:

An operating system (OS) provides the basic platform to develop, test, and run applications. For the development of the **SMS Food Ordering System**, the operating system needed to support full-stack web development tools including React for the frontend, Node.js and Express.js for the backend, and MySQL for database management.

6.1.1 Available Operating Systems:

Various operating systems were considered for the development of the system:

- **Windows 10/11:** Most widely used, supports a variety of development tools and has a user-friendly interface. Ideal for students due to familiarity and widespread availability.
- **Linux (Ubuntu):** Lightweight and commonly used for backend deployment, particularly in server environments. However, it requires terminal-based interaction, which may not be comfortable for all users.
- **macOS:** Stable and Unix-based, often preferred by frontend developers, but is restricted to Apple hardware, making it less accessible for all users.

6.1.2 Selected Operating System:

For the development of this project, **Windows 10** was selected as the primary operating system. It offers compatibility with all major development environments required for the SMS Food Ordering System, such as:

- **Visual Studio Code** for writing frontend and backend code.
- **Node.js** and **npm** for backend environment and package management.
- **MySQL Workbench** for managing databases.
- **Git Bash and Command Prompt** for terminal operations.

Windows also provides GUI-based tools that are easy to install and configure, making it suitable for student and beginner-level development. It allows running servers, emulators, browsers, and editors simultaneously with ease, offering a complete environment for full-stack development.

6.2 Development Approach:

The development approach defines the overall methodology or model used to build the software. For this project, several software development methodologies were considered to determine the most efficient way to complete the SMS Food Ordering System within time and quality constraints.

6.2.1 Available Development Approaches:

Some commonly used software development models that were initially considered include:

- **Waterfall Model**
A traditional linear approach, where each phase is completed before the next begins. It is rigid and doesn't accommodate changing requirements.
- **Agile Model**
A flexible, iterative approach focused on rapid delivery and constant feedback. Suitable for ongoing and evolving projects, but may require team collaboration tools and workflows.
- **Spiral Model**
Focuses on iterative development with risk analysis at each step. Suitable for large, complex projects but can be overkill for medium-scale academic systems.
- **Rapid Application Development (RAD) Model**
Emphasizes quick development with user feedback and component reuse. Ideal for projects with clear objectives and a focus on GUI-based interfaces.

6.2.2 Selected Development Approach (RAD Model):

The **RAD (Rapid Application Development)** model was selected for developing the Online Food Ordering System due to the following reasons:

- **Fast Prototyping:** The project required a fast turnaround time with a strong focus on user interface. RAD allows quick prototyping and changes based on feedback.
- **Reusable Components:** Components like login forms, food cards, cart management, and payment UI were reused across the system, aligning well with RAD practices.
- **User Involvement:** The system was designed in multiple phases with user (teacher/supervisor) feedback at each iteration, which RAD supports.
- **Parallel Development:** Frontend (React-based) and backend (Node.js & Express.js) development could proceed concurrently, improving efficiency.
- **Figma Design to Code:** RAD allowed easy implementation of the design prototypes created in Figma into actual functional UI components.

Thus, RAD proved to be the most effective model to meet project deadlines while ensuring quality and responsiveness in the final system.

6.3 Programming Language:

The selection of the programming language plays a crucial role in determining the system's performance, scalability, and ease of development. For this project, both **frontend** and **backend** technologies were considered and carefully selected.

6.3.1 Available Programming Languages:

Several programming languages were evaluated for both the frontend and backend layers of the application:

Frontend Options:

- **HTML, CSS, JavaScript:** The core technologies used to structure, style, and add basic interactivity to webpages.
- **React.js:** A powerful JavaScript library used for building component-based user interfaces.
- **Angular:** A complete framework for dynamic web apps, offering strong structure but higher complexity.
- **Vue.js:** Lightweight and flexible, great for small-to-medium applications.

Backend Options:

- **Node.js with Express.js:** A JavaScript runtime and framework combination, great for fast and scalable APIs.
- **PHP with Laravel:** Widely used for web apps but slightly heavier and less reactive compared to Node.js.
- **Python with Django:** Easy to write but more suited for data-heavy applications.

After careful consideration, the stack was chosen based on simplicity, reusability, and the developer's experience.

6.3.2 Selected Programming Language:

For this project, the following languages and frameworks were selected:

Frontend:

- **HTML5** – Used for structuring all pages.
- **CSS3** – Used for styling and layout to match the Figma design exactly.
- **JavaScript (ES6+)** – For adding dynamic behaviors and client-side interactivity.
- **React.js** – For creating reusable components and maintaining the application's state efficiently.

Backend:

- **Node.js** – Enables JavaScript on the server side, allowing uniform language use across the stack.
- **Express.js** – Simplifies building the server and API endpoints required for order placement, login, etc.

These selections were made due to:

- React's component-based structure and virtual DOM for high-performance rendering.
- Node.js and Express's speed, simplicity, and wide community support.
- Smooth integration between the backend APIs and frontend React components.

This tech stack offered modern web standards, ease of development, faster implementation of design prototypes, and scalability for future enhancements.

6.4 Platform:

The platform refers to the environment in which the application is developed, tested, and deployed. It includes the operating system, browsers, hosting services, and the development environment used to build and run the application.

6.4.1 Available Platforms:

A number of platforms were considered during the evaluation stage to determine the most efficient and compatible environment for development and deployment:

Local Development Platforms:

- Windows OS: Compatible with most development tools and commonly used by developers.
- Linux OS: Preferred for hosting and deployment due to its speed and reliability.
- macOS: Stable and popular among frontend developers but less common in deployment servers.

Hosting/Deployment Platforms:

- Heroku: Easy-to-use platform for hosting Node.js applications but with limited free-tier resources.
- Vercel: Ideal for frontend (React) hosting with seamless integration and fast deployment.
- Netlify: Good for frontend hosting, supports CI/CD pipelines.
- DigitalOcean / AWS EC2: Full control over server, better scalability, but requires more configuration.

6.4.2 Selected Platform:

After evaluating several options, the following platform configuration was selected: Development Environment:

- Operating System: Windows 10
- Code Editor: Visual Studio Code (VS Code)
- Browser: Google Chrome (for testing and debugging frontend)
- Package Manager: npm (Node Package Manager) for installing React and backend libraries

Frontend Deployment:

Platform: Vercel

Reason: Vercel provides fast, production-ready deployments for React apps with GitHub integration, custom domains, and automatic builds.

Backend Deployment:

Platform: Render / Cyclic.sh (if used)

Reason: These platforms are designed for Node.js hosting and integrate well with GitHub repositories. They offer free-tier support for smaller projects.

This combination ensures a smooth development experience, minimal setup time, and scalability for future feature upgrades.

6.5 Database:

The database is a crucial part of the SMS Food Ordering System as it stores all critical information such as user details, food items, orders, payment history, and more. Choosing the right database ensures efficient data management and seamless interaction between frontend and backend.

6.5.1 Available Databases:

Several database management systems (DBMS) were considered based on factors such as ease of use, community support, scalability, performance, and compatibility with Node.js.

Relational Databases:

- MySQL: Open-source, widely used, highly compatible with Node.js. Excellent for structured data and ACID compliance.
- PostgreSQL: Advanced features, reliable and highly scalable.

NoSQL Databases:

- MongoDB: Flexible, document-oriented database; suitable for rapid development but less structured than SQL.
- Firebase Realtime Database: Easy integration, ideal for real-time apps, but not relational.

6.5.2 Selected Database:

Database: MySQL

Reasons for Selection:

- Structured Data: Ideal for storing users, orders, menu items, payments in relational tables.
- Compatibility: Easily integrates with Node.js using libraries like mysql2 or Sequelize ORM.
- Reliability & Performance: Proven stability and high performance under load.

Database Tables (Sample Overview):

Table Name	Description
Users	Stores user login details and profile info
Restaurants	Stores restaurant profiles and credentials
menu_items	Food items offered by restaurants
Orders	Stores user orders, order status, and timestamps
order_details	Contains items within a specific order
Payments	Stores payment status and transaction details

Chapter 7:

Testing

Definition:

Testing is an essential phase in the software development life cycle that ensures the correctness, reliability, and performance of the application. For the **SMS Food Ordering System**, both **black-box testing** and **white-box testing** methods were used to thoroughly examine each component and ensure proper functionality under expected and unexpected conditions.

7.1 Test Case Specification:

Test case specification refers to the documentation of conditions, inputs, expected outputs, and the environment under which a test will be executed. The objective of writing test cases is to verify that each module of the SMS Food Ordering System functions according to the defined requirements.

In this system, major modules like user registration, login, browsing menu, placing an order, managing orders, and processing payments were considered for testing.

Each test case includes:

- Test Case ID
- Module Name
- Test Scenario
- Input Data
- Expected Output
- Actual Output
- Status (Pass/Fail)

7.2 Black Box Testing:

Black box testing focuses on verifying functionality without examining internal code structure. It is based solely on input and output.

7.2.1 Boundary Value Analysis (BVA):

In the registration and login modules, BVA was applied to input fields like username length, password length, and age (if applicable). The system was tested at boundary values (e.g., min 6 characters and max 16 characters for passwords).

Example:

- Input: 5-character password → Expected: Rejected
- Input: 6-character password → Expected: Accepted

7.2.2 Equivalence Class Partitioning:

The menu filtering feature was tested using valid and invalid categories. This reduced the number of test cases while ensuring coverage.

Example:

- Valid class: “Pakistani”, “Italian” → Pass
- Invalid class: “Unknown” → Fail

7.2.3 State Transition Testing:

This was applied to the order status flow:

- From: "Order Placed" → "Preparing" → "Out for Delivery" → "Delivered"
- Any invalid transitions (e.g., skipping "Preparing") were also tested.

7.2.4 Decision Table Testing:

The payment module was tested with combinations:

- Payment method (Credit/Debit/Online)
- Delivery address presence
- Cart not empty

Only if all conditions were satisfied, payment was processed.

7.2.5 Graph-Based Testing:

The navigation between pages such as Home → Menu → Cart → Checkout was treated as a graph. Each edge was tested to ensure proper redirection and no dead ends or broken links existed.

7.3 White Box Testing:

White box testing was conducted at the code level by the development team. Logical conditions, branches, and loops were tested.

7.3.1 Statement Coverage:

Ensured each line of JavaScript and Node.js backend code was executed at least once.

Example:

A function calculating order total was checked to ensure every statement (e.g., tax, subtotal, discount) executed during testing.

7.3.2 Branch Coverage:

Verified both conditions of if, else if, and else were tested.

For example:

javascript

CopyEdit

```
if (paymentSuccess) {  
    // show success  
} else {  
    // show failure  
}
```

Both paths were tested.

7.3.3 Path Coverage:

All possible paths through functions such as placeOrder() and addToCart() were tested for correctness in execution.

7.4 Test Cases:

Here are sample test cases from key modules:

Test Case TC_01

Test Case ID	TC_01
Description	Test user login with valid credentials
Input	Email: user@gmail.com, Password: 123456
Expected Output	User redirected to home page with welcome message
Actual Output	As expected
Status	Pass

Test Case TC_02:

Test Case ID	TC_02
Description	Login with incorrect credentials
Input	Email: user@gmail.com, Password: wrongpass
Expected Output	Error message: "Invalid credentials"
Actual Output	As expected
Status	Pass

Test Case TC_03

Test Case ID	TC_03
Description	Add an item to cart from menu
Input	Click on “Add to Cart” for Chicken Biryani
Expected Output	Cart shows 1 item with correct name and price
Actual Output	As expected
Status	Pass

Test Case TC_04

Test Case ID	TC_04
Description	Complete order placement with cart items
Input	Cart with 2 items, user logged in, click “Place Order”
Expected Output	Order confirmation screen shown
Actual Output	As expected
Status	Pass

Test Case TC_05

Test Case ID	TC_05
Description	Simulate successful payment via card
Input	Card details and click “Pay Now”
Expected Output	“Payment successful” message shown
Actual Output	As expected
Status	Pass

Test Case TC_06

Test Case ID	TC_06
Description	Admin logs in and adds a new dish
Input	Title: “Kebab Roll”, Price: 350, Image uploaded

Expected Output	New dish appears in menu list
Actual Output	As expected
Status	Pass

Test Case TC_07

Test Case ID	TC_07
Description	Logged-in user views past orders
Input	Click on “Order History”
Expected Output	List of past orders with status displayed
Actual Output	As expected
Status	Pass

Chapter 08:

Implementation

8.1 Component Diagram:

The component diagram provides a high-level overview of the software's physical components and their interactions. It visually represents how each part of the SMS Food Ordering System is organized and how components communicate with one another.

Components in the System:

- **Frontend (React.js):** Handles user interface and interactions.
- **Backend (Node.js, Express.js):** Manages business logic, user authentication, and API endpoints.
- **Database (MySQL):** Stores users, restaurants, orders, dishes, payments, etc.
- **Authentication Module:** Handles login, signup, and secure session management.
- **Payment Gateway Module:** Processes online payments securely.

Diagram:

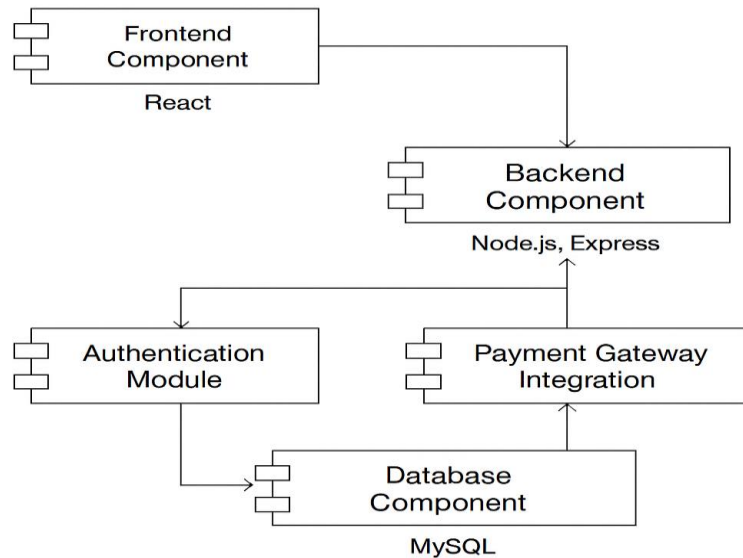


Fig.20: Component Diagram

8.2 Deployment Diagram:

A deployment diagram shows the physical deployment of software artifacts on nodes (servers, client devices). It demonstrates how the application is deployed in a real-world environment.

Deployment Environment:

- **Client Device (Web Browser):** Accesses the frontend via HTTP(S).
- **Web Server (Node.js):** Hosts the backend application, handles routing and API logic.

- **Database Server (MySQL):** Stores persistent data.
- **Cloud Hosting (e.g., Heroku, Vercel, AWS):** Hosts the full-stack application online.

Diagram:

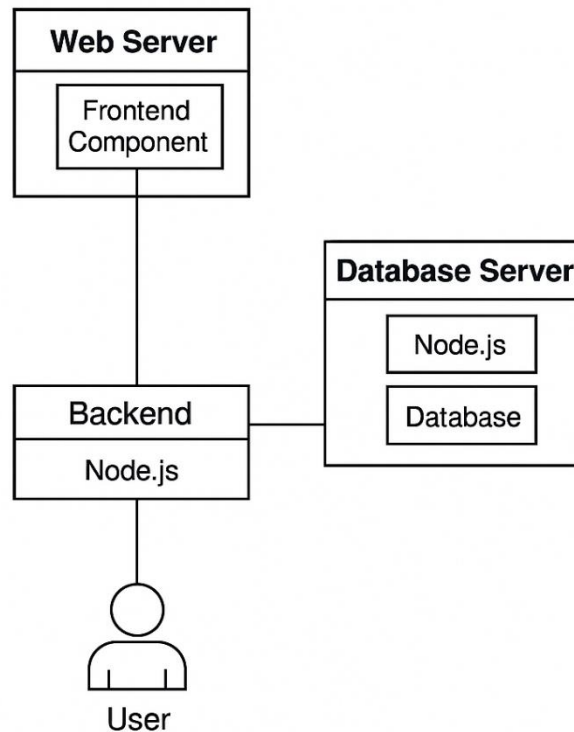


Fig.21: Deployment Diagram

8.3 Database Architecture:

In this project, a **Three-Tier Architecture** is used, which separates the system into three layers to improve maintainability, scalability, and performance.

1. Presentation Layer (Client/UI)

- Technology: HTML, CSS, JavaScript, React.js
- Function: Handles user interaction and displays data

2. Application Layer (Server/Logic)

- Technology: Node.js, Express.js
- Function: Processes requests, business logic, authentication, session management

3. Data Layer (Database)

- Technology: MySQL
- Function: Stores and retrieves data like user info, orders, menus, transactions

Advantages of 3-Tier Architecture:

- **Modularity:** Each layer is independent
- **Security:** Logic and data are isolated from frontend
- **Scalability:** Each tier can be scaled separately
- **Maintainability:** Easy to manage and upgrade

Diagram:

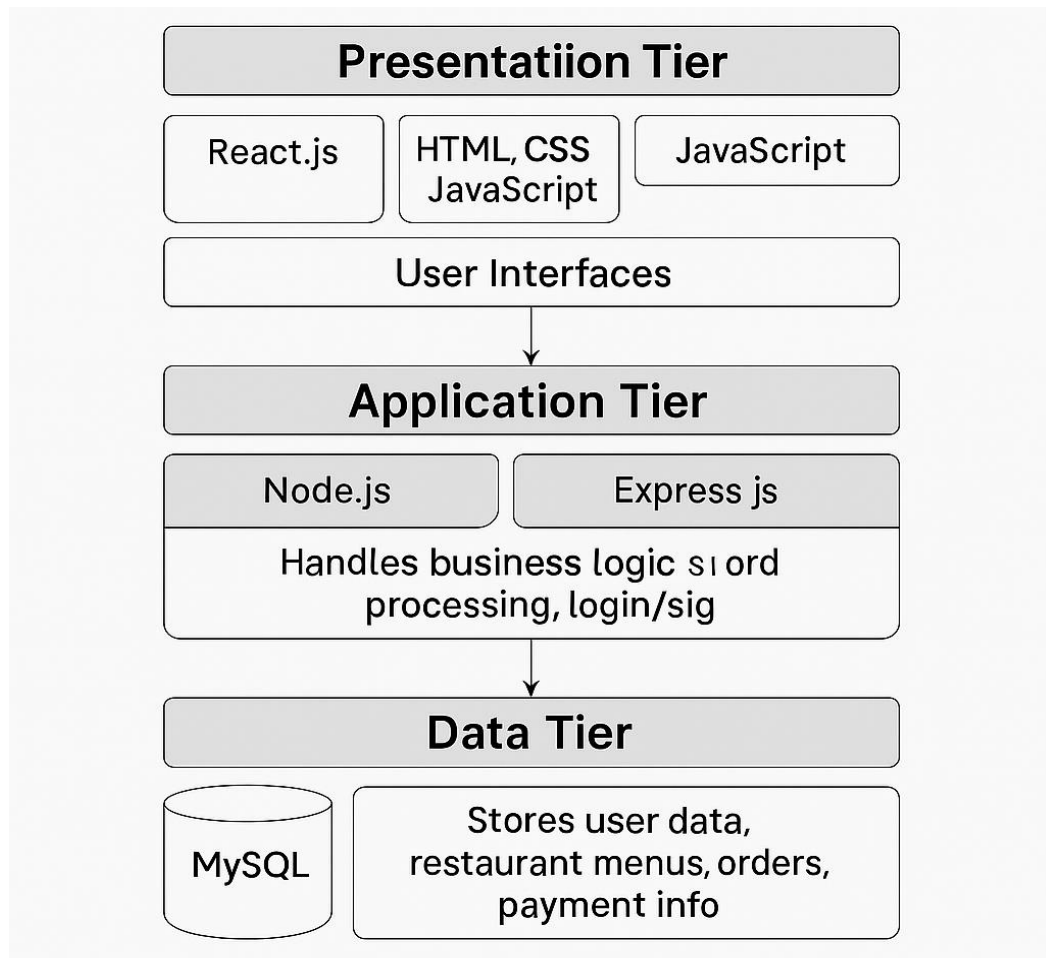


Fig 22: Three-Tier Architecture

Chapter09:

Tools and Technologies

9.1 Frontend Technologies:

The frontend (client-side) is the part of the system that interacts directly with the users. It was designed to be responsive, interactive, and easy to navigate.

ReactJS:

ReactJS is a popular JavaScript library developed by Facebook for building user interfaces, especially single-page applications (SPA). In our project, ReactJS was used to:

- Build reusable components such as the Navbar, ProductCard, FoodList, Cart, CheckoutForm, Reviews, etc.
- Manage state efficiently using useState, useEffect, and Context API for dynamic data rendering.
- Improve the performance and user experience by minimizing full-page reloads through virtual DOM and client-side routing (react-router-dom).

HTML5:

HTML5 is the standard markup language used to structure the content of the website. It was used to:

- Define semantic elements such as <header>, <footer>, <section>, <article>, and <nav>.
- Embed multimedia content, forms, and structure of the UI components.

CSS3:

CSS3 was used for styling the web pages, making them visually attractive and user-friendly. It helped in:

- Creating responsive layouts using Flexbox and Grid.
- Applying animations, transitions, shadows, and hover effects to improve UI.
- Using external stylesheets and modular CSS (SCSS/Modules) for organized styling.

JavaScript (ES6+):

JavaScript is the core scripting language of the frontend, responsible for making the website dynamic and interactive. It was used to:

- Handle events (e.g., button clicks, form submissions).
- Perform client-side validation for forms like login/signup and checkout.
- Fetch data from backend APIs and render it in real-time using asynchronous operations (fetch, axios, async/await).

9.2 Backend Technologies:

The backend (server-side) is responsible for handling business logic, processing data, managing user sessions, and communicating with the database.

Node.js:

Node.js is a powerful, event-driven, non-blocking I/O runtime environment that allows JavaScript

to be used on the server side. In this project, it is used to:

- Build a scalable and efficient backend server.
- Execute server-side scripts and connect frontend with database.
- Handle large numbers of concurrent requests without slowing down the server.

Express.js:

Express.js is a minimalist web framework for Node.js that simplifies the creation of web servers and APIs. It was used to:

- Design RESTful APIs to manage data (e.g., users, products, orders, payments).
- Route HTTP requests like GET, POST, PUT, and DELETE.

MySQL:

MySQL is a relational database management system (RDBMS) used to store and retrieve structured data. In our project, it was used to:

- Store essential data such as user accounts, food items, order details, delivery addresses, and payment records.
- Maintain data integrity through the use of foreign keys, indexing, and normalization.
- Run complex SQL queries to generate reports, summaries, and transaction logs.

9.3 Development and Testing Tools:

To ensure smooth development, effective version control, and thorough testing, the following tools were used:

- **Visual Studio Code (VS Code):**

A lightweight but powerful code editor that supports syntax highlighting, debugging, IntelliSense, Git integration, and extensions for React, Node.js, and MySQL development.

- **Git and GitHub:**

Git was used for version control to track code changes and collaborate effectively.

- Git allowed us to commit, branch, merge, and revert code safely.
- GitHub was used to host the repository, manage issues, and collaborate with team members.

- **Browser Developer Tools:**

These tools (built into Chrome, Firefox, etc.) helped with:

- Inspecting HTML and CSS layout issues.
- Debugging JavaScript errors.

- **Firebase Authentication (Optional/Integrated):**

Firebase Authentication was optionally integrated to allow users to log in using Google and Facebook accounts securely. It simplified:

- User account creation and session management.
- OAuth 2.0 integration for third-party sign-in.

APPENDIX A:

USER DOCUMENTATION

User Documentation for Restaurant Owner:

Introduction

Welcome to the **SMS Food Ordering System** – a web-based platform designed to help restaurant owners manage their menus, receive orders, and offer deals and discounts with ease.

This documentation will guide restaurant owners on how to use the system effectively to maintain their restaurant's presence and operations online.

Getting Started

- Visit the official web portal.
- Navigate to the “**Restaurant Login**” section.
- **Sign up by entering:**
 - Restaurant name
 - Email address
 - Password
 - Contact number
 - Restaurant address
- After successful registration, log in using the registered email and password.

Menu Management

- Access the “**Menu Management**” panel from the dashboard.
- You can:
 - **Add new menu items** by uploading images, names, prices, and descriptions.
 - **Edit existing items** to update prices, descriptions, or images.
 - **Delete items** that are no longer available.
- Use cuisine filters (e.g., Pakistani, Indian, Chinese, etc.) to organize the items.

Order Management

- View all **incoming customer orders** in the **Orders** tab.
- Each order shows:
 - Customer name and address

- List of ordered food items
- Quantity and price
- Special instructions
- You can:
 - Accept or reject orders based on availability.
 - Change the status of orders to *Preparing*, *Out for Delivery*, or *Completed*.
- Notifications alert you in real-time for new orders.

Managing Deals and Discounts

- Navigate to the “**Offers**” section from the admin dashboard.
- Create limited-time offers (e.g., 15% off for orders above ₹500).
- Set conditions such as:
 - Valid items or categories
 - Time duration
 - Promo codes (if any)

User Documentation for Customer:

Introduction

Welcome to SMS, your go-to web platform to browse menus, place orders, and enjoy your favorite meals from top restaurants.

This guide will help users understand how to navigate the system, explore food items, and complete their orders easily.

Getting Started

- Open the website.
- Sign up for a **new account** by entering:
 - Full Name
 - Email address
 - Password
 - Mobile number
- Log in with your credentials to begin exploring.

Menu Selection and Filtering

- Use filters such as cuisine type (e.g., Pakistani, Indian, Chinese, Thai, French, etc.).
- Browse through restaurants and view their menus.

- Add desired items to the cart.
- Modify item quantities or remove items from the cart before checkout.

Placing an Order

- After reviewing your cart, click on **“Place Order”**.
- Enter your delivery details or select saved address.
- Choose a **payment method**:
 - Cash on Delivery (COD)
 - Online Payment (Card or Wallet – if enabled)
- Confirm the order.

Tracking and Receiving Your Order

- Go to the **My Orders** section to track current orders.
- View real-time status: *Order Placed → Preparing → Out for Delivery → Delivered*
- Once the food is delivered, check the package, and enjoy your meal.

APPENDIX B:

References

The following sources were consulted during the research, analysis, and development of this project:

Tools & frameworks Documentation:

- **React.js Documentation** : <https://reactjs.org/docs/getting-started.html>
- Node.js Documentation <https://nodejs.org/en/docs/>
- Express.js Documentation : <https://expressjs.com/en/starter/installing.html>
- MySQL Official Documentation <https://dev.mysql.com/doc/>
- Figma Design Tool: <https://www.figma.com/resources/learn-design>
- React Router Documentation <https://reactrouter.com/en/main/start/tutorial>
- JWT (JSON Web Tokens) Authentication Guide <https://jwt.io/introduction>

Academic & Technical References:

- Acharya, K. (2024, May 2). Online Food Order System. SSRN. <https://ssrn.com/abstract=4814732>
- Ahmad, S., & Khan, M. (2022). A study on online food ordering systems and their impact on the restaurant industry. International Journal of Computer Applications, 184(34), 10–15. <https://doi.org/10.5120/ijca2022922644>
- Kaur, P., & Singh, H. (2021). Development of a web-based
- food ordering system. International Research Journal of Engineering and Technology (IRJET), 8(6), 4521–4526.
- Tandon, A., & Sharma, R. (2020). Design and implementation of an online food delivery system. Journal of Emerging Technologies and Innovative Research, 7(8), 154–160.

Online Sources & Articles:

- Statista. (2024). Revenue in the online food delivery market worldwide. <https://www.statista.com/outlook/dmo/eservices/online-food-delivery/worldwide>
- Food Fusion. (2023). Food ordering user interface design inspiration. <https://www.foodfusion.com/>
- W3Schools. (2025). React tutorial. <https://www.w3schools.com/react/>
- GeeksforGeeks. (2024). How to build an online food ordering system using MERN stack. <https://www.geeksforgeeks.org/>
- Medium. (2023). Best practices for developing an online food ordering app. <https://medium.com/>