INSTITUTE OF ENGINEERING AND MANAGEMENT

SALT LAKE SECTOR V, KOLKATA

ASSIGNMENT WORK OF DBMS LAB

GUIDED BY: Prof. DEEPSHUBRA GUHA ROY SIR

Online Food Delivery: Design a database for a food delivery service, including restaurants, menus, orders, and deliveries.

Submitted By:

Oindrila Paul:12022002016005

Debasmita Dutta: 12022002016055

ABSTRACT

With the emergence of such food delivery services through online platforms, the need for efficient and reliable database systems that can support dynamic complex structures of data has increased manifold. The project outlines a database system for an imaginary food delivery service through an online platform. All the main entities of restaurants, menus, orders, customers, and deliveries shall be discussed. Such an interaction process between the customers, restaurants, and delivery personnel will be made hassle-free while the information flows effectively. This project addresses all important issues regarding consistency, concurrency, and scalability in its design of an efficient relational database model. In this final implementation, all orders must be placed by the customers, the menus updated by the restaurants, and that all deliveries tracked and completed by the delivery personnel with reliability.

ACKNOWLEDGEMENT

I would like to extend my appreciation to everyone for their support in the successful completion of this project.

Firstly, I offer my thanks to my project supervisor, Prof. Deepshubra Guha Roy, for the guidance, support, and continuous encouragement he provided during the project. His support, patience, and insightful feedback dictated the direction of this work.

I would hereby take this opportunity to extend my sincere gratitude to the AIML Department and Institute of Engineering and Management, Kolkata for equipping me with all resources and making available very conducive learning atmosphere that has enabled me to successfully complete this project.

Special thanks go to my family and friends for their unwavering support, understanding, and encouragement during this journey. Their constant motivation and belief in my abilities made them instrumental in overcoming challenges.

TABLE OF CONTENT

CHAPTER: 1

INTRODUCTION

1.1 Overview

Online food delivery growth has been exponential, whereby most its users prefer to order food online or through applications rather than sitting down or cooking in the comfort of their homes. In turn, there has been a huge demand for strong database systems capable of handling big volumes of data such as restaurant information, menus, profiles of customers, orders, and deliveries. The objective of this project is to design a relational database model for the needs of the food delivery service with efficiency in data organization and reliability and easy interaction between customers, restaurants, and delivery persons.

The following are some factors of utmost importance while designing this database:

- Data Integrity: The database has to correctly capture and reflect customer profile information, order history, what's offered at restaurants, and status on delivery.
- Real-Time Data Updates:Customers can review real-time updates on the current options of a restaurant's menu, along with being able to track the status of their order and follow in real-time the preparations and delivery of orders.
- Concurrency and Scalability: The database has to smoothly accept and process concurrent transactions and scale with demand during busy hours when thousands of users are onboard. Good design should support thousands of concurrent users without any degradation of performance.
- Security and Access Control: Since the system has more than one user role (customers, restaurants, delivery agents, administrators), the database needs to have role-based access control so that only authorized users can view or update data relevant to their roles.
- Redundancy Minimization and Optimizing Performance: Using normalization techniques and optimizing query structures, the database design aims at minimizing data redundancy, making retrieval of data easier and optimizing the performance.

The report's approach is strictly based on the methodology of database design. Starting with a review of current industry practices, it moves through the stages of requirements gathering, schema design, and implementation, culminating in testing and performance analysis. This project demonstrates how a well-architected database can support a complex high-demand application by controlling data relationships and optimizing retrieval times to ensure prompt responsiveness for all users.

1.2 Problem Definition

The problem this project solves is to create a holistic and efficient database system for an online food delivery service that can manage the core aspects of such a service while at the same time ensuring there is real-time interaction among users. The system must, therefore, handle the following main challenges:

- Advanced data management: Dealing with a food delivery service requires handling highly varied data forms such as details related to restaurants, menu items, customer profiles, order histories, and all routes the delivery had to take. The system therefore has to be intelligent enough to handle different forms of data in such a way that it does not fragment or lose track.

- High Demand Concurrency: Peak hours, during meals, could bring high concurrent requests from customers, restaurants, and delivery agents. In such cases, the database had to support multiple simultaneous transactions preserving data integrity when different parts of the data are updated by a number of users at virtually the same time.

- Real-Time Data Requirements Customers request information related to menu items, pricing, estimated delivery times, and status about their orders in real-time. The menu should be updated in real time, and the status of orders should be tracked by delivery agents. The database must process data in real-time for updates to reflect any changes in real-time.

- Efficient order processing and tracking: Orders commonly involve multiple steps in a process such as confirmation, preparation, pickup, and delivery from placing to delivery. For this, it is important to trace in real-time at each stage while keeping customers and delivery staff accurately, up-to-date in the shortest possible time. Database support of an efficient order flow and tracking mechanism, whereby all parties concerned can view these updates at each step.

- Scalability for Future Growth: The restaurant database will grow in size-including restaurants, customers, and delivery personnel-as the food delivery service expands. One needs to ensure that scalability is provided in such a way that transaction volumes go up without major redesigns. Scalability is, thus, also horizontal-for instance, through replication or sharding-as well as vertical.

- Data security and role-based access: The database should ensure secure access to the data along with the control on role-based access. Each of these user roles, which include a customer, restaurant manager, delivery person, and administrator, should have varied levels of accesses along with different kinds of permissions to safeguard the data and prevent unauthorized changes.

The database will surmount the challenges and hence be reliable, efficient, and user-friendly to all users. Thus, it will serve the business objectives of the food delivery service. It lays a foundation that can be developed further: for example, using data analytics to predict demand, optimize routes of delivery, and make personalized recommendations.

1.3 Objective

The objective of this report is to develop and deploy a robust, scalable, and efficient database system for an online food delivery service. The proposed system has to meet complex requirements concerning real-time data management in order to provide continuous support in the seamless interaction between the customers, restaurants, and delivery personnel. The report will outline the design of a relational database model that captures specific aspects of operational challenges in delivering integrity both in data, security, and high performance. The aims can be summarized as follows:

- Establish an Integrated Database Structure: Define and design the database schema comprising all the core entities, such as restaurants, menus, customers, orders, and deliveries, including their inter-relations to promote logical flow and access.

- It ensures data integrity and consistency. Relational database principles such as normalization and foreign key constraints ensure that data entered into the database is accurate and free of redundancies. It also ensures data consistency associated with more than one entity, especially in a transaction.

- Support Real-Time Data Interactions: A system must be able to update real-time data seamlessly to enable order tracking delivery status and live updates for menus. The goal of this arrangement is to structure the database in a way that querying and updating can be done very fast to meet the expectation of users regarding instant feedback and correct information being retrieved.

- Ability to scale and high concurrency: Design the database with an ability for thousands of concurrent users, flexible scaling in both horizontal and vertical directions as demand increases. It needs support for multiple restaurants, thousands of orders at the same time, and hundreds of delivery agents without deteriorating performance.

- Implement Secure, Role-Based Access Control: Define the rights at which access can be granted to various user categories for instance customers, restaurant managers, delivery personnel, administrators). This would help in ensuring that data was secure and not accessed or changed by unauthorized persons.

- Optimize for Performance and Efficiency: Utilize indexing, query optimization and efficient transaction management in ways that improve query times and thus performance while withstanding heavy loads. This goal guarantees that users have the least possible delay when viewing or updating data.

- Future Improvements: Design the database to incorporate future capabilities of this kind in analytics, predictive models, machine learning algorithms, which can be used in demand forecasting, optimized routes, and personalized recommendations for customers.

These objectives would be achieved so that this report would present the complete outline of a working, flexible database which could meet the needs of an online food delivery service for the present needs while ensuring that all future expansions and technological up-gradations were taken care of.

1.4 Organization of the report

This report is to outline the whole process of designing, implementing, and testing a database for an online food delivery service. It guides the reader step by step in developing a system that meets the very complex requirements of a dynamic, user-interactive environment. The report has eight chapters, each focused on one aspect of the database development, ensuring logical progress from problem identification to solution deployment. Below is an overview of each chapter:

Chapter 1

This chapter introduces the emergence of the food delivery industry on the online platform and also picks on the databases to function well. The chapter describes the problems of large orders, customer management, and user-friendliness while browsing menus, ordering, and real-time tracking of orders. The discussion will cover this:

- Background of the online food delivery business.
- Incentive to design a scalable, reliable database.
- Essential features of the database that are integrity of data, high availability, and real-time updates
- A short declaration of who may use the database (customers, restaurants, delivery people, and administrators)

Chapter 2

The chapter is a literature review of previous studies, systems, and approaches related to database management for online food delivery and e-commerce platforms. This encompasses

- Common Database Structures and Comparing Systems Relational vs. NoSQL Databases Analysis
- Evaluation of Current Food Delivery Services, such as UberEats, Grubhub, and Potential Data Architectures
- Database Model Type-Pros and Cons of Various Models and Technologies: SQL vs. NoSQL Best Practices Regarding Data Normalization, Transactions Management, Indexing of Databases to Concurrency Problems Introduced by Scalability.
- Gaps and limitations in existing systems this project is meant to fill.

Chapter 3

In this chapter, a general description of the structure of the database is given. Every one of the core entities and their interrelations will be discussed. The core components are as follows:

- Entities and Attributes : Those are descriptions of the core entities involved in the system such as restaurants, menus, customers, orders, and deliveries and the respective attributes assigned to each.
- Restaurant : Restaurant name, address, contact numbers, hours of operation, etc .
- Menu Item : Item Id, name, description, price, and restaurant.
- Customer: Customer ID, name, address, contact information, payment method.
- Order: Order ID, customer ID, order status, order total, timestamps for order placement and completion.
- Delivery: Delivery ID, order ID, delivery personnel ID, status, timestamps for pickup and delivery.
- ERD: It is essentially a diagrammatic representation of a database showing the relationship between entities in it.
- Relationships: Explanation of relationship (one-to-many, many-to-many) of entities involved; eg, customers and orders, orders and delivery personnel, restaurants and menu items.

Chapter 4

The underlying theory of the database design that is being proposed will rely on some core principles of database design, along with a few of their implementations. A few of the discussed aspects are as follows:

- Normalization: Ensuring that the database is in a normalized state- usually to 3NF- thus, ensuring there is minimal redundancy, along with data integrity.
- Data Integrity: Discussion about the constraints and rules placed for maintaining data integrity, such as foreign keys, primary keys, and cascading delete rules.
- Indexes: Analysis of strategies used for indexing to make queries faster. Such discussions usually focus on the most important attributes such as customer ID, order status, and timestamps.
- Concurrency Control: Discussion of techniques needed to manage multiple users that may be accessing data simultaneously or trying to update data with techniques like transaction isolation levels.
- Security and Access Control: Ensure that user roles are established, e.g., customer, restaurant owner, administrator, with proper rights.

Chapter 5

This chapter details the methodology to be used in implementing the systems plus the tools and techniques used. Some of them are as follows:

- Requirement Gathering: Elucidating the functional and non-functional requirements of the database
- Technology Stack: Picking tools for managing the database such as MySQL/ PostgreSQL but possibly also using SQLAlchemy or Django ORM for interfaces.
- Schema Design: Translating the ERD into SQL scripts to create tables, relationship specifications, and constraints.
- Implementation: A step-by-step process in implementing the database, creating sample data, and testing.
- Testing and Validation: Techniques that would be used to validate the data integrity, such as sample queries, unit testing, and performance testing

Chapter 6

This will offer the results of the database implementation as well as a discussion of its performance. Sections include:

- Performance Metrics: The efficiency of the database in terms of speed, query performance, and efficiency under load will be discussed.
- Challenges and Solutions: Challenges encountered in design and implementation, such as managing relationships, concurrency, etc.
- Real-Time Testing: Real time results of scenarios being tested along the lines of a restaurant placing orders, tracking delivery, and updates on menus.
- Scalability Analysis: How the database can be scaled- for instance adding replicas or shards, as the number of users and restaurants grows with the service.

Chapter 7

This chapter summarizes overall effectiveness achieved in the database design by noting down whether such a design meets the objectives set at the start of the paper. The highlights include:

- A discussion of how the database could handle concurrent user actions and also ensure data integrity.
- Its ability to support real-time updates, efficient query performance, and scalability.
- Reflecting on the contributions of the project to the available databases with regard to online food delivery services.

Future improvements which would include advanced add-ons like analytics dashboards and usage of NoSQL databases for specific application use cases.

Chapter 8

This chapter compiles all the references used in the report. These include:

- Article and research papers on database design and food delivery systems
- Online resources and documents related to the tool and technologies applied.
- Books and Journals related to the topic, including database management systems and real-time application databases.

CHAPTER: 2

LITERATURE REVIEW

2.1 Introduction

The chapter on literature review reviews the existing studies, systems, and methodologies of designing and developing online food delivery services such as those with real-time transaction services. Industry practices, academic studies, and technological frameworks are consulted in this chapter to provide insight concerning the strengths and limitations of the current systems. Challenges that this project will address are identified, with which the online food delivery database will be able to yield a more robust and optimized solution.

Online food ordering is an essential component of the global economy, driven by the desire for convenience and accessibility. As people become more reliant on mobility and internet access, food delivery apps are mushrooming, with complex back-end systems needed to support this large number of users, orders, and interaction in near-real-time. The traditional restaurant ordering system used to focus on both dine-in and takeaway services, not requiring dynamic data management along with concurrent user interactions that are now an integral part of online delivery systems.

These studies show the need for effective database structure in handling such complex requirements. In theory, an optimized database is expected to manage different entities and their interrelation within various types of users such as customers, restaurants, and delivery people. Research these days is focused on effective data management focusing on data integrity, security, scalability, and real-time processing.

2.2 Existing System

This review of existing food delivery services provides information on the typical methodologies deployed to handle data in demand-intensive applications. Popular systems, such as UberEats, Grubhub, DoorDash, and Zomato, have used special database architectures to cater for millions of active users and thousands of simultaneous transactions. That way, they follow a combination of relational and non-relational database models in order to meet the demands of such high-traffic sites and diversified needs for data.

2.2.1. UberEats

Methodology:

- Hybrid Database Model: UberEats uses SQL (MySQL) for structured data, such as user profile and payment orders, and NoSQL (Cassandra) for unstructured data like real-time tracking of orders and personalize recommendations.
- Microservices Architecture: To be scalable UberEats, it follows a microservices model. Microservices model where several independent services-from executing the order operations; real-time delivery tracking; and processing payments-work together such that the level of scaling can be performed for each of these services based on demand and requirement.
- Data Partitioning and Sharding: UberEats distributed its database across different areas, localizing data access; it also brought down latency that may affect the performance among the various users.
- Messaging Queue for Real-time: As regarding real time updating orders and statuses UberEats employs Kafka as their messaging queue system that promotes real time communication between service ensuring a user is posted in a very instant manner.

Problems:

- Data Consistency Complexity: Another complication comes from using a hybrid database model in that there are challenges in the consistency of data between the SQL and NoSQL systems. For instance, it is quite challenging to update an order status in real time across both databases, which may lead to problems of synchronization.
- Latency At Peak Times: Although partitioning reduces latency in some respects, handling spikes in concurrent users, particularly at peak meal times, will still make services respond slower because access to and updates on the data have to be performed concurrently.
- Security over Data: Because there are several services accessing sensitive user information with a microservices architecture, it is difficult to secure such access and safeguard against improper leakage of data across services if there are many entry points.

2.2.2. Grubhub

Methodology:

- Grubhub mainly uses MySQL, a relational database to store customer details and order information and restaurant details besides handling transaction processing. It also has a pretty good guarantee of transactional consistency based on the properties of ACID.

- Apart from this, NoSQL is used selectively with MongoDB when something needs to be accessed speedily like in cases of real-time data needs such as actual time tracking of order status and delivery in real-time orders.
- Cache Management of Frequently Accessed Data : In Grubhub, the use of caching mechanisms like Redis is made for maintaining frequently accessed data such as popular restaurant information so that the load on the primary database could be minimized and the response times may be improved.
- Role-Based Access Control (RBAC): The role-based access ensures proper permission for data access to users, restaurants, and delivery personnel by Grubhub, hence improvements in security and less unauthorized access to the system.

Problems:

- Concurrency Issues At Peak Traffics: Heavy reliance on a single relational database causes bottlenecks at peak traffics because numerous users may simultaneously access the database, which has an effect on lower query response times.
- Data related inconsistencies in Hybrid Database: Although MongoDB is used for adding more online demand to data, as well as maintaining data consistency that is stored in both MySQL and MongoDB databases, is difficult, especially in terms of tracking orders and deliveries.
- Less Scalable Flexibility on Particular Services : As Grubhub's approach is relatively monolithic, flexibility in scaling particular components of the system like order management and payment processing has ramifications about efficiency of the overall system at high loads.

2.2.3. Zomato

Methodology:

- Relational Database (PostgreSQL): Zomato employs an advanced relational database PostgreSQL, for structured data regarding restaurants, menu items, profiles of users, and orders with potential for strong integrity of data and related querying functionalities.
- ElasticSearch for Rapid Searching: ElasticSearch with PostgreSQL is deployed by Zomato to index and enable fast searches of large datasets, including restaurants and items on their menus, thereby ensuring rapid response times when delivering search results to users.
- Geolocation Partitions: As most service providers do, Zomato maintains its data distribution by geographical locations, thus making it possible for the end-users to get exactly the restaurant listings as well as faster access to the data.

- Caching Layer for Frequently Accessed Data: For fast speed and minimizing the load on the database, Zomato uses Redis for frequently requested data, such as popular restaurants and their information.

Problems:

- Data Synchronization between PostgreSQL and ElasticSearch: Elasticsearch does indexing quite frequently to retrieve real-time updates from the PostgreSQL database. In case updates are not synchronized in time, there could be delay or data discrepancy.
- Scalability Issues During High Traffic: Though powerful at relational data management, PostgreSQL becomes unscalable if any substantial part of your read and write operations are done concurrently, especially during peak hours when several user bases touch the database simultaneously.
- Less Real-Time Capability: PostgreSQL is trans-actional by nature, and this guarantees data integrity but limits the database's ability to efficiently handle real-time tracking data. This is because updating order status and delivery tracking may not be instantaneous, which negatively impacts the user experience .

2.2.4. DoorDash

Methodology:

- Independent Databases Microservices: DoorDash uses a microservice architecture; it runs on services which have a database. This would mean that all of its services, whether it is for orders service or deliveries functions independently, thus scales much easier with less systemic failure.
- Scalability: Sharding and Replication. DoorDash uses sharding to spread data across multiple nodes and replication to ensure high availability and fault tolerance. This design supports high traffic loads and enables faster query responses.
- Real-Time Data Streaming: DoorDash uses real-time data streaming frameworks, which maintain immediate updates to the data and hence keep its customers posted about their orders.
- Role-Based Access and Security Control: DoorDash makes use of RBAC just like any other system, therefore the data is protected based on roles among users. Customers, restaurant people, and delivery agents must only receive relevant information.

Problems:

- Complexity Data Management Microservices Architecture Microservices architecture helps in improving scalability but it makes the management of data within services complex. Complex inter-service communication and coordination make data consistency more complicated to achieve.

- Data redundancy across databases: The separate usage of independent databases for every microservice in DoorDash results in the redundancy of data, raising the storage cost and further complicating updates that stretch across multiple services.
- Operational Overheads in Maintenance: Multiple databases per service mean more effort and resources on monitoring, backup, and synchronization.

CHAPTER: 3

SYSTEM DESCRIPTION

This chapter details the proposed database system for online food delivery as minutely elaborated on, including architectural design, flow diagrams that illustrate user and administrator interactions along with working principles that guide the system's functionality.

3.1 Introduction

This system serves to connect customers, restaurant management, and delivery staff in a robust and efficient platform for effective communication and smooth processing of orders. Customers can browse our menu, order, and track their orders in real-time from preparation to delivery. Restaurant staff can then manage the menu, accept orders, and monitor order fulfillment. The platform provides delivery personnel with the ability to view their assignments, update on their status, and optimize routes. It is built for handling high volumes of traffic, bringing real-time updates, and ensuring safe and secure data access to the parties involved-a more enhanced dining and delivery experience.

- Customers: The menu would be accessible, complete with item descriptions and price in easy-to-reach locations on the customer's device. All items to be selected, ordered, and submitted can be done at the touch of just a few buttons to ensure that the interface is simple and friendly for the user. Upon submission, customers will track the live status of their order for preparation to get dispatched to eventually arrive at their doorsteps. Real-time updates create transparency and assurance, thereby enhancing the customer experience at every touchpoint.

- Restaurant Management: The restaurant team has overall control over the menu and order processing. This includes updating what is on the menu, charge, and availability to best signify current stock and daily specials. An order is sent directly to the kitchen on placing; the order details, special instructions, are available for all staff members, and thus they can manage prioritization based on real-time demand. This reduces mistakes in the kitchen and accelerates order completion times. The team is also abreast of order status and can therefore interact with their customers in a responsive way to ensure a smooth service.

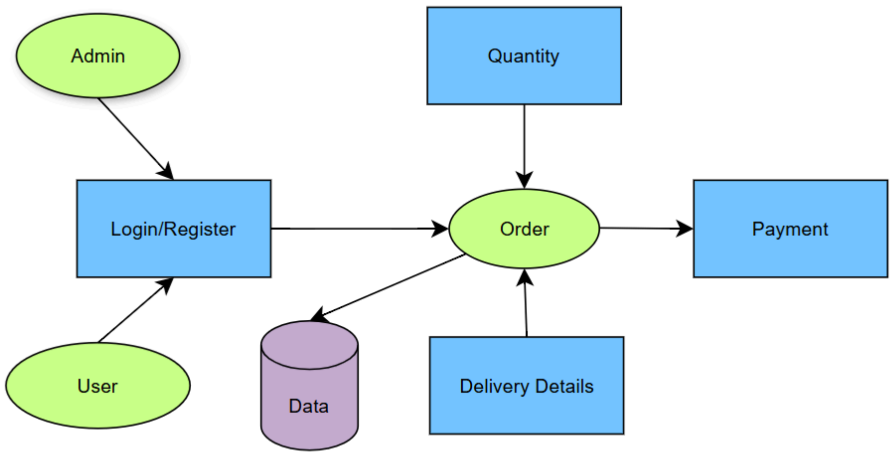
- Delivery Operations: The system offers clear and organized delivery views to delivery staff for all assigned deliveries, allowing them to see orders currently, view detailed delivery instructions, update the status of each order in real time ("on the way," "delivered"), which keeps customers informed while the restaurant team maintains oversight of the delivery process. Route management functionality helps delivery people design the best routes for them to be as short as possible and deliver goods as fast as possible, which means the number of successful deliveries on time rises.

- Real-time Updates and Notifications: It will run seamlessly with high traffic, and real-time updates are in place for all user types. The real-time updates notify customers, restaurant staff, and delivery personnel about changes in orders, so the whole world is in step with the stage of the order lifecycle. Customers will not be forced to wait and will appreciate the level of service provided, thus assuring them.

- Security and Data Integrity: The system is built in the perspective of securing the whole system, so that the data kept inside this application pertaining to the customers and the payment gateway details, as well as all the delivery addresses and other necessary information, will remain strictly secure. Role-based access control restricts the functionalities of each user, so that it will be safe and will make the system stability strong. The platform also prevents unauthorized access, so all the customers can be safe in their belief in handling the information securely.

3.2 Flow Diagram

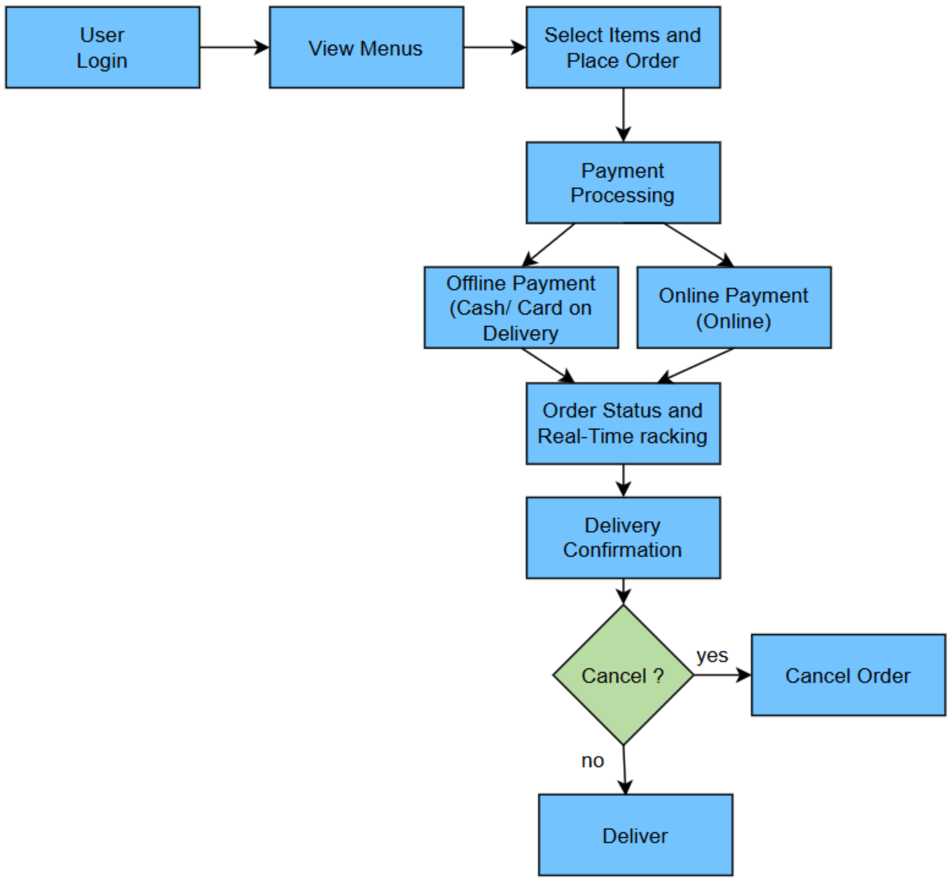
There is an architectural diagram that gives a high-level view of the system components and their interactions as follows:



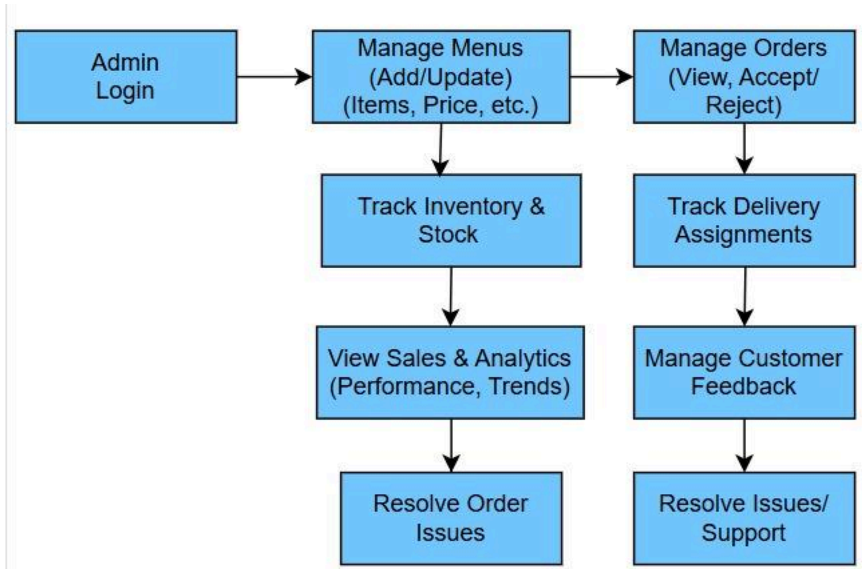
Explanation:

- Frontend: It is simply a web or mobile application where users interact with.
- API Gateway: It deals with all the API requests, routes them to the associated services, and applies role-based access control.
- Backend Application: This deals with core functionalities such as order processing, delivery tracking, and management of menu items and other information.
- Databases: Contains details related to users, orders, menus, and areas to deliver. The structure is designed to cope up with high concurrency and data integrity.
- Cache System: Redis is deployed to store frequently accessed items avoiding database load.
- Analytics Engine: This is data analytics that gives insights on what data is being analyzed to support such features as recommendations and demand forecasting.

3.3 Flow Diagram for User



3.4 Flow Diagram for Admin


Admin Login
Manage Menus (Add/Update) (Items, Price, etc.)
Manage Orders (View, Accept/ Reject)
Track Inventory & Stock
Track Delivery Assignments
View Sales & Analytics (Performance, Trends)
Manage Customer Feedback
Resolve Order Issues
Resolve Issues/ Support

3.5 Working Principle

The functioning of the online food ordering system relies on efficient data processing, real-time communication, and secure, role-based transactions.

- Relational is for structured data, and NoSQL is applied where there is high frequency, such as real-time tracking of parcels. Therefore, the relational model guarantees data consistency, while the NoSQL model facilitates fast scaling in data access.

- API-Driven Architecture: Every communication between frontend and backend shall be performed using RESTful APIs or GraphQL. The API Gateway shall be a secure gateway that ensures only authenticated and authorized users access the sensitive information regarding the restaurants.

- Workflow Order : After a customer has made the order, the system will process the order in relevance to the database of the restaurant, update the inventory as required, and inform the restaurant about the order. There will also be an assignment of the delivery people given their availability and location.

- For status changes and updates, the system uses message queues like Kafka. Thus real-time tracking is possible with the order status. The delivering personnel update the progress-status (pickup, en route, delivered), which is then automatically shown at the UI level.

- Data Caching: Frequently accessed data, such as popular restaurant listings and order status, can be cached with Redis, as it reduces the load on the database and improves response time for the user.

- Role-Based Access Control (RBAC): There are different roles such as a customer, restaurant, delivery personnel, and admin who have different rights. For example, only administrators can create new restaurants while customers can order and monitor the orders only.

- Analytics and Reporting: The system analyzes the related data including customer preference, peak ordering time, and how deliveries have been done. Reports would be generated and thus the admin could work out his decisions based on data for promotions, resources, and processes.

CHAPTER: 4

THEORETICAL ANALYSIS

4.1 Introduction to Tools used in Project

4.1.1 Introduction to HTML (Hypertext Markup Language)

HTML is the standard markup language that is used to create and structure content on web pages. HTML specifies the look and feel of a Web page through elements or tags, such as <h1>, <p>, <div>, etc. HTML is very important in building the skeleton of any website; and each webpage in the app relies on it to define headings, paragraphs, links, images, etc. The page's structure, such as menus, user profiles, and order summaries, is created using HTML in an online food delivery web application.

Advantages:

- Easy to learn and implement.
- Works fine in all major browsers.
- Structured and standardized definition of content.
- Good support across different platforms can be used with other languages as well for dynamic applications.

Syntax Example:

```
<html>
  <head>
    <title>Online Food Delivery</title>
  </head>
  <body>
    <h1>Annapurna ki Rasoi</h1>
    <p>Yes, We have the Best Main Course</p>
  </body>
</html>
```

Working: HTML defines parts of a document using elements known as tags. These tags are interpreted by the browser and then displayed on an ordered structure. Essentially, HTML only provides for structure that is later styled or manipulated by CSS and JavaScript.

4.1.2. Introduction to CSS- Cascading Style Sheets

CSS is the style sheet technology applied to style and beautify visually in HTML web pages. The contents of an HTML page can be controlled layout, color, font, and spacing, among other design-related elements, by using CSS. CSS can make a web application more user-friendly and also visually appealing, which is very important for the user experience of an online food delivery system. CSS can give a web application a professional and uniform look on each page and thus make it easier to use and enjoy the interface.

Benefits:

- Content is separated from presentation; hence, code is cleaner.
- It will be able to maintain a style throughout pages, and especially for pages that run many.
- Offer responsive designs and is compatible with all devices.
- Supports animations and transitions to make the page even more attractive.

Syntax Example:

```
</head>
<.[endif]-->

body {
  font-family: Arial, sans-serif;
  background-color: #ffffff;
}

h1 {
  color: #000000;
  text-align: center;
}
```

Working: CSS picks out the HTML elements that target them and aligns the style accordingly based on classes, IDs, or element tags. It uses selectors and properties to define the rules governing the style applied to each element of the page. CSS enables a web app to be beautiful and well-coordinated, where the same style is applied throughout all pages.

4.1.3. Introduction to JavaScript (JS)

JavaScript is the scripting language, which extends the richness and adds dynamic property of a web application. JavaScript is used in an online food delivery application to validate forms-for example, login/signup forms. JavaScript controls the functionality in the cart and dynamically loads new content without refreshing the page. The use of interactive elements will allow many things to happen, including real-time updates, such as the estimated delivery time, or enabling the searching and filtering of food items. It can be run both on the client side within the browser, as well as on the server-side by using technologies such as Node.js.

Advantages:

- Makes interactive and enhances user experience.
- It will reduce the server load because tasks are taken care of on the client side.
- Supports a lot of libraries and frameworks like jQuery, React, etc that make complex things easier.
- Supported by all modern browsers.

Syntax Example:

```
function showOrderSummary() {
  alert("Your order has been placed!");
}
```

Working: The web application can be used to respond to a user's interactions, like when the user clicks buttons or submits forms. The application can update content without a page refresh and can dynamically load data, so it is better for performance and user experience. It can also interact with the back end to fetch or send data asynchronously using AJAX.

4.1.4. Introduction to PHP (Hypertext Preprocessor)

PHP is a server-side scripting language that assists in developing dynamic content by having an interaction with databases and performing backend logic. For a food ordering app in an online delivery context, PHP can be used to handle user authentication, form processing, retrieving data from a database to show all the items available on the menu or their order history, and all other server-side operations. With PHP, you can dynamically generate HTML when there is user interaction or even some database query, which is very useful in giving personalized content for each of your users.

Advantages:

- It is possible to generate dynamic content.
- It integrates well with a wide variety of databases, especially MySQL.
- Codes are securely used with server-side scripting for protection of sensitive data.
- Open source and widely accepted by web hosting providers.
-

Syntax Example:

```
<?php
$username = "user123";
echo "Welcome, $username!";
?>
END
```

Working: PHP code is executed on the server, before it sends the web page to the browser. It may also access a database, process forms, and produce HTML that contains dynamic content. For instance, in food delivery web application, PHP would possibly perform user authentication, management of orders, and account operations by interacting with the database.

Connecting PHP Application with MySQL Database:

1. Creating the MySQL Database Confirm that MySQL has been successfully installed and is running. Create a database and mandatory tables, for example, you would have tables of users, orders, menu_items etc.
SQL command to create a database and a table:

```
CREATE DATABASE food_delivery;

USE food_delivery;

CREATE TABLE users (
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) NOT NULL,
password VARCHAR(255) NOT NULL,
   email VARCHAR(100) NOT NULL
);
```

2. Database Credentials in PHP: Store the database credentials, including your host, username, password, and database name, into your PHP file when you want to connect with MySQL. Always best to keep these credentials in an external configuration file for safety.

```
// db_config.php
$host = "localhost";     // Database host
$user = "root";          // Database username
$password = "";          // Database password
$database = "food_delivery"; // Database name
```

3. PHP Connection to MySQL Database : Use mysqli or PDO (PHP Data Objects) extension to connect to your database from PHP. Here's how you can do it with each of them:

Method 1: mysqli

```
// Include the database configuration file
include 'db_config.php';

// Connect to database
$conn = new mysqli($host, $user, $password, $database);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: ". $conn->connect_error);
}
echo "Connected successfully";
```

Method 2: Using PDO

```
// Include the database configuration file
include 'db_config.php';

try {

   $conn = new PDO("mysql:host=$host;dbname=$database", $user, $password);
   // Set the PDO error mode to exception
   $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   echo "Connected successfully";
} catch(PDOException $e)
echo "Connection failed: ". $e->getMessage();
}
```

4. Perform Database Operations

After establishing the connection, you can perform operations such as querying, inserting, updating, and deleting data.

Example: Fetching Data (Using mysqli)

```
$sql = "SELECT * FROM users";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
   // Output data of each row
   while ($row = $result->fetch_assoc()) {
echo "id: ". $row["user_id"]. " - Name: ". $row["username"]. "<br>;";
}
} else {
   echo "0 results";
}
```

Example: Inserting Data (Using PDO)

```
$username = "new_user";
$password = password_hash("secure_password", PASSWORD_DEFAULT);
$email = "user@example.com";
ylan;
$stmt = $conn->prepare("INSERT INTO users (username, password, email) VALUES
(:username, :password, :email)");
$stmt->bindParam(':username', $username);
$stmt->bindParam(':password', $password);
$stmt->bindParam(':email', $email);
$stmt->execute();
ylan;
echo "New record created successfully";
```

5. Close the Connection: After performing database operations, it is always a good practice to close the connection.
Using mysqli: $conn->close();
Using PDO: $conn = null;

4.1.5. Introduction to Bootstrap

Bootstrap is the open source front-end framework for responsive and aesthetically worthy development of web pages. Among several advantages of using Bootstrap, one is that it comes with an already defined CSS and JavaScript components like navigation bars, buttons, grids, and forms that can readily be spread across devices at the same platform-designer will maintain uniformity both on a desktop and tablet/smartphone platform. Implementation of such an online food delivery web application with the use of Bootstrap would quickly provide layouts that are cross-device responsive, using which any customer would seamlessly browse the menus, place the orders, and trace their deliveries.

Advantages:

- Responsive design is made accessible through the grid system.
- It has a library of pre-styled components that save on development time.
- Ensures the design is uniform in all devices.
- It can be used to match the branding and style of a given web application.

Syntax example:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

<div class="container">
  <div class="row">
    <div class="col">
      <h1>Order Food</h1>
    </div>
  </div>
</div>
```

Working: Bootstrap is a pre-fabricated collection of CSS and JavaScript components that can easily be added to your HTML. Classes are for grid layouts, buttons, forms, navigation bars, and much more and are pre-optimized for desktop and mobile screens. In the case of an online food delivery app, it may help designers significantly simplify the design process while ensuring that an app remains accessible on multiple screen sizes.

4.1.6. Introduction to MySQL

MySQL is a relational database management system with data stored in an organized and efficient way, so it can retrieve data quickly. In an online food delivery app, MySQL will store all user's details, various menu items, and orders along with their payment details. This has to be managed with huge volumes of data with good performance. PHP can interact with MySQL to add, update, or retrieve data to allow for a seamless back-end experience that supports smooth app functions with data integrity.

Benefits:

- Popular, open-source, and highly reliable.
- Scalable in handling large data volumes.
- Secure and interoperable with other languages like PHP.
- Efficient data management of web applications.

Syntax Example (SQL Query):

```
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  customer_name VARCHAR(50),
  order_date DATE,
  total_amount DECIMAL(10, 2)
);

INSERT INTO Orders (order_id, customer_name, order_date, total_amount)
VALUES (1, 'Alice', '2024-11-10', 29.99);
```

Working: MySQL MySQL is a relational database management system that stores data in tables. To illustrate this, for an online food delivery app, MySQL can handle user information, menu items, orders, and transaction histories. It accepts SQL queries from PHP or another server-side language, thus enabling the web app to add, retrieve, or update data as necessary.

4.1.7 Introduction to XAMPP

XAMPP is free, open-source software which helps in establishing a local web server for development. XAMPP is an abbreviation for Cross-Platform (X), Apache (A), MySQL (M), PHP (P), and Perl (P). These components are used for the development of web applications.

- Apache: This is one of the powerful and most used web servers that serves web pages to the users.
- MySQL: The most widely-used and popular open-source database management system for storing and retrieving data.

- PHP: A server-side scripting language that allows the generation of dynamic content along with communication with the server.
- Perl: Sometimes used for server-side scripting.

Benefits of XAMPP:

- Easy to Install: In a single package, XAMPP includes everything you'd require to start developing a PHP application locally.
- Cross-Platform: It supports Windows, Mac, as well as Linux.
- Local Development: You can run server-related work on your local machine without needing to be online.
- Easy Management: Using the control panel in XAMPP, you can easily open or shut down Apache, MySQL, and other services.

Usage in Online Food Delivery Web Application Development: XAMPP is a complete local server environment through which one can develop test and debug web applications. In the context of an online food delivery project testing for all functionality such as user registration placing of order and database queries would be possible on the local environment before they are deployed onto a live server.\

4.1.8 Introduction to phpMyAdmin

phpMyAdmin is a web-based tool for the administration of MySQL databases via XAMPP. It provides an interface to handle MySQL databases. Since it's a GUI-based software, you can interact with the database easily without having to type SQL commands.

Important Features of phpMyAdmin:

- Databases management: Develop, modify, or drop databases and tables
- User Management: Helps manage MySQL user accounts, permissions, and passwords
- Data Manipulation: Possibility to perform easy data entry, update, and delete operations.
- Query Execution allows you to execute SQL queries to your database directly and is helpful in advanced database management.
- Backup and Export provide options to back up or export databases. This feature can be helpful in migrating and protecting data.

Usage in Building an Online Food Delivery Web Application: for the online food delivery application, phpMyAdmin turns complicated jobs of database handling easy. They can create tables such as users, orders, menu_items, etc. also add data to them and see and update records whenever they want to. It is very helpful to non-technical users, as it lets them have a user-friendly interface to create, update, and manage the database backend for the application.

How XAMPP and phpMyAdmin Interact
When you have XAMPP installed and running, it gives access to a local Apache server and a MySQL database. phpMyAdmin can even be accessed using a web browser at http://localhost/phpmyadmin/. phpMyAdmin then provides the ability to interact with the MySQL databases. Thus, XAMPP and phpMyAdmin are the perfect pair for developing, testing, and managing the database of a local PHP web application - like the online food delivery app.

4.1.9 Introduction to Apache Server

The Apache HTTP Server, simply known as Apache, is one of the most widely used servers worldwide. It is free to use, open-source, and maintained by the Apache Software Foundation. In essence, Apache fills the gap between the actual server and the client's browser in delivering web content to users.

How Apache Works: When a user types a URL inside the browser to open it, this would send a request to the server hosting the website. Apache receives this request, locates the requested files, like for example HTML, CSS, PHP, or other resources, and would then send them back to the requesting user's browser for purposes of viewable display. It is the task of Apache to process the request, execute any server-side script as might be required, and deliver the content.

Key Features of the Apache Server:

- Apache Server is cross-platform; it can be installed in one of the most widely used systems, Windows, Linux, and macOS.
- Multilingual support: The server supports scripts in PHP, Python, Perl, and even other server-side languages.
- Modular design: Any module can be added to a server to extend its functions, including SSL for HTTPS and URL rewriting and lots more.
- Configurability: The httpd.conf configuration file can be customized to create settings and control access to set up virtual hosts and define rules governing security.

Advantages of Apache Server:

- Reliability and Stability: Stableness is engraved in the product; it is one of the most trusted web servers for small to large-scale applications.
- Customizability: Apache has a modular framework, which provides extensive documentation to customize as per different application needs.
- Security: Provides excellent security functionalities such as SSL/TLS encryption and access controls.
- Community Support: It is one of the oldest web servers ever developed and is supported by a big community that ensures appropriate resources and support.

Usage in Online Food Ordering Web Application: In the case of an online food ordering application, Apache would also work as its web server. It will process incoming requests for loading pages, login processing for users, order processing, and fetching data to display the menu. With the use of PHP for backend processing and MySQL for storage of its database, Apache brings a responsive and secure user experience in the way it serves web pages along with dynamic content.

Apache can be bundled with XAMPP. This makes it relatively easy to use as a local server during the development and testing phases of a web application project.

CHAPTER: 5

METHODOLOGY

- Using the data from MySQL, the admin and by user will store that.
- By using the Search Option, the user can search for the goods.
- For front-end development – HTML, CSS, JS, BOOTSTRAP.
- Back End – PHP, MySQL & APACHE SERVER.

Modules -

Users can :

- Search Food items
- Filter Food items
- Add items in cart, update items in cart, delete items from cart
- Make items payable, Checkout, Shipping
- Cancel Order
- Raise Support Tickets
- Edit their own profile

Admin can :

- Create, Retrieve, Update Delete Food items
- Create, Retrieve, Update Delete Users
- Process Orders, Cancel Orders, Refund Money
- Search Food items
- Filter Food items
- Add items in cart, update items in cart, delete items from cart
- Make items payable, Checkout, Shipping
- Resolve Support Tickets

CHAPTER 6:

RESULT AND DISCUSSION

The well-structured database will allow the management of all critical aspects such as user profiles, restaurant information, orders, payments, and delivery. This is because Users Database stores customer details for account management and personal experiences. Restaurants Database stores restaurant information, menús, pricing, and customer reviews so that users can browse and select their preferences. This database records each and every order placed till its fulfillment, about items and status of an order. This database deals with transaction data so that the transaction as well as billing processes run smoothly. The delivery personnel and tracking are managed by the Delivery Database so that the overall process goes down to the last second and without any errors for the timely and accurate order fulfillment. The databases used are as follows:

6.1 PHP

The PHP files of this Online Food Ordering Web App project address several major areas such as user authentication, orders management, ticketing, and connectivity to the database. There are significant PHP files identified below, and most likely roles for them:

1. Core Application Files
   - onlineorder.php: it likely contains the primary interface that users interact with in the application; the content might involve order selection and processing.
   - send-contact.php: the file handles user inquiries or contact form submissions.

2. User Account and Authentication
   - account/login.php: Provides the functionality of user login.
   - account/register.php: Manages a new account's registration for users.
   - account/logout.php: Logs out users and provides management of session expiration.

3. Order Management
   - account/orders.php: Demonstrates orders unique for each user.
   - account/all-orders.php: Perhaps a page where admins get the view of all orders within the system.
   - account/place-order.php: Provides functionality for placing a new order for users.
   - account/confirm-order.php: Manages confirmation of an order from the user's side.
   - routers/order-router.php: Router file that might be responsible for processing any backend requests with regard to orders, probably in the form of AJAX.

4. Ticketing and Suuport
   - account/tickets.php: Controls the support tickets from users.
   - account/view-ticket.php and view-ticket-admin.php: Provides the ticket details to the user as well as to the admin
   - routers/add-ticket.php and ticket-status.php: Adds new tickets and controls status updates on tickets.

5. Admin and Management Pages
   - account/admin-page.php: Probably the central hub for admins to administer the application
   - account/users.php: User management tool for administrators
   - routers/add-users.php: The central hub also allows admins to add users from within the control panel

6. Database Connection and Wallet
   - includes/connect.php: Central file for connecting the database.
   - includes/wallet.php. Apparently, it contains wallet features which involve keeping track of user transactions' balances.

6.2 SQL

This database schema supports a basic e-commerce or inventory management application that incorporates order processing, management of items, user interaction, and handling support tickets. The schema consists of eight primary tables, each targeted to hold and manage different kinds of data integral to the functionality of the application at hand. Here's a quick overview of each:

- items: This table includes details of available products, such as its ID, name, price, and deletion flag.
- orders: Orders table helps to know the customer details, address, describing the order, date of order made, type of payment, total cost, status, and a deletion flag.
- order_details: It holds the line item information of an order. This table establishes the connection of each item with its quantity to a particular order.
- tickets: It manages support tickets generally raised by users and has fields for its subject, description, status, and type.
- ticket_details: Stores the number of updates or responses in each ticket with the relationships of the involved users.

- users: It stores user information like ID, role, name, username, password, email, contact details, and verification status.
- wallet: Every customer/user is linked to a wallet which links a customer ID with a wallet ID representing such a wallet.
- wallet_details: It contains all the details of every wallet including wallet ID, account number, CVV, and balance.

6.3. Workings

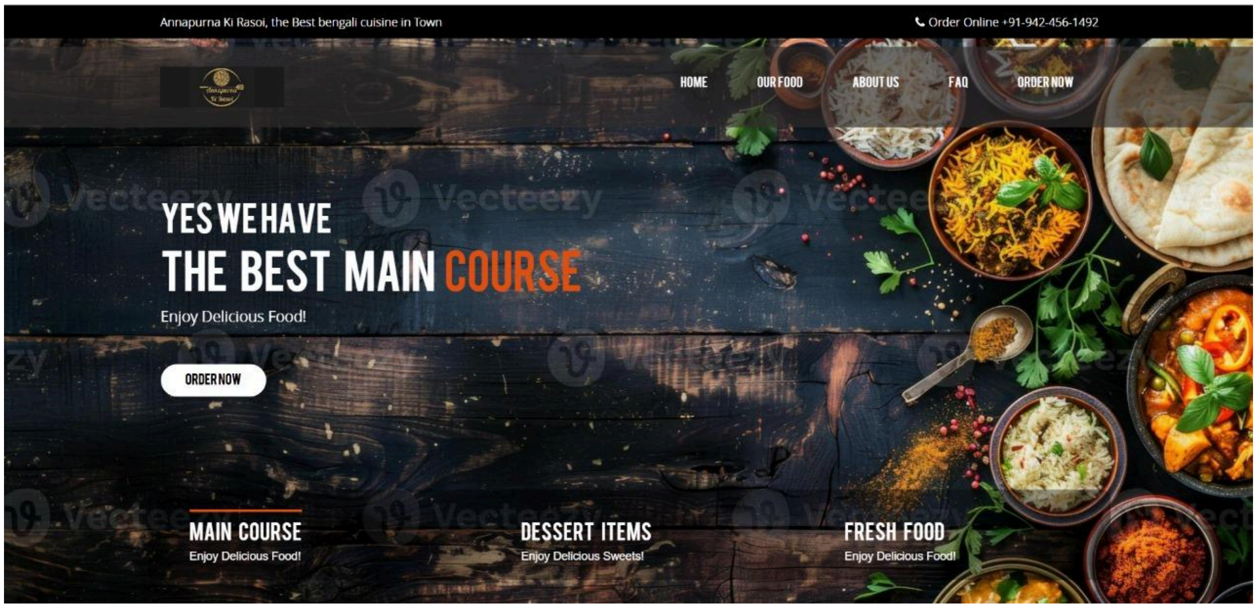
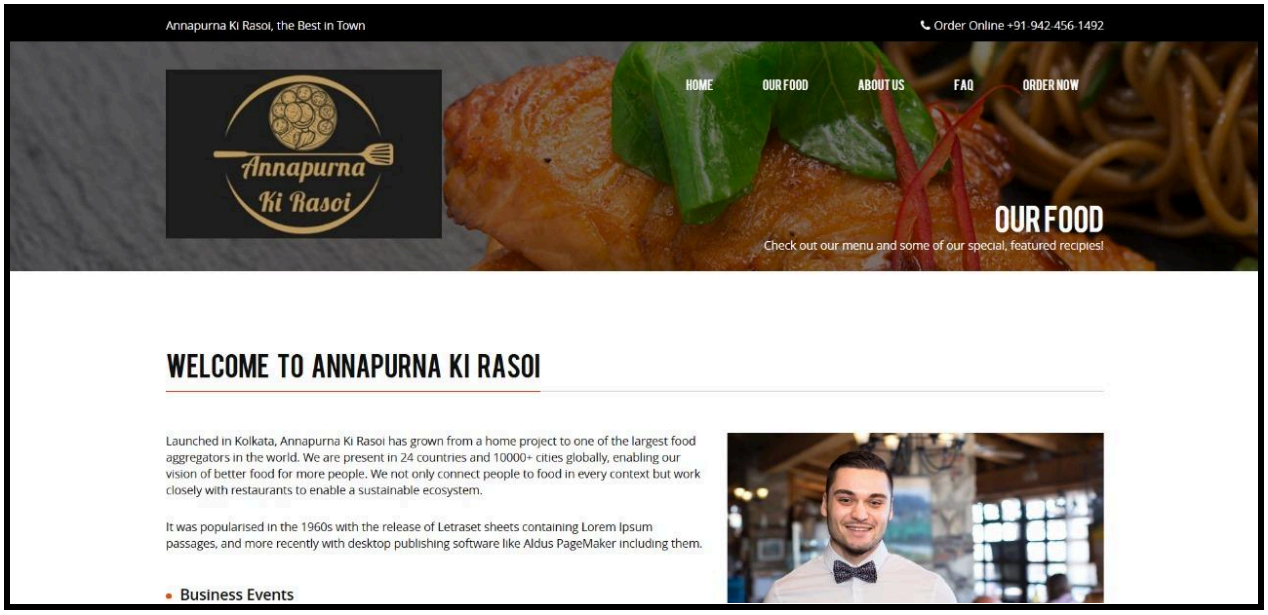


Fig. 6.3.1. Index


Annapurna Ki Rasoi, the Best in Town

Order Online +91-942-456-1492

OUR FOOD

Check out our menu and some of our special, featured recipies!

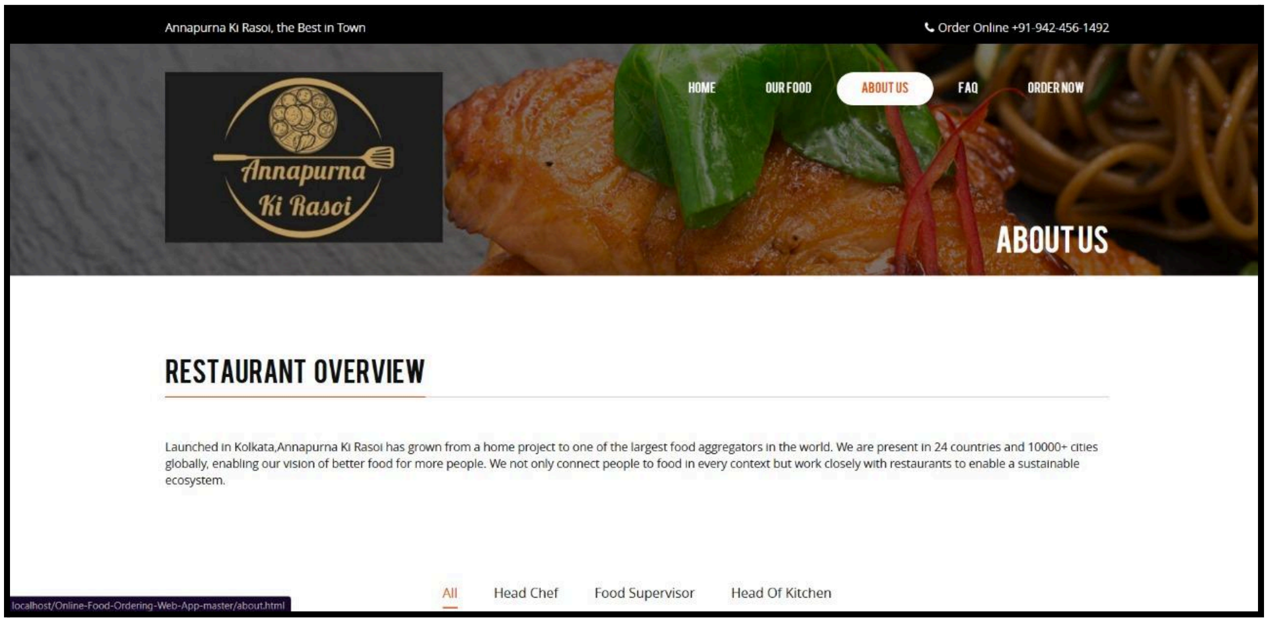
WELCOME TO ANNAPURNA KI RASOI

Launched in Kolkata, Annapurna Ki Rasoi has grown from a home project to one of the largest food aggregators in the world. We are present in 24 countries and 10000+ cities globally, enabling our vision of better food for more people. We not only connect people to food in every context but work closely with restaurants to enable a sustainable ecosystem.

It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including them.

- Business Events

Fig. 6.3.2 Our Food


Annapurna Ki Rasoi, the Best in Town

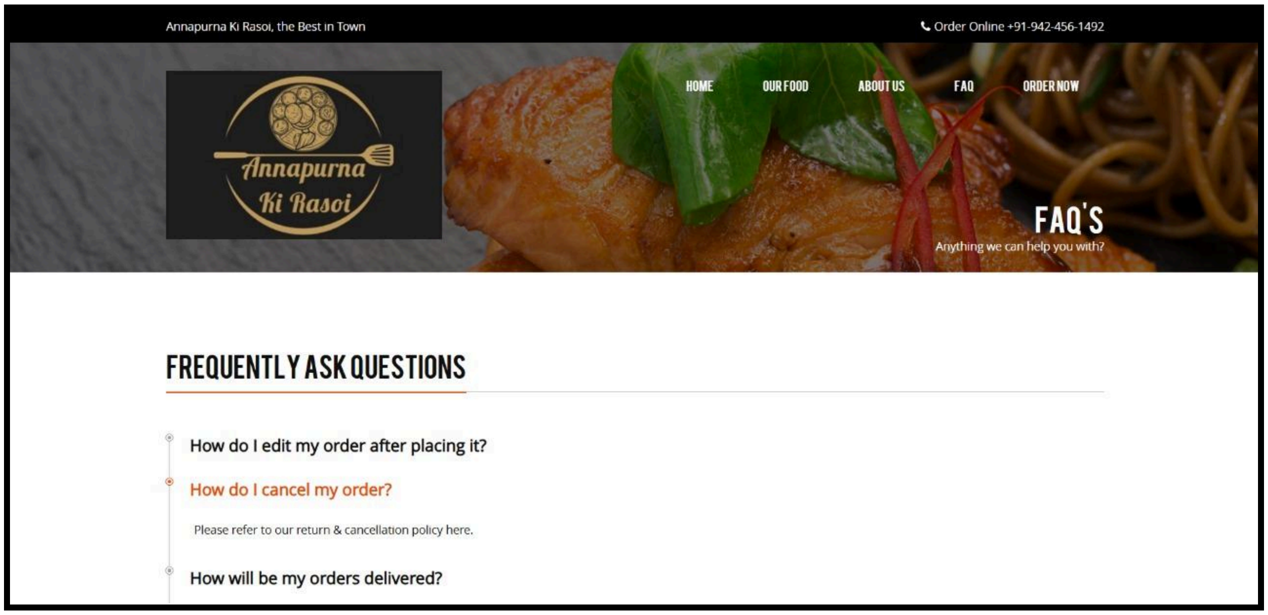
Order Online +91-942-456-1492

ABOUT US

RESTAURANT OVERVIEW

Launched in Kolkata,Annapurna Ki Rasoi has grown from a home project to one of the largest food aggregators in the world. We are present in 24 countries and 10000+ cities globally, enabling our vision of better food for more people. We not only connect people to food in every context but work closely with restaurants to enable a sustainable ecosystem.

Fig. 6.3.3. About us


Annapurna Ki Rasoi, the Best in Town
Order Online +91-942-456-1492
Annapurna Ki Rasoi
HOME
OUR FOOD
ABOUT US
FAQ
ORDER NOW
FAQ'S
Anything we can help you with?

FREQUENTLY ASK QUESTIONS

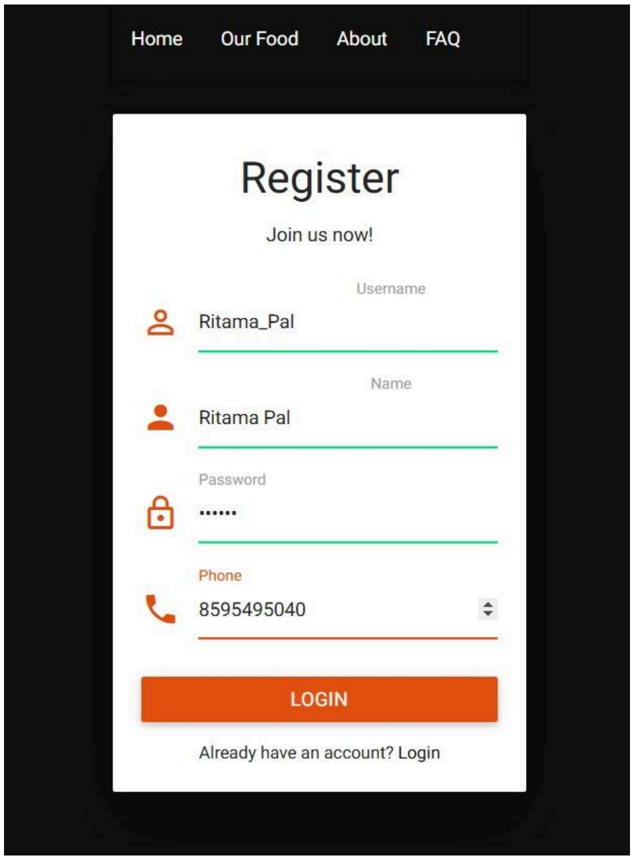
How do I edit my order after placing it?

How do I cancel my order?

Please refer to our return & cancellation policy here.

How will be my orders delivered?

Fig. 6.3.4. Frequently Asked Questions (FAQs)



Register

Join us now!

Username	Ritama_Pal
Name	Ritama Pal
Password	••••••
Phone	8595495040

Already have an account? Login

Fig. 6.3.5. User Register

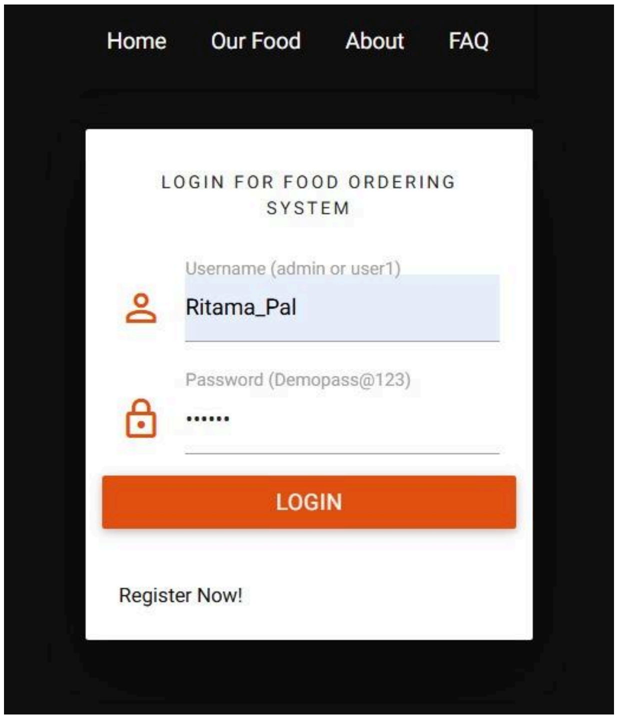
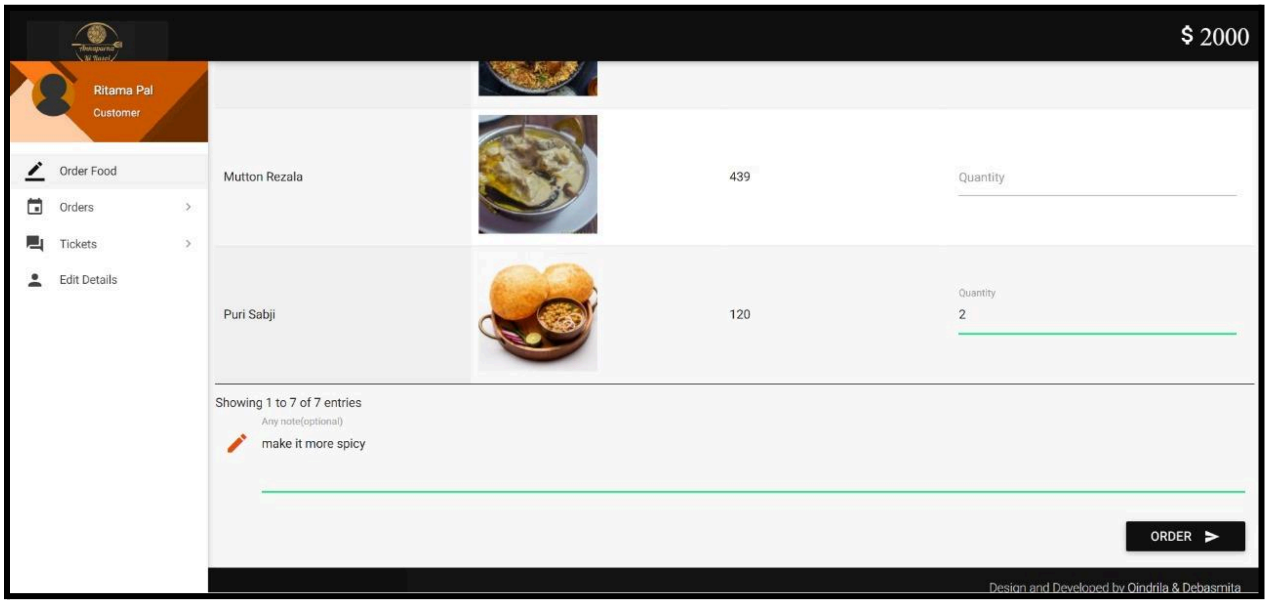


Fig. 6.3.6. Login



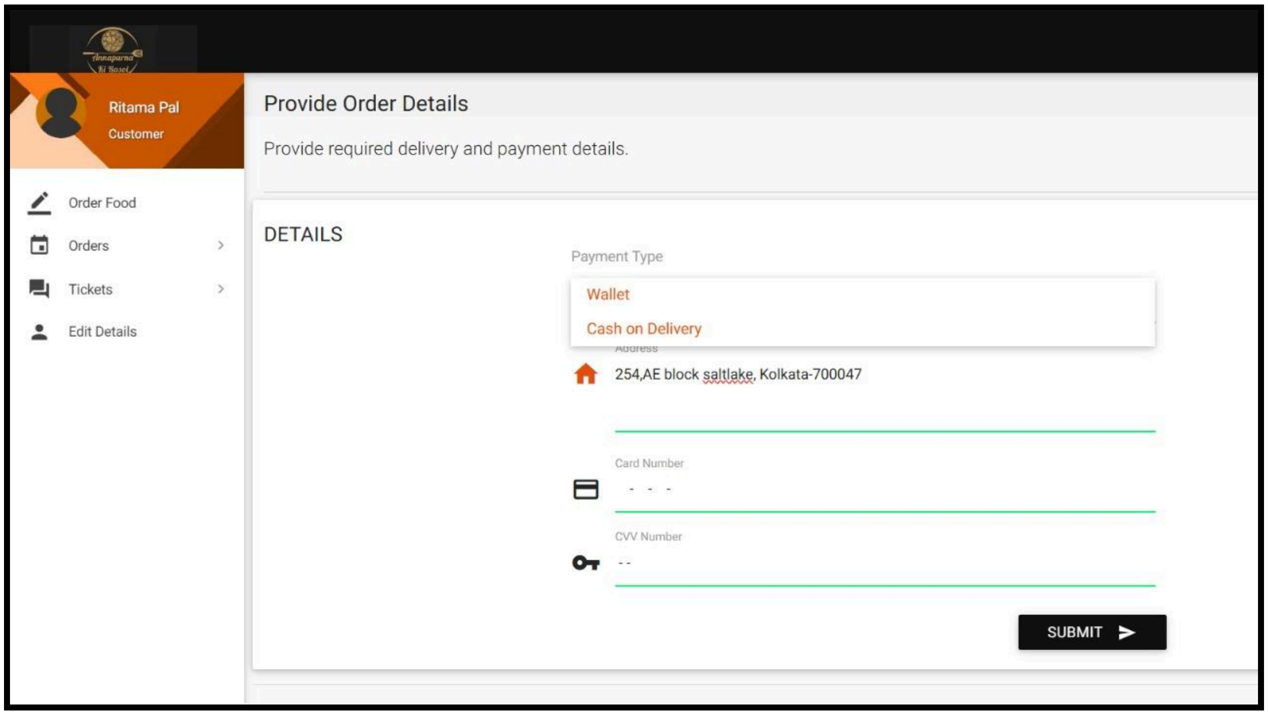
Mutton Rezala		439	Quantity
Puri Sabji		120	Quantity 2

Showing 1 to 7 of 7 entries

Any note(optional)

make it more spicy

Fig 6.3.7. User Order



Provide Order Details

Provide required delivery and payment details.

DETAILS

Payment Type

Wallet

Cash on Delivery

Address

254,AE block saltlake, Kolkata-700047

Card Number

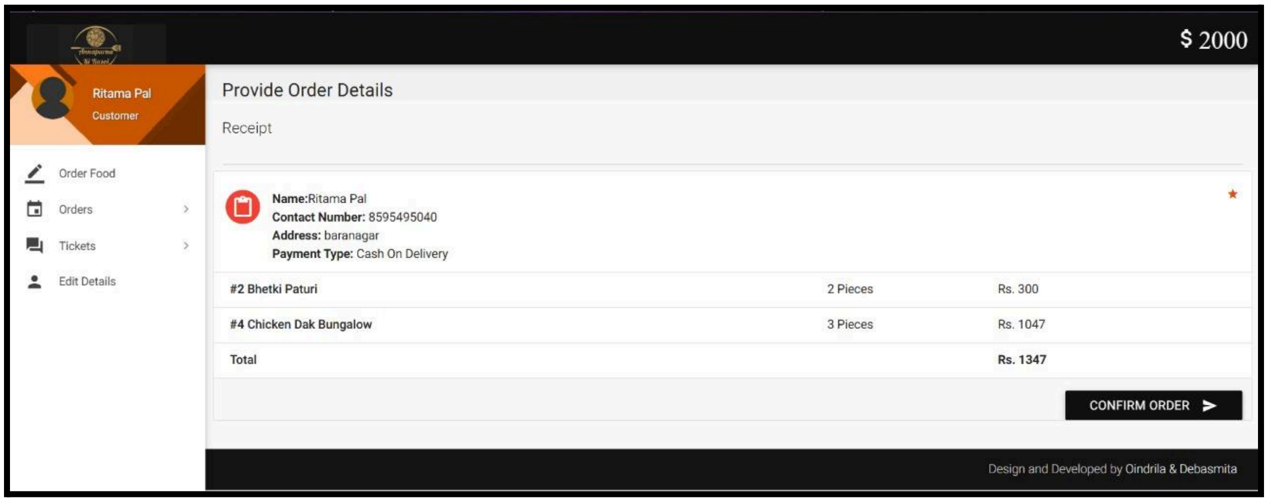
- - -

CVV Number

- -

SUBMIT

Fig 6.3.8. Payment Details



Provide Order Details

Receipt

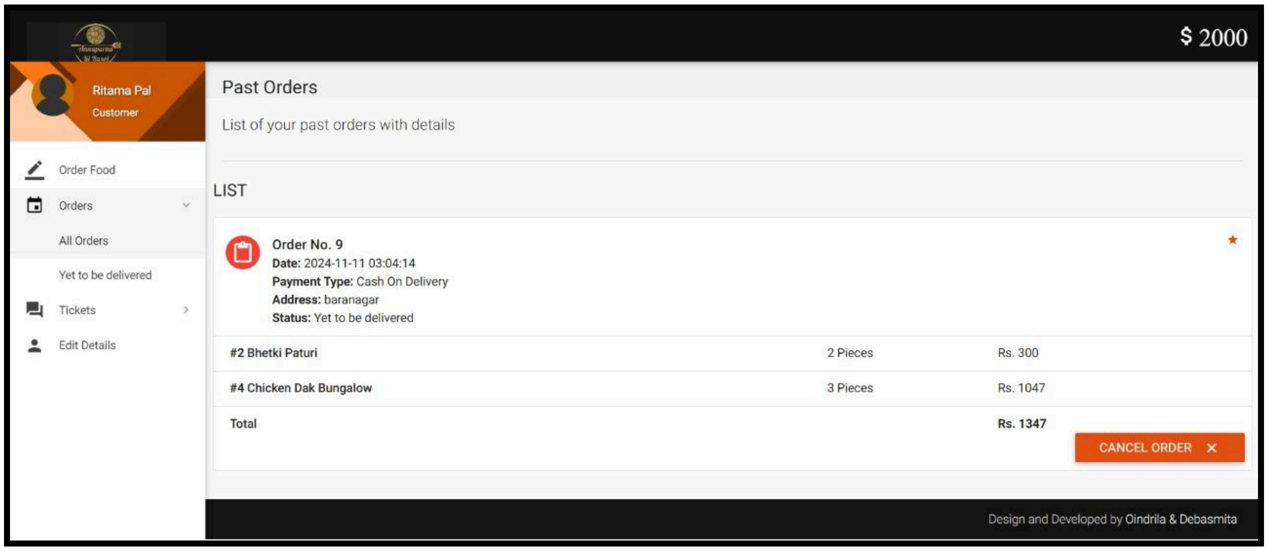
Name: Ritama Pal
Contact Number: 8595495040
Address: baranagar
Payment Type: Cash On Delivery

#2 Bhetki Paturi	2 Pieces	Rs. 300
#4 Chicken Dak Bungalow	3 Pieces	Rs. 1047
Total		Rs. 1347

CONFIRM ORDER

Design and Developed by Oindrila & Debasmita

Fig 6.3.9. COD Payment



$ 2000

Ritama Pal
Customer

Past Orders

List of your past orders with details

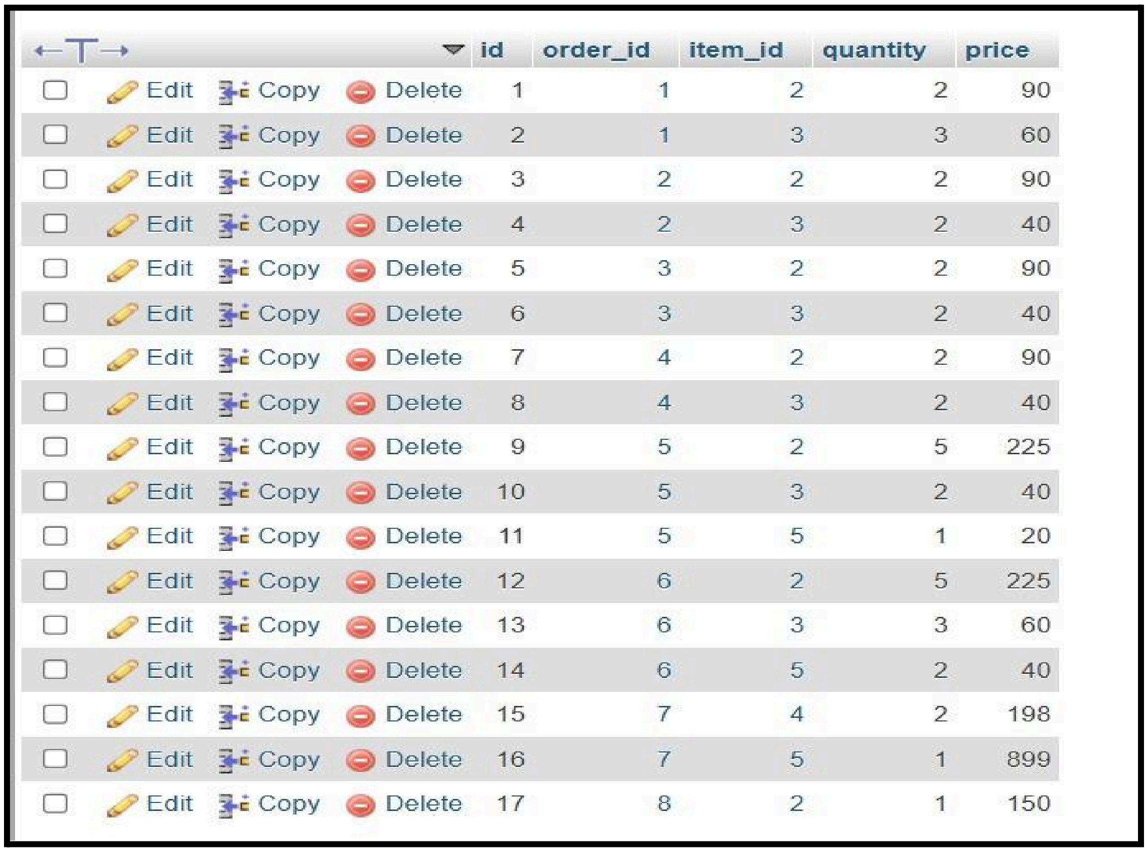
LIST

Order No. 9
Date: 2024-11-11 03:04:14
Payment Type: Cash On Delivery
Address: baranagar
Status: Yet to be delivered

#2 Bhetki Paturi	2 Pieces	Rs. 300
#4 Chicken Dak Bungalow	3 Pieces	Rs. 1047
Total		Rs. 1347

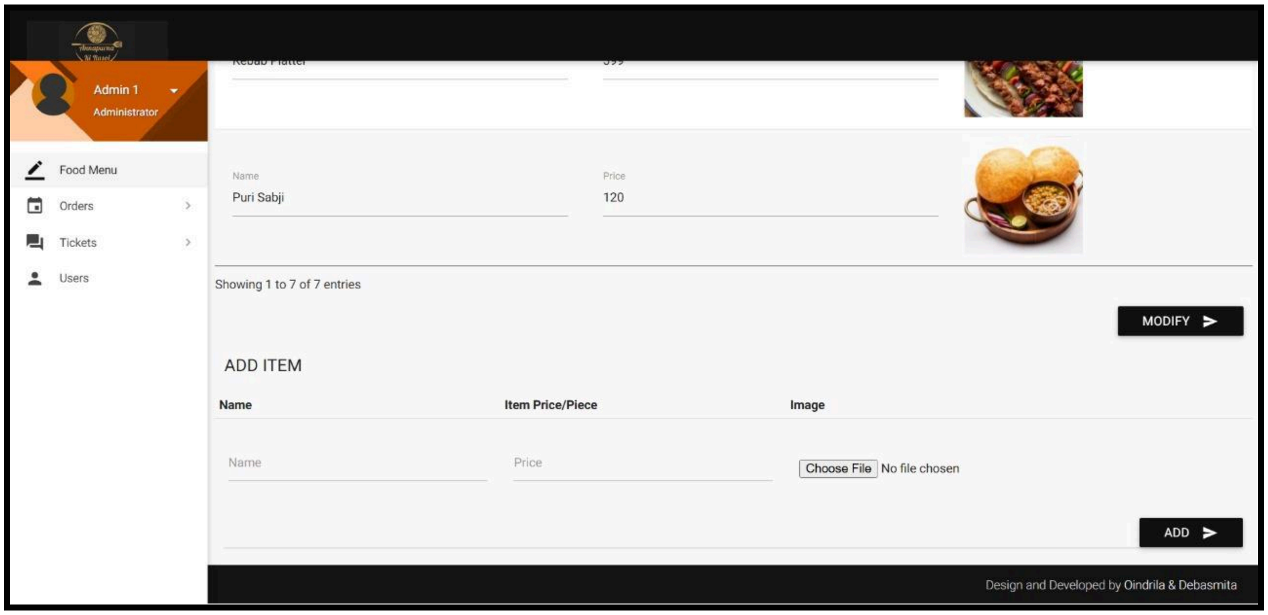
Design and Developed by Oindrila & Debasmita

Fig 6.3.10. COD Order Confirmed



	id	order_id	item_id	quantity	price
Edit Copy Delete	1	1	2	2	90
Edit Copy Delete	2	1	3	3	60
Edit Copy Delete	3	2	2	2	90
Edit Copy Delete	4	2	3	2	40
Edit Copy Delete	5	3	2	2	90
Edit Copy Delete	6	3	3	2	40
Edit Copy Delete	7	4	2	2	90
Edit Copy Delete	8	4	3	2	40
Edit Copy Delete	9	5	2	5	225
Edit Copy Delete	10	5	3	2	40
Edit Copy Delete	11	5	5	1	20
Edit Copy Delete	12	6	2	5	225
Edit Copy Delete	13	6	3	3	60
Edit Copy Delete	14	6	5	2	40
Edit Copy Delete	15	7	4	2	198
Edit Copy Delete	16	7	5	1	899
Edit Copy Delete	17	8	2	1	150

Fig 6.3.11. Database of Orders



Name
Puri Sabji
Price
120

Showing 1 to 7 of 7 entries

MODIFY

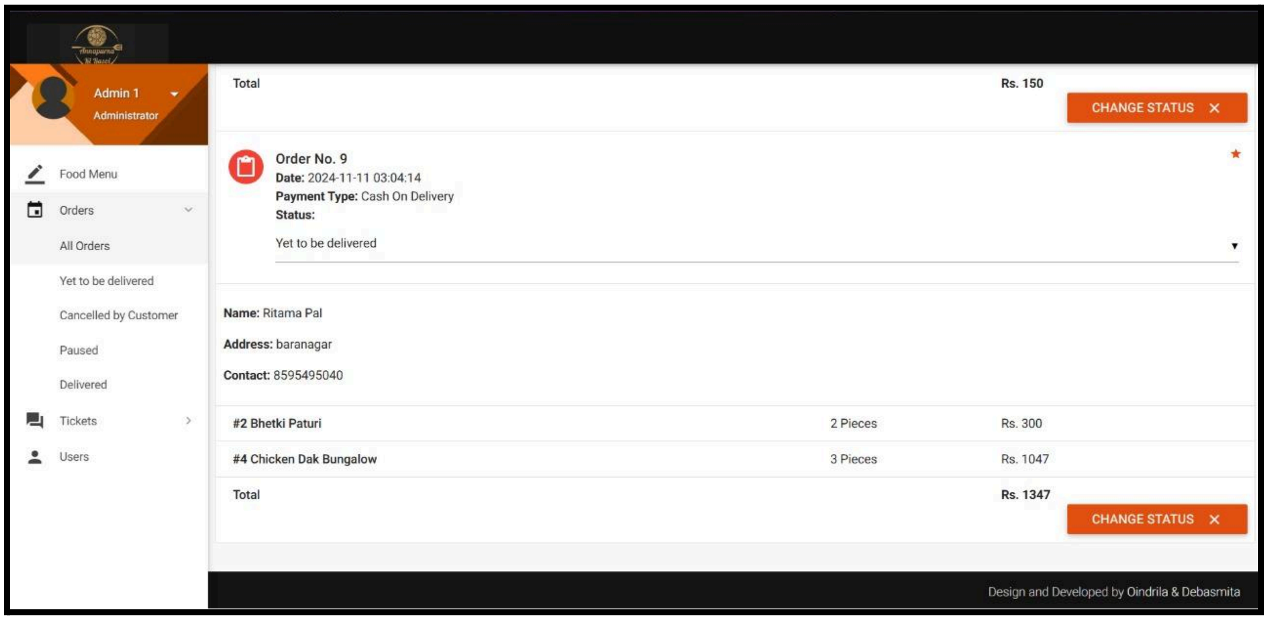
ADD ITEM

Name	Item Price/Piece	Image
Name	Price	Choose File No file chosen

ADD

Design and Developed by Oindrila & Debasmita

Fig 6.3.12. Admin menu



Total Rs. 150

CHANGE STATUS

Order No. 9
Date: 2024-11-11 03:04:14
Payment Type: Cash On Delivery
Status:
Yet to be delivered

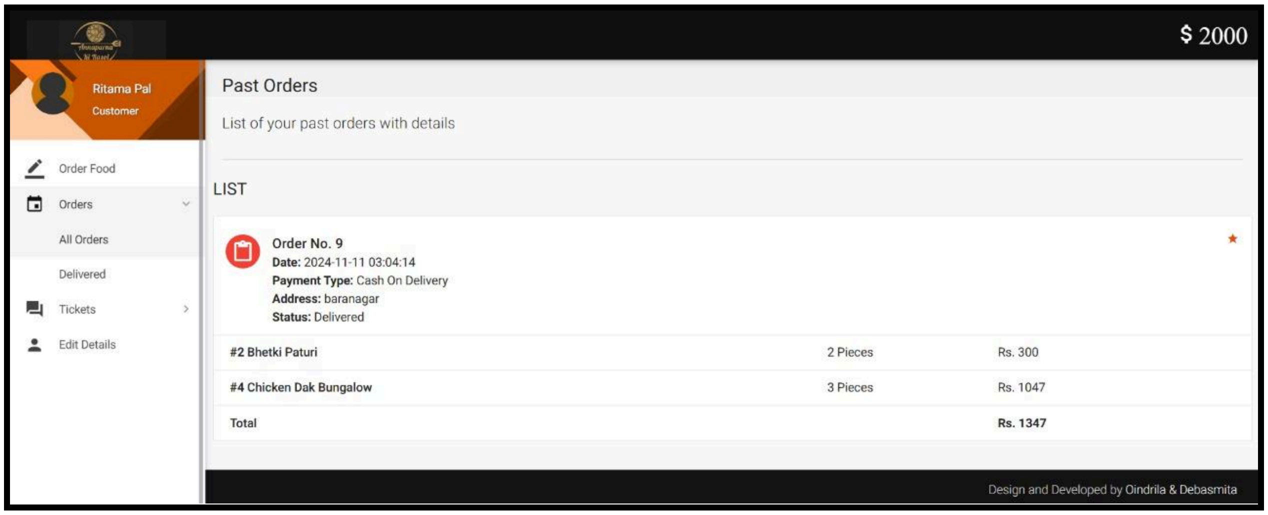
Name: Ritama Pal
Address: baranagar
Contact: 8595495040

#2 Bhetki Paturi	2 Pieces	Rs. 300
#4 Chicken Dak Bungalow	3 Pieces	Rs. 1047
Total		Rs. 1347

CHANGE STATUS

Design and Developed by Oindrila & Debasmita

Fig 6.3.13. Admin all orders



Past Orders

List of your past orders with details

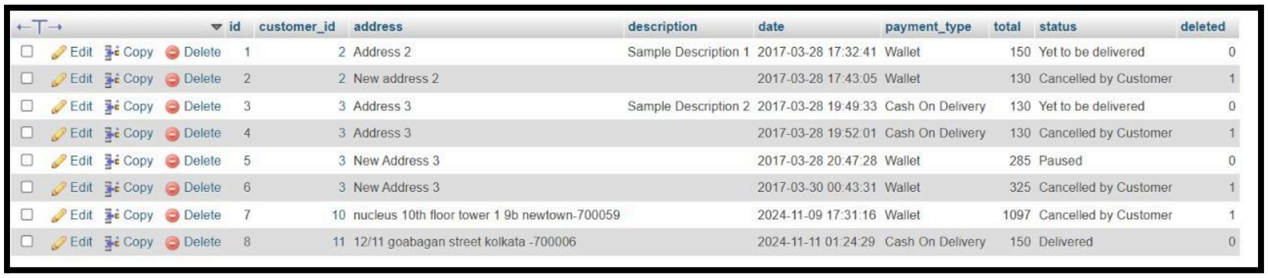
LIST

Order No. 9
Date: 2024-11-11 03:04:14
Payment Type: Cash On Delivery
Address: baranagar
Status: Delivered

#2 Bhetki Paturi	2 Pieces	Rs. 300
#4 Chicken Dak Bungalow	3 Pieces	Rs. 1047
Total		Rs. 1347

Design and Developed by Oindrila & Debasmita

Fig 6.3.14. Order Delivered



	id	customer_id	address	description	date	payment_type	total	status	deleted
Edit Copy Delete	1	2	Address 2	Sample Description 1	2017-03-28 17:32:41	Wallet	150	Yet to be delivered	0
Edit Copy Delete	2	2	New address 2		2017-03-28 17:43:05	Wallet	130	Cancelled by Customer	1
Edit Copy Delete	3	3	Address 3	Sample Description 2	2017-03-28 19:49:33	Cash On Delivery	130	Yet to be delivered	0
Edit Copy Delete	4	3	Address 3		2017-03-28 19:52:01	Cash On Delivery	130	Cancelled by Customer	1
Edit Copy Delete	5	3	New Address 3		2017-03-28 20:47:28	Wallet	285	Paused	0
Edit Copy Delete	6	3	New Address 3		2017-03-30 00:43:31	Wallet	325	Cancelled by Customer	1
Edit Copy Delete	7	10	nucleus 10th floor tower 1 9b newtown-700059		2024-11-09 17:31:16	Wallet	1097	Cancelled by Customer	1
Edit Copy Delete	8	11	12/11 goabagan street kolkata -700006		2024-11-11 01:24:29	Cash On Delivery	150	Delivered	0

Fig 6.3.15. Database of Customers

6.4. Limitations and Further works

Limitations :

- Scalability Issues: The given database structure is ineffective with the large volume of users and restaurants and orders, thus affecting performance and speed.
- Lack of Personalization: In the basic profile of user preference it would be somewhat difficult to gather more profound insights-based personalization regarding the ordering habits.
- Basic Payment Security: The transactions are made, but the security features are not robust to thwart a number of frauds and sophisticated security threats.
- Deliver Tracking Accuracy: Real-time tracking is limited due to the lack of updates sent with regard to the status of deliveries thereby affecting the scores for user satisfaction.

- Less Support for Advanced Analytics: There is only basic analytics concerning the behavior of a user, popular items and restaurants, hence impeding decisions for improvement.

Future Works :

- Scalable Database Solutions: Using more scalable databases such as NoSQL for big, unstructured data or sharded databases would take on high data volumes.
- Advanced Personalization: Including machine learning models that analyse user behavior and preferences will make for personalized recommendations and targeted promotions.
- Advanced Payment Security: Multi-layered security protocols like encryption and fraud detection algorithms would improve the security of transactions by the user.
- Real-Time Tracking. GPS-based tracking for real-time delivery updates will greatly enhance the accuracy of estimated times of deliveries and the user satisfaction levels.
- Comprehensive Analytics Dashboard. Restaurant and user trends would be provided by the comprehensive dashboard for administrators and partners to help them in better decision-making with data, thus increasing the efficiency and profitability quotient of the venture.

CHAPTER: 7

CONCLUSION

Based on the integration of the food ordering system into the model, this application shall strive to make food ordering experience much easier and efficient for its users by connecting them with their favorite restaurants in a user-friendly way. By structuring databases for the necessary entities—users, restaurants, orders, and payment details—the application shall therefore be able to handle user preferences, order histories, and also payment transactions. Although not scalable, fail-safe, or having real-time tracking for users, the company has set its base. Further advanced features from scalable databases to real GPS tracking, enhanced security, and robust data analytics will further enhance the service to deliver seamless, personalized user experience. These enhancements will be crucial in serving changing user demand, supporting restaurant partners, and further growing a safe, scalable, and reliable food delivery ecosystem as the company scales.

CHAPTER: 8

REFERENCES

1. Designing Data-Intensive Applications by Martin Kleppmann – Good on this one, as it dwells deep into data management and scalable database designs, perhaps necessary in the management of large orders and user information in an online food delivery application.

2. Database System Concepts by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan is a useful textbook that provides practical applications in database management principles that can be applied to an understanding of relational database structuring and optimization relevant for user data, order data, and restaurant data.

3. Web Application Architecture: Principles, Protocols and Practices by Leon Shklar and Rich Rosen – The book describes the principles of web architecture and server-client interactions to support the structure and design of a food delivery system implemented on the web.

4. Restaurant Delivery Channel Study 2020, by Deloitte – A study on the trend, consumer preference, and challenges of the food delivery sector, thereby informing the reader about user expectations and potential areas to be developed in food delivery applications.

5. Scaling Laravel by Antonio Ribeiro – This is a resource on scaling web applications built with Laravel as the underlying framework and will help build scalable high-performing food delivery applications.

6. Google Maps API – Google Map API is, so far the most popular method used in the development of geolocation and real-time tracking. It is quite useful in implementing location-based features with food delivery applications.

7. Payment Card Industry Data Security Standard (PCI DSS) Documentation – Official documentation on handling payment information, which is very important for safe handling of users' transactions and payment information within the application.

8. Usability Engineering by Jakob Nielsen – Outlines knowledge on how to design for the user and make the interface intuitive to use-a very important consideration for the food delivery applications that will increase the consumer's satisfaction.